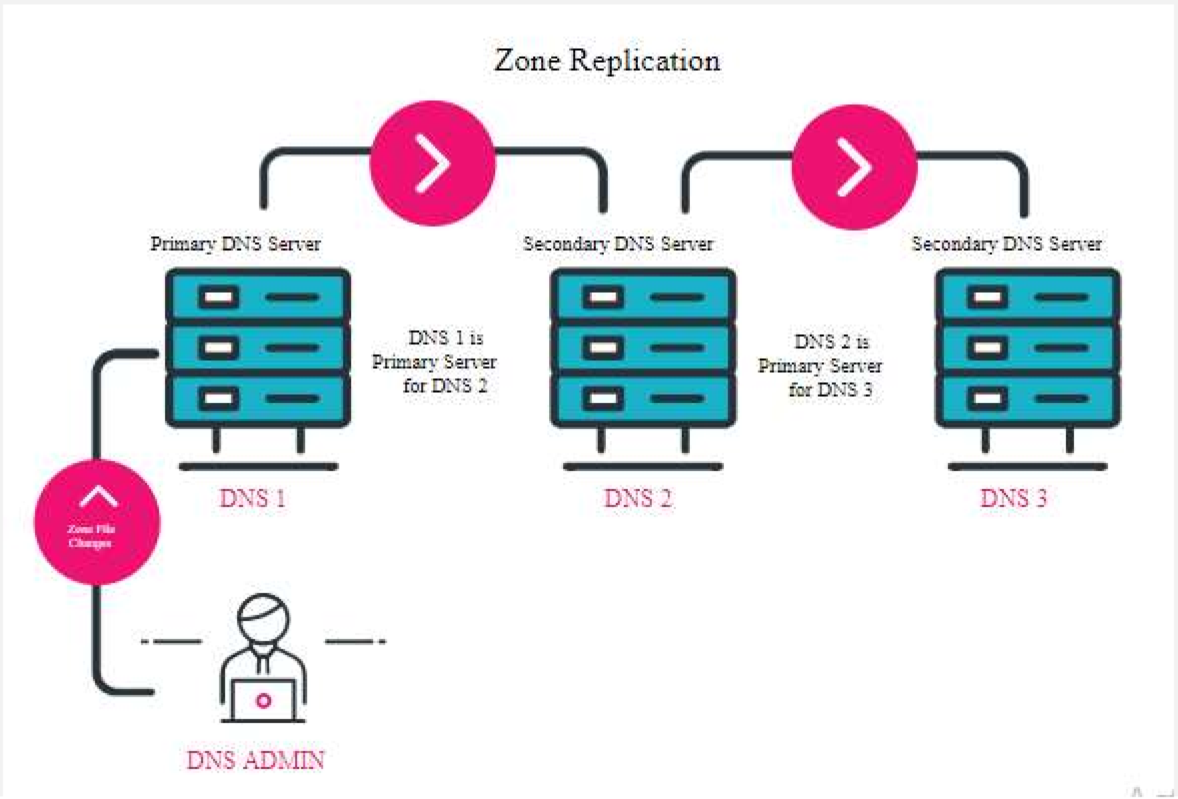
Secondary DNS Server



Why do you need to configure Master Slave DNS Server?

In production environment we are always concerned about uptime and redundancy so with Master Slave DNS server configuration our master server data is automatically replicated to the slave server hence anytime if the master server goes down for some reason then your slave server can act as the primary DNS server by them time you fix the master server.

So you can again ask,

why not have two or more than two DNS server in the network and if one goes down then other one will any how act as primary?

This is correct but in such case you will have to manually perform the same changes on all your DNS servers in the network but if you configure master slave DNS server, any change you do on master DNS server is automatically replicated on the slave DNS server so there is little chance of human error and no re-work needed hence it is recommended to configure **Master Slave DNS Server**.

Benefits of having a secondary DNS server for a domain:

- **Provides redundancy** in case the primary DNS server goes down. If there is no secondary server, when the primary fails, the website will become unavailable at its human-readable domain name (although it will still be accessible by its IP).
- **Distributes the load** between primary and secondary servers. Some resolvers use the Smooth Round Trip Time (SRTT) algorithm to prefer the lowest latency name server from the available pool of servers (primary and one or more secondaries).
- **Part of a secure DNS strategy**—DNS servers are exposed to security threats, first and foremost Distributed Denial of Service attacks (DDoS). Setting up an external DNS provider with DDoS protection as a secondary DNS, is a common way to deflect DDoS attacks.

➪ **Primary DNS Side Configuration:**

==============================

#vi /etc/named.conf

```
// Adding forward zone

zone "atcomputer.net" IN {

 type master;

 file "forward.atcomputer.net";

 allow-transfer { 192.168.3.3; };        Allow Secondary Server.

};

// Adding Reverse zone

zone "3.168.192.in-addr.arpa" IN {

 type master;

 file "revese.atcomputer.net";

 allow-transfer { 192.168.3.3; };         Allow Secondary Server.

};

:x
```

#cd /var/named/

#vim forward.atcomputer.net

```
NS  ns2.atcomputer.net.                 Add Secondary Server Info.

A   192.168.3.3                         Add Secondary Server Info.

ns1    IN    A   192.168.3.2

ns2    IN    A   192.168.3.3            Add Secondary Server Info.
```

:x

#vim reverse.atcomputer.net

```
                    NS  ns2.atcomputer.net.

         A   192.168.10.10
```

:x

Secondary DNS Side Configuration:

Step-1: Time, Date and Time Zone Configuration:
#timedatectl set-timezone Asia/Dhaka
#timedatectl

Step-2: Set SELinux Off:
#vi /etc/selinux/config
SELINUX=disabled
:x

#reboot **mendetory**

Step-3: Hostname Configuration:
#hostnamectl set-hostname ns1.atcomputer.net
# logout

Login again to check the effect
#hostname
ns1.atcomputer.net

Step-4: Network Configuration:

#nmtui

A DNS Server must have a static IP address, let's verify is the case:

```
#cat /etc/sysconfig/network-scripts/ifcfg-enp0s3|egrep -i
"boot|ipaddr|mask|gateway"
```

Which, for instance, yields the below results:

```
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.0.0.63
NETMASK=255.255.255.0
GATEWAY=10.0.0.1
```

```
#  systemctl reload NetworkManager
```

or

```
#nmcli connection down enp0s3
#nmcli connection up enp0s3
# ifconfig
```

Step-5: Update hosts file:

```
# vim /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
192.168.3.3 ns2.atcomputer.net ns2
:x
```

To check the effect

```
[root@mail ~]# hostname
ns1.atcomputer.net
[root@mail ~]# hostname -d
atcomputer.net
[root@mail ~]# hostname -f
ns1.atcomputer.net
```

Step-6: SSH Server Configuration:

=>Create a new user for ssh secure login:

```
    #adduser sysadmin
    #passwd sysadmin
```

=> Necessary Packages Install:

```
#dnf -y install openssh-clients openssh-server
```

=> Edit the SSH Configuration File:

```
#cd /etc/ssh/
#vim sshd_config

Port 6622 Change ssh port
PermitRootLogin no Deny root login
AllowUsers system zimbra Permit SSH Users

:x
```

=> Allow ssh port into the Firewall:

```
#firewall-cmd –permanent –add-port=6622/tcp
#firewal-cmd –reload
```

=> Restart SSH Service:

```
#systemctl status sshd
#systemctl restart sshd
```

Try to login remotely

```
#ssh system@192.168.10.5 –p 6622
```

Step-07: DNS Server Configuration:

=> Necessary Packages Installation:

```
# dnf install –y bind bind-utils
```

=> Edit the DNS Configuration Files:

```
# cd /etc/

# cp named.conf named.conf.ori

# vim /etc/named.conf
```

```
options {
        listen-on port 53 { 127.0.0.1; 192.168.3.3; };          Provide the Secondary DNS Server IP Address

//      listen-on-v6 port 53 { ::1; };                          Comments this line

allow-query             { any; }
allow-recursion { localhost; 192.168.3.0/24; };

//Adding Forward Zone Info

zone "atcomputer.net" IN {
type slave;
masters { 192.168.3.2; };
file “slaves/forward.atcomputer.net”;
};

//Adding Reverse Zone Info
zone “3.168.192.in-addr.arpa” IN {
                type slave;
masters { 192.168.3.3; };
file "slaves/reverse.atcomputer.net";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

```
:X
```

```
#systemctl restart named
#systemctl enable named

#systemctl status named
```

Add a allow rule in the firewall to let clients can connect to DNS server for name resolution.

```
firewall-cmd --permanent --add-port=53/udp

firewall-cmd --reload
```

Verify Master to Slave Data Replication

Create automatically on the slave server under `/var/named/slaves/`

```
[root@slave ~]# ls -l /var/named/slaves/

total 8

-rw-r--r-- 1 named named 386 Jun 22 02:30 example.com.rzone

-rw-r--r-- 1 named named 410 Jun 22 02:47 example.com.zone
```

NOTE:

These zone files will not be in readable format and should not be modified as they get updated automatically every time the `named-chroot` service is reloaded or restarted on the Master DNS Server.

Now since our `named-chroot` service is running on the Slave DNS Server, the zones related data will be instantly transferred from master to slave under `/var/named/data/named.run`

Below is a sample content from my `/var/named/data/named.run`

```
managed-keys-zone: loaded serial 2

zone 0.in-addr.arpa/IN: loaded serial 0

zone 2.0.10.in-addr.arpa/IN: loaded serial 2

zone 1.0.0.127.in-addr.arpa/IN: loaded serial 0

zone example.com/IN: loaded serial 6
zone localhost.localdomain/IN: loaded serial 0

zone 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa/IN: loaded serial 0

zone localhost/IN: loaded serial 0

all zones loaded

running
```

We can verify this by modifying our forward zone file on the Master DNS Server by adding a new **A record**

```
[root@master ~]# vim /var/named/forward.atcomputer.net

$TTL 1D

@       IN SOA  example.com       root (

                                        7       ; serial

                                        1D      ; refresh

                                        1H      ; retry

                                        1W      ; expire

                                        3H )    ; minimum

                IN NS   master

master          IN A    10.0.2.32

localhost       IN A    127.0.0.1

client          IN A    10.0.2.30

slave           IN A    10.0.2.31

test            IN A    10.0.2.33
```

Here we have added a new **A record** for "**test**" and updated the **serial number**.

IMPORTANT NOTE:

Make sure you update the serial number every time you modify the zone files or else the slave will fail to get an update.

Next reload the `named-chroot` service and observe the logs under `/var/named/data/named.run` on slave server

```
[root@master ~]# systemctl reload named-chroot
```

Below logs are seen on slave server under `/var/named/data/named.run`.

So our transfer was successfully completed:

```
client 10.0.2.32#25912: received notify for zone 'example.com'

zone example.com/IN: Transfer started.

transfer of 'example.com/IN' from 10.0.2.32#53: connected using
10.0.2.31#39338

zone example.com/IN: transferred serial 7

transfer of 'example.com/IN' from 10.0.2.32#53: Transfer completed: 1
messages, 8 records, 247 bytes, 0.002 secs (123500 bytes/sec)

zone example.com/IN: sending notifies (serial 7)
```

Next verify the **A record** resolution on slave server

```
[root@slave ~]# nslookup test.example.com

Server:         10.0.2.32

Address:        10.0.2.32#53



Name:   test.example.com

Address: 10.0.2.33
```

So the data has been transferred successfully so our master to slave replication is working successfully.