Configuring DNS

Here is the process used to configure my DNS servers using Bind9 on Debian, original tutorial: <https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-a-private-network-dns-server-on-debian-9>

**Example Hosts used:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Host** | **Role** | **Private FQDN** | **Private IP Address** |
| apollo | Host Windows PC | Apollo.elgboks.com | 192.168.0.14 /24 |
| ns1 | Primary DNS server | ns1.elgboks.com | 192.168.0.26 /24 |
| ns2 | Secondary DNS server | ns2.elgboks.com | 192.168.0.28 /24 |
| ftp1 | File transfer server | ftp1.elgboks.com | 192.168.0.25 /24 |

**Prerequisite:**

Need to ensure that all hosts are in the same domain. To do this, either specify the domain during the host’s install or edit the ***/etc/hosts*** file post install. To edit the file, add (or replace) the following line:

127.0.0.1 hostname.full.domainname hostname

Which will look like the following:

Text

Description automatically generated

**Installing Bind:**

On both DNS servers, **ns1** and **ns2**, update *apt* package cache with:

Sudo apt install bind9 bind9utils bind9-doc

Then, as we’re only using IPv4 we can edit ***/etc/default/bind9*** by adding “-4” to the end of options parameter:



Finally, we then restart Bind so we can begin configuring the primary DNS server:

Sudo systemctl restart bind9

**Configuring the Primary DNS server:**

Configuring the Options file:

On **ns1**, we are going to open ***/etc/bind/named.conf.options***

Above the existing ***options*** block, we will create a new ACL block called “trusted”. This block will specify clients allowed to recursively query. Here we specify the IP addresses of each of our clients: **ns1, ns2, ftp1, Apollo**.

acl “trusted” {

192.168.0.26; # ns1 – primary DNS

192.168.0.28; # ns2 – Secondary DNS

192.168.0.25; # FTP server

192.168.0.14; # Apollo – host Windows machine

};

options {

...

}

Then we will edit the ***options*** block to enable recursion and to specify, from our ACL, who can use it, i.e. “trusted” clients. Then we specify ns1’s IP address that they’ll listen on for queries. Finally, we then specify public DNS servers that we will forward any unresolved requests to:

options {

directory “/var/cache/bind”;

recursion yes;

allow-recursion { trusted; };

listen-on { 192.168.0.26; };

allow-transfer { none; };

forwarders {

8.8.8.8;

8.8.4.4;

};

...

};

Configuring the Local file:

Again, on **ns1**, we open ***/etc/bind/named.conf.local***

Here we will be specifying the **forward** and **reverse lookup zones**. We will use our domain name, “elgboks.com” to define the forward zone:

zone “elgboks.com” {

type master;

file “etc/bind/zones/db.elgboks.com”; # zone file path

allow-transfer { 192.168.0.28; }; # ns2 IP

};

For our reverse lookup zone, we will follow the standard naming convention of reversing the first two octets of the IP and then suffixing with “.in-addr.arpa”:

zone “168.192.in-addr.arpa” {

type master;

file “/etc/bind/zones/db.192.168; }; # 192.168.0.0 subnet

allow-transfer { 192.168.0.28; }; # ns2 IP

};

If our servers spanned across multiple subnets, then we would have to specify zones for each subnet, but this isn’t the case for this example.

Creating the Forward Zone file:

The forward zone is responsible for resolving queries like [ftp1.elgboks.com](ftp://ftp1.elgboks.com) to that host’s IP address. So, to begin with, we need to create the directories and files that we specified in the local file using the following commands:

Sudo mkdir /etc/bind/zones

Then we’ll copy the sample ***db.local*** file and base our own zone file off of it:

Sudo cp /etc/bind/db.local /etc/bind/zones/db.elgboks.com

We can then begin editing the file. To start with, we’ll replace “*localhost”* in the SOA record with our own details. It is very **important to increment the *serial* field each time we edit** the file, i.e. here we increase it from 2 to 3:

$TTL 604800

@ IN SOA ns1.elgboks.com. admin.elgboks.com. (

3 ; Serial

...

After the SOA record, we add our own name server records. Then we add the corresponding A records for these name servers and our clients:

; name servers – NS records

IN NS ns1.elgboks.com.

IN NS ns2.elgboks.com.

; name servers – A records

ns1.elgboks.com. IN A 192.168.0.26

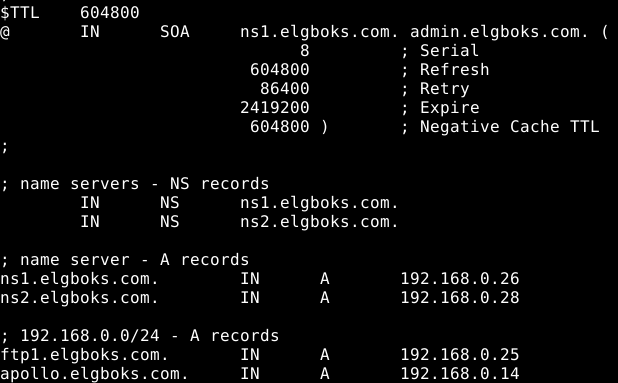
ns2.elgboks.com. IN A 192.168.0.28

;192.168.0.0 – A records

ftp1.elgboks.com IN A 192.168.0.25

apollo.elgboks.com IN A 192.168.0.14

The final file should look something like this:



Creating the Reverse Zone file:

The reverse DNS zone is responsible for resolving queries with an IP address and returning the FQDN of the corresponding host, e.g. “192.168.0.25” would resolve to “ftp1.elgboks.com”.

So, like with the forward zone file, we must create the corresponding file specified in “named.conf.local”. Again, we will copy a sample reverse zone file provided and edit it as needed:

Sudo cp /etc/bind/db.127 /etc/bind/zones/db.192.168

Like before, we need to edit the SOA record to replace localhost with our domain and **increment the serial** as appropriate.

$TTL 604800

@ IN SOA ns1.elgboks.com. admin.elgboks.com. (

3 ; Serial

...

At the end of the file, we specify the NS records for our DNS servers and then specify the PTR records for each host, following the reversal naming standard.

; name servers – NS records

IN NS ns1.elgboks.com

IN NS ns2.elgboks.com

; PTR records

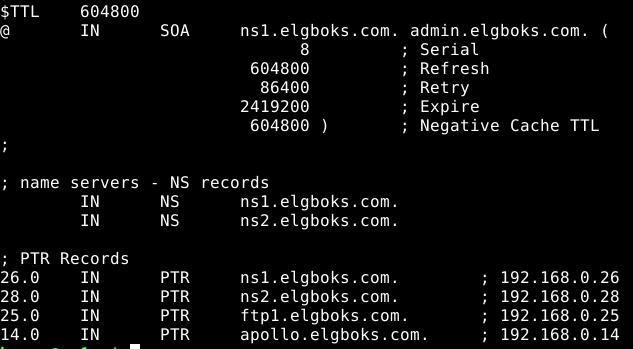
26.0 IN PTR ns1.elgboks.com. ; 192.168.0.26

28.0 IN PTR ns2.elgboks.com. ; 192.168.0.28

25.0 IN PTR ftp1.elgboks.com. ; 192.168.0.25

14.0 IN PTR apollo.elgboks.com. ; 192.168.0.14

With the final file looking like this:



Checking Configuration Syntax

We’ve now finished configuring the primary DNS server, so we need to check our configs for any syntax errors with the following commands:

Sudo named-checkconf

Sudo named-checkzone elgboks.com /etc/bind/zones/db.elgboks.com

Sudo named-checkzone 168.192.in-addr.arpa /etc/bind/zones/db.192.168

If none of these commands return any errors then we are good to go, and can restart Bind with:

Sudo systemctl restart bind9

If we are using UFW firewall, then we enable access to Bind with:

Sudo ufw allow Bind9

**Configuring the Secondary DNS server:**

It is generally a good idea to use a secondary DNS server in case the primary is unavailable. For this home lab, a second DNS server may be overkill but for the sake of completeness and simulation of a real environment, a second DNS server has been configured.

So, like before, on **ns2**, we begin by editing ***/etc/bind/named.conf.options*** to specify our trusted hosts and what options they have:

acl “trusted” {

192.168.0.26; # ns1

192.168.0.28; # ns2

192.168.0.25; # ftp1

192.168.0.14; # apollo – Windows host

};

options {

directory “/var/cache/bind”;

recursion yes;

allow-recursion { trusted; };

listen-on { 192.168.0.28; }; # ns2 IP

allow-transfer { none; }; # Disable zone transfers

forwarders {

8.8.8.8;

8.8.4.4;

};

...

};

Now we’ll edit ***/etc/bind/named.conf.local*** to define our zones and that this server is a slave to our primary (master) DNS server:

zone “elgboks.com” {

type slave;

file “db.elgboks.com”;

masters { 192.168.0.26; }; # ns1 IP

};

zone “168.192.in-addr.arpa” {

type slave;

file “db.192.168”;

masters { 192.168.0.26; }; # ns1 IP

};

Once again, we check that the syntax is correct and if it is, we can restart the service and allow it through the firewall and our secondary DNS server is configured:

Sudo named-checkconf

Sudo systemctl restart bind9

Sudo systemctl ufw allow Bind9

**Configuring DNS clients**

For our clients to be able to access our DNS servers, they need to be configured to query them. There is conflicting information on how best to achieve this, especially given how many modern Linux distributions (such as Debian10, which I am using) have dropped support for the very popular, but no longer supported, ***ifconfig***set of packages in favour of the ***ip*** packages which are seen as more powerful.

Ultimately, I edited ***/etc/resolvconf/resolv.conf.d/head*** on each client. To do this you have to first install the *resolvconf* package with:

Sudo apt update

Sudo apt install resolvconf

Then you can edit the ***/etc/resolvconf/resolv.conf.d/head*** file, so that the file looks like so:

Nameserver 192.168.0.26

Nameserver 192.168.0.28

Nameserver 8.8.8.8

Nameserver 8.8.4.4

Search elgboks.com

Domain elgboks.com

Finally restart the resolv.conf service for the changes to take effect by entering the command:

Sudo systemctl restart resolvconf

Testing DNS clients

To test the forward zone, I used the nslookup command to send a query using the hostname where I expected to receive the IP address. I then tested the reverse lookup zone, by using nslookup with the IP address, where I expected to receive the hostname.

Nslookup ns1

Nslookup 192.168.0.26

For good measure, I then pinged various hosts using both their FQDN and IP address which also returned positive results, meaning that the DNS servers were fully up and working.