Instalación de Servidor DNS

Luigi Guarino
14/11/2017

Contenido

Servidor maestro en CentOS 7

1. Introducción

En el siguiente manual se presenta la instalación de dos servidores DNS para CentOS 7 y Debian 9 respectivamente.

Se realizara la instalación de un servidor DNS maestro en CentOS y posteriormente, servidor DNS esclavo en Debian.

¿Qué es DNS?

El **sistema de nombres de dominio** o DNS (sigla en inglés de *Domain Name System)***,** es el **protocolo** que permite la **resolución de dominios a direcciones IP e inversamente.**

Por tanto, un **servidor DNS** (sigla en inglés de *Domain Name System* – *S*istema de nombres de dominio) es, el servidor encargado de **traducir** nombres de dominio a la dirección IP del servidor designado al mismo, y viceversa... Además, el protocolo DNS trabaja bajo los **puertos 53 UDP y TCP**.

¿Donde usamos DNS?

Miles de personas utilizan servidores DNS cada segundo y el la mayoría de los casos sin darse cuenta. ¿Y por qué? Bueno, podríamos decir que, en general, los humanos no somos muy buenos recordando números...

Imaginaros que cada vez que quisiéramos navegar a **www.google.es**, tendríamos que escribir en la barra de direcciones **216.58.205.67** o para ir a **www.wordpress.org** recordar la IP **66.155.40.250** o **66.155.40.549**

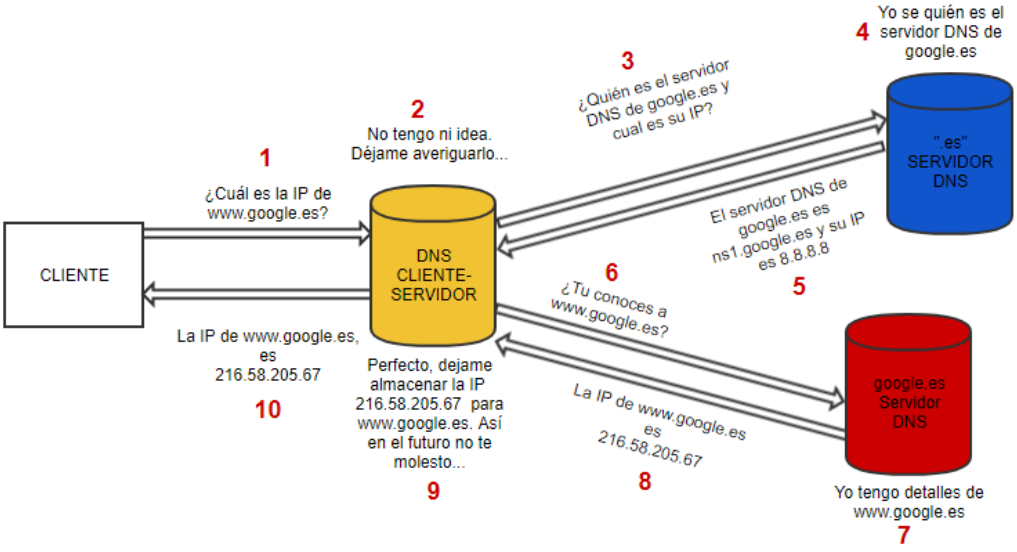
Entonces Internet no habría tenido mucho éxito... Es por ello que tenemos los DNS. En **esta entrada** nos vamos a centrar en los servidores **DNS a nivel local**. Es decir, servidores DNS que actuaran sobre una **zona de red concreta**.

¿Bueno, pero cómo funciona DNS?

Sencillo. Primero vamos a aclarar que un servidor DNS en sí mismo, **funciona como servidor** pero también como **cliente**. Es decir, un servidor DNS **no dispone** de todas las **direcciones IP del mundo**, eso sería una locura. Por tanto, cada servidor DNS dispone de una **base de datos** donde recoge todas las **direcciones IP** y todos los **nombres de los PCs** pertenecientes a su **dominio**.

Es por ello, que los servidores DNS están construidos de forma **jerárquica**, para que sí, el servidor DNS **más cercano** al cliente no pueda **resolver la petición** de traducción, este servidor, **traslade la petición** a un **DNS superior** (y así, sucesivamente si fuera necesario).

Pongamos un ejemplo:



En el anterior esquema, se muestra la petición de un usuario para navegar a **www.google.es**. Digamos que es la **primera vez** que nuestro servidor DNS recibe esta **petición**. Por tanto, no tiene ni idea de la **resolución** de IP. Para resolverla, **redirige la petición** a otros servidores DNS, y estos, le devuelven la resolución. Por último, nuestro servidor recibe la dirección IP y la **almacena** en su **base de datos**.

En el caso que vamos a realizar nosotros, únicamente tendremos las **resoluciones** que nosotros como administradores **especifiquemos**, ya que, se trata de un **DNS privado**, que actuara solamente en nuestra red.

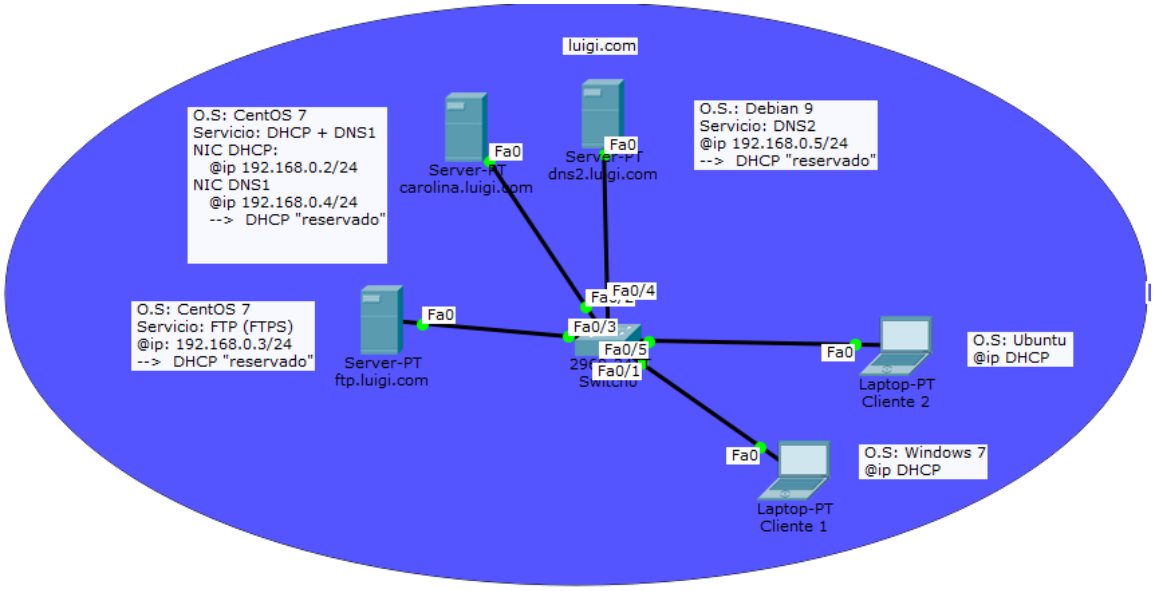
2. Objetivos

- Instalar y configurar un servidor DNS (servidor maestro) → DNS1
- Configurar zonas de búsqueda directa e inversa

3. Esquema de red

Para este manual, partimos de los servidores DHCP y FTP. Para completar esta red de servidores, instalaremos el **servicio DNS** (bind9) en el mismo **servidor DHCP**. Posteriormente crearemos el servidor **DNS2** en **Debian**, y lo integramos a su vez a la **red**.

El **dominio** donde trabajare se nombrara como: **luigi.com.** Además, realizaremos los cambios oportunos sobre las **tarjetas de red** (siendo nuestro servidor **DHCP** el encargado de **repartir las @ip**) para trabajar sobre el siguiente **esquema de red**:



4. Instalación y configuración del DNS Primario (maestro)

4.1 Instalar BIND9

Para el funcionamiento del servicio DNS trabajaremos sobre la plataforma más completa para servidores de este tipo en sistemas Linux, **Bind**, en su versión 9.11.2.

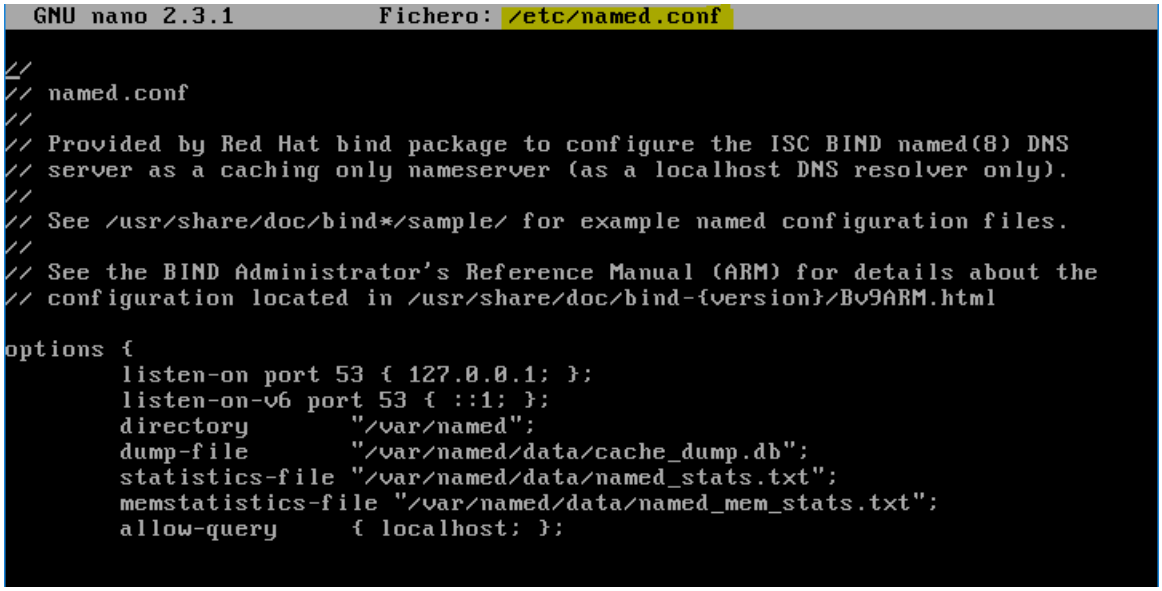
Lo primero de todo, **sudo su,** y actualizamos el repositorio de paquetes: **yum update** y posteriormente realizamos la instalación de los **paquetes bind**: **sudo yum install bind bind-utils**

Una vez realizada la instalación, se crearan **varios ficheros de configuración** sobre **/etc**.

4.2 Configuración del fichero principal

El fichero principal de configuración se ubica sobre /etc/named.conf. Como siempre, buen administrador, buen "respaldador". Realizamos la copia de seguridad de nuestro fichero antes de editarlo: cp /etc/named.conf /etc/named.conf.backup.

Accedemos a él: **nano /etc/named.conf :**



Ya sabéis que no me gustan los ficheros ya redactados... así que como siempre, a borrar y desde 0:

```
[root@carolina /]# rm /etc/named.conf
rm: ¿borrar el fichero regular «/etc/named.conf»? (s/n) s
[root@carolina /]# nano /etc/named.conf_
```

Comenzamos a configurar:

4.2.1 Configuraciones generales del DNS (options)

directory: definimos el directorio de actuación del servicio

dump-file: fichero dónde se almacenara la información de cache

```
options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        };
```

4.2.2 Configuración de zonas DNS

Como bien hemos comentado anteriormente, nuestro servidor **DNS** debe ser capaz de resolver direcciones tanto de **forma directa e indirecta**. Es decir, si recibe una consulta acerca de **quién es ftp.luigi.com**, deberá devolver su **IP** a **192.168.0.3/24** Además, si recibe una petición de **quién es 192.168.0.3/24** tiene que responder forma inversa, es decir, **ftp.luigi.com**.

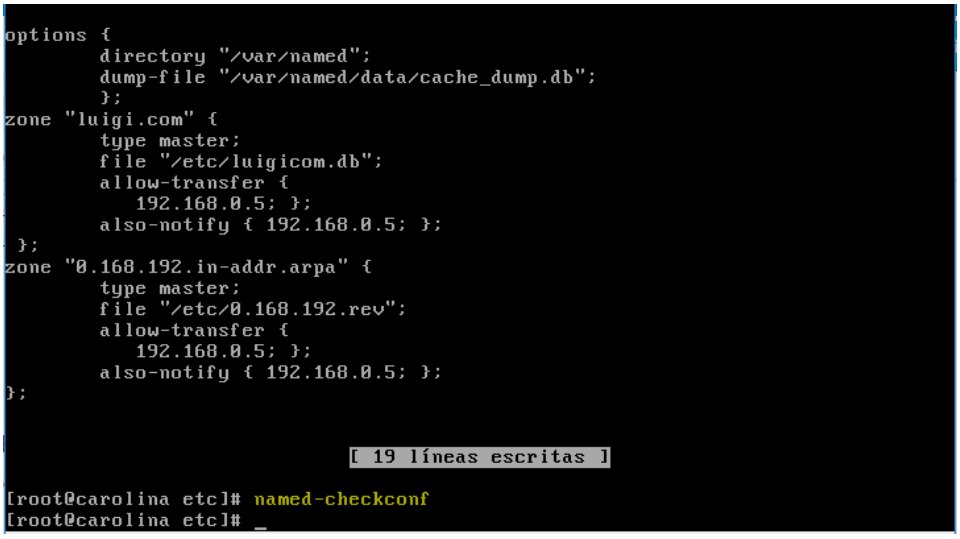
Utilizamos la siguiente sintaxis:

- **zone "nombre_de_zona"**: definimos el nombre con el que caracterizamos la zona de actuación.
- **type**: definimos si se trata de un servidor maestro o esclavo. Es decir, si el propio servidor **almacena las resoluciones** (**master**) o sí, en cambio, se le **trasfiere** las resoluciones desde otro servidor (**slave**)
- **file**: parámetro que define el fichero donde se hayan las resoluciones.
- **allow-transfer**: especificamos la @ip del servidor **DNS espejo** (slave). Este parámetro permite trasferir las resoluciones al DNS secundario.
- **also-notify**: volvemos a especificar la @ip del **DNS secundario** para permitir, la **notificación** de cualquier **cambio** en el **fichero de resoluciones** del **DNS maestro**.

Así queda nuestro servidor:

Vemos como en el fichero se redactan dos zonas. Es decir, una zona para resoluciones **directas (luigi.com)** e **indirectas (0.168.192.in-addr.arpa).**

Para comprobar que no existe ningún **error sintáctico**, contamos con la herramienta **named-chekconf**. Si no existen errores, continuamos. Si existe algún error, comprobamos el fichero en busca de algún error (un espacio, ; , tabulaciones...)



Una vez hecho, ya tenemos la **configuración base** del DNS. Ahora, vamos a **crear** y **configurar** los ficheros de las zonas, redactados anteriormente. En nuestro caso son: **luigicom.db,** para directas y **0.168.192.rev,** para indirectas.

4.3 Generar las zonas

4.3.1 Configuración general de la zona directa

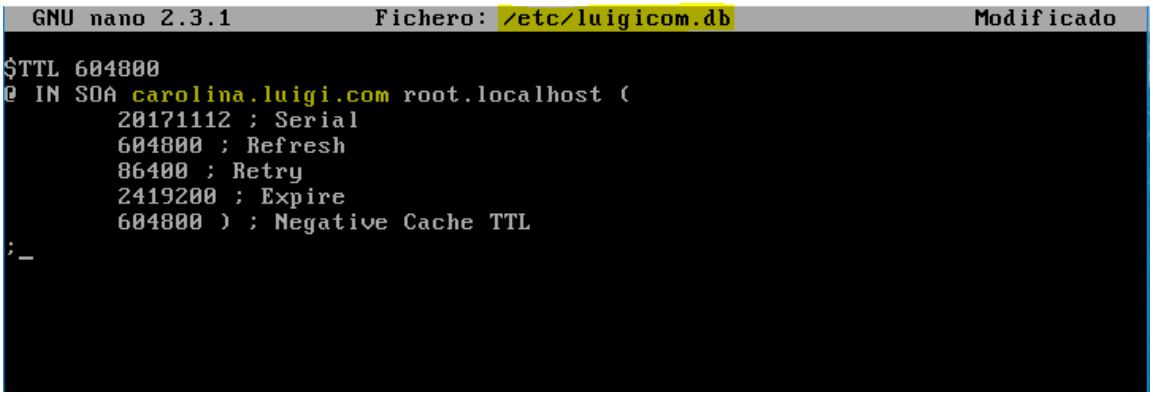
En este fichero de configuración, se hallaran las r**esoluciones de dominios a direcciones IP**. Lo primero de todos será **crear** el fichero:

nano /etc/luigicom.db

Ahora lo configuramos utilizando la siguiente sintaxis:

- **TTL**: es el **tiempo** que trascurre hasta el fichero de configuración se da por "**caducado**". En este momento, el **servidor esclavo solicita la trasferencia de zona**, es decir, **actualiza** sus ficheros de resoluciones (en segundos)
- **SOA**: son una serie de parámetros que definen **metainformación** del servidor. Estos incluyen:
- **Serial**: Identificador "arbitrario" del fichero. Se recomienda que siga la estructura **AAAAMMDD**.
- **Refresh**: Tiempo de espera para que el servidor secundario soliciten **actualizar** sus registros de resolución.
- **Retry**: Tiempo de espera de un servidor secundario si ha **fallado** la trasferencia de datos.
- **Expire**: cantidad de tiempo que el servidor secundario **gasta** para intentar realizar la trasferencia de datos. Este tiempo indica, la cantidad del mismo que se utilizan **resoluciones "caducadas"**.
- **Negative cache TTL**: volvemos a definir el tiempo de caducidad del fichero.

Resultado:

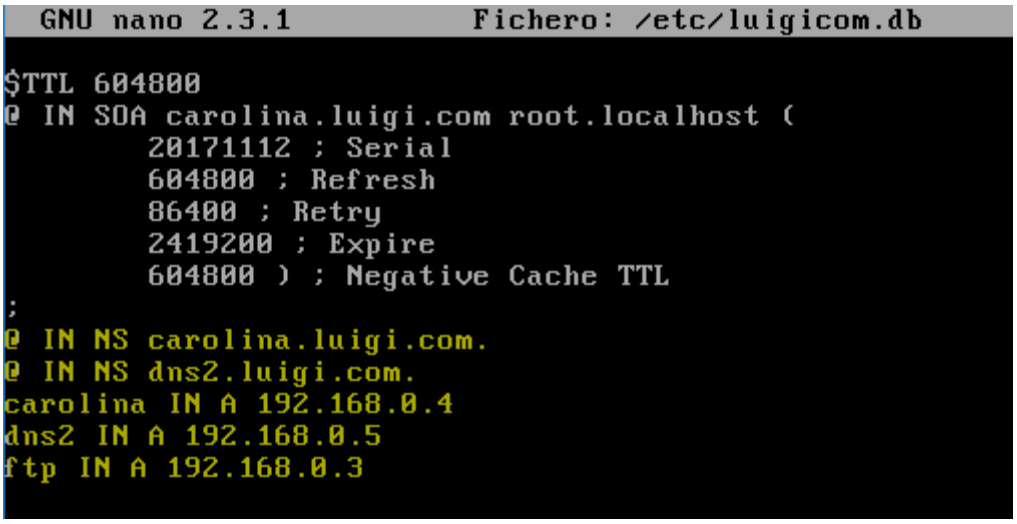


4.3.2 Configuración de resoluciones directas

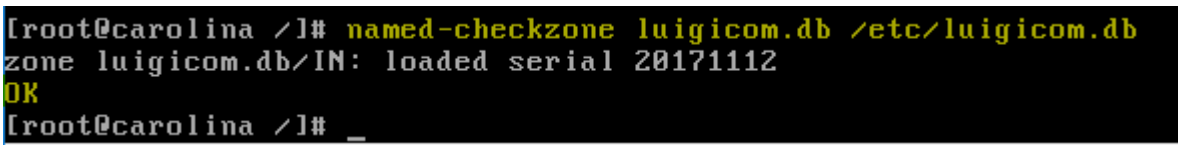
Utilizamos la siguiente sintaxis para definir las resoluciones directas:

- ***nombre***: es el nombre a resolver. Por ejemplo, *ftp*. En el caso de ser un **servidor DNS**, se identifica como ***@ IN NS "nombre_dominio"***
- ***clase:*** es el tipo de dirección IP. La palabra reservada IN indica qué es IPv4
- ***tipo:*** existen varios:
- **A**: traduce nombres a direcciones IPv4
- **AAAA**: traduce nombres a direcciones IPv6
- **NS**: define servidor DNS del dominio
- **MX**: define servidor de correo del dominio

Nuestra configuración:



Comprobamos si existe algún error sintáctico con **named-checkconf :**



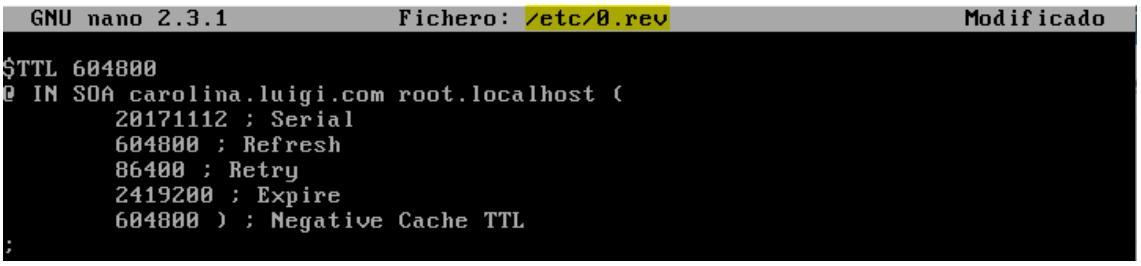
Una vez comprobado, procedemos a crear el fichero de resoluciones indirectas:

nano /etc/0.168.192.rev

Lo configuramos utilizando la siguiente sintaxis:

4.3.3 Configuración general de la zona indirecta

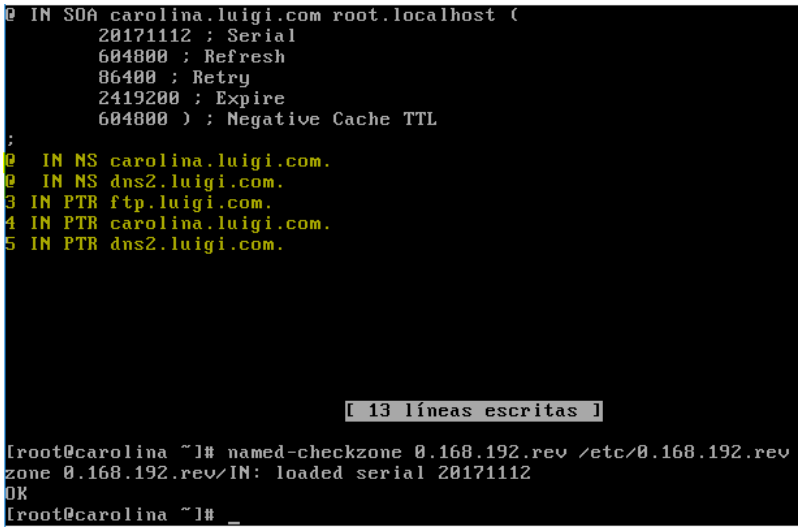
Vamos a copiar literalmente lo escrito anteriormente en el fichero **luigicom.db,** ya que utilizaremos los mismos **parámetros generales** de configuración:



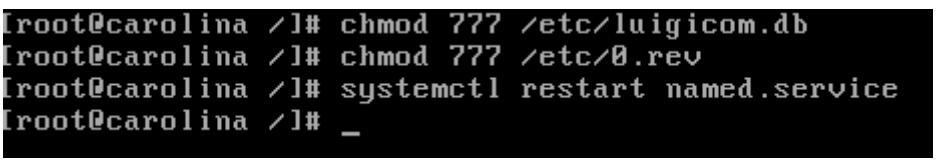
4.3.4 Configuración de resoluciones indirectas

Utilizamos la misma **sintaxis** que para el **anterior fichero**. Únicamente variamos la parte ***nombre***, ya que, esta vez hacemos referencia a la **parte de host** de la @ip.

Realizamos la configuración y comprobamos de nuevo la sintaxis:



Por último, vamos a dar **permisos absolutos** a los dos ficheros de resoluciones (así, no existirá problema de acceso para el DNS2) y reiniciamos el servicio:



5. Pruebas de contexto

Ya tenemos todo configurado. Ahora vamos a comprobar que el servicio funciona correctamente....

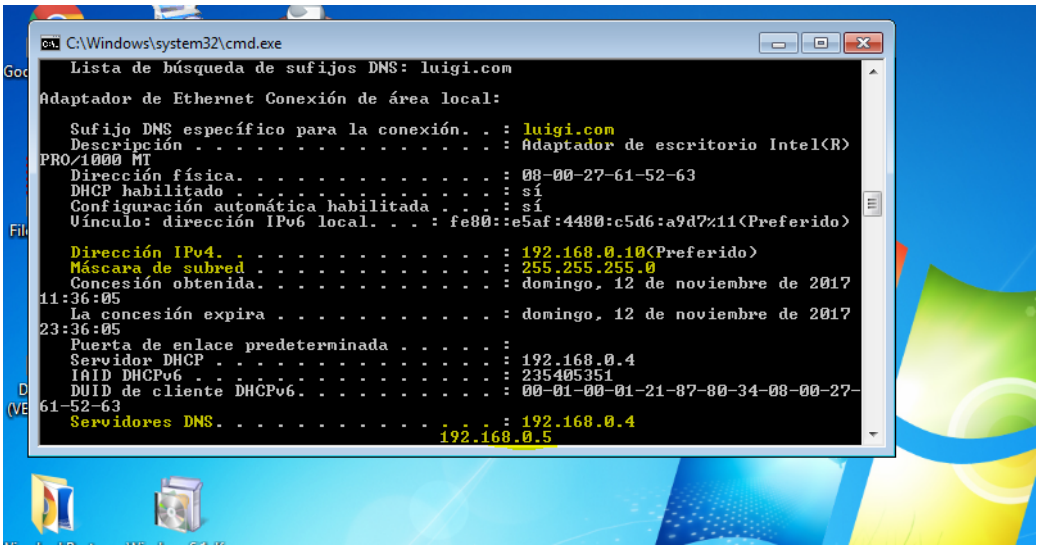
Comprobamos si el propio servidor **Carolina resuelve el nombre**. Para ello utilizamos el comando **host "@ip/dominio":**

```
[root@carolina ~]# host 192.168.0.4
4.0.168.192.in-addr.arpa domain name pointer carolina.luigi.com.
[root@carolina ~]# host 192.168.0.3
3.0.168.192.in-addr.arpa domain name pointer ftp.luigi.com.
[root@carolina ~]# host dns2.luigi.com
dns2.luigi.com has address 192.168.0.5
[root@carolina ~]# _
```

Una vez hecho lo fácil, vamos a utilizar a nuestros clientes Windows 7 y Ubuntu para comprobar si Carolina funciona de forma correcta. Por supuesto las **NIC de las maquinas** están configuradas para recoger la **@ip mediante DHCP**.

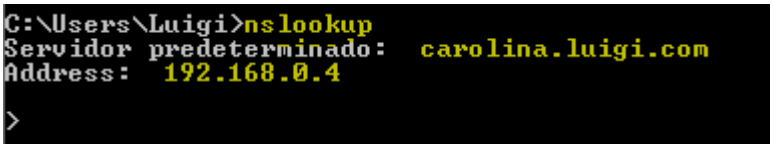
Windows 7

Comenzamos con Windows. Vamos a utilizar la herramienta **nslookup**, para analizar cómo se comporta el servidor. Lo primero, vemos si Carolina, en su papel de DHCP, está repartiendo correctamente. Abrimos consola y ejecutamos: **ipconfig /all**



Correcto! Ahora vamos a emitir un par de resoluciones...

Ejecutamos la herramienta ***nslookup*:**



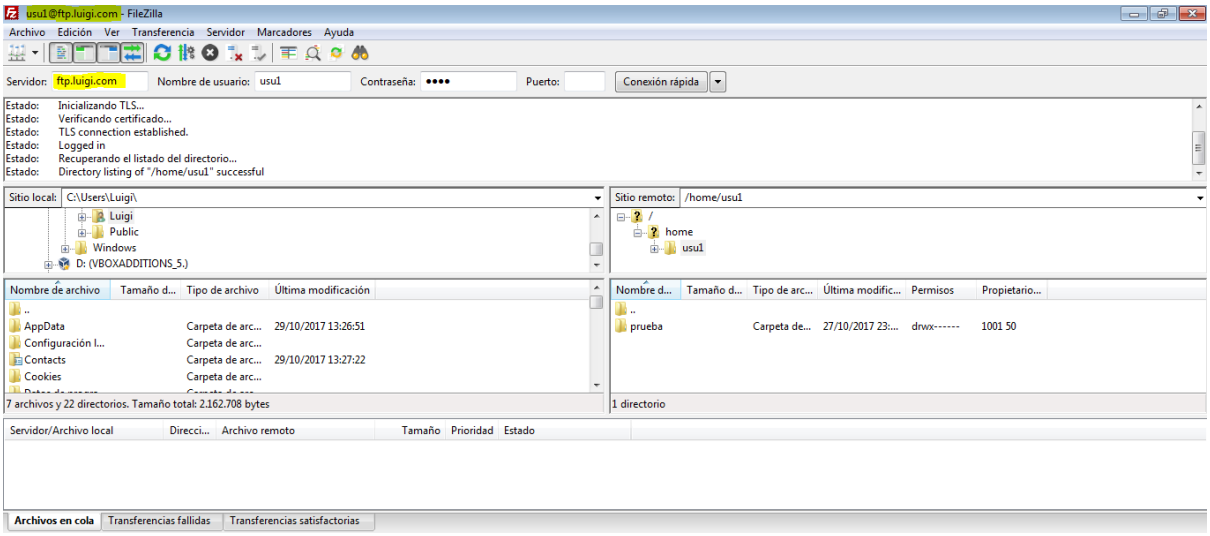
Podemos comprobar cómo nos ha devuelto la **@ip de nuestro servidor DNS1** y su correspondiente **nombre**.

Ahora vamos a resolver el dominio **ftp.luigi.com**:

Perfecto! Vamos a pedir que nos resuelva la **@ip 192.168.0.5**:

Desde el **navegador** también resolveremos. Vamos a probar a acceder a nuestro **FTP** de **forma anónima**:

¿Y nuestro **FTP seguro**? También...



Ubuntu

Por último, vamos a comprobar si nuestro viejo amigo, Ubuntu es capaz de **resolver las peticiones**:

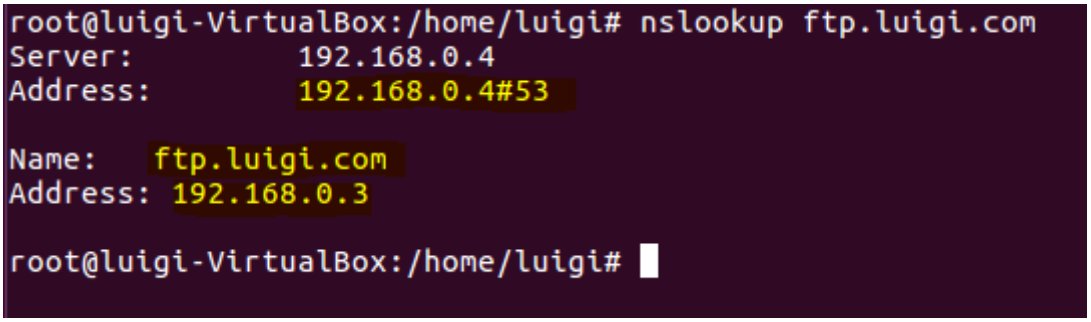
Primero comprobamos que se le ha repartido un @ip valida con **ifconfig**:

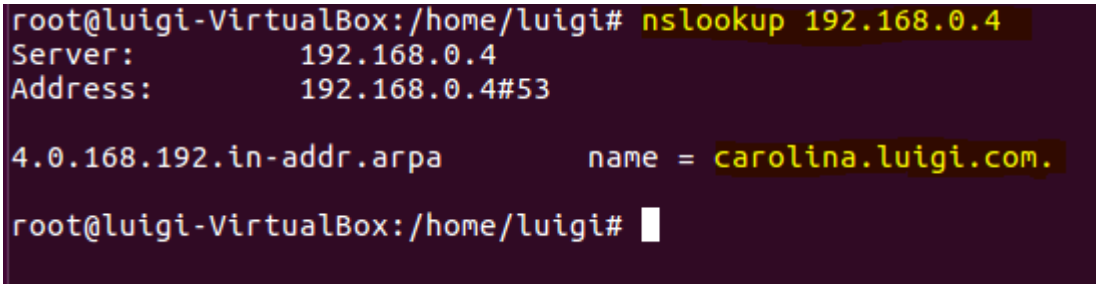
```
root@luigi-VirtualBox:/home/luigi# ifconfig
enp0s3    Link encap:Ethernet  direcciónHW 08:00:27:f5:40:0f
          Direc. inet:192.168.0.11  Difus.:192.168.0.255  Másc:255.255.255.0
          Dirección inet6: fe80::ff78:1576:b823:342a/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:8 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:63 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:1328 (1.3 KB)  TX bytes:7552 (7.5 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:4 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:4 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:240 (240.0 B)  TX bytes:240 (240.0 B)
```

Y el servidor DNS: **nano /etc/resolv.conf**

Y volvemos a utilizar la herramienta **nslookup**:





Servidor esclavo en Debian 9

1. Una pequeña introducción

¿Qué es un servidor DNS secundario?

Un servidor **DNS secundario** puede ser un servidor DNS en sí mismo o, como en nuestro caso, un servidor de **respaldo**.

Es decir, los servidor DNS secundarios se especifican en los sistemas para **garantizar** que, si el **DNS primario cae**, el cliente pueda seguir **resolviendo dominio y/o direcciones IP.**

Además, debemos saber que cualquier **sistema**, sobre todo **Linux**, se puede especificar tantos servidores DNS como uno quiera, pudiendo entonces existir DNS terciario, cuaternario, quinario... ?

¿Y un servidor esclavo?

Bueno, un **servidor esclavo** es el que **vamos a montar** nosotros. Su función principal es **respaldar** a su servidor maestro (Carolina, en este caso), y actuar como servidor de resoluciones en la red privada, sí este mismo cae.

Por si mismo, el servidor esclavo es **incapaz** de resolver ninguna petición, por lo que, el servidor maestro c**ede sus resoluciones** (fichero luigicom.db y 0.168.192.rev, en este caso) para que el **servidor esclavo pueda funcionar si es necesario**.

2. Objetivos

- Instalar y configurar el servidor DNS (servidor esclavo) → DNS2
- Integrar y comprobar el funcionamiento del servidor en la red

3. Instalación y configuración del DNS Secundario (esclavo)

3.1 Iniciar sesión como administrador

```
Debian GNU/Linux 9 dns2 tty1

dns2 login: root
Password:
Linux dns2 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@dns2:~# _
```

3.2 Instalar Bind9

Lo primero de todo, es recomendable, actualizar nuestro repositorio de paquetes. Para ello: **apt-get update** y **apt-get upgrade.** Posteriormente procedemos a la instalación del paquete y herramientas de **Bind9** (el mismo que utilizamos en la primera parte):

apt-get install bind9 bind9-doc dnsutils

3.3 Comprobar dirección IP

Recordando el anterior **esquema de red**, debemos **verificar** que el **servidor DHCP** (Carolina), ha repartido la **@ip 192.168.0.5/24** para nuestro nuevo servidor:

Ejecutamos **ifconfig** para comprobarlo:

```
root@dns2:~# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.5  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe64:842f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:64:84:2f  txqueuelen 1000  (Ethernet)
        RX packets 10014  bytes 2830095 (2.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8310  bytes 683773 (667.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 440  bytes 35287 (34.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 440  bytes 35287 (34.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.4 Configuración de Bind9

El paquete **Bind9** funciona de forma un poco diferente en Debian con respecto a CentOS. Esta vez al **instalar el paquete**, se han **creado** los diferentes **archivos de configuración** sobre **/etc/bind.**

El fichero que utilizaremos para la configuración será: /etc/bind/named.conf.options. Únicamente haremos uso de este fichero, ya que, la configuración predeterminada del servicio nos sirve para realizar el servidor esclavo.

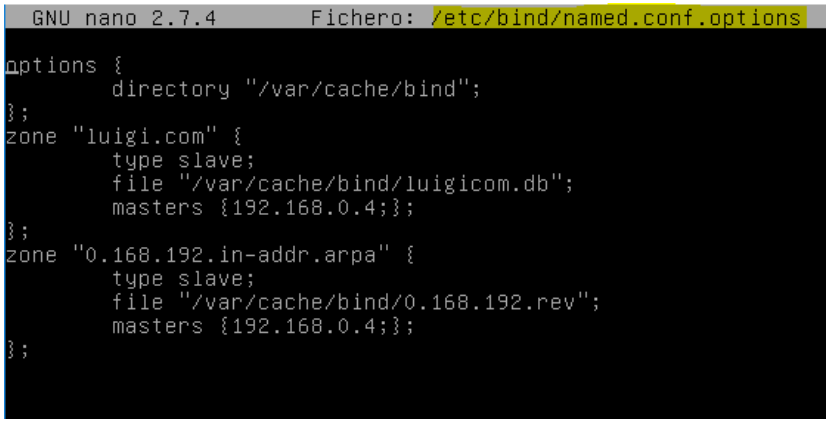
Como siempre realizamos la **copia de seguridad del fichero** antes de editarlo: cp /etc/bind/named.conf.option /etc/bind/named.conf.option.backup

3.4.1 Configuración del fichero de zonas esclavas

Antes de comenzar con la configuración, vamos a concretar algunos conceptos. Utilizamos el fichero named.conf.options y NO named.conf.local, ya que, el fichero conf.options realiza las resoluciones desde cache (/var/cache/bind) y no desde local, como conf.local.

Esto es así porque, las trasferencias de zonas desde el servidor maestro al esclavo son almacenadas en cache, a la espera de algún cambio o alguna nueva resolución.

Una vez aclarado, vamos a configurar:



- **options - directory**: directorio donde se almacena la cache del servicio DNS.
- **zone**: nombre de las zonas directa e inversa
- **type slave**: parámetro que define la zona como esclava
- **file**: fichero cache donde se almacenara la información recibida por el servidor maestro
- **masters**: @ip del servidor DNS maestro que trasferirá los datos

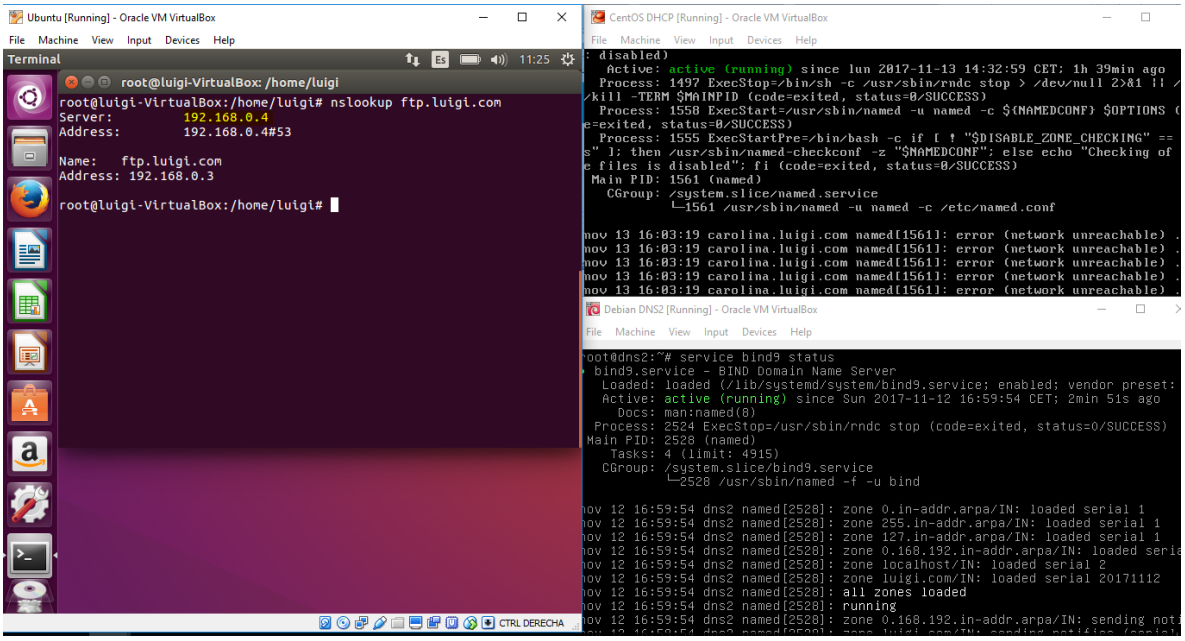
Listo! Una vez reiniciamos el servicio: **service bind9 restart,** comprobamos como se comporta los servidores en el red....

4. Pruebas de contexto

Para realizar las pruebas de funcionamiento, vamos a utilizar nuestra maquina cliente Ubuntu. ¿Por qué? Bueno, ya sabemos que Windows es confuso... Al intentar realizar las pruebas en la maquina Windows, no utiliza el servidor secundario aunque el primario se encuentre caído... cosas de Microsoft.

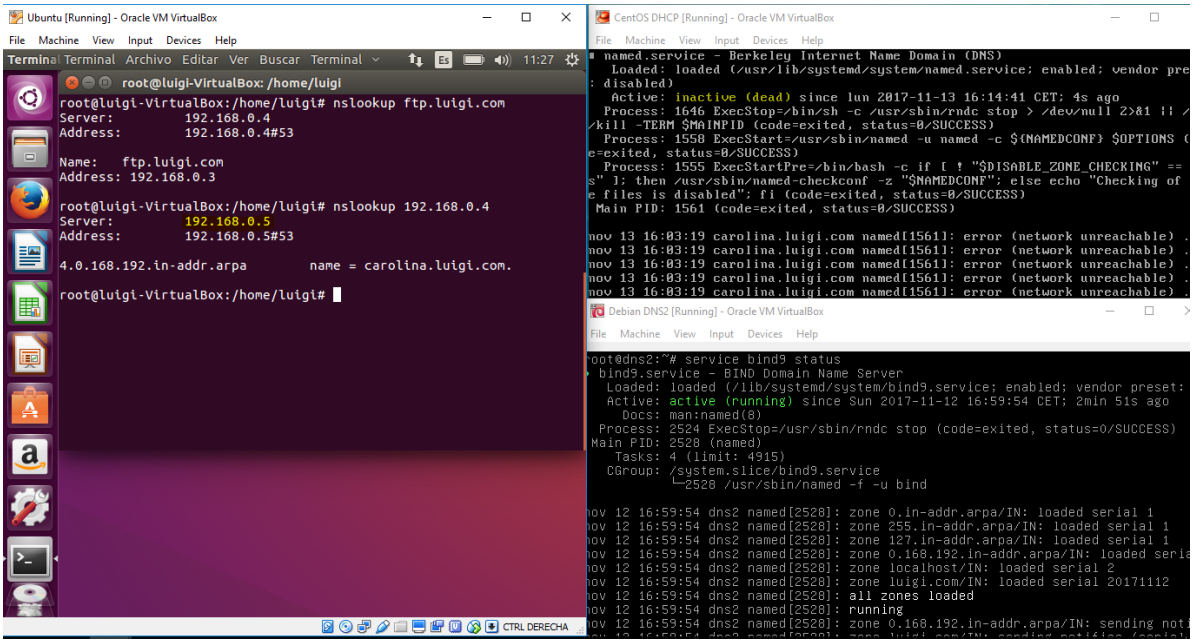
Vamos a ello...

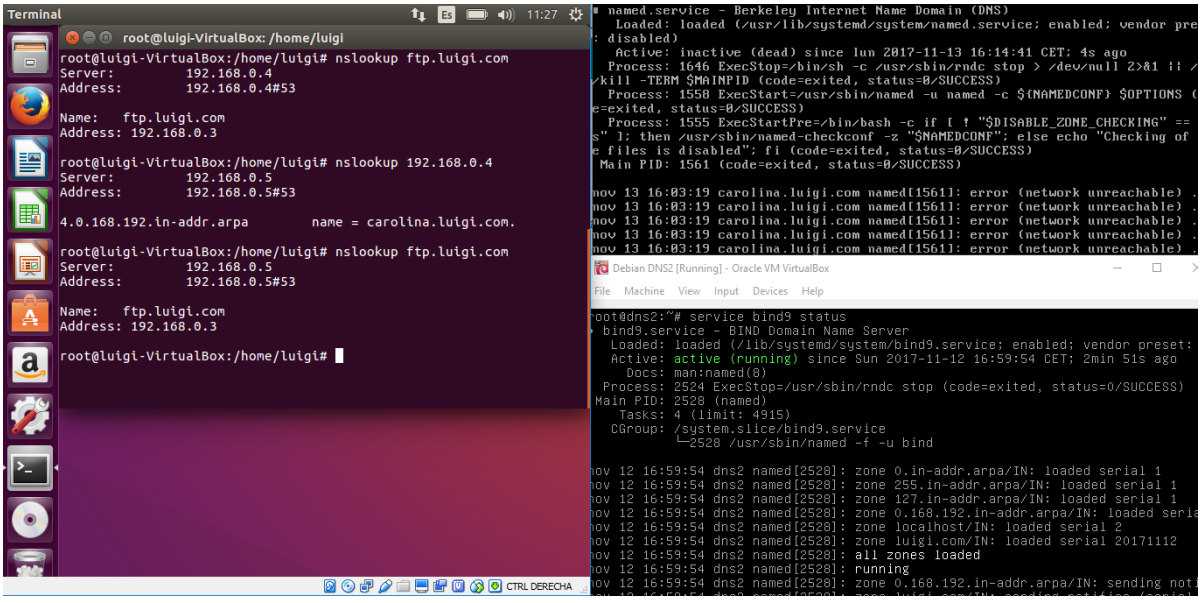
Nslookup con los dos servidores DNS en funcionamiento:



Comprobamos como resuelve el dominio de forma normal, a través del servidor DNS maestro.

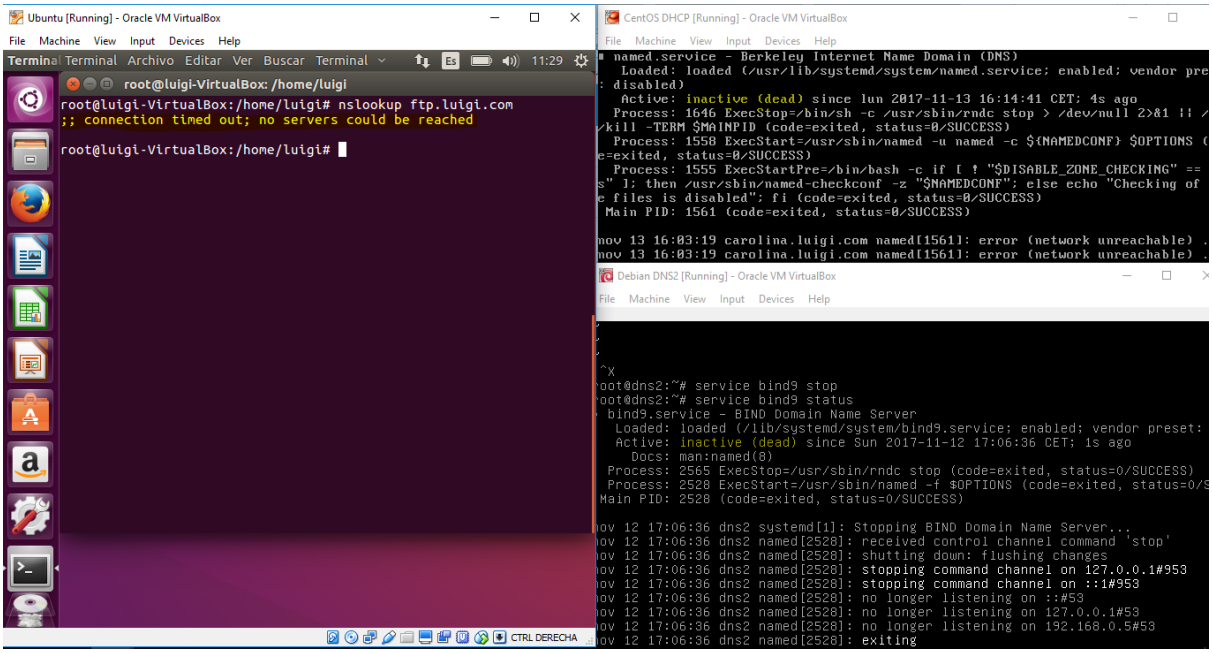
Nslookup con el servidor DNS maestro parado:





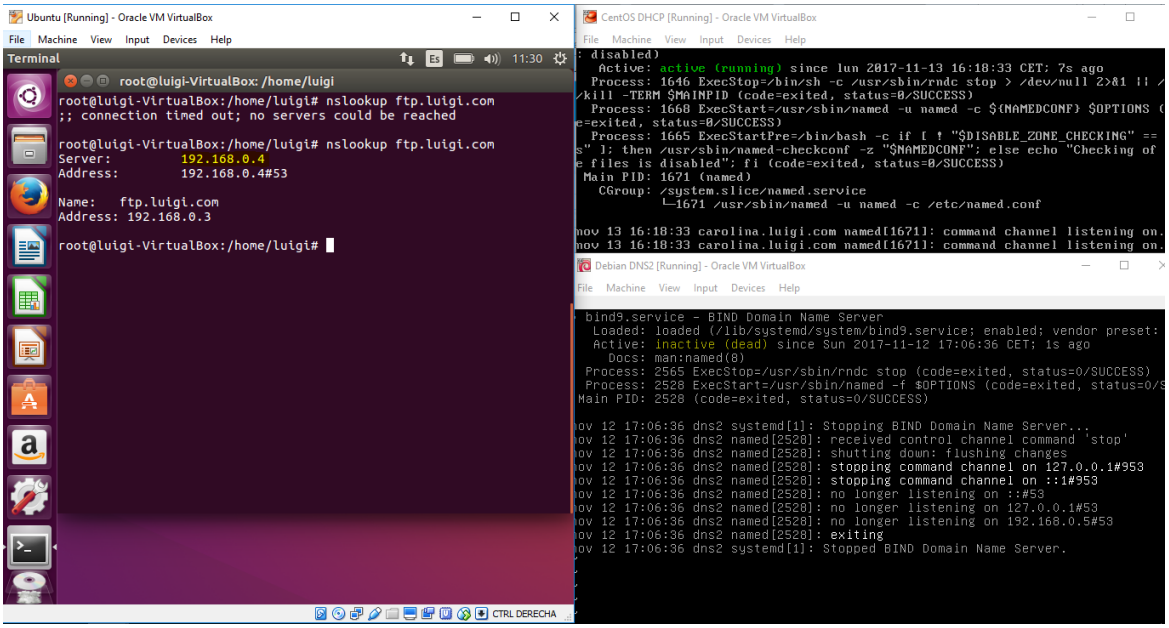
El cliente comprueba que el servidor DNS con @ip 192.168.0.4/24 no está. Así que reenvía la petición a nuestro servidor secundario con @ip 192.168.0.5/24 y lo resuelve.

Nslookup con los dos servidores parados:



Verificamos como si no existe ningún servicio levantado, es incapaz de resolver la petición de resolución.

Si volvemos a levantar los dos servicios, todo vuelve a la normalidad:



Conclusión

Terminado este manual, tendremos dos servidores DNS totalmente integrados en la red privada, funcionales y estables. Además cubrimos el problema de posible caída de uno de los servidores, pudiendo tener siempre uno disponible para realizar las resoluciones.