BIND9 for DNS Management on RHEL

BIND (Berkeley Internet Name Domain) is typically provided by the bind package along with several supporting packages:
sudo dnf install bind bind-utils bind-libs bind-chroot bind-dyndb-ldap bind-ldap-schema

- bind: The main BIND DNS server package.
- bind-utils: Utilities for querying DNS servers.
- bind-libs: Libraries for BIND.
- bind-chroot: Allows running BIND in a chroot jail for added security.
- bind-dyndb-ldap: A dynamic plug-in to use LDAP as a backend for BIND.
- bind-ldap-schema: LDAP schema extensions for BIND to integrate with LDAP

Configuration locations and notable files

/etc/named - A directory that typically holds zone files and additional configuration files.
/etc/rndc.key - The key file for secure communication between rndc and named.

/etc/named.conf	Central config file for the DNS server (global options, logging config, zone declarations
/var/named/*.zone	Zone files, domain-based name (example.com.zone); define a domain's DNS records
/etc/named/keys/	Directory for DNSSEC keys
/etc/named.rfc1912.zones	Used to include addl. zone files and settings
/etc/rndc.conf and /etc/rndc.key	Configuration and key files for the rndc utility, to securely control the BIND daemon
bind-dyndb-ldap	LDAP back-end for BIND, to enable storage of DNS data in LDAP

When using the bind-chroot package, the configuration files are located in the /var/named/chroot/etc/ directory.

Logs:
Syslog: BIND can be configured to send its logs to the system syslog (/var/log/messages or /var/log/syslog
Log Files: If configured explicitly in the BIND configuration file (named.conf),
A few pages into this, I put instructions on setting up SNMP for logs and network monitoring tool

DNS Configuration and File Locations: Debian vs. RHEL 9

File/Directory Type	Debian Default Location	RHEL 9 Default Location
BIND Main Config File	/etc/bind/named.conf	/etc/named.conf
BIND Addl Config Files	/etc/bind/named/	/etc/named/
Zone Files	/etc/bind/zones/ or /var/lib/bind/	/var/named/
Root Hints File	/usr/share/dns/root.hints	/var/named/named.ca
RNDC Configuration File	/etc/bind/rndc.conf	/etc/rndc.conf
RNDC Key File	/etc/bind/rndc.key	/etc/rndc.key
BIND Log Directory	/var/log/bind/	/var/log/named/
DNSSEC Key Files	/etc/bind/keys/ (if specified) or /var/lib/bind/	/etc/named/keys/ or /var/named/
DNSSEC KSK and ZSK Files	Typically in /var/lib/bind/	Typically in /var/named/
DNSSEC Signing Policies	/etc/bind/keys/ or defined in the named.conf	/etc/named/keys/ or defined in named.conf
TSIG Key Files	/etc/bind/ or /etc/bind/tsig/	/etc/named/ or /etc/named/tsig/
Cache Directory	/var/cache/bind/	/var/named/data/
Pid File	/run/named/named.pid	/run/named/named.pid

Zone Files: Can be stored where specified in named.conf but usually /etc/bind/ or /var/lib/bind/ (Debian) or /var/named/ (RHEL).
DNSSEC Files: KSK and ZSK keys sometimes put with zone files, but more common is /etc/bind/keys/ or /etc/named/keys/.
RNDC Key Files: By default, the RNDC key is placed in /etc/rndc.key, but can be overridden in rndc.conf or named.conf.
Config Mgmt: Debian often separates config files into smaller chunks in /etc/bind/, while RHEL puts them in /etc/named/.

DNS Records Review

A	IPv4 Address	Maps a domain name to an IPv4 address	example.com IN A 93.184.216.34
AAAA	IPv6 Address	Maps a domain name to an IPv6 address	example.com IN AAAA 2001:0db8::abcd
CNAME	Canonical Name	Creates an alias for a domain name	www.example.com IN CNAME example.com
MX	Mail Exchange	Specifies the mail server for a domain	example.com IN MX 10 mail.example.com
NS	Name Server	Indicates authoritative DNS servers for a domain	example.com IN NS ns1.example.com
PTR	Pointer	Map IP address to domain name (reverse DNS)	34.216.184.93.in-addr.arpa IN PTR example.com
SRV	Service Locator	Specifies the location of services within a domain	_sip._tcp.example.com IN SRV 10 60 5060 sipserver.example.com
TXT	Text	Stores arbitrary text strings, often for validation	example.com IN TXT "Client ID = 436HJK54J"
DNSKEY	DNS Public Key	Publishes the public key for DNSSEC	example.com IN DNSKEY 256 3 8 AwEAA...
RRSIG	Resource Record Sig	Contains the digital signature for DNS records	example.com IN RRSIG A 5 2 3600 202406270 2023062701 12345 example.com. AwEAA...
DNAME	Delegation Name	Redirects subtree of DNS NS to another domain	sub.example.com IN DNAME newsub example.com
SPF	Sender Policy Framewk	Specifies permitted mail servers for a domain	example.com IN SPF "v=spf1 include:_spf.example.com"
NSEC	Next Secure	Proof of non-existence for DNSSEC records	example.com IN NSEC a.example.com A RRSIG NSEC
NSEC3	Next Secure Version 3	Proof w/ hashed names - see also NSEC3PARAM	3a2a3b4d3b3a4d5b.example.com IN NSEC3 1 0 1
CDNSKEY	Child DS	DNSSEC, Securely delegate child zones	Example.com. IN CDNSKEY 256 3 8 (truncated pub key)
DS	Delegation Signer	Links a DNSSEC-signed child zone to its parent	example.com N DS 12345 8 2 49FD46E6C...
CDS	Child DS	Digest of a CDNSKEY	Example.com IN CDS 256 3 8 (truncated SHA)
CAA	CA Authorization	Specifies allowed CAs for issuing certificates	example.com IN CAA 0 issue "letsencrypt.org"

Installed Executables [/usr/sbin/ and /usr/bin/]

named — Main BIND DNS server daemon and executables

named-pkcs11 — Extended version of named; adds PKCS#11 support for HSM hardware-backed keys for DNSSEC

-c <config-file> Use the specified configuration file
-g Run in the foreground and print logs to stdout
-u <user> Run named as a specified user.
-t <directory> Specify a directory for chroot operation
-p <port> Listen on the specified port (default is 53)
-m <policy> Memory allocation policy
-4 / -6 Use IPv4 or IPv6 exclusively
-S Use a single processing thread
-T <number> Set the number of worker threads
-n <number> Set the number of UDP listeners per interface
-D <directory> Set the working directory for the server
-d <level> Set the debug level (higher levels increase verbosity)
-H Use HSM for cryptographic operations (named-pkcs11 only)
-l Log to syslog instead of stderr (named-pkcs11 only)

```
sudo named -c /etc/named.conf -g
sudo named-pkcs11 -c /etc/named-pkcs11.conf -u named -g -p 53 -t /var/named -k /etc/pki/keys/dnssec -H pkcs11 -l
```

rndc — Remote Name Daemon Control utility to manage named

start, stop, status; reload for reloading config and zones; reconfig to reload config but not zones
flush: Flush all caches
addzone and delzone <zone> Add or delete a zone

named-checkconf — Check syntax of named.conf; use -z to also check the validity of the zone files

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

named-checkzone — Check the syntax and consistency of a zone file

```
sudo named-checkzone <zone> <zone-file>
sudo named-checkzone example.com /var/named/example.com.zone
```

named-compilezone — Compile a zone file into a more efficient binary format.

<zone>: The zone name.
<input-file>: The input zone file.
<output-file>: The output file to write the compiled zone.

```
sudo named-compilezone -f <format> -o <output-file> <zone> <input-file>
sudo named-compilezone -f raw -o /var/named/example.com.zone.db example.com /var/named/example.com.zone
```

dnssec-keygen — Generate DNSSEC keys

-a <algorithm>: Specify the key algorithm
-b <keysize>: Specify the key size
-n <nametype>: Specify the type of owner name (zone, host, etc.)

```
sudo dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.com
```

dnssec-signzone — Sign a zone file with DNSSEC

-o <zone>: Specify the zone name
-k <key>: Specify a key for signing

```
sudo dnssec-signzone -o example.com -k Kexample.com.+008+12345 /var/named/example.com.zone
```

Common options for dnssec-dsfromkey, dnssec-importkey, and dnssec-verify:

-K <directory> Directory where key files are stored.
-l <ttl> Set TTL for the DS record (not in dnssec-verify)
-f <format> Output format (text, full)
-v <level> Set the verbosity level
-c <class> DNS class- default is IN (not in dnssec-dsfromkey)

dnssec-dsfromkey — Creates Delegation Signer (DS) records from DNSKEY records

-2: Use SHA-256 as the hash algorithm.
-a <algorithm>: Specify the hash algorithm (SHA-1, SHA-256, SHA-384).
-f <format>: Output format (text, full).
-T <type> Specify the DNS record type (default is DS)

```
dnssec-dsfromkey -a SHA-256 -f text example.com.dnskey
```

dnssec-importkey — Imports DNSSEC keys into BIND's DNSSEC mgmt system

-c <class>: Specify the DNS class (default is IN).
-e <epoch>: Specify the end of the key's validity period.
-t <type>: Specify the key type (KSK, ZSK).

```
dnssec-importkey -K /var/named/keys -t ZSK example.com example.com.key
```

dnssec-verify — Verifies the signatures in a zone, ensuring zone is correctly signed and validating integrity.

-k <key-file>: Specify a key file to use for verification.
-o <origin>: Specify the zone origin (domain name).
-t <directory>: Specify a directory for temporary files.
-x: Perform a DNSSEC signature expiry check.

```
dnssec-verify -o example.com -K /var/named/keys example.com.zone
```

Review: Zone Transfer Types
AXFR (Authority Full Zone Transfer): Transfers the entire zone file, containing all DNS records for a domain, from the primary authoritative server to a secondary server. This is typically done periodically or after significant changes to the zone file. Due to its size, full transfers can be resource-intensive, especially for large domains.

IXFR (Incremental Zone Transfer): Transfers only the updated portions of the zone file since the last successful transfer. This is the preferred method for frequent updates as it reduces bandwidth consumption and server load. Signed IXFR utilizes DNSSEC to cryptographically sign the transferred data, ensuring its authenticity and preventing tampering.

Review: Root Servers, TLD Servers, Etc
Internet Assigned Numbers Authority (IANA) manages the root zone, that is comprised of 13 sets of root servers globally. They don't perform zone transfers due to the immense load they manage- instead relying on efficient replication. They delegate responsibilities of top-level domains (TLDs) like .com and .org to specific registries such as Verisign (.com, .net, and .name TLDs); ICANN (.gov, .edu, and .mil); Public Interest Registry (PIR) oversees the .org TLD, and IANA (for ICANN) internet infrastructure TLDs like .arpa. The root zone data contains information about the authoritative name servers for each TLD. When a DNS resolver initiates a query for a domain name, the root servers simply point the resolver in the direction of the appropriate TLD servers

It is the TLD server layer in the hierarchy where zone transfers start being relevant. A primary TLD server holds and distributes the authoritative zone file for the TLD, while secondary TLD servers regularly perform zone transfers to obtain the latest zone data from the promary. When a change is made to the zone file on the primary TLD server, a notification is sent to the secondary servers using a mechanism like SIGNOTIFY (part of DNSSEC). Secondary servers receive the notification and examine its content to understand the nature of the change, like details about the specific resource that was modified (e.g., a new domain added or an existing record updated). Based on the notification details and potentially a configured time interval, the secondary server initiates a zone transfer to retrieve the update from the primary server.

If changes to the TLD zone file occur frequently, updates might be more frequent (potentially every few minutes). Each TLD registry might have its own policies regarding update intervals, since there is also a trade-off between speed and efficiency: frequent updates can create more traffic, less frequent updates might introduce a propagation delay.

TLD servers actually experience lower traffic compared to root servers, making zone transfers more manageable. Techniques like zone change incrementalism (ZCI) can further optimize zone transfers by minimizing the amount of data transferred based on the specific changes made. Secondary servers play a crucial role in efficiently answering DNS queries, often handle a large portion of the overall traffic volume, reducing the load on the primary server and root servers.

Review: DNS behavior - a seldom-visited webpage in a new browser window:
You type the domain name into your web browser's address bar or click a hyperlink
Your PC's DNS cache is checked - Does the OS have the IP or domain name cached?
Home router: have the IP or domain name cached?
Recursive resolver at ISP takes responsibility for finding the answer to your query, including contacting other name servers.
- Iterative Resolution Process:
  - Checks its own cache to see if it has the info; if not, ask the appropriate TLD servers
    - If not found, it asks root nameservers for the right TLD servers.
      - Root nameservers point the recursive resolver to the proper TLD nameservers
    - The recursive resolver then queries the TLD servers it was referred to
    - TLD servers will know the authoritative nameservers for this domain
  - The recursive resolver contacts the authoritative nameservers for domain name
- An authoritative nameserver checks its zone file containing the domain's IP address.

Once the recursive resolver receives the IP address, it sends it back to your web browser.
The recursive resolver likely will cache this information for future queries.
Finished- website connection established.

Sample named.conf for Red Hat BIND

The original of this in that shipped in RHEL was over-commented. It was been trimed to save space)

```
*/
options
{
// Directory for writable files
        directory "/var/named";
        dump-file "data/cache_dump.db";
        statistics-file "data/named_stats.txt";
        memstatistics-file "data/named_mem_stats.txt";
        secroots-file "data/named.secroots";
        recursing-file "data/named.recursing";

// Listen on these interfaces
        listen-on port 53 { 127.0.0.1; };
        listen-on-v6 port 53 { ::1; };

// Restrict access
        allow-query { localhost; };
        allow-query-cache { localhost; };

// Recursion settings
        recursion yes;

// DNSSEC validation
        dnssec-validation yes;

// File paths for system specifics
        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
        managed-keys-directory "/var/named/dynamic";

// Use system-wide Crypto Policy
        include "/etc/crypto-policies/back-ends/bind.config";
};
logging
{
// Debugging log
        channel default_debug {
            file "data/named.run";
            severity dynamic;
        };
};
// Views for different client types
view "localhost_resolver"
{
// Localhost resolver (caching only)
        match-clients { localhost; };
        recursion yes;

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };

// Zones for localhost
        include "/etc/named.rfc1912.zones";

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };
};
view "internal"
{
// Zones for internal clients (localnets)
        match-clients { localnets; };
        recursion yes;

// Zones for localhost
        include "/etc/named.rfc1912.zones";

// Authoritative internal zones
        zone "my.internal.zone" {
            type primary;
            file "my.internal.zone.db";
        };
        zone "my.slave.internal.zone" {
            type secondary;
            file "slaves/my.slave.internal.zone.db";
            masters { 127.0.0.1; };
        };
        zone "my.ddns.internal.zone" {
            type primary;
            allow-update { key ddns_key; };
            file "dynamic/my.ddns.internal.zone.db";
        };
};
key ddns_key
{
        algorithm hmac-sha256;
        secret "use /usr/sbin/ddns-confgen to make TSIG keys";
};
view "external"
{
// Zones for external clients
        match-clients { any; };
        recursion no;

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };
// Authoritative external zones
        zone "my.external.zone" {
            type primary;
            file "my.external.zone.db";
        };
};
/* DNSSEC keys (trusted anchors) */
trust-anchors {
 // Root Key
  . initial-key 257 3 8
"AwEAAaz/tAm8yTn4Mfeh5eyI96WSVexTBAvkMgJzkKTOiW1vkI
bzxeF3   [truncated]     R1AkUTV74bU=";

 // Key for forward zone
  example.com. static-key 257 3 8
"AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/Rl9VQW
            [truncated]  NWUla4fWZbbaYQzA93mLdrng+M=";

 // Key for reverse zone.
  2.0.192.IN-ADDRPA.NET. initial-ds 31406 8 2 "F78CF3344F72
[truncated]  6D";
};
*/
```

Simple quick solutions for smaller DNS setups

IPv6 and Dual-Stack Support

```
options { listen-on { 192.168.0.1; }; listen-on-v6 { 2001:db8::1; }; allow-query { any; }; allow-query-v6 { any; };  };
        # listen-on are the IP addresses of the interfaces to get info from, allow-query opens availability
```

Blackhole Lists - Blocking Queries

```
options { blackhole { 192.0.2.0/24; 203.0.113.0/24; };  };
        # putting contents of a bracketed block on one line is frowned upon but I need the space to fill the page neatly!
```

Granular Access Control Lists (ACLs)

```
acl "trusted" {
   192.168.1.0/24;  // Local network
   10.0.0.0/16;     // Another trusted network
   localhost;       // Localhost
};
options {
   allow-query { trusted; };  // Restrict queries to trusted networks
   allow-recursion { trusted; };  // Restrict recursion to trusted networks
};
```

Query Rate Limiting to Mitigate DoS Attacks

```
options {
   rate-limit {
      responses-per-second 10;  // Limit to 10 responses per second per client
      window 5;  // Time window in seconds for rate limiting
      log-only yes;  // Log but don't drop excess responses (for monitoring)
   };
};
```

Basic High Availability and Replication (provision secondary DNS server)

On Primary DNS Server (named.conf):

```
zone "example.com" {
   type primary;
   file "zones/example.com.db";
   allow-transfer { secondary_dns_ips; };  // Allow transfers to secondary servers
   also-notify { secondary_dns_ips; };     // Notify secondary servers of zone updates
};
```

On Secondary DNS Server (named.conf):

```
zone "example.com" {
   type secondary;
   file "slaves/example.com.db";
   masters { primary_dns_ip; };  // IP of the primary DNS server
};
```

Secure Internal DNS Forwarding (internal.corp)

```
zone "internal.corp" {              # Configure forwarding for "internal.corp" domain
   type forward;                    # Zone type to forward- forward queries for this domain to specified servers.
   forwarders {                     # Internal DNS servers that will handle queries for "internal.corp" domain
      192.168.0.1;                  # Primary internal DNS server IP
      192.168.0.2;                  # Secondary
   };
                                    # Define client networks allowed to forward queries to internal DNS servers.
   allow-query { internal-net; };   # Only allow queries from the defined ACL (internal-net).
   forward only;                    # Use only the forwarders- do NOT try to resolve the queries if forwarders fail
};
acl "internal-net" {                # Has to be defined outside the zone { } block
   192.168.1.0/24;                  # Example of an internal subnet
   10.0.0.0/16;                     # Another internal subnet
};
```

Chroot Configuration

```
options {
   directory "/var/named/chroot";  // Chroot directory
   pid-file "/var/named/chroot/run/named.pid";  // Adjust paths for chroot environment
   session-keyfile "/var/named/chroot/run/session.key";
};
```

Geo-Location Based Routing

```
view "us_clients" {
   match-clients { 192.0.2.0/24; 198.51.100.0/24; };  // US clients
   zone "example.com" {
      type primary;
      file "zones/us.example.com.db";  // Zone file for US clients
   };
};
view "eu_clients" {
   match-clients { 203.0.113.0/24; 203.0.114.0/24; };  // EU clients
   zone "example.com" {
      type primary;
      file "zones/eu.example.com.db";  // Zone file for EU clients
   };
};
```

Dynamic DNS Updates Across Multiple Views (with TSIG Key)

```
zone "dynamic.example.com" {                          # DNS zone where dynamic updates are allowed
   type primary;                                      # Primary (authoritative) source for this zone
   file "zones/dynamic.example.com.db";               #  File where the zone data is stored
   update-policy {                                    #Configure the update policy for this zone.
      // Grant permissions to the TSIG key (ddns_key)
      grant ddns_key wildcard *.dynamic.example.com. A;   # Any A record (map hostnames to IP) in zone can be updated
   };
};
key ddns_key {                                        #  Define TSIG key used for securing dynamic DNS updates
   algorithm hmac-sha256;                             # Hashing algorithm used to create the MAC
   secret "base64-encoded-secret-key";                # Actual base64-encoded key for making HMAC signature
};
```

Advanced Logging and Monitoring

```
logging {                          # BIND's part: In named.conf,
   channel query_log {
      file "/var/log/named/queries.log" versions 10 size 100M;
      severity info;
      print-time yes;
   };
   channel security_log {
      file "/var/log/named/security.log" versions 10 size 50M;
      severity notice;
      print-time yes;
   };
   category queries { query_log; };
   category security { security_log; };
};

agentAddress  udp:161          # SNMP's part - in /etc/snmp/bind-snmp.conf)
rocommunity public
view all    included  .1 80
group MyROGroup v1    all
group MyROGroup v2c   all
group MyROGroup usm   all
access MyROGroup ""      any       noauth    exact  all    none   none
```

Minimizing Unnecessary Transfers

Allow-Transfer: This directive specifies which hosts or networks are allowed to initiate zone transfers. By default, it might be set to any, allowing anyone to request a transfer. Here's an example to restrict transfers to specific IP addresses:

```
zone "yourdomain.com" {
  allow-transfer { 192.168.1.10; 10.0.0.2; };  # Replace with authorized IP addresses
};
```

Also-Notify: This directive informs secondary servers when the zone file is updated. This can help automate transfer requests from authorized secondary servers, reducing unnecessary manual transfers.

```
zone "yourdomain.com" {
  also-notify { 192.168.1.20; };  # Replace with secondary server IP
};
```

Schedule Transfers During Off-Peak Hours
While named.conf doesn't directly schedule transfers initiated by secondary servers, you can achieve a similar effect on the primary server. The transfer-source directive specifies the IP address the server uses to initiate outgoing zone transfers. You can combine this with firewall rules to restrict outbound traffic during peak hours, indirectly influencing transfer timing.

```
zone "yourdomain.com" {
  transfer-source { 10.0.0.1; };  # Replace with appropriate IP for outbound transfers
};
```

A safer and easier method would be to leverage cron to schedule it
This example usies a cron job to reload the BIND rndc service to initiate a zone transfer during off-peak hours:

```
        # Edit the cron job file for the BIND user (usually named or bind)
sudo crontab -e -u bind
# Add the following line to schedule a transfer at 2:00 AM
0 2 * * * /usr/sbin/rndc reload example.com
```

DNSSEC Key Pairs, ZSK and KSK, Secure Zone Transfers
Key Creation: dnssec-keygen, tsig-keygen

```
        # Generate a Zone Signing Key (ZSK)
dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.com
        # Generate a Key Signing Key (KSK) helps validating DNSKEY records
dnssec-keygen -a RSASHA256 -b 4096 -n ZONE -f KSK example.com
        # This will create key files with .key and .private extensions. These files are used for signing the zone.
tsig-keygen -a hmac-sha256 tsig_key > /etc/bind/tsig_key.conf
        # This will create the TSIG file for actual zone transfers
```

Signing the zone file, updating named.conf

```
dnssec-signzone -o example.com -k Kexample.com.+008+12345 example.com.db
```

-o example.com (zone domain), -k Kexample.com.+008+12345 (the KSK) example.com.db (zone file to sign)

```
        # Put keys and signed zone file in named.conf
zone "example.com" {
   type master;
   file "zones/example.com.db.signed";         // Use the signed zone file
   allow-transfer { key /etc/bind/tsig_key.conf; };          // Secure transfers - reference key defined in tsig_key.conf
   notify yes;        // Enable notifications
   also-notify { 192.168.1.10; };        // Secondary server to notify
   inline-signing yes;          // Enable inline signing for automatic DNSSEC signing
   auto-dnssec maintain;   // Automatically maintain DNSSEC records
   max-transfer-time-in 60;// Limit the maximum transfer time to 60 seconds
   transfer-format many-answers;    // Optimize transfer format for efficiency
};
        # Setting up secondary server
Copy keys and signed zone(s) from the primary to secondary
scp /var/named/zones/example.com.db.signed secondary:/var/named/zones/example.com.db.signed
scp /var/named/keys/Kexample.com.+008+12345.key secondary:/var/named/keys/Kexample.com.+008+12345.key
scp /etc/named/tsig_key.conf secondary:/etc/named/tsig_key.conf
scp /var/named/keys/example.com.db.signed secondary:/var/named/keys/example.com.db.signed
```

Above I am copying all these manually to start off fresh without waiting (copy then restart named on secondary). ZSK and KSK are managed on the primary, the secondary server only gets public keys to verify signatures, are included in DNSKEY records sent during zone transfers. Auto-dnssec and inline-signing features simplifies this by managing keys and signatures automatically.

Add matching secondary server configuration (named.conf)

```
zone "example.com" {
   type slave;
   file "zones/example.com.db";
   masters { 192.168.0.1; };  // IP of the primary server
   allow-notify { 192.168.0.1; };  // Allow notifications from the primary server
   key "/etc/bind/tsig_key.conf";  // Referencing a defined TSIG key for secure communication
};
```

Above the secondary server also has been given the allow-notify line to get notifications from the primary. The lines for the primary were already added here, but you'll need these three lines in the primary for notifications:

```
   allow-transfer { key /etc/named/tsig_key.conf; };      // Secure transfers - reference key defined in tsig_key.conf
   notify yes;
   also-notify { 192.168.0.2; };          // Secondary server IP
```

Enable RNDC for Secure Remote Commands
RNDC (Remote Name Daemon Control) keys allow a secure secure control channel for commands to be issued to BIND (e.g., for restarting or reloading configurations).

Generating RNDC Keys
Use the rndc-confgen tool to generate an RNDC key. This command generates a key with 512 bits
rndc-confgen -a -b 512 -c /etc/rndc.key

Primary server congiuration
Configure rndc.conf and named.conf for RNDC:
rndc.conf:

```
key "rndc-key" {
    algorithm hmac-sha256;
    secret "base64-encoded-secret-key";  // Your RNDC secret key
};
options {
    default-key "rndc-key";
    default-server 127.0.0.1;  // RNDC server
    default-port 953;  // RNDC port
};
```

named.conf:

```
include "/etc/rndc.key";
controls {
    inet 127.0.0.1 port 953 {
        allow { 127.0.0.1, <secondary_serv_IP>, <workstation_IPs>; };   // Add IPs for machines that should have RNDC access
        keys { "rndc-key"; };
    };
};
```

Configuration on secondary server:
The rndc.conf file is generally identical so it can also be copied over. Changes to named.conf are the same as the primary.
Both the primary and secondary servers should have the same RNDC key if RNDC commands should control both servers.
Transfer RNDC key: scp /etc/rndc.key user@secondary-server:/etc/bind/rndc.key
Transfer RNDC config file: scp /etc/rndc.conf user@secondary-server:/etc/rndc.conf

Update named.conf to include the RNDC key:

```
include "/etc/rndc.key";
controls {
    inet 127.0.0.1 port 953 {
        allow { 127.0.0.1, <primary_serv_IP>, <workstation_IPs>; };
        keys { "rndc-key"; };
    };
};
```

Any machines that should be able to send RNDC commands to the servers will need BIND configured on them, have the key and have rndc conf files configured to do so. When running RNDC commands, specify the IP of the server:

```
rndc -s 192.168.1.10 -k /etc/rndc.key status
```

Centralized Management with LDAP

1. Adding LDAP in named.conf

```
dynamic-db "example" {
  library "ldap.so";
  arg "uri ldapi:///";  # No trailing slash here
  arg "base cn=dns,dc=example,dc=com";
  arg "auth_method sasl";
  arg "security sasl";  # Add this line for improved security
};
```

2. On the LDAP server, create a schema file, (e.g., /etc/dirsrv/slapd-instance_name/schema/99-dns.ldif) with this in it:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 2.16.840.1.113730.3.1.16 NAME 'dNSDomain' EQUALITY caseIgnoreIA5Match ORDERING
caseIgnoreIA5OrderingMatch SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.26' USAGE
userApplications X-ORIGIN 'LDAP schema for DNS' )
attributeTypes: ( 2.16.840.1.113730.3.1.17 NAME 'dNSTTL' EQUALITY integerMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.27'
USAGE userApplications X-ORIGIN 'LDAP schema for DNS' )
attributeTypes: ( 2.16.840.1.113730.3.1.18 NAME 'dNSRecord' EQUALITY caseIgnoreIA5Match ORDERING
caseIgnoreIA5OrderingMatch SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.26' USAGE
userApplications X-ORIGIN 'LDAP schema for DNS' )
objectClasses: ( 2.16.840.1.113730.3.2.3 NAME 'dNSZone' SUP top STRUCTURAL MUST dNSDomain MAY ( dNSTTL $
dNSRecord ) X-ORIGIN 'LDAP schema for DNS' )
```

3. Create files for zones you want LDAP to handle

This file is for 2 zones. Add another identical block under these to add more. Save this in multiple-dns-zones.ldif

```
# Zone for xyz321.com
dn: dc=xyz321,dc=com
objectClass: top
objectClass: dNSZone
dNSDomain: xyz321.com
dNSTTL: 86400
dNSRecord: SOA ns1.xyz321.com. hostmaster.xyz321.com. (
        2024062601 ; Serial
        3600       ; Refresh
        1800       ; Retry
        1209600    ; Expire
        86400 )    ; Minimum TTL
dNSRecord: NS ns1.xyz321.com.
dNSRecord: NS ns2.xyz321.com.

# www.xyz321.com record
dn: cn=www,dc=xyz321,dc=com
objectClass: top
objectClass: dNSZone
dNSDomain: www.xyz321.com
dNSTTL: 86400
dNSRecord: A 192.0.2.1

# Zone for abc123.com
dn: dc=123abc,dc=com  [etc., just like the blocks above- do that for each one you want to add]
```

4. When completed, run this line of code (replace "cn=admin,dc=example,dc=com" with your actual LDAP admin DN)

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -f multiple-dns-zones.ldif
```

You can verify the successful addition of these entries by using the ldapsearch command

```
ldapsearch -x -b "dc=xyz321,dc=com" -D "cn=admin,dc=example,dc=com" -W
```

Later you can update that same ldif file with more zones (or changes), run ldapadd again to update the database

This deletes the www record, not the zone

"ldapmodify -D "cn=admin,dc=example,dc=com" -W << EOF dn: cn=www,dc=xyz321,dc=com changetype: delete EOF"

I

f you delete a zone it will delete that zone's records too, so, to delete both you could just run the ldapmodify delete on "dn: dc=xyz321,dc=com" (the zone) and it will zap both on one shot!

Multi-Master DNS with Anycast:Load Balancing, High Availability, and Scalability
DNS Failover for fallback for continuous service

Multi-Master DNS ensures that DNS queries can be answered by any of the master servers, providing redundancy. If one server fails, others can continue to serve the zone data. Anycast allows multiple servers to share the same IP address, routing client requests to the nearest or best-performing server (for load balancing, distributing the query load evenly across servers. High Availability is enhanced because the system is resilient to individual server failures. Anycast addresses also improve performance by reducing latency, directing users to the closest server. DNS Failover involves monitoring the availability of services and rerouting traffic to backup servers when a primary server becomes unavailable.

Assume you have three servers:
DNS Server 1: 192.168.0.1 - DNS Server 2: 192.168.0.2 - Anycast IP: 203.0.113.10

On Server 1:

```
// named.conf on 192.168.0.1
options {
   directory "/var/named";                          // Default directory for zone files
   allow-transfer { 192.168.0.2; 203.0.113.10; };   // Allow zone transfers to other masters and Anycast
   also-notify { 192.168.0.2; 203.0.113.10; };      // Notify other masters and Anycast server of zone updates
   listen-on port 53 { 192.168.0.1; };              // Listen on the specific IP address for DNS queries
   listen-on-v6 { none; };                          // Disable IPv6 if not needed
};
logging {
   channel default_log {
      file "/var/log/named/default.log" versions 3 size 10M;
      severity info;
      print-time yes;
      print-severity yes;
   };
   category default { default_log; };
};
zone "example.com" {
   type primary;                                    // Master server type
   file "zones/db.example.com";                     // Location of the zone file
   allow-transfer { 192.168.0.2; 203.0.113.10; };   // Allow transfers to other masters and Anycast
   also-notify { 192.168.0.2; 203.0.113.10; };      // Notify other masters and Anycast server of changes
```

On DNS Server 2 (192.168.0.2):

```
// named.conf - all except what is below is identical to Server 1
options {
   allow-transfer { 192.168.0.1; 203.0.113.10; };
   also-notify { 192.168.0.1; 203.0.113.10; };
   listen-on port 53 { 192.168.0.2; };
};
zone "example.com" {
   allow-transfer { 192.168.0.1; 203.0.113.10; };
   also-notify { 192.168.0.1; 203.0.113.10; };
};
```

On Anycast DNS Server (203.0.113.10):

```
// named.conf - all except what is below is identical to Server 1
options {
   allow-transfer { 192.168.0.1; 192.168.0.2; };
   also-notify { 192.168.0.1; 192.168.0.2; };
   listen-on port 53 { 203.0.113.10; };
};
zone "example.com" {
   allow-transfer { 192.168.0.1; 192.168.0.2; };
   also-notify { 192.168.0.1; 192.168.0.2; };
};
```

Create and Configure the Zone Files
Ensure the zone file db.example.com is synchronized across all servers.

```
$TTL 86400 ; 1 day
@       IN SOA  ns1.example.com.  admin.example.com. (
                                  2024062901   ; serial
                                  3600         ; refresh (1 hour)
                                  1800         ; retry (30 minutes)
                                  1209600      ; expire (2 weeks)
                                  3600         ; minimum (1 hour)
                                  )
        NS      ns1.example.com.
        NS      ns2.example.com.
ns1     A       192.168.0.1
ns2     A       192.168.0.2
anycast A       203.0.113.10
www     A       192.168.0.3
```

Scalability and Performance Considerations
Allow-Transfer, Also-Notify: Expand these with additional IPs for new servers or Anycast instances as things scale.
Logging: Adjust log file sizes and the number of retained versions based on monitoring needs and storage capacity.
Listen-on, Listen-on-V6: Update these to include additional IPs or enable IPv6 (no IPv6 here just to save page space)

Testing and Validation
- DNS Query Testing:
  - Use dig to test DNS resolution from each server, including the Anycast IP
    If one isn't working you will likely get an NXDOMAIN instead of an IP address.
    dig @192.168.0.1 example.com +short; dig @192.168.0.2 example.com+short; dig @203.0.113.10 example.com
- Zone File Synchronization:
  - Update the zone file on one server and verify that the changes propagate to the others.
- Anycast Routing:
  - Confirm that queries to the Anycast IP are correctly distributed to the nearest or least-loaded server.

Response Policy Zones (RPZ): Policy-based managing and modifying DNS responses

Blocking, redirecting, or altering DNS queries to enforce security policies, control content, and manage network behavior dynamically.

RPZ Components
RPZ Zone Definition:
-- A DNS zone specifically created to hold RPZ data, generally named with a .rpz suffix. Can be primary or secondary

RPZ Policy Association:
-- The response-policy directive links a DNS zone to RPZ, specifying which zones' data will be used to apply policies.

RPZ Data File:
-- Contains SOA records and policy rules that dictate how to handle DNS queries based on matching criteria. They should not be placed with other zone files, but should be in the bind/named directories. On RHEL, I am using /etc/bind/rpz/

Basic RPZ Configuration
-- Define the RPZ zone (named.conf)

```
   # Create a DNS zone with an appropriate .rpz suffix
zone "example.rpz" {
   type master;                    # Define as master (primary)
   file "/etc/bind/db.example.rpz";   # Specify the file containing RPZ data
};
```

-- Associate the RPZ policy (named.conf)

```
   # Link the RPZ zone to BIND's response policy using the response-policy directive.
options {
   response-policy {
      zone "example.rpz";
   };
};
```

-- Populate the RPZ data file (example.rpz)

```
   # Define rules and actions in the RPZ data file, typically starting with SOA records followed by specific policy rules.

$TTL 60
@   IN SOA   ns1.example.com.  admin.example.com. (
                  2024062901          ; serial
                  3600                ; refresh
                  1800                ; retry
                  1209600             ; expire
                  60                  ; minimum TTL
                  )
      NS         ns1.example.com.

*.malicious-domain.com.   60   IN   CNAME   rpz-nxdomain.
*.old-service.com.        60   IN   CNAME   new-service.com.
```

Expanded RPZ Rules Table

response-policy	Specifies the policy zone(s) used for RPZ. Lists the zones and the order in which they apply.	Priority: Determines the order of zone application.
policy-zone	Defines a named policy zone that holds the RPZ rules.	Zone Source: File or feed from which the zone data is loaded.
match-clients	Matches client IP addresses or address ranges. Used to apply policies based on the source of the DNS query.	ACLs: Predefined Access Control Lists for matching clients.
match-destination	Matches based on the destination IP addresses in the DNS queries.	IP Range: Range or specific IP to match against.
match-subdomain	Matches queries based on subdomains. Useful for wildcard matching within a domain.	Wildcards: Supports wildcard entries for flexible matching.
match-type	Matches DNS queries based on their type (e.g., A, AAAA, MX). Allows policies to be applied to specific query types.	Query Type: List specific DNS query types (e.g., A, AAAA).
match-regex	Uses regular expressions to match DNS queries. Provides advanced matching criteria for complex policies.	Regex Pattern: Specify the regex pattern for query matching.
policy	Defines the action to be taken when a rule matches (e.g., NXDOMAIN, NODATA, CNAME). Actions dictate how matched queries are handled.	Actions List: Possible actions include NXDOMAIN, CNAME, etc.
qname-wait-recurse	Configures whether the RPZ should wait for recursion to complete before applying policies.	Boolean: True or False to enable/disable waiting.
qname-wait-time	Sets the maximum time to wait for a recursive query before applying the RPZ policies.	Time Value: Duration to wait for recursion (e.g., 2s).
break-dnssec	Specifies if DNSSEC validation should be disabled for RPZ responses.	Boolean: True (disable DNSSEC) or False (keep DNSSEC).
log	Enables or configures logging for RPZ matches, providing visibility into policy enforcement and aiding in troubleshooting.	Logging Level: Defines verbosity or specific logging actions.
rate-limit	Applies rate limiting to queries matching the RPZ rules, useful for mitigating abuse or attacks.	Rate Parameters: Max queries per second, burst sizes, etc.
feed-update	Specifies how often to update the RPZ zone from external feeds.	Update Frequency: Interval for feed updates (e.g., hourly).
source-policy	Defines policies based on the origin of the RPZ data, allowing differentiation of policies from different sources.	Source Specification: Identify sources and their policies.
inform-action	Configures additional actions (e.g., notify) when an inform action is taken, useful for alerting systems or administrators about policy matches.	Notification Methods: Email, SNMP traps, etc.

Expanded RPZ Actions Table

NXDOMAIN	Returns a "no such domain" response, effectively blocking the domain.	Default Action: Common for blocking malicious domains.
NODATA	Returns no data for the requested domain, blocking specific records while allowing others.	Use Case: Selective blocking of record types.
CNAME	Redirects the query to another domain by providing a canonical name.	Redirection Target: Specify the target domain for redirection.
PASSTHRU	Allows the query to bypass RPZ processing and be resolved normally.	Exception Handling: Useful for whitelisting specific queries.
DROP	Silently drops the query, providing no response, effectively blackholing it.	Application: Stops traffic to/from known bad actors.
TCP-Only	Forces the DNS query to be resolved over TCP instead of UDP.	Performance Impact: Adds overhead due to TCP's nature.
Redirect	Responds with a specific IP address, redirecting the traffic.	Redirection IP: Target IP for redirection.
Inform	Logs the query without modifying the response, used for monitoring and auditing DNS traffic.	Logging Level: Determines the verbosity of the logs.
Alter	Modifies the DNS response data, such as changing the TTL or other record attributes.	Modification Details: Specify the changes to be applied.
Override	Replaces the response with a predefined answer, useful for internal redirections or custom responses.	Override Content: The custom response data.
Fake-IP	Responds with a false IP address, typically used to redirect traffic to a controlled or null destination.	Fake IP Address: The IP to be returned.
Geo-Location	Uses geographical information to tailor the response, often used for content localization or restriction.	Geo-Parameters: Regions or countries to target.
Notify	Sends notifications or triggers actions when specific RPZ rules are matched.	Notification Methods: Email, webhook, etc.
Rewrite	Alters parts of the query or response, such as changing the domain name or resource record being queried.	Rewrite Rules: Specify how the query or response is rewritten.
Throttle	Limits the rate of responses to certain queries to mitigate abuse or DDoS attacks.	Throttle Parameters: Max queries per second, burst sizes, etc.

Advanced Use Cases for RPZ

Using External Blacklist Data
Community-maintained RPZ feeds can be used to block known malicious domains. DNSBLs can use thesame method
Many sites with RPZs want you to sign up for a trial of thier product- this one didn't - https://urlhaus.abuse.ch/downloads/rpz/
sudo wget -O /etc/bind/rpz/urlhaus.rpz https://urlhaus.abuse.ch/downloads/rpz/

```
        # Configure RPZ zone
zone "urlhaus.rpz" {
   type slave;
   file "/etc/bind/rpz/urlhaus.rpz";
   masters { 192.0.2.1 key "local_keyfile.key"; };       // Defines the master server and the key for secure zone transfers
};
        # Policy directive
options {
   recursion yes;
   response-policy { zone "urlhaus.rpz" policy nxdomain; };   // Matches get a NXDOMAIN response, effectively blocking them.
};
        # RPZ Data - /etc/bind/rpz/urlhaus.rpz
$TTL 30
@ SOA rpz.urlhaus.abuse.ch. hostmaster.urlhaus.abuse.ch. 2407010019 300 1800 604800 30
 NS localhost.
; abuse.ch URLhaus Response Policy Zones (RPZ)
; Last updated: 2024-07-01 00:19:23 (UTC)
; Terms Of Use: https://urlhaus.abuse.ch/api/
; For questions please contact urlhaus [at] abuse.ch
testentry.rpz.urlhaus.abuse.ch CNAME . ; Test entry for testing URLhaus RPZ
1.bdl99down.kukulaa.cn CNAME . ; Malware download (2024-05-30), see https://urlhaus.abuse.ch/host/1.bdl99down.kukulaa.cn/
139520.aioc.qbgxl.com CNAME . ; Malware download (2024-05-06), see https://urlhaus.abuse.ch/host/139520.aioc.qbgxl.com/
```

Internal DNS Redirection
Redirect queries for internal domain names to appropriate external addresses.

```
        # Configure RPZ zone
zone "internal.rpz" {
   type master;
   file "/etc/bind/rpz/db.internal.rpz";
};
        # Policy directive
options {
   response-policy {
      zone "internal.rpz";
   };
};
        # RPZ Data - /etc/bind/rpz/db.internal.rpz
```

Define redirection rules in the RPZ data file. These assume internal private DNS names inside company are *.corp

```
$TTL 60
@   SOA  ns1.example.com. admin.example.com. (
      2024062901; serial
      3600        ; refresh
      1800        ; retry
      604800      ; expire
      60 )        ; TTL
;
server1.corp  60  IN  CNAME  server1.example.com.
server2.corp  60  IN  CNAME  server2.example.com.
```

Rate Limiting DNS Queries
This configuration limits responses to 10 per second for queries matching the abuse domain.

```
        # Configure RPZ zone
zone "rtlimit-ddos.rpz" {
   type master;
   file "/etc/bind/rpz/rtlimit-ddos.rpz";
};
```

```
            # Policy directive
options {
   response-policy {
            zone "/etc/bind/rpz/rtlimit-ddos.rpz" policy passthru;
            rate-limit { responses-per-second 10; };
        };
            # RPZ Data -/etc/bind/rpz/rtlimit-ddos.rpz
$TTL 60
@    SOA  ns.example.com. admin.example.com. (
       2023062503; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
         ;
abuse.example.com  A  127.0.0.1
```

Geolocation-Based Content Control
Redirect queries based on the geographical location of the request.

```
            # Configure RPZ zone
zone "geo.rpz" {
   type master;
   file "/etc/bind/rpz/db.geo.rpz";
};
            # Policy directive
options {
   response-policy {
      zone "geo.rpz";
   };
};
            # RPZ Data - /etc/bind/rpz/db.geo.rpz
```
Route traffic to different servers based on region-specific subdomains.
```
$TTL 60
@   SOA  ns1.example.com. admin.example.com. (
       2024062901; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
;
us.region.example.com  60  IN  CNAME  us-content.example.com.
eu.region.example.com  60  IN  CNAME  eu-content.example.com.
```

Redirecting Traffic
Redirect traffic from a deprecated service to a new domain.

```
zone "redirects.rpz" {
   type master;
   file "/etc/bind/db.redirects.rpz";
};
            # Policy directive
response-policy {
   zone "redirects.rpz";          // internal policy rules
};
            # RPZ Data /etc/bind/rpz/redirects.rpz
$TTL 60
@    IN SOA ns1.example.com. hostmaster.example.com. (
       2024062901; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
   )
   IN NS ns1.example.com.

*.old-service.com. 60 IN CNAME new-service.com.
```

DNS Rcodes (Return Codes)

NOERROR	No Error. Query successful; valid response.
FORMERR	Format Error (unable to interpret query format.)
SERVFAIL	Server error processing request;
NXDOMAIN	FQDN doesn’t exist; "NXDOMAIN"
NOTIMPL	Server doesn't support query type
REFUSED	Server refused query; likely a policy, zone transfer
NOTAUTH	Server not authoritative for the zone.
NOTZONE	Name not found in the zone.
PREREQ	Failed prerequisites: YXDomain, YXRRSet, NXRRSet.

Domain Statuses (reported in whois queries)

Depending on the Top Level Domain (TLD), there can be different status names used by the underlying registry.

.COM and .NET domain names

ACTIVE	Domain registered; functioning for websites or email.
REGISTRAR-HOLD	On hold by registrar; contact registrar.
REGISTRY-HOLD	On hold by registry; contact registrar.
REGISTRAR-LOCK	Registrar locked domain; settings can't change.
REGISTRY-LOCK	Registry locked domain; settings can't change.
REDEMPTIONPERIOD	Domain expired; 30-day hold before release.
PENDINGRESTORE	Expired domain being restored to ACTIVE.
PENDINGDELETE	Domain expired 75 days ago; deleting soon.

.ORG, .BIZ, and .INFO domain names

OK	Domain active; usable for websites, email, or name servers.
CLIENT_DELETE_PROHIBITED	Registrar locked domain; cannot delete.
SERVER_DELETE_PROHIBITED	Registry locked domain; cannot delete.
CLIENT_HOLD	Registrar on hold; domain unusable.
SERVER_HOLD	Registry on hold; domain unusable.
CLIENT_RENEW_PROHIBITED	Registrar locked domain; cannot renew.
SERVER_RENEW_PROHIBITED	Registry locked domain; cannot renew.
CLIENT_TRANSFER_PROHIBITED	Registrar locked domain; cannot transfer.
SERVER_TRANSFER_PROHIBITED	Registry locked domain; cannot transfer.
CLIENT_UPDATE_PROHIBITED	Registrar locked domain; settings can't change.
SERVER_UPDATE_PROHIBITED	Registry locked domain; settings can't change.
INACTIVE	Domain unusable; name server issues.
PENDING_DELETE	Domain registration about to delete.
PENDING_TRANSFER	Domain transferring; no modifications allowed.
PENDING_VERIFICATION	Registry creating domain record.

DNS message format

Message ID									
QR	OPCODE	AA	TC	RD	RA	RE	AD	CD	RCODE
QDCOUNT									
ANCOUNT									
NSCOUNT									
ARCOUNT									

QR	Query () or response ()	Is the message is a query (0) or a response (1)?
OPCODE	Type	Type of message being sent: Query (0), Inverse Query (1), Status Request (2)
AA	Authoritative answer	Indicates the name server is authoritative for queried domain
TC	Truncation	Response was truncated due to exceeding the size limit (UDP)
RD	Recursion desired	The client requests recursive resolution
RA	Recursion available	The DNS server supports recursive queries
RE	Reserved for future use	
AD	Authenticated data (DNSSEC)	Answer and authority sections were authenticated
CD	Checking disabled (DNSSEC)	Client requests the DNS server to disable DNSSEC validation
RCODE	Response type	[NoError (0), FormatError (1), ServerFailure (2), NameError (3), etc.]
QDCOUNT	Question Count	Number of client queries to server in the Question section
ANCOUNT	Answer Count	Number of resource records in the Answer (actual answers)
NSCOUNT	Authority Count	Number of resource records (authoritative name servers) in Authority section
ARCOUNT	Additional Count	Number of resource records in the Additional section

Utilities in the bind-utils package: dig, nslookup, host

The **host** command for DNS lookup (using OS resolver libraries for queries.
-a: Displays all records for a domain (same as -v). -l: Lists all hosts in the specified zone (zone transfer).
-C: Checks DNS configuration and zone files. -t type: DNS record to query (e.g., A, MX, NS, etc.).
-W <seconds> (timeout), -R retries, -T: Forces TCP instead of UDP, -4: IPv4 only, -6: IPv6 onl, -v: verbose

The **nslookup** command for DNS queries directly to DNS servers
-type or -query: DNS record to query (e.g., A, MX, TXT). -retry: number of retries.
-server: DNS server to query -timeout <seconds> (query timeout)
-vc: Use TCP for queries (a virtual circuit) -debug: Debugging mode for detailed output.

The **whois** command for info about domain registrations and IP address allocations (not part of bind-utils)
-h: Connects to a specified WHOIS server instead of the default. -l: Recursive domain registration lookup
-p: Connects to a specified port on the WHOIS server. -r: Does a raw WHOIS query (no special handling of data).
-I: Searches for IP networks. -H: Disables the display of the legal disclaimers in the output.

The **dig** command to query domain records, name servers, and troubleshooting DNS

dig example.com ANY	Fetches all available DNS records (A, MX, NS, TXT, etc.)
dig example.com +trace	Show full path of servers in resolving domain from root servers to the authoritative DNS servers
dig example.com +dnssec	Get DNSSEC-related information for example.com (DNSKEY, RRSIG, etc.)
dig -x 192.0.2.1	Reverse DNS lookup of 192.0.2.1, getting associated domain name
dig @8.8.8.8 example.com +noall +answer +stats	Query 8.8.8.8 for example.com, show only the answer section, stats

Dig Query Options

+[no]additional	Displays or omits the additional section data in responses. Default is to display it.
+aaflag	Sets the AA (Authoritative Answer) flag in the query.
+adflag	Sets the AD (Authentic Data) bit in the query to indicate DNSSEC validation status.
+[no]all	Sets or clears all display flags. Default is not to clear any flags.
+[no]authority	Displays or omits the authority section of a reply. Default is to display it.
+[no]cmd	Toggles the printing of the initial comment in the output with version and query options info.
+[no]comments	Toggles the display of comments in the output, (packet headers and OPT pseudosection info).
+[no]crypto	Toggles the display of cryptographic fields in DNSSEC records.
+[no]dns64prefix	Looks up IPV4ONLY.ARPA AAAA and prints any DNS64 prefixes found.
+dnssec	Requests DNSSEC records and sets the DO (DNSSEC OK) bit in the query.
+domain=NAME	Sets the search list to contain a single domain NAME.
+edns[=#]	Specifies the EDNS version to query with. Default is 0.
+[no]ednsflags[=#]	Sets or clears the EDNS flags (Z bits) in the query.
+[no]ednsnegotiation	Enables or disables EDNS version negotiation. Default is enabled.
+ednsopt[=code[ ]]	Specifies an EDNS option with code and optional value in hexadecimal.
+[no]expire	Sends/ does not send an EDNS Expire option.
+[no]fail	Indicates to retry or not retry with another server if a SERVFAIL is received. Default is not to retry.
+[no]identify	Shows or hides IP addr and port of the responding server when +short is enabled. Default is to hide.
+[no]idnout	Converts/ does not convert punycode on output. Default is to process when output is a tty.
+[no]multiline	Prints records in a verbose, multi-line format or not. Default is not to print in multiline.
+ndots=D	Number of dots that must be in name before is considered absolute. Default is in /etc/resolv.conf or 1.
+[no]nssearch	Finds/ does not find authoritative name servers for the zone and displays their SOA records.
+opcode=value	Sets the DNS message opcode to the specified value. Default is QUERY (0).
+padding=value	Pads the size of the query packet to blocks of value bytes. Default is 0 (no padding).
+qid=value	Specifies the query ID to use for sending queries.
+recurse	Enables recursion in the query. This bit is set by default.
+short	Provides terse answer format or not. Default is verbose.
+[no]tcp	Uses/ does not use TCP for querying name servers. Default is to use UDP unless required otherwise.
+[no]trace	Traces/ does not trace the delegation path from the root name servers. Default is not to trace.
+time=T	Sets the query timeout to T seconds. Default is 5 seconds.
+[no]tls	Uses/ does not use DNS over TLS (DoT) for querying. Default port is 853.
+tls-ca[=file-name]	Validates server TLS certificates using the specified CA file. Uses default CA store if not specified.
+tls-hostname=hostname	Uses the specified hostname for TLS certificate verification.
+[no]tries=T	Sets the number of times to try UDP and TCP queries. Default is 3.
+[no]split=W	Splits long hex- or base64-formatted fields into chunks of W characters. Default is 56 or 44 in multiline.
+[no]stats	Toggles the printing of statistics. Default is to print statistics.
+[no]subnet=addr/prefix	Uses the EDNS Client Subnet option with the specified address and prefix.
+subnet=addr[@src...	+subnet=addr[@src-prefix]/prefix Specifies EDNS Client Subnet addr, source and destination prefix.
+tls-only	Forces the use of DNS over TLS (DoT) only for the query.