

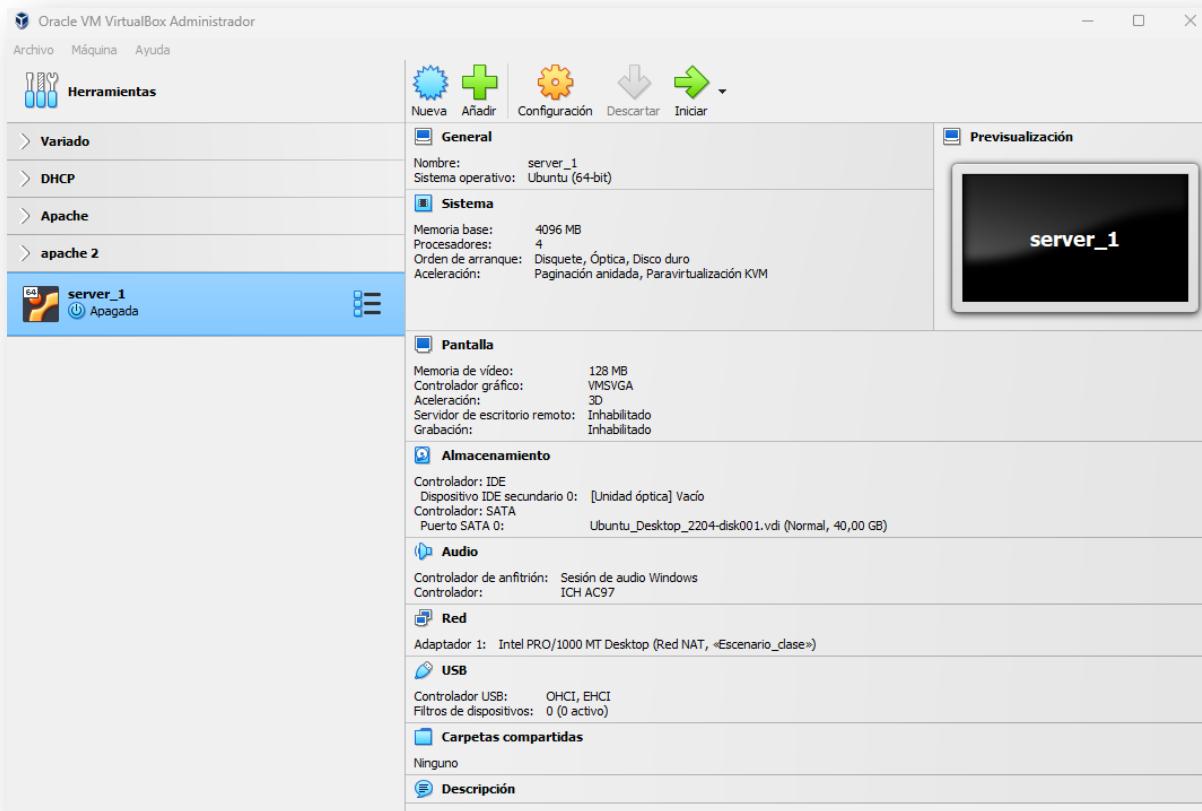


# Repasso segundo parcial

En este parcial hemos visto como configurar las IPs en los equipos de una red. También hemos visto algunos de los servicios habituales en redes como el DHCP para configuración automática de IPs, DNS para la resolución de nombres y APACHE como servidor web.

## Configuración

Configuración de la máquina virtual para “server\_1” (Ubuntu 22.04 Desktop). Ojo a la configuración de red porque todas las máquinas virtuales que usaremos deben estar en la misma red.



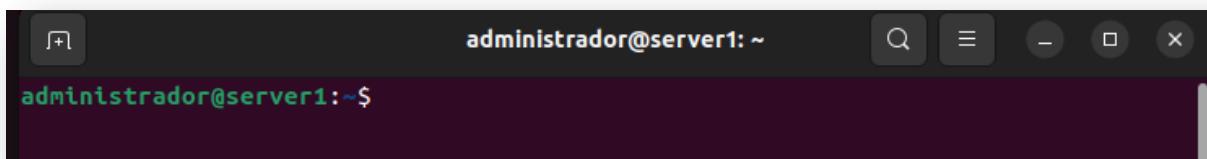
Para configurar de manera personalizada la máquina virtual que tenemos en la carpeta compartida realizaremos los siguientes pasos:

- Para desactivar el apt-cacher que apunta al repositorio de teis ejecutamos el siguiente comando: `sudo rm /etc/apt/apt.conf.d/01apt-cacher-ng`
- Para cambiar el nombre del equipo en el prompt ejecutamos el siguiente comando: `sudo nano /etc/hostname` y cambiamos el nombre por “server1”



- Para eliminar la referencia del nombre antiguo en el archivo “hosts” ejecutamos el siguiente comando: sudo nano /etc/hosts y sustituimos el nombre anterior al nuevo de “server1”
- Para parar el gestor de redes incorporado en Ubuntu Desktop escribimos el siguiente comando: sudo systemctl stop NetworkManager.service
- Para que no vuelva a lanzarse el NetworkManager en cada arranque ejecutamos el siguiente comando: sudo systemctl disable NetworkManager.service
- Con todos esos cambios hechos reiniciamos el equipo con sudo reboot

Desde este momento ya tenemos el equipo listo para ser configurado.



## Configuración IP del servidor

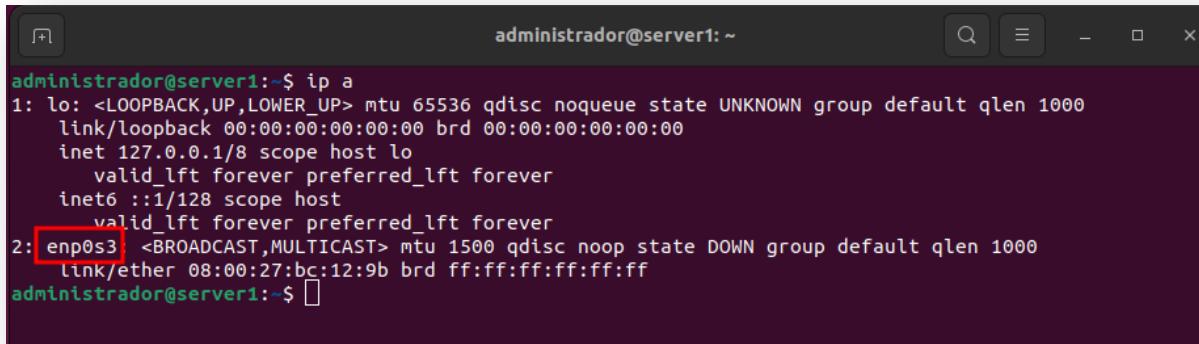
Los servidores siempre deberían tener una IP fija. En este caso, esta máquina es la única que tenemos en nuestra red por el momento así que debe ser configurada mediante NETPLAN.

Este programa viene incluido en todas las distribuciones Linux actuales. Anteriormente se usaba el programa “ifconfig” que ya no viene de serie, pero aún se puede instalar desde los repositorios de Ubuntu. Algunas versiones antiguas de “netplan” no incluyen algunas palabras reservadas como “default” y han eliminado otras como “gateway”

Antes de configurar una IP fija debemos tener claros los datos que vamos a incluir aunque es posible que aún no los tengamos todos y necesitemos editar de nuevo más adelante. Algunos de los datos que debemos tener claros son:

- Nombre interfaz de red
- IP
- Máscara
- Puerta de enlace
- Servidores DNS
- Dominio de búsqueda

La única excepción aquí es el nombre de la interfaz. Esto es algo hardware que depende del tipo y lugar de la tarjeta de red dentro del equipo. Para averiguar esta información debemos ejecutar el comando ip a.

```

administrador@server1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:bc:12:9b brd ff:ff:ff:ff:ff:ff
administrador@server1:~$ 

```

El resto de los parámetros son algo “administrativo” que debes decidir tú en función de la red que quieras configurar teniendo en cuenta la red NAT que has creado y/o la posible división en subredes, etc...

Para configurar netplan debemos editar el primer archivo (alfabéticamente) de la carpeta “/etc/netplan” con sudo nano /etc/netplan/01-network-manager-all.yaml

En mi caso configuraré lo siguiente:

- Nombre interfaz de red: enp0s3 (según captura anterior)
- IP: 192.168.100.10
- Máscara: 255.255.255.0 o /24
- Puerta de enlace: 192.168.100.1
- Servidores DNS: por ahora 8.8.8.8 para poder navegar, pero lo cambiaré después cuando configure mi propio DNS
- Dominio de búsqueda: red.clase



```
GNU nano 6.2          /etc/netplan/01-network-manager-all.yaml *
# Let NetworkManager manage all devices on this system
network:
  version: 2
  ethernets:
    enp0s3:
      addresses:
        - 192.168.100.10/24
      routes:
        - to: default
          via: 192.168.100.1
      nameservers:
        addresses:
          - 8.8.8.8
      search:
        - red.clase
```

Para probar la configuración ejecutamos un **sudo netplan try** y confirmamos con un **ip a**

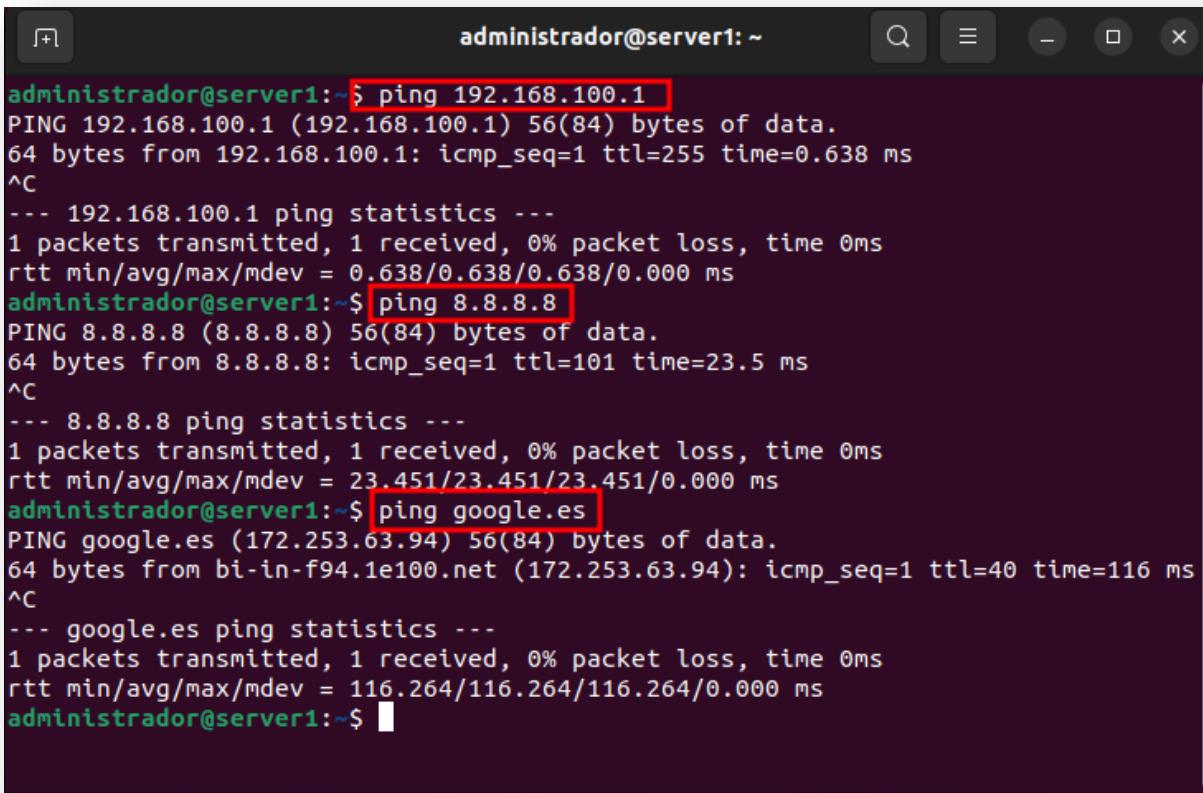
```
Do you want to keep these settings?

Press ENTER before the timeout to accept the new configuration

Changes will revert in 112 seconds
Configuration accepted.
administrador@server1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:bc:12:9b brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.10/24 brd 192.168.100.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:febcb129b/64 scope link
        valid_lft forever preferred_lft forever
administrador@server1:~$
```



Y para un 100% de seguridad hacemos un ping a la puerta de enlace, a un IP externa a nuestra red y a un nombre de dominio externo a nuestra red.



```

administrador@server1:~$ ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=255 time=0.638 ms
^C
--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.638/0.638/0.638/0.000 ms
administrador@server1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=101 time=23.5 ms
^C
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 23.451/23.451/23.451/0.000 ms
administrador@server1:~$ ping google.es
PING google.es (172.253.63.94) 56(84) bytes of data.
64 bytes from bi-in-f94.1.e100.net (172.253.63.94): icmp_seq=1 ttl=40 time=116 ms
^C
--- google.es ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 116.264/116.264/116.264/0.000 ms
administrador@server1:~$ 

```

## Instalación y configuración del servidor DHCP.

Este servicio se usa para poder configurar automáticamente las IPs en los equipos clientes de la red que lo soliciten. Normalmente encontraremos en estos equipos que está activado el modo “automático” o “DHCP”. Si es así, los equipos no tienen configurada ninguna IP en local (nadie va a editar su NETPLAN) y en cada arranque solicitarán a la red ayuda para configurarse una IP correcta. Esto lo hacen mediante una petición a la IP de broadcast puerto 67. El servidor que esté ejecutando el servicio DHCP estará escuchando en el puerto 67 peticiones de ayuda y contestará con la configuración de red pertinente para cada equipo. El programa que usaremos para el servidor DHCP se llama KEA. Lo instalaremos con el siguiente comando: **sudo apt install kea-dhcp4-server**

De la misma manera que antes usando NETPLAN, para la configuración de KEA también deberemos tener claro qué datos son los que se van a servir a los clientes que así lo soliciten. Pensemos que lo que estamos automatizando con KEA lo tendríamos que hacer con NETPLAN en cada uno de otros equipos de la red.



Los datos que debemos pasar son los mismos que anteriormente con NETPLAN a excepción del nombre de la interfaz (que usará el mismo de la tarjeta cliente que está solicitando configuración)

- IP: alguna del rango 192.168.100.100 al 192.168.100.200
- Máscara: 255.255.255.0 o /24
- Puerta de enlace: 192.168.100.1
- Servidores DNS: por ahora 8.8.8.8 para poder navegar, pero lo cambiaré después cuando configure mi propio DNS
- Dominio de búsqueda: red.clase

Cuando se tengan estos datos claros procedemos a configurar KEA mediante la edición del archivo de configuración con el comando: **sudo nano /etc/kea/kea-dhcp4.conf**



```
GNU nano 6.2                               /etc/kea/kea-dhcp4.conf *
{
  "Dhcp4": {
    "interfaces-config": { "interfaces": [ "enp0s3" ] },
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.100.0/24",
        "pools": [
          { "pool": "192.168.100.100 - 192.168.100.200" }
        ],
        "option-data": [
          { "name": "routers", "data": "192.168.100.1" },
          { "name": "domain-name-servers", "data": "8.8.8.8" },
          { "name": "domain-search", "data": "red.clase" }
        ]
      }
    ]
  }
}
```

Después de guardar la configuración debemos reiniciar el servicio de KEA y comprobar su estado. Esto lo hacemos con los comandos: **sudo systemctl restart kea-dhcp4-server.service** y **sudo systemctl status kea-dhcp4-server.service**.

Una salida correcta del estado del servicio sería la siguiente:

- kea-dhcp4-server.service - Kea IPv4 DHCP daemon

```
Loaded: loaded (/lib/systemd/system/kea-dhcp4-server.service; enabled; vendor preset: enabled)
```



Active: active (running) since Wed 2025-03-19 20:46:49 CET; 3s ago

Docs: man:kea-dhcp4(8)

Main PID: 5633 (kea-dhcp4)

Tasks: 5 (limit: 4625)

Memory: 2.6M

CPU: 17ms

CGroup: /system.slice/kea-dhcp4-server.service

```
└─5633 /usr/sbin/kea-dhcp4 -c /etc/kea/kea-dhcp4.conf
```

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.hosts/5633.140486245166720] HOSTS\_BACKENDS\_REGISTERED the following host backend types are available: mysql postgresql

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_CFGMGR\_ADD\_IFACE listening on interface enp0s3

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_CFGMGR\_SOCKET\_TYPE\_DEFAULT "dhcp-socket-type" not specified , using default socket type raw

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_CFGMGR\_NEW\_SUBNET4 a new subnet has been added to configuration: 192.168.100.0/24 with params: valid-lifetime=7200

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCP4\_CONFIG\_COMPLETE DHCPv4 server has completed configuration: added IPv4 subnets: 1; DDNS: disabled

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_MEMFILE\_DB opening memory file lease database: type=memfile universe=4

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.180 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_MEMFILELEASE\_FILE\_LOAD loading leases from file /var/lib/kea/kea-leases4.csv

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.181 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCPSRV\_MEMFILE\_LFC\_SETUP setting up the Lease File Cleanup interval to 3600 sec

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.196 WARN [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCP4\_MULTI\_THREADING\_INFO enabled: no, number of threads: 0, queue size: 0

Mar 19 20:46:49 server1 kea-dhcp4[5633]: 2025-03-19 20:46:49.196 INFO [kea-dhcp4.dhcpsrv/5633.140486245166720] DHCP4\_STARTED Kea DHCPv4 server version 2.0.2 started

Sería interesante ganar experiencia cometiendo errores en el archivo de configuración para saber interpretar las salidas del “status” y poder arreglarlo.

Para confirmar que tenemos el servicio funcionando podemos usar el comando ss que nos dará información sobre los puertos de escucha en el equipo (si lo ejecutamos con sudo nos dirá también el programa que escucha en el puerto)

**sudo ss -ltunp**



```
administrador@server1:~$ sudo ss -tulpn
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 0.0.0.0:631 0.0.0.0:*
0.0.0.0:*
users:(("cups-browsed",pid=700,fd=7))
0.0.0.0:45716 0.0.0.0:*
users:(("avahi-daemon",pid=517,fd=14))
0.0.0.0:5353 0.0.0.0:*
users:(("avahi-daemon",pid=517,fd=12))
0.0.0.0:53 0.0.0.0:*
users:(("systemd-resolve",pid=446,fd=13))
0.0.0.0:192.168.100.10:67 0.0.0.0:*
users:(("kea-dhcp4",pid=5633,fd=10))
0.0.0.0:55850 [:]:*
users:(("avahi-daemon",pid=517,fd=15))
0.0.0.0:5353 [:]:*
users:(("avahi-daemon",pid=517,fd=13))
tcp LISTEN 0 4096 127.0.0.53%lo:53 0.0.0.0:*
users:(("systemd-resolve",pid=446,fd=14))
tcp LISTEN 0 128 127.0.0.1:631 0.0.0.0:*
users:(("cupsd",pid=650,fd=7))
tcp LISTEN 0 128 [:]:631 [:]:*
users:(("cupsd",pid=650,fd=6))
administrador@server1:~$
```

La mejor manera de probarlo es usando otra máquina virtual que haga de equipo cliente. ¡Debemos prestar atención a configurar la máquina dentro de la misma red del servidor! Al usar una máquina virtual del IES Teis estaría bien eliminar el apt-cacher y cambiar el nombre del equipo a “cliente”. No será necesario modificar nada del networkManager porque este equipo es cliente y se configurará automáticamente gracias a nuestro nuevo servidor DHCP.

```
administrador@cliente:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:44:95:e3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.100.255 scope global dynamic noprefixroute enp0s3
        valid_lft 7172sec preferred_lft 7172sec
    inet6 fe80::89ba:345b:ce0e:632b/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
administrador@cliente:~$
```

Si todo ha ido bien tendremos nuestro equipo cliente configurado con los datos correctos. Además, podemos hacer unas pruebas mínimas para confirmar que el equipo navega correctamente haciendo pines a la puerta de enlace (funciona la red local), a una ip externa (funcionan las rutas) y a un dominio externo (funciona la resolución DNS)



```
administrador@cliente:~$ ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=255 time=0.297 ms
^C
--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.297/0.297/0.297/0.000 ms
administrador@cliente:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=101 time=23.4 ms
^C
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 23.406/23.406/23.406/0.000 ms
administrador@cliente:~$ ping google.es
PING google.es (172.253.63.94) 56(84) bytes of data.
64 bytes from bi-in-f94.1e100.net (172.253.63.94): icmp_seq=1 ttl=40 time=116 ms
^C
--- google.es ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 116.295/116.295/116.295/0.000 ms
administrador@cliente:~$
```

## Servidor DNS

Usaremos el programa BIND9 para nuestro servidor DNS. En esta parte hay que tener claro que una zona se corresponde con un determinado dominio y gestiona todos sus subdominios. En este ejemplo usaré varios dominios (red.clase, javi.es, javi.com, javi.net, javi.pt) que usaré tal cual. En algunos utilizaré también un subdominio www para conseguir URLs como [www.javi.es](http://www.javi.es)

Las zonas deben definirse en un archivo y enlazarlas con el archivo donde guardaremos los registros concretos de esa zona.

Como mínimo debemos tener 3 registros por zona, el SOA, un NS y un A con la IP del NS. A partir de ahí ya todos son registros A equivalentes a entradas de una agenda.

Hay una zona especial que se llama zona inversa que relaciona IPs con nombres. Se define igual con la diferencia de que el “nombre de dominio” se compone de la parte de red de la IP escrita a la inversa y acabada en “in-addr.arpa”

La instalación de BIND9 tan solo requiere del siguiente comando: **sudo apt install bind9**

Antes de configurar zonas vamos a adaptar BIND9 para el uso que le vamos a dar en este escenario.

Configuramos el archivo “/etc/default/named” para que use el RESOLVCONF y solo gestione IPv4



GNU nano 6.2 /etc/default/named

```
# # run resolvconf?
RESOLVCONF=yes

# startup options for the server
OPTIONS="-u bind -4"
```

También configuramos el archivo “/etc/bind/named.conf.options” para indicar la ruta de los archivos temporales de cesiones, los forwarders que usará si no puede resolver, y las interfaces (y redes) desde las que permite preguntar. El archivo quedará así:

GNU nano 6.2 /etc/bind/named.conf.options

```
options {
    directory "/var/cache/bind";

    forwarders { 8.8.8.8; };

    dnssec-validation no;

    listen-on { any; };

    allow-query { localhost; 192.168.100.0/24; };
};
```

La primera zona que voy a configurar es la de mi propia intranet que se llama “red.clase”. Si revisamos las configuraciones anteriores veremos que ya en el NETPLAN y en el DHCP tenía como dominio de búsqueda que completara con “red.clase” porque quiero que ese sea el nombre de mi dominio interno.

Definimos las zonas en el archivo “/etc/bind/named.conf.local”. En este caso voy a declarar la zona “red.clase” y su zona inversa.



GNU nano 6.2 /etc/bind/named.conf.local

```

zone "red.clase" IN {
    type master;
    file "/etc/bind/zonas/db.red.clase";
};

zone "100.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/zonas/db.100.168.192";
};

```

Recordemos que la zona inversa es la IP de red invertida y finalizando en “.in-addr.arpa”

Para cada zona debemos crear un archivo de registros en la misma ruta que hemos indicado.

Por ahora mi “red.clase” solo tiene una máquina y un router (que pone VirtualBox en la 192.168.100.1) así que solo dejaré dos entradas con nombre en esta zona, la del “server1” y la de “router” cada una apuntando a su IP correspondiente. Además, debo escribir los registros mínimos necesarios para la zona como el registro SOA, al menos un NS y al menos un A con la IP del NS.



GNU nano 6.2 /etc/bind/zonas/db.red.clase

```

; BIND data file for red.clase
;
$TTL    604800
@       IN      SOA    server1.red.clase. correo.red.clase. (
                        20250320      ; Serial
                        604800        ; Refresh
                        86400         ; Retry
                        2419200       ; Expire
                        604800 )      ; Negative Cache TTL
;
          IN      NS     server1.red.clase.
          IN      A      192.168.100.10
server1 IN      A      192.168.100.10
router  IN      A      192.168.100.1

```

La zona inversa es muy parecida teniendo en cuenta que en ella relacionamos IP con nombre. Hay que prestar atención a que los dominios de la última columna sean completos y por ello hay que terminarlos con punto.

GNU nano 6.2 /etc/bind/zonas/db.100.168.192

```

; BIND data file for inversa red 192.168.100.X
;
$TTL    604800
@       IN      SOA    server1.red.clase. correo.red.clase. (
                        20250320      ; Serial
                        604800        ; Refresh
                        86400         ; Retry
                        2419200       ; Expire
                        604800 )      ; Negative Cache TTL
;
@       IN      NS     server1.red.clase.
10     IN      PTR    server1.red.clase.
1      IN      PTR    router.red.clase.

```

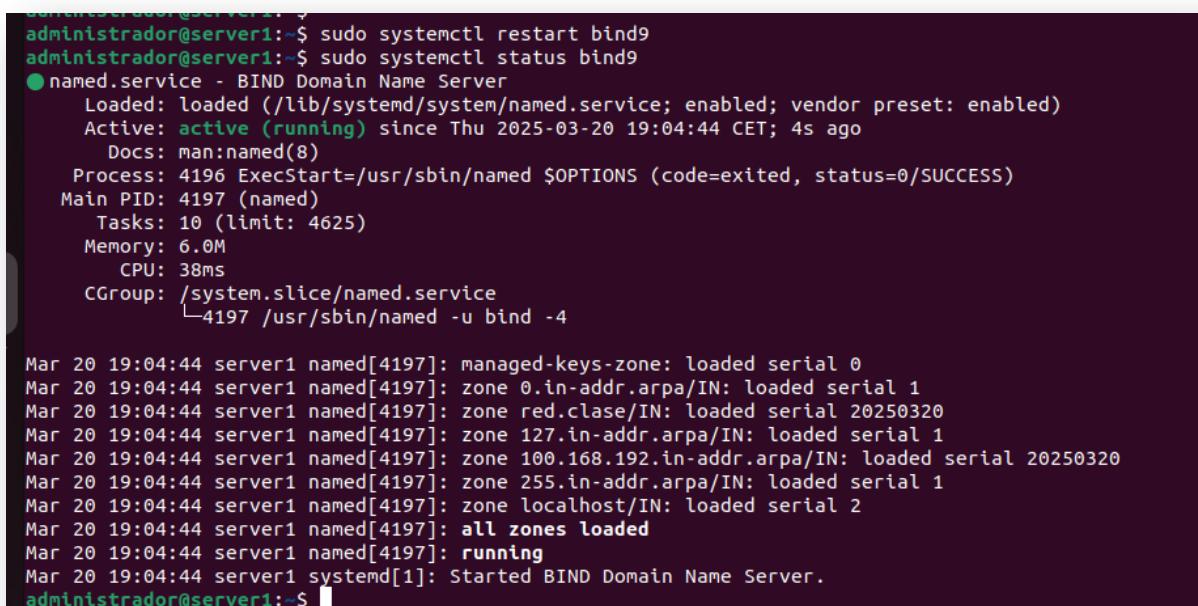


Tenemos los comandos named-checkconf y named-checkzone para comprobar sintácticamente los archivos de configuración y los de zona. Una salida correcta debería verse así:



```
administrador@server1: ~
administrador@server1: $ named-checkconf
administrador@server1: $ named-checkzone red.clase /etc/bind/zonas/db.red.clase
zone red.clase/IN: loaded serial 20250320
OK
administrador@server1: $ named-checkzone 100.168.192.in-addr.arpa /etc/bind/zonas/db.100.168.192
zone 100.168.192.in-addr.arpa/IN: loaded serial 20250320
OK
administrador@server1: $
```

Una vez confirmemos que los archivos de configuración están bien reiniciamos el servicio BIND9 con **sudo systemctl restart bind9** y comprobamos su estado con **sudo systemctl status bind9**. Una salida correcta se debería ver así:



```
administrador@server1: ~
administrador@server1: ~$ sudo systemctl restart bind9
administrador@server1: ~$ sudo systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2025-03-20 19:04:44 CET; 4s ago
     Docs: man:named(8)
   Process: 4196 ExecStart=/usr/sbin/named $OPTIONS (code=exited, status=0/SUCCESS)
    Main PID: 4197 (named)
      Tasks: 10 (limit: 4625)
     Memory: 6.0M
        CPU: 38ms
       CGroup: /system.slice/named.service
               └─4197 /usr/sbin/named -u bind -4

Mar 20 19:04:44 server1 named[4197]: managed-keys-zone: loaded serial 0
Mar 20 19:04:44 server1 named[4197]: zone 0.in-addr.arpa/IN: loaded serial 1
Mar 20 19:04:44 server1 named[4197]: zone red.clase/IN: loaded serial 20250320
Mar 20 19:04:44 server1 named[4197]: zone 127.in-addr.arpa/IN: loaded serial 1
Mar 20 19:04:44 server1 named[4197]: zone 100.168.192.in-addr.arpa/IN: loaded serial 20250320
Mar 20 19:04:44 server1 named[4197]: zone 255.in-addr.arpa/IN: loaded serial 1
Mar 20 19:04:44 server1 named[4197]: zone localhost/IN: loaded serial 2
Mar 20 19:04:44 server1 named[4197]: all zones loaded
Mar 20 19:04:44 server1 named[4197]: running
Mar 20 19:04:44 server1 systemd[1]: Started BIND Domain Name Server.
administrador@server1: ~$
```

Ahora mismo no es sencillo probarlo porque tanto “server1” como “cliente” tiene configurado el DNS en la 8.8.8.8 así que usaremos el comando nslookup para las pruebas.

Dentro de NSLOOKUP cambiaremos el servidor DNS al que hacer peticiones apuntando a nuestra IP para que se use este en lugar del 8.8.8.8. Esto lo haremos mediante “server 192.168.100.10”

Después haremos consultas a “router.red.clase” para confirmar que nuestro DNS resuelve bien (debe dar respuesta autoritativa), también probaremos a consultar solamente “router” para confirmar que el dominio de búsqueda que configuramos en NETPLAN funciona bien (debe dar



respuesta autoritativa) y también probaremos a resolver “Google.es” para confirmar que los resolvers configurados en BIND9 funcionan bien (la respuesta debe ser NO autoritativa porque nuestro DNS no tiene los registros de la zona Google.es). Por último, probaremos una ruta inversa preguntando qué nombre tiene la IP “192.168.100.10”.

```

administrador@server1:~$ nslookup
> server 192.168.100.10
Default server: 192.168.100.10
Address: 192.168.100.10#53
>
> router.red.clase
Server:      192.168.100.10
Address:     192.168.100.10#53

Name:   router.red.clase
Address: 192.168.100.1
>
> router
Server:      192.168.100.10
Address:     192.168.100.10#53

Name:   router.red.clase
Address: 192.168.100.1
>
> google.es
Server:      192.168.100.10
Address:     192.168.100.10#53

Non-authoritative answer:
Name:   google.es
Address: 172.253.63.94
Name:   google.es
Address: 2607:f8b0:4004:c08::5e
>
> 192.168.100.10
10.100.168.192.in-addr.arpa      name = server1.red.clase.
>

```

Es hora de cambiar NETPLAN para que use nuestro DNS en lugar del 8.8.8.8. Este cambio en NETPLAN solo afectará a nuestro servidor.



```

GNU nano 6.2                               /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  ethernets:
    enp0s3:
      addresses:
        - 192.168.100.10/24
      routes:
        - to: default
          via: 192.168.100.1
      nameservers:
        addresses:
          - 192.168.100.10
      search:
        - red.local

```

También cambiaremos KEA para que informe a los clientes que el DNS que usaremos en esta red es el 192.168.100.10

```

GNU nano 6.2                               /etc/kea/kea-dhcp4.conf
{
  "Dhcp4": {
    "interfaces-config": { "interfaces": [ "enp0s3" ] },
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.100.0/24",
        "pools": [
          { "pool": "192.168.100.100 - 192.168.100.200" }
        ],
        "option-data": [
          { "name": "routers", "data": "192.168.100.1" },
          { "name": "domain-name-servers", "data": "192.168.100.10" },
          { "name": "domain-search", "data": "red.clase" }
        ]
      }
    ]
  }
}

```

Todos estos cambios deben ser realizados en los archivos de configuración correspondientes y los servicios reiniciados para que lean la nueva configuración.



Es necesario tener en cuenta que los clientes que ya estén configurados mediante DHCP no solicitarán una nueva configuración y, por tanto, no recibirán los nuevos valores. Para tener la certeza de que la nueva configuración se transmita adecuadamente podemos apagar todo, arrancar el servidor y después el cliente.

Para comprobar que el nuevo DNS está funcionando en los clientes podemos usar la herramienta “nslookup” o simplemente hacer pines a nombres que solo conozca nuestro DNS (y a otros para comprobar los forwarders)

```

administrador@cliente:~$ ping router.red.clase
PING router.red.clase (192.168.100.1) 56(84) bytes of data.
64 bytes from router.red.clase (192.168.100.1): icmp_seq=1 ttl=255 time=0.296 ms
^C
--- router.red.clase ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.296/0.296/0.296/0.000 ms
administrador@cliente:~$ 
administrador@cliente:~$ ping router
PING router.red.clase (192.168.100.1) 56(84) bytes of data.
64 bytes from router.red.clase (192.168.100.1): icmp_seq=1 ttl=255 time=0.274 ms
^C
--- router.red.clase ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.274/0.274/0.274/0.000 ms
administrador@cliente:~$ 
administrador@cliente:~$ ping google.es
PING google.es (172.253.63.94) 56(84) bytes of data.
64 bytes from bi-in-f94.1e100.net (172.253.63.94): icmp_seq=1 ttl=40 time=116 ms
^C
--- google.es ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 116.091/116.091/116.091/0.000 ms
administrador@cliente:~$ 

```

## Instalación de APACHE2

Vamos a crear la página web de nuestra “red.clase” y la alojaremos en el “server1”. Para ello instalamos el servidor web Apache2 con el comando “sudo apt install apache2”.

Para alojar una web debemos crear un VirtualHost de Apache que es una instancia dedicada a atender las peticiones a un dominio o IP. Por defecto, Apache2 viene con un VirtualHost por defecto configurado en el archivo “000-default.conf” que atiende peticiones a la IP del servidor.

Nosotros vamos a crear un nuevo VirtualHost que atenderá las peticiones del dominio “red.clase”. Crearemos un archivo llamado “red.clase.conf” en la carpeta “sites-available” e indicaremos lo siguiente:

- ServerName: nombre del dominio (red.clase)
- DocumentRoot: ruta donde están los archivos HTML de la web de red.clase



- ErrorLog y CustomLog: ruta donde se guardarán los logs del servidor.

```
GNU nano 6.2                               /etc/apache2/sites-available/red.clase.conf

<VirtualHost *:80>

    ServerName red.clase
    DocumentRoot /var/www/red.clase/html
    ErrorLog /var/www/red.clase/logs/error.log
    CustomLog /var/www/red.clase/logs/access.log combined

</VirtualHost>
```

Probablemente las rutas que hemos elegido para guardar la web aún no existen y será necesario crearlas (comando mkdir). Dentro de la carpeta indicada en “DocumentRoot” dejaremos un archivo “index.html” con un “Hola mundo desde red.clase”

Para publicar la web debemos hacer un enlace desde “sites-available” a “sites-enabled” y para ello Apache2 dispone del comando: “sudo a2ensite red.local.clase”

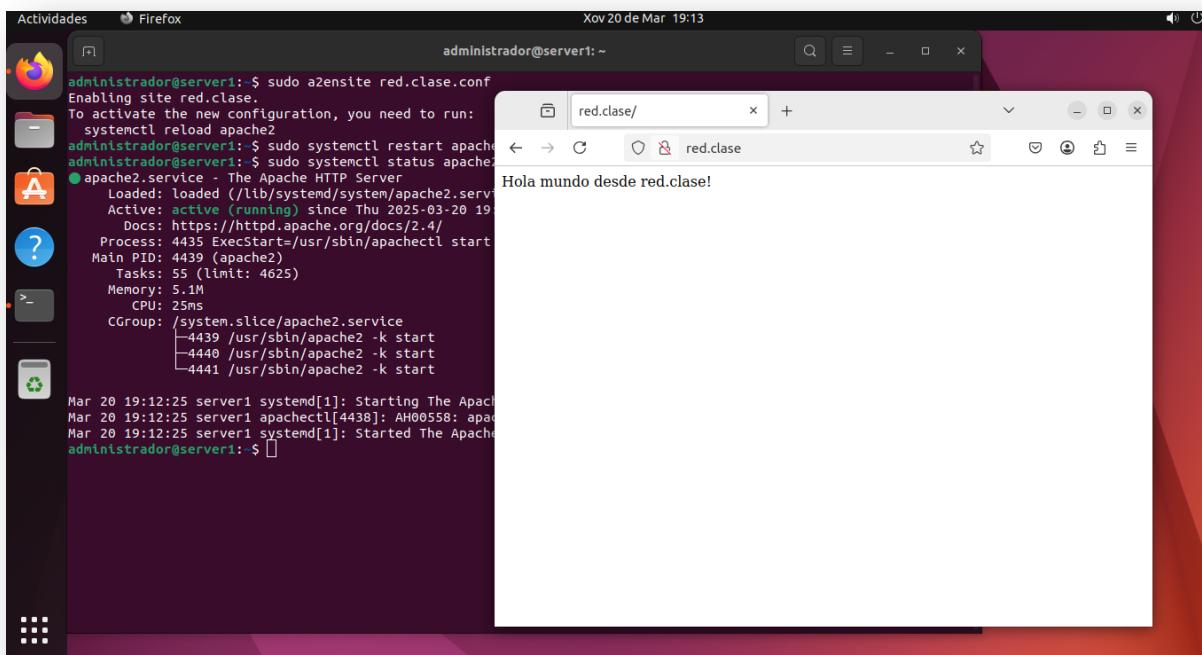
```
administrador@server1: ~
administrador@server1:~$ sudo a2ensite red.clase.conf
Enabling site red.clase.
To activate the new configuration, you need to run:
    systemctl reload apache2
administrador@server1:~$ sudo systemctl restart apache2
administrador@server1:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2025-03-20 19:12:25 CET; 7s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 4435 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 4439 (apache2)
   Tasks: 55 (limit: 4625)
   Memory: 5.1M
      CPU: 25ms
     CGroup: /system.slice/apache2.service
             └─4439 /usr/sbin/apache2 -k start
                  ├─4440 /usr/sbin/apache2 -k start
                  ├─4441 /usr/sbin/apache2 -k start

Mar 20 19:12:25 server1 systemd[1]: Starting The Apache HTTP Server...
Mar 20 19:12:25 server1 apachectl[4438]: AH00558: apache2: Could not reliably determine the server's fully qualified name, using server1. Set the 'ServerName' or 'ServerAlias' directive.
Mar 20 19:12:25 server1 systemd[1]: Started The Apache HTTP Server.
administrador@server1:~$
```

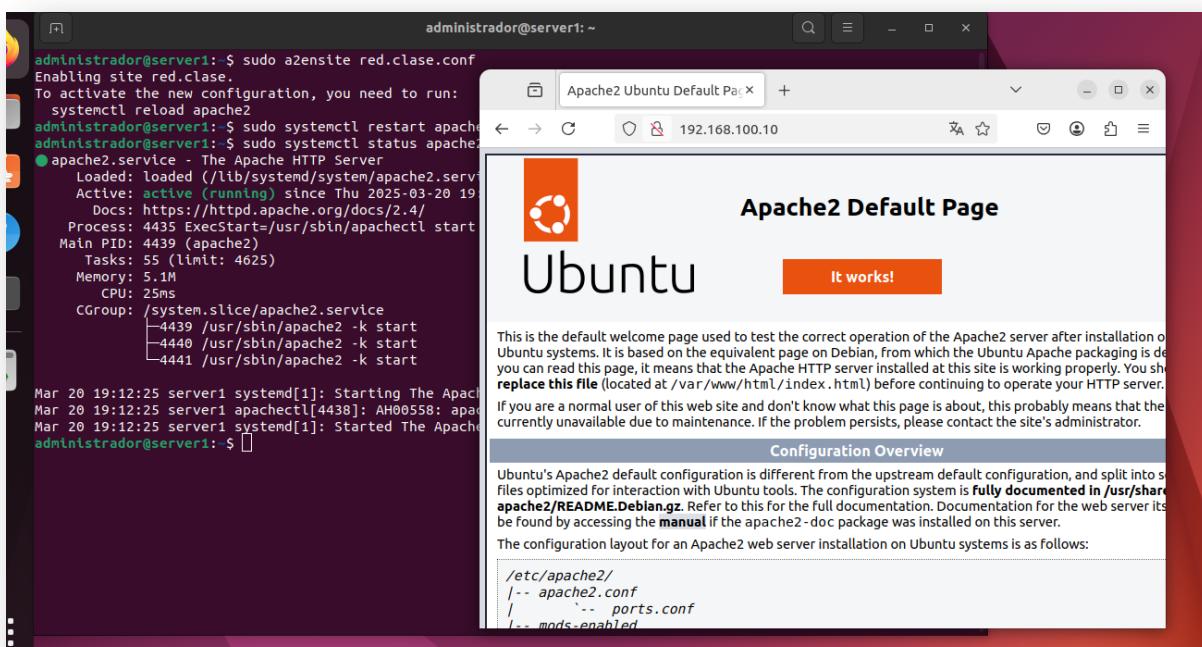
Al publicar una nueva web es necesario reiniciar el servicio o bien hacer un reload. La diferencia reside en que este último no corta las otras conexiones con otros VirtualHost que pudieran estar ejecutándose ahora mismo.



Para probarlo abrimos un navegador y escribimos en la barra de direcciones <http://red.clase>. Si todo va bien tendremos la web que hemos creado.



Si en lugar de escribir el dominio, escribimos directamente la IP del servidor (192.168.100.10) veremos la web por defecto porque ningún otro VirtualHost encaja con el dominio solicitado (ninguno porque hemos puesto una IP)





## AVANZADO

Vamos a crear un nuevo servidor web en nuestra red que usaremos para alojar otras páginas pertenecientes a otros dominios como javi.es, javi.com, etc.

Nos descargamos un Ubuntu Server 22.04. Si usamos el de IES Teis y lo personalizamos cambiando el nombre en el archivo “/etc/hostname”, la ruta propia en “/etc/hosts” y eliminando el “apt-caher” con el comando “sudo rm /etc/apt/apt.conf.d/01apt-cacher-ng”

Un servidor debería tener una IP fija. Ahora que tenemos un DHCP en “server1” podríamos aprovecharlo para hacer allí la reserva y que se configure solo. Hablaremos en clase sobre ventajas en inconvenientes de hacerlo mediante DHCP o NETPLAN.

Antes configuraremos el “server1” mediante NETPLAN. Ahora configuraremos el “server2” mediante una reserva en el DHCP. Necesitaremos la MAC de la interfaz de red del “server2”, la podremos obtener mediante el comando ip a.

```
administrator@server2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c2:90:30 brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.102/24 metric 100 brd 192.168.100.255 scope global dynamic enp0s3
            valid_lft 7044sec preferred_lft 7044sec
        inet6 fe80::a00:27ff:fec2:9030/64 scope link
            valid_lft forever preferred_lft forever
administrator@server2:~$
```

Con esta información ya podemos ir al “server1” donde está el servidor DHCP y registrar la reserva de esta MAC a una IP fija como por ejemplo la 192.168.100.20.

Una vez incluidos los nuevos datos, el archivo de configuración de KEA debería quedar así:



```

GNU nano 6.2                               /etc/kea/kea-dhcp4.conf

{
  "Dhcp4": {
    "interfaces-config": { "interfaces": [ "enp0s3" ] },
    "subnet4": [
      {
        "id": 1,
        "subnet": "192.168.100.0/24",
        "pools": [
          { "pool": "192.168.100.100 - 192.168.100.200" }
        ],
        "option-data": [
          {"name": "routers", "data": "192.168.100.1"},  

          {"name": "domain-name-servers", "data": "192.168.100.10"},  

          {"name": "domain-search", "data": "red.clase"}
        ],
        "reservations": [
          {
            "hw-address": "08:00:27:c2:90:30",
            "ip-address": "192.168.100.20"
          }
        ]
      }
    ]
  }
}

```

Para que los cambios surjan efecto debemos reiniciar el servicio DHCP en el “server1” con **sudo systemctl restart kea-dhcp4-server**.

También será necesario reiniciar “server2” con sudo reboot o forzar que vuelva a cargar la configuración de red con un **sudo netplan try**



```

        valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:c2:90:30 brd ff:ff:ff:ff:ff:ff
    1 | 192.168.100.102/24 metric 100 brd 192.168.100.255 scope global dynamic enp0s3
        valid_lft 7044sec preferred_lft 7044sec
    inet6 fe80::a00:27ff:fedc:9030/64 scope link
        valid_lft forever preferred_lft forever
administrador@server2:~$ sudo netplan try
[sudo] password for administrador:
Do you want to keep these settings?

Press ENTER before the timeout to accept the new configuration

Changes will revert in 118 seconds
Configuration accepted.
administrador@server2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:c2:90:30 brd ff:ff:ff:ff:ff:ff
    3 | 192.168.100.20/24 metric 100 brd 192.168.100.255 scope global dynamic enp0s3
        valid_lft 7196sec preferred_lft 7196sec
    inet6 fe80::a00:27ff:fedc:9030/64 scope link
        valid_lft forever preferred_lft forever
administrador@server2:~$
```

Ahora que ese nuevo servidor tiene una IP fija es el momento de asignarle un nombre en nuestro DNS interno del dominio red.clase, le llamaremos “server2” con IP 192.168.100.20. Para ello debemos editar el archivo donde guardamos los registros de la zona “red.clase” (“/etc/bind/zonas/db.red.clase”) y añadir el subdominio “server2” de la siguiente manera:



```
GNU nano 6.2                               /etc/bind/zonas/db.red.clase *

;
; BIND data file for red.clase
;

$TTL    604800
@       IN      SOA    server1.red.clase. correo.red.clase. (
                      20250320      ; Serial
                      604800        ; Refresh
                      86400         ; Retry
                      2419200       ; Expire
                      604800 )       ; Negative Cache TTL
;
          IN      NS     server1.red.clase.
          IN      A      192.168.100.10
server1 IN      A      192.168.100.10
server2 IN      A      192.168.100.20
router  IN      A      192.168.100.1
```

También aprovechamos y modificamos la zona inversa de la misma manera dejando su archivo de registros DNS de la siguiente manera:

```
GNU nano 6.2                               /etc/bind/zonas/db.100.168.192 *

;
; BIND data file for inversa red 192.168.100.X
;

$TTL    604800
@       IN      SOA    server1.red.clase. correo.red.clase. (
                      20250320      ; Serial
                      604800        ; Refresh
                      86400         ; Retry
                      2419200       ; Expire
                      604800 )       ; Negative Cache TTL
;
          IN      NS     server1.red.clase.
10      IN      PTR    server1.red.clase.
20      IN      PTR    server2.red.clase.
1       IN      PTR    router.red.clase.
```

Será necesario reiniciar el servicio DNS para que los cambios se carguen y tengan efecto. Este cambio funcionará sin problemas porque server2 es nuevo y nadie lo tiene cacheado. Si queremos cambiar la IP de un nombre que ya estaba asignado a otra IP podemos tener problemas con los caches...



```

administrador@server1:~$ sudo nano /etc/bind/zonas/db.red.clase
administrador@server1:~$ sudo nano /etc/bind/zonas/db.100.168.192
administrador@server1:~$ ping server2
ping: server2: Fallo temporal na resolución de nomes
administrador@server1:~$ sudo systemctl restart bind9
administrador@server1:~$ ping server2
PING server2.red.clase (192.168.100.20) 56(84) bytes of data.
64 bytes from server2.red.clase (192.168.100.20): icmp_seq=1 ttl=64 time=1.01 ms
^C
--- server2.red.clase ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.009/1.009/1.009/0.000 ms
administrador@server1:~$ 

```

En la captura anterior había modificado los archivos de registros de zonas e intenté hacer un ping al nombre del nuevo equipo. No funcionó porque aún no se había cargado los cambios. Reinicié el servicio y el ping ya funcionó.

El “server2” será un servidor web por lo que necesitamos instalar Apache2 allí. Lo haremos de la misma manera que en “server1”. Para tener una interfaz un poco más amigable que la que ofrece el Ubuntu server de “server2” podemos conectar a través de ssh desde cualquier equipo con interfaz gráfica mediante el comando ssh administrador@server2

Desde el equipo “server2” ejecutamos sudo apt install apache2.

Si queremos probarlo podemos abrir un navegador en un cliente y escribir en la barra de direcciones la IP del “server2” (<http://192.168.100.20>). Deberíamos ver la página por defecto de apache2



Ya lo tenemos todo listo para crear nuestro nuevo dominio “javi.es” alojado en este nuevo servidor. Aquí tendremos que resolver dos frentes:



- Configurar el DNS de “server1” para que redirija las peticiones de “javi.es” al “server2” que es donde está el servidor web que atenderá esas peticiones.
- Configurar el servidor web APACHE del “server2” para que tenga un VirtualHost que coincida con el dominio “javi.es” y sirva la web que corresponda.

Podemos hacerlo en cualquier orden, pero deben estar las dos cuestiones solucionadas para poder probarlo bien.

Empezaremos por el DNS creando una nueva zona llamada “javi.es”.

```
GNU nano 6.2                               /etc/bind/named.conf.local *
zone "red.clase" IN {
    type master;
    file "/etc/bind/zonas/db.red.clase";
};

zone "100.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/zonas/db.100.168.192";
};

zone "javi.es" IN {
    type master;
    file "/etc/bind/zonas/db.javi.es";
};
```

Además de crear la zona debemos especificar los registros propios de esta zona (de la inversa no) en la ruta que hayamos indicado.

En esta zona (“javi.es”) utilizaremos como SOA y como NS “server1.red.clase” que no tendremos que resolver porque eso ya lo hace su zona y además añadiremos una línea vacía (para indicar javi.es.) apuntando a la 192.168.100.20



```
GNU nano 6.2                               /etc/bind/zonas/db.javi.es

; BIND data file for javi.es
;

$TTL    604800
@       IN      SOA     server1.red.clase. correo.red.clase. (
                        20250320          ; Serial
                        604800            ; Refresh
                        86400             ; Retry
                        2419200           ; Expire
                        604800 )          ; Negative Cache TTL
;
        IN      NS      server1.red.clase.
        IN      A       192.168.100.20
```

Después de guardar, reiniciamos y hacemos alguna comprobación.

```
administrador@server1:~$ sudo nano /etc/bind/zonas/db.javi.es
administrador@server1:~$ named-checkzone javi.es /etc/bind/zonas/db.javi.es
zone javi.es/IN: loaded serial 20250320
OK
administrador@server1:~$ sudo systemctl restart bind
Failed to restart bind.service: Unit bind.service not found.
administrador@server1:~$ sudo systemctl restart bind9
administrador@server1:~$ ping javi.es
PING javi.es (192.168.100.20) 56(84) bytes of data.
64 bytes from server2.red.clase (192.168.100.20): icmp_seq=1 ttl=64 time=0.437 ms
64 bytes from server2.red.clase (192.168.100.20): icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from server2.red.clase (192.168.100.20): icmp_seq=3 ttl=64 time=0.990 ms
^C
--- javi.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2023ms
rtt min/avg/max/mdev = 0.437/0.870/1.183/0.316 ms
```

Ahora que ya está solucionado lo del DNS pasamos a configurar el VirtualHost en APACHE y para ello nos vamos al “server2”.

Debemos crear dentro de “sites-available” un archivo llamado “javi.es.conf” donde definimos el VirtualHost



```
GNU nano 6.2                               /etc/apache2/sites-available/javi.es.conf

<VirtualHost *:80>

    ServerName javi.es
    DocumentRoot /var/www/javi.es/html
    ErrorLog /var/www/javi.es/logs/error.log
    CustomLog /var/www/javi.es/logs/access.log combined

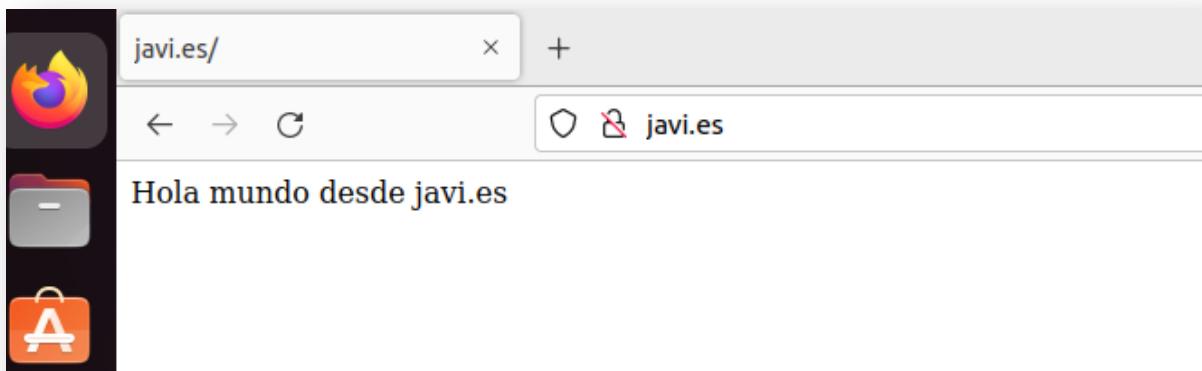
</VirtualHost>
```

Tenemos que crear las rutas que hemos definido (mkdir) y crear el archivo “index.html” con un “Hola mundo desde javi.es”.

Una vez hecho esto tendremos que publicar el sitio con el comando “sudo a2ensite javi.es.conf” y reiniciar el servidor apache con un “sudo systemctl reload apache2”

```
administrador@server2:~$ sudo nano /etc/apache2/sites-available/javi.es.conf
administrador@server2:~$ sudo mkdir /var/www/javi.es/html -p
administrador@server2:~$ sudo mkdir /var/www/javi.es/logs -p
administrador@server2:~$ sudo nano /var/www/javi.es/html/index.html
administrador@server2:~$ 
administrador@server2:~$ sudo a2ensite javi.es.conf
Enabling site javi.es.
To activate the new configuration, you need to run:
    systemctl reload apache2
administrador@server2:~$ sudo systemctl reload apache2
administrador@server2:~$
```

Solo queda probarlo en un navegador de otro equipo.





Toca repetir los mismos pasos para crear una web en inglés bajo el dominio javi.com. otra en gallego bajo el dominio javi.gal...

Primero: crear zonas.

```
GNU nano 6.2
zone "red.clase" IN {
    type master;
    file "/etc/bind/zonas/db.red.clase";
};

zone "100.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/zonas/db.100.168.192";
};

zone "javi.es" IN {
    type master;
    file "/etc/bind/zonas/db.javi.es";
};

zone "javi.com" IN {
    type master;
    file "/etc/bind/zonas/db.javi.com";
};

zone "javi.gal" IN {
    type master;
    file "/etc/bind/zonas/db.javi.gal";
};
```

Segundo: definir los registros de cada zona.



GNU nano 6.2 /etc/bind/zonas/db.javi.com

```
;;
; BIND data file for javi.gal
;
$TTL    604800
@       IN      SOA     server1.red.clase. correo.red.clase. (
                        20250320          ; Serial
                        604800            ; Refresh
                        86400             ; Retry
                        2419200           ; Expire
                        604800 )          ; Negative Cache TTL
;
IN      NS      server1.red.clase.
IN      A       192.168.100.20
```

### Tercero: comprobar y reiniciar el servicio DNS

administrador@server1:~\$ named-checkconf  
administrador@server1:~\$ named-checkzone javi.com /etc/bind/zonas/db.javi.com  
zone javi.com/IN: loaded serial 20250320  
OK  
administrador@server1:~\$ named-checkzone javi.gal /etc/bind/zonas/db.javi.gal  
zone javi.gal/IN: loaded serial 20250320  
OK  
administrador@server1:~\$ sudo systemctl restart bind9  
administrador@server1:~\$ sudo systemctl status bind9

● named.service - BIND Domain Name Server  
 Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)  
 Active: active (running) since Thu 2025-03-20 20:35:43 CET; 7s ago  
 Docs: man:named(8)  
 Process: 5388 ExecStart=/usr/sbin/named \$OPTIONS (code=exited, status=0/SUCCESS)  
 Main PID: 5389 (named)  
 Tasks: 10 (limit: 4625)  
 Memory: 6.2M  
 CPU: 35ms  
 CGroup: /system.slice/named.service  
 └─5389 /usr/sbin/named -u bind -4

```
Mar 20 20:35:43 server1 named[5389]: zone localhost/IN: loaded serial 2
Mar 20 20:35:43 server1 named[5389]: zone javi.es/IN: loaded serial 20250320
Mar 20 20:35:43 server1 named[5389]: zone 100.168.192.in-addr.arpa/IN: loaded serial 20250320
Mar 20 20:35:43 server1 named[5389]: zone 255.in-addr.arpa/IN: loaded serial 1
Mar 20 20:35:43 server1 named[5389]: zone javi.com/IN: loaded serial 20250320
Mar 20 20:35:43 server1 named[5389]: zone javi.gal/IN: loaded serial 20250320
Mar 20 20:35:43 server1 named[5389]: zone 0.in-addr.arpa/IN: loaded serial 1
Mar 20 20:35:43 server1 named[5389]: all zones loaded
Mar 20 20:35:43 server1 named[5389]: running
Mar 20 20:35:43 server1 systemd[1]: Started BIND Domain Name Server.
```

administrador@server1:~\$ █



Cuarto: en APACHE crear un archivo acabado en “.conf” para cada dominio donde se define su VirtualHost específico.

```
GNU nano 6.2                                         administrador@server2: ~
<VirtualHost *:80>                                /etc/apache2/sites-available/javi.com.conf

    ServerName javi.com
    DocumentRoot /var/www/javi.com/html
    ErrorLog /var/www/javi.com/logs/error.log
    CustomLog /var/www/javi.com/logs/access.log combined

</VirtualHost>
```

Quinto: Crear la estructura de carpetas y el archivo index.html de cada dominio.

```
administrador@server2:~$ sudo nano /etc/apache2/sites-available/javi.com.conf
administrador@server2:~$ sudo nano /etc/apache2/sites-available/javi.gal.conf
administrador@server2:~$ sudo mkdir /var/www/javi.com/html -p
administrador@server2:~$ sudo mkdir /var/www/javi.com/logs -p
administrador@server2:~$ sudo mkdir /var/www/javi.gal/html -p
administrador@server2:~$ sudo mkdir /var/www/javi.gal/logs -p
administrador@server2:~$ sudo nano /var/www/javi.com/html/index.html
administrador@server2:~$ sudo nano /var/www/javi.gal/html/index.html
administrador@server2:~$
```

Sexto: Publicas ambos dominios y reiniciar el servidor web



```
administrador@server2:~$ sudo a2ensite javi.com.conf
Enabling site javi.com.
To activate the new configuration, you need to run:
    systemctl reload apache2
administrador@server2:~$ sudo systemctl reload apache2
administrador@server2:~$ sudo a2ensite javi.gal.conf
Enabling site javi.gal.
To activate the new configuration, you need to run:
    systemctl reload apache2
administrador@server2:~$ sudo systemctl reload apache2
administrador@server2:~$
```

### Séptimo: Probar desde un navegador

