Assignment 1: **Background:** I am a system administrator responsible for managing the IT infrastructure of a medium-sized company. My company is expanding its online presence and needs a reliable DNS server to manage internal and external domain name resolution. I have decided to implement this on Red Hat Enterprise Linux due to its stability, security features, and robust support.

Step 1: Server Setup

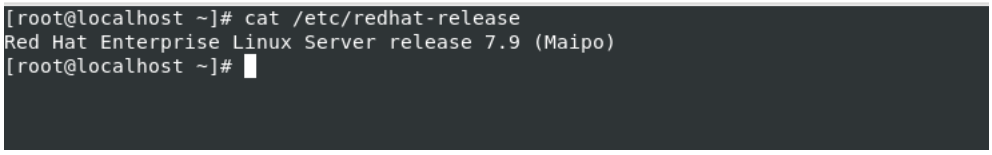
CPU Core = 2

RAM = 2 GB

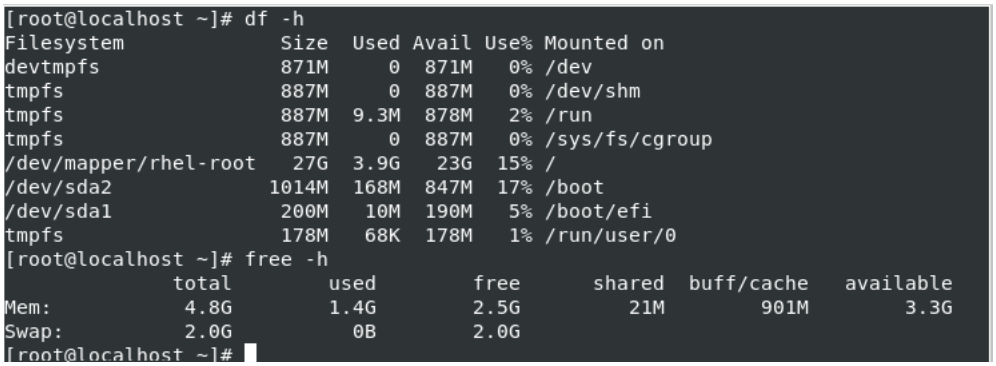
Disk Space = 30 GB

commands and paths to show the CPU RAM and Disk Space

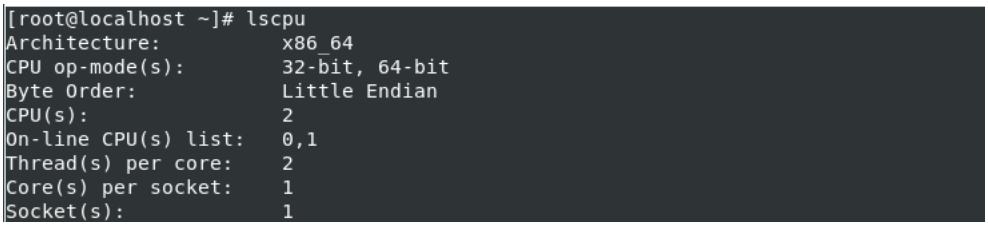
- **`cat /etc/redhat-release`** → Displays the Red Hat OS version



- **`df -h`** → Shows available disk space in a human-readable format
- **`free -h`** → Displays memory (RAM) usage in a human-readable format



- **`lscpu`** → Provides detailed information about the CPU



- To create a DNS server, it is necessary to set a **Fully Qualified Domain Name (FQDN)**. Use the following commands to configure and verify the hostname:

  `hostnamectl set-hostname` – Use this command to set a fully qualified domain name. **`hostnamectl`** – Check the current status of the hostname.

```
[root@localhost ~]# hostnamectl set-hostname linuxpc.mofi61.com
[root@localhost ~]# hostnamectl
   Static hostname: linuxpc.mofi61.com
         Icon name: computer-vm
           Chassis: vm
        Machine ID: bbdb4b18355a45419b108109d26a0c85
           Boot ID: e2142ee37f4a43d38eb046e7b4bb99f4
    Virtualization: microsoft
  Operating System: Red Hat Enterprise Linux Server 7.9 (Maipo)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:7.9:GA:server
            Kernel: Linux 3.10.0-1160.el7.x86_64
      Architecture: x86-64
[root@localhost ~]#
```

- **`dnsdomainname`** – Check the domain name portion of the FQDN.

```
[root@localhost ~]# dnsdomainname
mofi61.com
[root@localhost ~]#
```

Step 2: Installation of DNS Software

- **`yum`** – Package manager used in RHEL-based systems
- **`bind`** – DNS server software
- **`bind-utils`** – Utilities like `dig`, `nslookup`, etc.
- **`-y`** – Automatically confirms the installation without prompting

```
[root@localhost ~]# yum install bind* -y
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package bind.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-chroot.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-dyndb-ldap.x86_64 0:11.1-7.el7_9.1 will be installed
---> Package bind-export-libs.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-libs.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-libs-lite.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-license.noarch 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-pkcs11.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-pkcs11-libs.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-pkcs11-utils.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
---> Package bind-utils.x86_64 32:9.11.4-26.P2.el7_9.16 will be installed
```

Step 3: Configure the Main DNS Configuration File

To define the DNS server's behavior, zones, and resolution options, you need to edit the BIND configuration files. These files are typically located in the **`/etc/named/`** directory and include:

- **named.conf**
- **named.conf.options**
- **named.conf.local**
- **Other zone-specific configuration files**
- • **listen-on port 53 { 192.168.0.100; };**

  ➤ Specifies that the DNS server listens for connections on port **53** at the IP address **192.168.0.100**.
- • **allow-query { 192.168.0.0/24; };**

```
options {
        listen-on port 53 { 192.168.0.100; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        recursing-file  "/var/named/data/named.recursing";
        secroots-file   "/var/named/data/named.secroots";
        allow-query     { localhost; 192.168.0.0/24; };
```

Configure main configuration file **vi /etc/named.rfc1912**

```
zone "mofi61.com" IN {
        type master;
        file "mofi61.forward";
        allow-update { none; };
};
zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "mofi61.reverse";
        allow-update { none; };
};
```

- **forwarders { 8.8.8.8; };**

  ➤ Sets up DNS forwarding to an external resolver, such as **Google DNS**, for resolving external domain names.

- **dnssec-enable yes;**

  ➤ Enables DNSSEC support to help verify the authenticity of DNS responses.
- **dnssec-validation yes;**

  ➤ Activates DNSSEC validation to ensure that data has not been tampered with.

```
        recursion yes;
        forwarders {
                8.8.8.8;
                8.8.4.4;
        };

        dnssec-enable yes;
        dnssec-validation yes;
```

In this step, we will create custom zone data files for our DNS server by copying the default templates provided by BIND.

The default zone files are located in the **`/var/named/`** directory. These include:

- **`named.localhost`** – Used for loopback (forward) zone
- **`named.loopback`** – Used for reverse zone

These commands:

- Change the current directory to **`/var/named`**
- Copy **`named.localhost`** to create **`mofi61.forward`**
- Copy **`named.loopback`** to create **`mofi61.reverse`**

```
[root@localhost ~]# cd /var/named/
[root@localhost named]# ls
chroot  dynamic      named.ca     named.localhost  slaves
data    dyndb-ldap   named.empty  named.loopback
[root@localhost named]# cp named.localhost mofi61.forward
[root@localhost named]# cp named.loopback mofi61.reverse
[root@localhost named]# ls
chroot  dynamic      mofi61.forward  named.ca     named.localhost  slaves
data    dyndb-ldap   mofi61.reverse  named.empty  named.loopback
[root@localhost named]#
```

Step 4: Creating DNS Zones

Generate DNSSEC Keys for Zone Signing

To secure the zone **`mofi61.com`** with DNSSEC, generate a **Key Signing Key** (KSK) and a **Zone Signing Key** (ZSK):

Generate Key Signing Key (**KSK**) : **dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK mofi61.com**

Generate Key Signing Key **(KSK**) : **dnssec-keygen -a RSASHA256 -b 2048 -n ZONE mofi61.com**

- **`-a RSASHA256`** → Algorithm used for key generation
- **`-b 2048`** → Key size in bits
- **`-n ZONE`** → Specifies the key is for a DNS zone
- **`-f KSK`** → Flags the key as a Key Signing Key (for the first command only)

```
[root@localhost named]# dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK mofi61.c
om
Generating key pair.................+++ .................+++
Kmofi61.com.+008+34231
[root@localhost named]# dnssec-keygen -a RSASHA256 -b 2048 -n ZONE mofi61.com
Generating key pair.....+++ .....+++
Kmofi61.com.+008+38370
[root@localhost named]#
```

- DNSSEC Key Generation for Reverse Zone

To secure the **reverse lookup zone** (`0.168.192.in-addr.arpa`) with **DNSSEC**, we need to generate a **Key Signing Key (KSK)** and a **Zone Signing Key (ZSK)** specifically for this zone.

Generate Key Signing Key **(KSK**): **dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK 0.168.192.in-addr.arpa**

Generate Zone Signing Key **(ZSK): dnssec-keygen -a RSASHA256 -b 2048 -n ZONE 0.168.192.in-addr.arpa**

- **`-a RSASHA256`** → Uses the RSASHA256 algorithm for key generation
- **`-b 2048`** → Specifies a key size of 2048 bits
- **`-n ZONE`** → Indicates the key is being generated for a DNS zone
- **`-f KSK`** → Flags the key as a Key Signing Key (used only with the first command)

These keys will be used later to digitally sign the reverse zone file and enable **DNSSEC protection** for reverse lookups.

```
[root@localhost named]# dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK 0.168.19
2.in-addr.arpa
Generating key pair.....................................................................
.................+++ .................+++
K0.168.192.in-addr.arpa.+008+10114
[root@localhost named]# dnssec-keygen -a RSASHA256 -b 2048 -n ZONE 0.168.192.in-ad
dr.arpa
Generating key pair.....................................+++ ..................
.....................................................................................
.....................................................................................
.........................+++
K0.168.192.in-addr.arpa.+008+06330
[root@localhost named]#
```

Step :5 Edit the Forward Zone File (`mofi61.forward`)

In this step, we will configure the **forward zone file** by including the DNSSEC key files and adding necessary DNS records such as **A**, **MX**, and **CNAME** records.

Open the Forward Zone File: vim mofi61.com

Include DNSSEC Public Keys:

Add the following lines at the beginning or appropriate place in the zone file to include the public keys:

- **`$INCLUDE`** tells BIND to read and include the contents of another file.

- The `KSK.key` and `ZSK.key` files contain the public portions of the **Key Signing Key** and **Zone Signing Key** generated earlier.

```
$TTL 1D
$INCLUDE "Kmofi61.com.+008+34231.key"
$INCLUDE "Kmofi61.com.+008+38370.key"


@       IN SOA  linuxpc.mofi61.com.     root.mofi61.com. (
                                        2       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
@       IN      NS      linuxpc.mofi61.com.

primary IN      A       192.168.0.100
linuxpc IN      A       192.168.0.100
mail    IN      A       192.168.0.100
```

- Edit the Reverse Zone File (**`mofi61.reverse`**)

Now that the forward zone is configured, we will edit the **reverse zone file** to include DNSSEC public key files and define **PTR records** for reverse DNS resolution.

Open the Reverse Zone File: vim mofi61.reverse

Include DNSSEC Public Keys:

Add the following lines to load the public keys required for DNSSEC:

- `$INCLUDE` instructs BIND to import the content of the specified key files.
- These key files

must be generated previously using **`dnssec-keygen`** and stored in **`/var/named/.`**

```
$TTL 1D
$INCLUDE "K0.168.192.in-addr.arpa.+008+10114.key"
$INCLUDE "K0.168.192.in-addr.arpa.+008+06330.key"


@       IN SOA  linuxpc.mofi61.com.     root.mofi61.com. (
                                        5       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
@       IN      NS      linuxpc.mofi61.com.

100     IN      PTR     linuxpc.mofi61.com.
```

- **SOA (Start of Authority):** Begins the reverse zone file.
- **NS Record:** Points to the nameserver **(`ns1.mofi61.com`).**
- **PTR Records:** Map IP addresses (last octet) back to domain names.

Modify permission for zone files chmod:

change file permissions

```
[root@localhost named]# chmod 777 mofi61.forward
[root@localhost named]# chmod 777 mofi61.reverse
[root@localhost named]#
```

Sign the Forward Zone

Sign the Forward Zone File**: dnssec-signzone -A -N INCREMENT -o mofi61.com -t mofi61.forward**

- **`-A`** → Appends DNSSEC records to the existing zone file.
- **`-N INCREMENT`** → Automatically increments the serial number in the SOA record.
- **`-o mofi61.com`** → Specifies the origin of the zone (i.e., the domain name).
- **`-t`** → Prints signing statistics and progress.
- **`mofi61.forward`** → Name of the zone file to be signed.

```
[root@localhost named]# dnssec-signzone -A -N INCREMENT -o mofi61.com -t mofi61.fo
rward
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                      ZSKs: 1 active, 0 stand-by, 0 revoked
mofi61.forward.signed
Signatures generated:                       11
Signatures retained:                         0
Signatures dropped:                          0
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Signing time in seconds:                 0.017
Signatures per second:                 618.985
Runtime in seconds:                      0.034
[root@localhost named]#
```

Sign the Reverse Zone

Sign the Reverse Zone File: dnssec-signzone -A -N INCREMENT -**o 0.168.192**.in-addr.arpa -t **mofi61.reverse**

- `-A` → Appends DNSSEC records (RRSIG, NSEC, etc.) to the zone file.
- `-N INCREMENT` → Automatically increases the serial number in the SOA record.
- `-o 0.168.192.in-addr.arpa` → Sets the origin of the reverse DNS zone.
- `-t` → Displays signing statistics during the process.
- `mofi61.reverse` → The reverse zone file to be signed.

```
[root@localhost named]# dnssec-signzone -A -N INCREMENT -o 0.168.192.in-addr.arpa
-t mofi61.reverse
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                      ZSKs: 1 active, 0 stand-by, 0 revoked
mofi61.reverse.signed
Signatures generated:                       7
Signatures retained:                        0
Signatures dropped:                         0
Signatures successfully verified:           0
Signatures unsuccessfully verified:         0
Signing time in seconds:                0.011
Signatures per second:                624.888
Runtime in seconds:                     0.026
[root@localhost named]#
```

Update the **/etc/named.rfc1912.zones** for forward zone,

```
[root@linuxpc ~]# vim /etc/named.rfc1912.zones
[root@linuxpc ~]# systemctl restart named
zone "mofi61.com" IN {
        type master;
        file "mofi61.forward.signed";
        allow-transfer { none; };
        allow-update { none; };
};
```

update for reverse zone,

```
zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "mofi61.reverse.signed";
        allow-transfer { none; };
        allow-update { none; };
};
```

Modify permission for zone files

chmod: change file permissions Owner

(root): Read and write (can edit the file)

```
[root@linuxpc ~]# chown root:named /var/named/mofi61.forward.signed
[root@linuxpc ~]# chown root:named /var/named/mofi61.reverse.signed
[root@linuxpc ~]# chmod 640 /var/named/mofi61.forward.signed
[root@linuxpc ~]# chmod 640 /var/named/mofi61.reverse.signed
[root@linuxpc ~]#
```

Allow DNS Port Through Firewall

To ensure that DNS queries can reach your server, you need to **allow DNS port (53)** through the system firewall.

```
[root@localhost named]# firewall-cmd --permanent --add-service=dns
Warning: ALREADY_ENABLED: dns
success
[root@localhost named]# firewall-cmd --reload
success
[root@localhost named]#
```

Restrict Zone Transfers

Add **`allow-transfer`** Directive to **`named.conf`**: vim /etc/named.conf & vim /etc/named.rfc1912.conf

Inside the zone definition block or global options, add the following directive: allow-transfer { none; };

- `allow-transfer` controls which clients can request a zone transfer.
- Setting it to `{ none; };` completely disables zone transfers, enhancing server security.

```
zone "mofi61.com" IN {
        type master;
        file "mofi61.forward";
        allow-transfer { none; };
        allow-update { none; };
};
zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "mofi61.reverse";
        allow-transfer { none; };
        allow-update { none; };
};
```

Step 6: Testing and Validation of DNS Server

Once all configurations, zone setups, and DNSSEC settings are complete, it is essential to **test the DNS server** to ensure it is functioning correctly and securely

Testing DNS Resolution Using **`dig`**

The **`dig`** (Domain Information Groper) tool is a powerful utility used to query DNS servers and inspect their responses. dig linuxpc.mofi61.com

```
[root@localhost named]# dig linuxpc.mofi61.com

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el7_9.16 <<>> linuxpc.mofi61.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4372
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;linuxpc.mofi61.com.            IN      A

;; ANSWER SECTION:
linuxpc.mofi61.com.     86400   IN      A       192.168.0.100

;; AUTHORITY SECTION:
mofi61.com.             86400   IN      NS      linuxpc.mofi61.com.

;; Query time: 0 msec
;; SERVER: 192.168.0.100#53(192.168.0.100)
;; WHEN: Wed Apr 09 00:08:07 +06 2025
;; MSG SIZE  rcvd: 77
```

Validating with **`nslookup`**

Another user-friendly tool for DNS testing is **`nslookup`**, which works in interactive mode.

nslookup linuxpc.mofi61.com

```
[root@localhost named]# nslookup linuxpc.mofi61.com
Server:         192.168.0.100
Address:        192.168.0.100#53

Name:   linuxpc.mofi61.com
Address: 192.168.0.100

[root@localhost named]#
```

step :7 Set Up DNS Logging

To monitor DNS queries, server activity, and errors, you should **configure logging for `named`**. This helps with troubleshooting and maintaining server health.

Create Logging Directory

```
[root@linux ~]# mkdir -p /var/log/named
[root@linux ~]# touch /var/log/named/named.log
[root@linux ~]# chown named:named /var/log/named/named.log
[root@linux ~]# chmod 644 /var/log/named/named.log
```

Configure Logging in **`named.conf`**

To enable DNS query logging, you need to define a logging section inside the main configuration file.

Add the Following Logging Block:

```
logging {
        channel default_log {
                file "/var/log/named/named.log";
                severity info;
                print-time yes;
                print-severity yes;
                print-category yes;
        };

        category default { default_log; };
        category queries { default_log; };
        category config { default_log; };
        category resolver { default_log; };
        category notify { default_log; };
        category client { default_log; };
```

- **`file`**: Specifies the path to the log file **(`/var/log/named/named.log`).**
- **`severity info`**: Logs informational messages.
- **`print-time yes`**: Includes a timestamp in each log entry.
- **`print-severity yes`**: Shows the severity level (e.g., info, warning, error).

Test DNS Log Output

After configuring logging in **named.conf** and ensuring proper permissions for the log directory, you can monitor DNS activity in real-time using the following command: **tail -f /var/log/named/named.log**

```
[root@linuxpc ~]# tail -f /var/log/named/named.log
13-Apr-2025 21:08:10.752 lame-servers: info: network unreachable resolving './NS/IN'
: 2001:500:a8::e#53
13-Apr-2025 21:08:10.752 lame-servers: info: network unreachable resolving './NS/IN'
: 2001:dc3::35#53
13-Apr-2025 21:08:10.752 lame-servers: info: network unreachable resolving './NS/IN'
: 2001:500:9f::42#53
13-Apr-2025 21:08:11.270 resolver: info: resolver priming query complete
13-Apr-2025 21:08:11.270 general: warning: checkhints: b.root-servers.net/A (170.247
.170.2) missing from hints
13-Apr-2025 21:08:11.270 general: warning: checkhints: b.root-servers.net/A (199.9.1
4.201) extra record in hints
13-Apr-2025 21:08:11.270 general: warning: checkhints: b.root-servers.net/AAAA (2801
:1b8:10::b) missing from hints
13-Apr-2025 21:08:11.270 general: warning: checkhints: b.root-servers.net/AAAA (2001
:500:200::b) extra record in hints
13-Apr-2025 21:08:11.326 general: info: managed-keys-zone: Key 20326 for zone . acce
ptance timer complete: key now trusted
13-Apr-2025 21:08:11.326 general: info: managed-keys-zone: Key 38696 for zone . acce
ptance timer complete: key now trusted
13-Apr-2025 21:15:30.420 queries: info: client @0x7fe32403c040 192.168.0.100#42761 (
localhost.mofi61.com): query: localhost.mofi61.com IN A + (192.168.0.100)
13-Apr-2025 21:15:30.420 queries: info: client @0x7fe32403c040 192.168.0.100#45814 (
localhost): query: localhost IN A + (192.168.0.100)
```

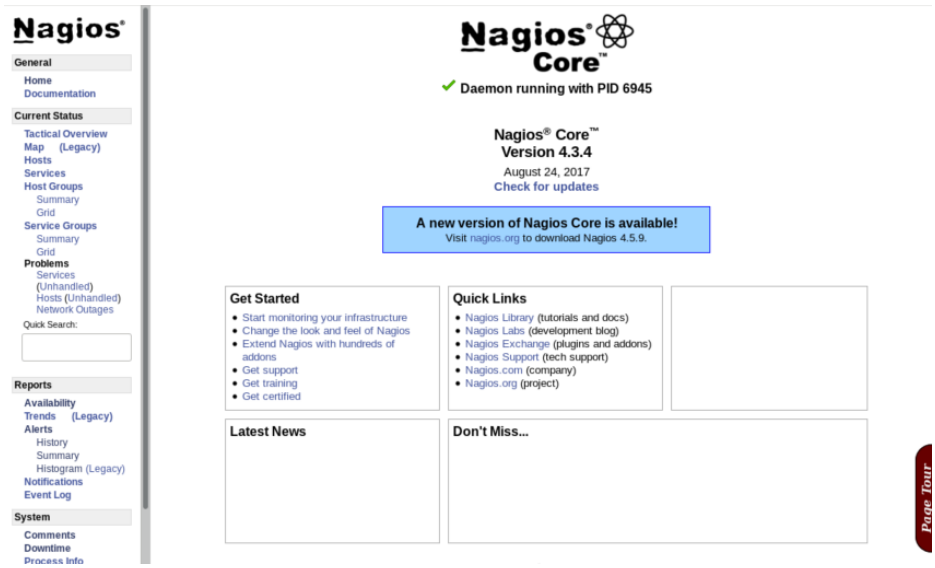
We can also check dns server error or critical event on /**var/log/** directory with **tail –f /var/log/messages** command

```
[root@linuxpc ~]# tail -f /var/log/messages | grep named
Apr 13 21:08:10 linuxpc named[3035]: command channel listening on 127.0.0.1#953
Apr 13 21:08:10 linuxpc named[3035]: configuring command channel from '/etc/rndc.key
'
Apr 13 21:08:10 linuxpc named[3035]: command channel listening on ::1#953
```

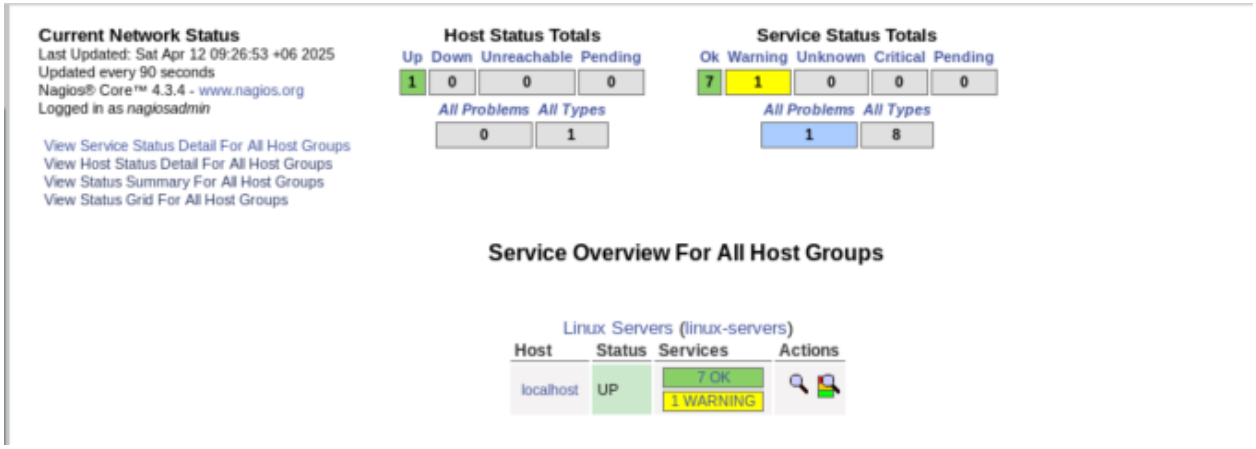
Configure DNS Monitoring and Alerting

To ensure the continuous availability and performance of your DNS server, it is essential to implement a monitoring and alerting system

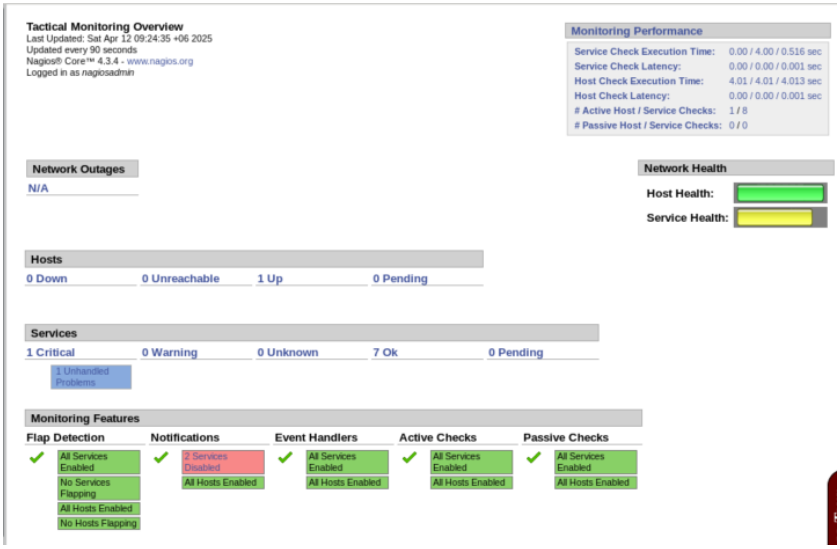
Set Up Monitoring with Tools like Nagios: Integrate monitoring solutions such as **Nagios**, **Zabbix**, or **Prometheus** to continuously observe the health of the DNS server and receive immediate alerts in case of issues.



Configure DNS Monitoring and Performance Tracking



Monitoring Host and Service Status in Nagios : Nagios is a powerful monitoring solution that continuously checks the **availability**, **performance**, and **health** of both systems (hosts) and the services running on them.



Step :8 Backup and Disaster Recovery for DNS Server

Create a Local Backup Directory**: mkdir -p /dns_backups**

Create the DNS Backup Script:

Use a text editor to create a backup script file: **vim /usr/local/bin/dns_backup.sh**

Make the Script Executable: chmod +x /usr/local/bin/dns_backup.sh

```
[root@linuxpc ~]# mkdir -p /dns_backups
[root@linuxpc ~]# vim /usr/local/bin/dns_backups.sh
[root@linuxpc ~]# chmod +x /usr/local/bin/dns_backups.sh
```

I have created a backup and disaster recovery script for my own machine. In addition to local backups, I am using a **remote server** to store the backup files. The **remote server IP is `192.168.0.110`**, and it automatically receives backup files **every night at 2:00 (2:00 AM)**.

To enable this, I have ensured **SSH is properly configured** between both servers for secure file transfer. The backup is done using the **`SCP` could also be used** as an method.

On the **remote server**, I created a backup directory at: **/home/dns_backup**

```
#!/bin/bash
BACKUP_DIR="/dns_backups"
NAMED_CONF="/etc/named.conf"
ZONE_DIR="/var/named"
DATE=$(date +%Y-%m-%d_%H-%M-%S)
BACKUP_FILE="dns_backups_$DATE.tar.gz"
LOG_FILE="/var/log/dns_backup.log"
REMOTE_USER="root"
REMOTE_HOST="192.168.0.110"
REMOTE_DIR="/home/dns_backup"

mkdir -p $BACKUP_DIR
echo "[$DATE] Backup started." >> $LOG_FILE

tar -czf $BACKUP_DIR/$BACKUP_FILE $NAMED_CONF $ZONE_DIR

if [ $? -eq 0 ]; then
    echo "[$DATE] Backup created: $BACKUP_FILE" >> $LOG_FILE
else
    echo "[$DATE] Backup failed" >> $LOG_FILE
    exit 1
fi
```

Automating Backup with `cron`: To ensure daily automated backups for the DNS server, I have scheduled cron jobs using the **`crontab`** utility

Editing the Cron Table

Open the crontab for the current user with the following command: **crontab –e**

Add the Following Lines to Schedule Backup Jobs: 0 2 * * * /usr/local/bin/dns_backups.sh

0 2 * * * scp /usr/local/bin/dns_backup.sh root@192.168.0.110:home/dns_backup

To check all scheduled cron jobs for the current user, run**: crontab -l**

```
[root@linuxpc ~]# crontab -e
crontab: installing new crontab
[root@linuxpc ~]# crontab -l
0 2 * * * /usr/local/bin/dns_backups.sh
[root@linuxpc ~]#
```

Now check the backup in dns server to dive into the **cd /dns_backups** directory

```
[root@linuxpc ~]# /usr/local/bin/dns_backups.sh
tar: Removing leading `/' from member names
The authenticity of host '192.168.0.110 (192.168.0.110)' can't be established.
ECDSA key fingerprint is SHA256:Vwefp1XfwfNeLrZlSJiv1ST8QgMiVdnsLzo7O3x2KoM.
ECDSA key fingerprint is MD5:7d:9e:2d:4e:a2:02:96:83:9c:4a:5d:ea:63:22:4a:21.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.110' (ECDSA) to the list of known hosts.
root@192.168.0.110's password:
dns_backups_2025-04-12_20-09-36.tar.gz        100% 2332   275.1KB/s   00:00
[root@linuxpc ~]# cd /dns_backups/
[root@linuxpc dns_backups]# ls
dns_backups_2025-04-12_20-09-36.tar.gz
[root@linuxpc dns_backups]#
```

Now check the backup in remote server to dive into the **cd /home/dns_backup** and also verify the remote server IP 192.168.0.110

```
[root@linuxpc ~]# mkdir -p /home/dns_backup
[root@linuxpc ~]# cd /home/dns_backup
[root@linuxpc dns_backup]# ls
dns_backups_2025-04-12_20-09-36.tar.gz
[root@linuxpc dns_backup]#
```

Step 09:DNS Server Setup Documentation

This document details the installation, configuration, and customization of a DNS server using BIND on RHEL 7.9. It includes all essential steps, file modifications, and tips for troubleshooting and maintenance

System Information

- **OS**: Red Hat Enterprise Linux 7.9
- **DNS Software**: BIND (Berkeley Internet Name Domain)
- **Hostname**: `dns.example.com`
- **Domain**: `example.com`
- **Forward Zone File**: `/var/named/example.com.zone`
- **Reverse Zone File**: `/var/named/1.168.192.in-addr.arpa.zone`
- **Backup Directory**: `/dns_backups`

troubleshooting Tips

Check syntax: `named-checkconf` and `named-checkzone`

named-checkconf

named-checkzone example.com /var/named/example.com.zone

named-checkzone 1.168.192.in-addr.arpa /var/named/1.168.192.in-addr.arpa.zone

Check logs for errors: tail -f /var/log/messages

- **Scheduled Maintenance and Update Policy**
  Regular Maintenance Tasks

Task	Purpose	Frequency
Update BIND Packages	Apply patches to BIND (`bind`, `bind-utils`)	Weekly
Apply RHEL Security Updates	Update core system packages and dependencies	Weekly
Backup DNS Configuration	Backup `/etc/named.conf` and all zone files	Daily
Verify Zone File Syntax	Ensure zone file correctness with `named-checkzone`	After changes
Review Logs	Check `/var/log/messages` for DNS or system errors	Daily
Firewall Review	Verify only required ports (53/tcp & 53/udp) are open	Monthly

How to Apply Updates (Manual)

yum update bind bind-utils –y

yum update -y