Bind9. Servidor primario, secundario y resolución inversa

Descripción

Vamos a instalar el servidor Bind9 para poder dar nombres a los equipos de una red interna en lugar de tener que utilizar las IPs.

Dar nombres (FQDNs) a los equipos facilita la configuración de servicios y aplicaciones, así como el mantenimiento de la propia red.

Instalar Bind9

Partimos de una máquina virtual con **Ubuntu 22.04 LTS**.

Instalamos el servidor con:

```
sudo apt update && sudo apt install bind9 -y
```

Tras instalar bind9, el servicio está activo

```
service bind9 status
```

nslookup y dig

Podemos verificar que está funcionando correctamente con:

```
nslookup google.com 127.0.0.1
```

Con este comando le estamos diciendo a nuestra máquina "127.0.0.1" que resuelva la dirección "google.com".

La respuesta será:

```
ubuntu@ubuntu-2204:~$ nslookup google.es 127.0.0.1
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.200.78
Name:   google.com
Address: 2a00:1450:4003:80d::200e
```

La herramienta **nslookup** es muy popular, pero es posible que nos aparezca un mensaje diciendo que está obsoleta (**deprecated**) y se nos recomienda utilizar **dig**, que es una herramienta desarrollada por el **Internet Systems Consortium** (ISC), quienes están detrás del desarrollo de ISC DHCP o de Bind9.

Así que vamos a hacerles caso y a repetir la pregunta utilizando **dig**:

```
dig @127.0.0.1 google.es
```

Y obtendremos la siguiente salida:

```
; <<>> DiG 9.16.1-Ubuntu <<>> @127.0.0.1 google.es
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41997
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ee7c03774c74ca2c0100000063624015403328 77f5a29f64 (good)
;; QUESTION SECTION:
;google.es.                     IN      A

;; ANSWER SECTION:
google.es.              300     IN      A       142.250.184.163

;; Query time: 407 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 02 06:01:57 EDT 2022
;; MSG SIZE  rcvd: 82
```

Otra herramienta que podemos utilizar para realizar consultas es **host**.

Si escribimos:

```
host google.es 127.0.0.1
```

Obtendremos la siguiente salida:

```
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

google.es has address 142.250.184.163
google.es has IPv6 address 2a00:1450:4003:80c::2003
google.es mail is handled by 0 smtp.google.com.
```

Para resoluciones simples (nombres a IPs), podemos utilizar también la herramienta **ping**. Aunque ping es una herramienta para verificar la conectividad entre equipos:

```
ping google.es
```

Obtendremos la siguiente salida:

```
PING google.es (142.250.178.163) 56(84) bytes of data.
64 bytes from mad41s08-in-f3.1e100.net (142.250.178.163): icmp_seq=1 ttl=116 time=17.5 ms
64 bytes from mad41s08-in-f3.1e100.net (142.250.178.163): icmp_seq=2 ttl=116 time=18.6 ms
...
```

Además, a **ping** no le hemos especificado el servidor DNS que debe de utilizar, por lo que en este caso para la traducción estará utilizando el **resolver** de la máquina virtual y no **Bind9**.

> Este tutorial está centrado en la **instalación y configuración de Bind9**, no en las herramientas de consulta y diagnóstico DNS. De todas formas, para más información, se puede leer la documentación de cada herramienta con:
>
> ```
> man <nombre-herramienta>
> ```

Servidor caché

Si realizamos la siguiente consulta:

```
dig @127.0.0.1 facebook.com
```

Observaremos que el servidor tarda un tiempo en realizarla, en este caso **736 msec**.

```
...
;; Query time: 736 msec
...
```

Si repetimos la consulta, la resolución será instantánea:

```
...
;; Query time: 0 msec
...
```

Esto ilustra claramente el comportamiento del servidor como **caché**.

Configuración de IP estática y servidor DNS

Los equipos especiales como son los servidores suelen tener una configuración estática o una dirección reservada en DHCP. Nosotros vamos a configurar este servidor en **netplan** con una **IP estática**. Para ello creamos el archivo **/etc/netplan/01-cfg-static.yaml**.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
      - 192.168.31.2/24
      routes:
      - to: default
        via: 192.168.31.1
      nameservers:
        addresses: [192.168.31.2]
```

- Le asignamos como dirección IP **192.168.31.2**.
- Le asignamos como gateway la dirección del router **192.168.31.1**.
- Le asignamos como servidor DNS la propia IP del equipo **192.168.31.2**.

Y aplicamos la configuración:

```
sudo netplan apply
```

Y comprobamos que ya tenemos la dirección IP con el comando **ip a**.

```
ubuntu@ubuntu2004:/etc/bind$ 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 08:00:27:52:b9:a0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.2/24 brd 192.168.31.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe52:b9a0/64 scope link tentative
       valid_lft forever preferred_lft forever
```

Establecer Bind9 como el resolutor del equipo

Antiguamente los resolutores de DNS se establecían a mano en el archivo **/etc/resolv.conf**.

Si hacemos un **cat** al archivo **/etc/resolv.conf** veremos lo siguiente:

```
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
```

Se nos indica que **no se debe de editar este archivo** ya que está manejado por el resolver de **systemd-resolved**. Y que podemos emplear el comando **resolvectl status** para ver detalles acerca de los servidores DNS en uso.

Ubuntu dispone de un **servidor DNS Stub** en la dirección **127.0.0.53**, que simplemente reenvía todas las consultas que le lleguen a otro (por ejemplo, al que indiquemos en la configuración de netplan o de NetworkManager, en el caso de que configuremos los servidores a través del entorno gráfico). Por

eso, cuando realicemos **consultas DNS** la respuesta nos vendrá de este servidor DNS stub que actúa como intermediario. Hace el papel del **resolver**.

Para ver a dónde este servidor DNS Stub envía las consultas podemos teclear:

```
resolvectl status
```

Y obtendremos como salida algo parecido a:

```
Global
       LLMNR setting: no
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
          DNSSEC NTA: 10.in-addr.arpa
                      16.172.in-addr.arpa
                      168.192.in-addr.arpa
                      17.172.in-addr.arpa
                      18.172.in-addr.arpa
                      19.172.in-addr.arpa
                      20.172.in-addr.arpa
                      21.172.in-addr.arpa
                      22.172.in-addr.arpa
                      23.172.in-addr.arpa
                      24.172.in-addr.arpa
                      25.172.in-addr.arpa
                      26.172.in-addr.arpa
                      27.172.in-addr.arpa
                      28.172.in-addr.arpa
                      29.172.in-addr.arpa
                      30.172.in-addr.arpa
                      31.172.in-addr.arpa
                      corp
                      d.f.ip6.arpa
                      home
                      internal
                      intranet
                      lan
                      local
                      private
                      test

Link 2 (enp0s3)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 192.168.31.2
         DNS Servers: 192.168.31.2
          DNS Domain: ~.
```

Donde vemos que el servidor DNS para la interfaz **enp0s3** es el que configuramos en **netplan**, el **192.168.31.2**.

En Internet podemos encontrar **guías o tutoriales anticuados** con los que utilizar Bind9:

- Algunos pueden indicar que se debe editar el archivo **/etc/resolv.conf** a mano y establecer ahí los servidores DNS.
  - Esto ya no se hace desde **Ubuntu 12.04**.
- Otros pueden indicar que se debe editar el archivo **/etc/default/bind9** y poner **RESOLVCONF=yes** para utilizar el propio Bind9 como resolutor.
  - Esto no funciona en sistemas modernos que utilicen **systemd**.
  - Además, el archivo ya no se llama **/etc/default/bind9**, se llama **/etc/default/named**, que es el nombre del daemon que usa Bind9.

Configuración de Bind9

Ahora que ya sabemos que nuestro servidor funciona correctamente, tendremos que configurarlo de acuejrdo al uso que le vayamos a dar.

Una regla para el firewall

Pero para evitar problemas, añadimos una regla en el **firewall** que permita trabajar a **Bind9** a través de él:

```
sudo ufw allow Bind9
```

Archivos de configuración

Los archivos de configuración de Bind9 se encuentra en la carpeta **/etc/bind/**. Podemos listarlos con el comando:

```
ls -l /etc/bind
```

Nos encontraremos con los siguientes archivos:

```
total 48
-rw-r--r-- 1 root root 1991 Sep 20 08:05 bind.keys
-rw-r--r-- 1 root root  237 Dec 17  2019 db.0
-rw-r--r-- 1 root root  271 Dec 17  2019 db.127
-rw-r--r-- 1 root root  237 Dec 17  2019 db.255
-rw-r--r-- 1 root root  353 Dec 17  2019 db.empty
-rw-r--r-- 1 root root  270 Dec 17  2019 db.local
-rw-r--r-- 1 root bind  463 Dec 17  2019 named.conf
-rw-r--r-- 1 root bind  498 Dec 17  2019 named.conf.default-zones
-rw-r--r-- 1 root bind  165 Dec 17  2019 named.conf.local
-rw-r--r-- 1 root bind  846 Dec 17  2019 named.conf.options
-rw-r----- 1 bind bind  100 Nov  1 07:33 rndc.key
-rw-r--r-- 1 root root 1317 Dec 17  2019 zones.rfc1918
```

Algunos de ellos comienzan con **named** porque es el nombre del **daemon** que corre Bind9.

bind.keys

- Cuando el daemon **named** comienza, necesita cierta información como por ejemplo **cómo llegar a los servidores raíz**, antes de que pueda responder a consultas recursivas.
- Si **named** está configurado para realizar la validación de **DNSSEC**, también debe tener **trust anchors** (anclajes de confianza) iniciales.
- Esta información se puede configurar a través del archivo **named.conf**, pero **ISC** ha intentado simplificar los archivos de configuración compilando esta información a parte.
- No es un archivo que se tenga que editar a mano.
- **Más información**: bind.keys

```
#
# This file is NOT expected to be user-configured.
#
# Servers being set up for the first time can use the contents of this file
# as initializing keys; thereafter, the keys in the managed key database
# will be trusted and maintained automatically.
#
# These keys are current as of Mar 2019.  If any key fails to initialize
# correctly, it may have expired.  In that event you should replace this
# file with a current version.  The latest version of bind.keys can always
# be obtained from ISC at https://www.isc.org/bind-keys.
#
# See https://data.iana.org/root-anchors/root-anchors.xml for current trust
# anchor information for the root zone.

trust-anchors {
        # This key (20326) was published in the root zone in 2017.
        . initial-key 257 3 8 "AwEAAaz/tAm8yTn4Mfeh5eyI96WSVexTBAvkMgJzkKTOiW1vkIbzxeF3
                +/4RgWOq7HrxRixHlFlExOLAJr5emLvN7SWXgnLh4+B5xQlNVz8Og8kv
                ArMtNROxVQuCaSnIDdD5LKyWbRd2n9WGe2R8PzgCmr3EgVLrjyBxWezF
                0jLHwVN8efS3rCj/EWgvIWgb9tarpVUDK/b58Da+sqqls3eNbuv7pr+e
                oZG+SrDK6nWeL3c6H5Apxz7LjVc1uTIdsIXxuOLYA4/ilBmSVIzuDWfd
                RUfhHdY6+cn8HFRm+2hM8AnXGXws9555KrUB5qihylGa8subX2Nn6UwN
                R1AkUTV74bU=";
};
```

db.0 y db.255

- Son **db.0** y **db.255** son los archivos de resolución inversa para la **zona de broadcast**.

db.0

```
;
; BIND reverse data file for broadcast zone
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
```

db.255

```
;
; BIND reverse data file for broadcast zone
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
```

db.local y db.127

- **db.local** es el **archivo de resolución** y **db.127** es el **archivo de resolución inversa** para la **interface loopback local**.

db.local

```
;
; BIND reverse data file for broadcast zone
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
```

db.127

```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
1.0.0   IN      PTR     localhost.
```

db.empty

- Es el archivo de resolución inversa para la **zona vacía** especificada en la rfc1918 (**IPs Privadas**).
- Es un archivo que no se debe de editar a mano.

Bind proporciona una serie de **zonas vacías** que se cargan y configuran automáticamente cuando se inicia **named**. El propósito de estas zonas es **evitar que los servidores recursivos envíen consultas sin sentido** a los servidores de Internet que no pueden manejaras (lo que crea retrasos y respuestas **SERVFAIL** a los clientes que las consultan). Las zonas vacías garantizan que, en su lugar, se devueltan respuestas **NXDOMAIN** inmediatas y autorizadas.

- El mensaje **NXDOMAIN** es aquél que recibe un resolver (cliente DNS) cuando una consulta no puede ser resuelta a una dirección IP.

Más información: Automatic Empty zones

```
; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL    86400
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
```

Este archivo lo utilizaremos como **plantilla** para crear **nuestros propios archivos de zona**.

named.conf

El archivo **/etc/bind/named.conf** es el archivo de configuración principal de Bind9, pero **no lo editaremos a mano**.

Fijándonos en su contenido podemos ver que divide la configuración en otros archivos que sí editaremos.

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

En este enlace se puede consultar la **sintaxis y las declaraciones** que se pueden llevar a cabo en el fichero.

Hay que tener mucho ojo y evitar los errores sintácticos, ya que pueden impedir que el servicio **named** arranque.

De todas formas, con el comando **named-checkconf** podemos comprobar si hemos cometido errores al editar el fichero. El comando verificará si el archivo de configuración es correcto.

```
named-checkconf /etc/named.conf
```

named.conf.options

En este archivo se definen las **opciones de configuración** del servidor DNS.

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

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation auto;

        listen-on-v6 { any; };

        recursion { localhost; };
};
```

named.conf.local

En este archivo se lleva a cabo la **configuración local**, y aquí se pueden añadir las zonas vacías de la RFC 1918 si no están siendo usadas por la organización.

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

named.conf.default-zones

En este archivo se configuran las **zonas por defecto**: servidores raíz, zonas de reenvío o zonas de broadcast.

```
// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/usr/share/dns/root.hints";
};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};
```

rndc.key

Es un archivo relacionado con el Remote Name Daemon Control. Nosotros no lo utilizaremos.

zones.rfc1918

Es un archivo con las direcciones de las redes privadas de la RFC 1918 para prevenir a los servidores recursivos mandar consultas que no van a ser resueltas, como ya se comentó antes.

```
zone "10.in-addr.arpa"      { type master; file "/etc/bind/db.empty"; };

zone "16.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "17.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "18.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "19.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "20.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "21.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "22.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "23.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "24.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "25.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "26.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "27.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "28.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "29.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "30.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };
zone "31.172.in-addr.arpa"  { type master; file "/etc/bind/db.empty"; };

zone "168.192.in-addr.arpa" { type master; file "/etc/bind/db.empty"; };
```

Editando la configuración

Editando named.conf.options

Vamos a editar el archivo **named.conf.options** y configurar algunas opciones:

listen-on

Con la directiva **listen-on** podriamos especificar las redes donde el servidor va a **escuchar peticiones** y el puerto.

Así podríamos decirle que escuche en todas las interfaces:

```
listen-on { any; };
```

Si quisiéramos que escuche solo en algunas redes podríamos poner:

```
listen-on {
        10.10.10.0/24;
        10.1.0.0/16;
...
};
```

Nosotros vamos a poner:

```
listen-on { any; };
```

Para IPv6 sería:

```
listen-on-v6 { any; };
```

allow-query

Bind9 permite **peticiones locales** por defecto. Podríamos especificar qué redes o equipos pueden utilizar el servidor para realizar consultas.

Así permitiríamos solo las consultas del equipo 192.168.31.54:

```
allow-query { 192.168.31.54; };
```

Y así permitiríamos la consulta de cualquier equipo.

```
allow-query { any; };
```

Nosotros permitiremos la consulta de cualquier equipo.

forwarders

La directiva **forwarders** contiene la dirección de los servidores a los que Bind9 **redirigirá la petición en el caso de que no pueda darnos una respuesta**.

```
forwarders {
        8.8.8.8;
        8.8.4.4;
};
```

En este caso estaríamos configurando como forwarders los servidores **DNS de Google**.

Nuestra configuración

Nosotros dejaremos el archivo de configuración con el siguiente contenido:

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

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation auto;

        allow-query { any; };
        listen-on { any; };
        listen-on-v6 { any; };

};
```

Podemos comprobar que la configuración está bien con el comando:

```
named-checkconf
```

Si no aparecen errores, podemos reiniciar el servicio y la configuración comenzará a tener efecto:

```
sudo service bind9 restart
```

Testeando el servicio

Para ver si el servidor está funcionando correctamente podremos utilizar el siguiente comando desde otra máquina. Lo haré desde mi **máquina host** con el comando **nslookup**.

```
nslookup ubuntu.com 192.168.31.2
```

```
Server:  UnKnown
Address:  192.168.31.2

Non-authoritative answer:
Name:    ubuntu.com
Addresses:  2620:2d:4000:1::27
          2620:2d:4000:1::26
          2620:2d:4000:1::28
          185.125.190.21
          185.125.190.20
          185.125.190.29
```

Ahora que ya tenemos configurado un servidor Bind9. Vamos a apagar la máquina y utilizarla como "base" para los siguientes pasos.

Servidores primarios y secundarios

Configuración de equipos

Vamos ahora a configurar dos servidores, uno **primario** y otro **secundario** en la red **192.168.31.0/24**.

- La IP de mi **servidor DNS primario** será: **192.168.31.2**
- La de mi **servidor DNS secundario** será: **192.168.31.3**
- La IP del **gateway** (mi router) es: **192.168.31.1**
- La IP de mi **equipo host** (donde ejecuto las máquinas virtuales) me ha sido concedida por el **servidor DHCP** del router. Y no tiene mayor importancia, salvo que coincida con alguno de los servidores DNS. En general, muchos routers comienzan a conceder IPs dentro de un rango a partir de un número (por ejemplo del .30), por lo que en principio no debería de haber problema.

El router también actúa como servidor web ya que ofrece una interfaz web de acceso a su configuración. Así que me valdré de eso para poder darle un nombre más adelante como si de un servidor web se tratara.

Clonación de máquinas

- Clonamos nuestra máquina Bind9 "base" dos veces, una para el servidor primario y otra para el servidor secundario.
- Verificamos que su modo de red esté en **Adaptador Puente** (Bridged Adapter).

Para la primera máquina (servidor primario) ya tendríamos la **dirección IP estática configurada en netplan**. Para la segunda (servidor secundario), tendríamos que editar netplan, configurar su

dirección IP y aplicar los cambios.

Configurando el servidor primario

Editando named.conf.local

Editamos el archivo **/etc/bind/named.conf.local** y añadimos la zona **asir-sri.com**, quedando el fichero de la siguiente manera:

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
zone "asir-sri.com" {
        type master;
        file "/etc/bind/db.asir-sri.com";
        allow-transfer { 192.168.31.3; };
        also-notify { 192.168.31.3; };
};
```

- **type**: Tiene que ser "master" ya que es el servidor primario.
- **file**: Indica la ruta al fichero de zona, que crearemos más adelante.
- **allow-transfer**: Las IPs de los servidores que aquí se especifiquen podrán realizar una **transferencia de zona**. Nos interesará referenciar aquí al servidor secundario.
- **also-notify**: Las IPs de los servidores que aquí se especifiquen serán notificados de los cambios en la zona. Nos interesará referenciar aquí al servidor secundario.

Creando el archivo de zona db.asir-sri.com

Ahora tenemos que crear un fichero de zona con la información de nuestra zona. Utilizaremos como base la plantilla de la zona local: **/etc/bind/db.local**.

Realizamos una copia de la plantilla de zona local para nuestra zona asir-sri.com:

```
sudo cp /etc/bind/db.local /etc/bind/db.asir-sri.com
```

Y ahora editamos el fichero **/etc/bind/db.asir-sri.com**:

```
;
; asir-sri.com
;

$TTL    604800
@       IN      SOA     ns.asir-sri.com. root.asir-sri.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.asir-sri.com.
@       IN      NS      ns2.asir-sri.com.
@       IN      A       192.168.31.1
ns      IN      A       192.168.31.2
ns2     IN      A       192.168.31.3
router  IN      A       192.168.31.1
www     IN      CNAME   router
```

- La **@** sustituye al dominio "asir-sri.com"
- **root.asir-sri.com**: Sería el email del administrador del dominio (aunque en este caso no existe). Ojo, se pone "." en vez de "@".
- **IN**: Es la clase, en nuestro caso "Internet". Es el tipo de red.
- **A**: Es un registro de recursos para indicar IPs.
- **NS**: Es un registro de recursos para indicar los servidores de nombres autorizados.
- **ns** y **ns2**: Indican las IPs de los servidores de zona. Podríamos haber puesto en su lugar "ns.asir-sri.com." y "ns2.asir-sri.com.".
- **router**: Le damos un nombre al router.
- **www**: Creamos un alias CNAME para el router.

Una vez editado el archivo, comprobamos que no hayamos cometido ningún error:

```
named-checkzone asir-sri.com /etc/bind/db.asir-sri.com
```

Y reiniciamos el servicio:

```
sudo service bind9 restart
```

Y desde nuestro ordenador host podemos probar que funciona realizando las siguientes peticiones:

```
nslookup asir-sri.com 192.168.31.2
nslookup ns.asir-sri.com 192.168.31.2
nslookup ns2.asir-sri.com 192.168.31.2
nslookup www.asir-sri.com 192.168.31.2
nslookup router.asir-sri.com 192.168.31.2
```

El servidor responderá y la respuesta será **autoritativa**, ya que en nslookup no nos añadirá el texto de "*Non-authoritative answer*".

Configurando el servidor secundario

Los **servidores secundarios** se crean para **tolerancia a fallos**, **balanceo de carga** y para que las **respuestas sean más rápidas**.

Clonamos la **máquina virtual base** con el servidor Bind9 antes creada y en **netplan** le configuramos la IP **192.168.31.3**, ya que la máquina virtual tenía configurada la IP estática **192.168.31.2** y colisionará con la de nuestro servidor primario.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
      - 192.168.31.3/24
      routes:
      - to: default
        via: 192.168.31.1
      nameservers:
        addresses: [192.168.31.3]
```

He aprovechado y cambiado su servidor de nombres en la configuración de netplan a su propia IP.

Y aplicamos los cambios en **netplan**.

```
sudo netplan apply
```

Podemos verificar que tiene la IP **192.168.31.3** con el comando **ip a**.

```
ip a
```

Ahora editaremos el archivo de configuración **/etc/bind/named.conf.local**.

```
sudo nano /etc/bind/named.conf.local
```

Y lo dejamos de la siguiente forma:

```
zone "asir-sri.com" {
        type slave;
        file "/etc/bind/db.asir-sri.com";
        masters { 192.168.31.2; };
};
```

El parámetro **masters** contiene la IP del servidor primario.

Salvamos el archivo y reiniciamos Bind9:

```
sudo service bind9 restart
```

Podemos ver el estado del servicio en el servidor secundario y ver que se ha transferido la zona:

```
sudo service bind9 status
```

Veremos unas líneas en las que se nos confirma que se ha transferido al zona parecidas a:

```
Nov 02 07:07:46 ubuntu2004 named[1992]: zone asir-sri.com/IN: transferred serial 2
Nov 02 07:07:46 ubuntu2004 named[1992]: transfer of 'asir-sri.com/IN' from 192.168.31.2#53: Transfer status: succ
Nov 02 07:07:46 ubuntu2004 named[1992]: transfer of 'asir-sri.com/IN' from 192.168.31.2#53: Transfer completed: 1
Nov 02 07:07:46 ubuntu2004 named[1992]: zone asir-sri.com/IN: sending notifies (serial 2)
```

Podemos ahora ir a nuestro **ordenador host** y realizar peticiones para ver que el servidor secundario es capaz de responder a las consultas sobre la zona **asir-sri.com**:

```
nslookup asir-sri.com 192.168.31.3
nslookup ns.asir-sri.com 192.168.31.3
nslookup ns2.asir-sri.com 192.168.31.3
nslookup www.asir-sri.com 192.168.31.3
nslookup router.asir-sri.com 192.168.31.3
```

Las respuestas del servidor secundario también son **autoritativas** para la zona.

Ya podemos **apagar el servidor secundario** puesto que en esta práctica no lo utilizaremos más.

Configurando la zona inversa

La **resolución inversa** no se utiliza de forma muy frecuente. Normalmente nos interesará traducir nombres a IPs y no IPs a nombres. De todas formas, es interesante configurarla, especialmente si se va a utilizar un **servidor de email** puesto que la resolución inversa puede ser utilizada para verificación del dominio y **clasificar correos o no como spam**.

Para ello vamos al servidor primario y editamos el archivo **named.conf.local** y añadimos:

```
zone "31.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.31.168.192.in-addr.arpa";
};
```

Vamos a crear ahora nuestro archivo "**db.0.168.192.in-addr.arpa**". Como es muy parecido al de la zona, vamos a copiarlo.

```
sudo cp /etc/bind/db.asir-sri.com /etc/bind/db.31.168.192.in-addr.arpa
```

Y ahora lo editamos y lo dejamos con el siguiente contenido:

```
; Zona inversa
;
$TTL    604800
@       IN      SOA     ns.asir-sri.com. root.asir-sri.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@       IN      NS      ns.asir-sri.com.
@       IN      NS      ns2.asir-sri.com.

1.31.168.192.in-addr.arpa.      IN      PTR     asir-sri.com.
2.31.168.192.in-addr.arpa.      IN      PTR     ns.asir-sri.com.
3.31.168.192.in-addr.arpa.      IN      PTR     ns2.asir-sri.com.
```

En lugar de poner **1.31.168.192.in-addr.arpa.** podríamos poner simplemente **1**. Aunque es de buena práctica poner el FQDN.

Reiniciamos el servicio:

```
sudo service bind9 restart
```

Y podemos probar a realizar resoluciones inversas desde nuestro **equipo host**:

```
nslookup 192.168.31.1 192.168.31.2
nslookup 192.168.31.2 192.168.31.2
nslookup 192.168.31.3 192.168.31.2
```