



Linux Network Services: Domain Name Server (DNS)

*This material is based on work supported by the
National Science Foundation under Grant No. 0802551*



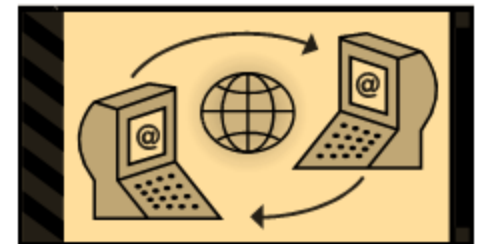
*Any opinions, findings, and conclusions or recommendations expressed in this material are those of
the author (s) and do not necessarily reflect the views of the National Science Foundation*

Lesson Overview

Billions of computers across the world are able to communicate with each other using Internet protocol (IP) addresses that function as unique identifiers for each computer. Unfortunately, it is not easy to remember a series of numbers that form an IP address. It is much easier to remember a catchy name such as www.google.com rather than one of its numerical addresses of 74.125.65.104.

To facilitate the conversion of domain names like *google.com* to their numeric equivalents, domain name servers (DNS) were invented. DNS servers are similar to large phone directories or databases that translate an IP address into a domain name that computers prefer to use.

In this lesson, you will explore DNS service and learn to install and configure DNS on a computer. Learning this critical task is important for Linux administrators responsible for networking resources.



Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

Lesson Objective:

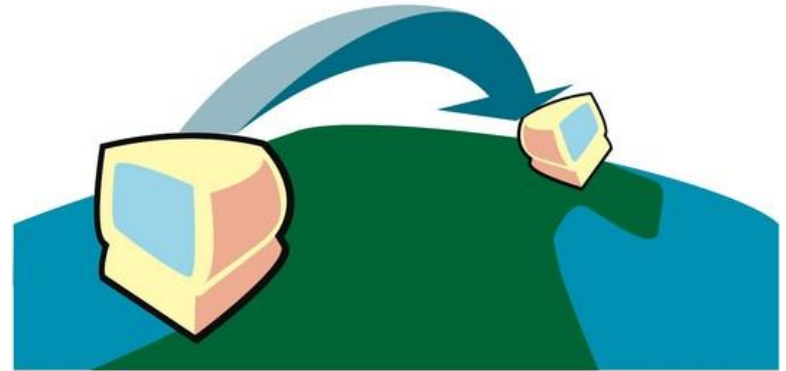
Given a need for a domain name system (DNS) server, a student will be able to assess the implications of hosting a DNS server and follow recommended procedures to install a DNS server as per industry standards.



Lesson Outline

In this lesson, you will explore:

- ❖ Need for DNS
- ❖ Function of DNS
- ❖ Installation of DNS



Resources and Notes

This lesson uses Fedora Linux for demonstration. To complete this lesson successfully, you must have access to:

- ❖ Fedora Linux on bare metal or as a virtual install
- ❖ 10 Gb of hard drive space dedicated to the operating system's use
- ❖ A command shell
- ❖ Internet for research
- ❖ Word processor

Use the resources on the right to configure your system for Fedora.

Resources:

- [Download Virtualbox](#)
- [Virtualbox for Linux Hosts](#)
- [Install Fedora to Virtualbox](#)
- [Virtualbox manual](#)
- [Using Virtualbox with Ubuntu](#)
(process similar to Fedora)

Introduction

Many of you may not remember life before Google, the Internet, and electronic phone books, but some of you may. Imagine trying to contact a neighbor via telephone without knowing his or her phone number. You would need to start dialing at 000-000-0000 and work your way up to 999-999-9999, hopefully entering the correct number along the way. It should be noted that 10^9 translates to over a billion telephone numbers, and a human being can typically dial a phone number in about 30 seconds. Consequently, it would take you over 833,333 hours to dial the various phone numbers, and you would annoy a lot of people.

Imagine dialing your cell phone with no idea who you are going to reach. You are trying to reach your best friend, but instead you reach someone in Japan who does not even speak English. Imagine someone trying to reach you without knowing your phone number and dialing randomly in hopes he will enter the correct number!

If there were no phone books, directory assistance, database, or Internet resource to connect two or more computers, how would you go about reaching anyone?

Required Reading

- [DNS Overview](#)

Introduction Contd

The IP version 4 protocol addresses that computers use to communicate with each other ranges from 1.0.0.1 to 254.254.254.254, or 254^4 or approximately 4,162,314,256 possibilities. If you did not know the correct IP address for the computer you wish to access, it would be very difficult to find the information you need in a reliable fashion.

Domain Name Servers (DNS) were developed to address this very need. Think of DNS as a great big phonebook directory that associates names with their correct IP addresses so that data can be routed correctly from one computer or device to the next.

Recommended Reading

- [DNS Overview](#)

Domain Name System

The practice of using a human friendly name as a to represent a host's numerical address on the network dates back to the Arpanet era. Before the DNS was invented in 1983, each computer on the network retrieved a file called *HOSTS.TXT* from a computer at SRI (now SRI International). The *HOSTS.TXT* file mapped names to numerical addresses.

A hosts file still exists on most modern operating systems by default and generally contains a mapping of the IP address 127.0.0.1 to "localhost". You can find this file on your own machine in the */etc* directory.

Many operating systems use name resolution logic that allows the administrator to configure selection priorities for available name resolution methods. But, what about machines located on a large corporate or educational network as well as on the public Internet?

Recommended Reading

- [How DNS Works](#)

Purpose of DNS

My machine lives behind a firewall. It has an IP address of 192.168.2.3 which is an internal IP address. The firewall maintains an external IP address that is tied to (or resolved to) a domain name, for example *psc.edu*.

The IP address tied to *psc.edu* is one of the millions that are directly connected to the Internet. Every domain name must resolve to or be identified by an IP address which is the equivalent of the phone number for the domain.

So, now we have a “phone number” or IP address, and we have a machine or domain name. With the large numbers of IP addresses and their associated domain names, there has to be a way for a machine to make the translation and look each other up. This is the role of the Domain Name Service (DNS) server.

But, with the ability for anyone to install a name server, as you are going to do in this lesson, who is actually in charge?

Recommended Reading

- [Importance of DNS](#)
- [How DNS Works](#)

Authoritative Name Server

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain.

Authoritative name servers are assigned to be responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. This mechanism has made the DNS distributed and fault tolerant and has helped avoid the need for a single central register to be continually consulted and updated.

Another role of the DNS server is to help mail flow from one location to another. Many times the IP address that receives the web traffic for a domain is not the one that receives the mail traffic.

For example, many companies contract with Google to handle their email and contract with someone else to handle their web servers. The DNS server contains the various IP addresses for each of these servers to make sure that the correct traffic is routed to the correct IP address.

Required Reading

- [Requirements for Authoritative Name Servers](#)

Domain Name Assignment

The right to use a domain name is delegated by domain name registrars that are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN)— the organization charged with overseeing the name and number systems of the Internet.

In addition to ICANN, each top-level domain (TLD) is maintained and serviced technically by an administrative organization, operating a registry.

A registry, such as *GoDaddy.com* or *Register.com*, is responsible for maintaining the database of names registered within the TLD it administers. The registry receives registration information from each domain name registrar authorized to assign names in the corresponding TLD and publishes the information using a special service called the *WHOIS* protocol.

Recommended Reading

- [DNS Considerations](#)

Assigning Domain Names

ICANN publishes the complete list of TLD registries and domain name registrars. Registrant information associated with domain names is maintained in an online database accessible with the WHOIS service.

For most of the more than 240 country code top-level domains (ccTLDs), the domain registries maintain the WHOIS (Registrant, name servers, expiration dates, etc.) information. For instance, DENIC, Germany NIC, holds the DE domain data.

Since 2001, most gTLD registries have adopted this so-called *thick* registry approach, i.e. keeping the WHOIS data in central registries instead of registrar databases.

For COM and NET domain names, a *thin* registry model is used: the domain registry (e.g. VeriSign) holds basic WHOIS (registrar and name servers, etc.) data. One can find the detailed WHOIS (registrant, name servers, expiry dates, etc.) at the registrars.

Recommended Reading

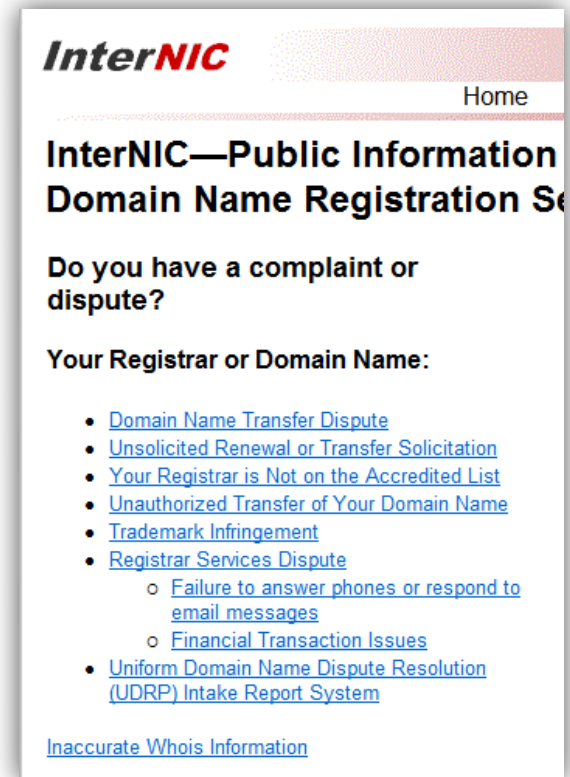
- [Top Level Domains](#)

Consumer Domains

Some domain name registries, often called *network information centers* (NIC), also function as registrars to end-users.

The major generic top-level domain registries, such as for the COM, NET,ORG, INFO domains, use a registry-registrar model consisting of many domain name registrars. In this method of management, the registry only manages the domain name database and the relationship with the registrars.

The *registrants* (users of a domain name) are customers of the registrar, in some cases through additional layers of resellers.



The screenshot shows the InterNIC website interface. At the top left is the InterNIC logo. To the right is a 'Home' link. Below the navigation bar is the main heading 'InterNIC—Public Information Domain Name Registration Services'. A sub-heading asks 'Do you have a complaint or dispute?'. Underneath, it says 'Your Registrar or Domain Name:' followed by a bulleted list of dispute categories. The categories are: Domain Name Transfer Dispute, Unsolicited Renewal or Transfer Solicitation, Your Registrar is Not on the Accredited List, Unauthorized Transfer of Your Domain Name, Trademark Infringement, Registrar Services Dispute (with sub-items: Failure to answer phones or respond to email messages, Financial Transaction Issues), and Uniform Domain Name Dispute Resolution (UDRP) Intake Report System. At the bottom of the list is a link for 'Inaccurate Whois Information'.

InterNIC Home

InterNIC—Public Information Domain Name Registration Services

Do you have a complaint or dispute?

Your Registrar or Domain Name:

- [Domain Name Transfer Dispute](#)
- [Unsolicited Renewal or Transfer Solicitation](#)
- [Your Registrar is Not on the Accredited List](#)
- [Unauthorized Transfer of Your Domain Name](#)
- [Trademark Infringement](#)
- [Registrar Services Dispute](#)
 - [Failure to answer phones or respond to email messages](#)
 - [Financial Transaction Issues](#)
- [Uniform Domain Name Dispute Resolution \(UDRP\) Intake Report System](#)

[Inaccurate Whois Information](#)

Taken from [Internic.net](http://internic.net)

Misconfigured Domains

Problems can occur because of misconfigured domain name servers. For example, if the DNS server has the wrong IP address in it, it would be possible to route secure or confidential information to the wrong system, otherwise known as “spoofing”. This has happened on a few occasions to major corporations and resulted in millions of dollars of fines and legal expenses.

Another possible outcome is that too much traffic may be routed to a server and cause the server to fail because of an overload. The moral of the story is that if you plan on configuring a public DNS server, make sure you follow the directions and do it correctly each time.

Additionally, it is a good idea to always test your system before you make it live.



Word of warning as you complete the examples that are a part of this lesson:

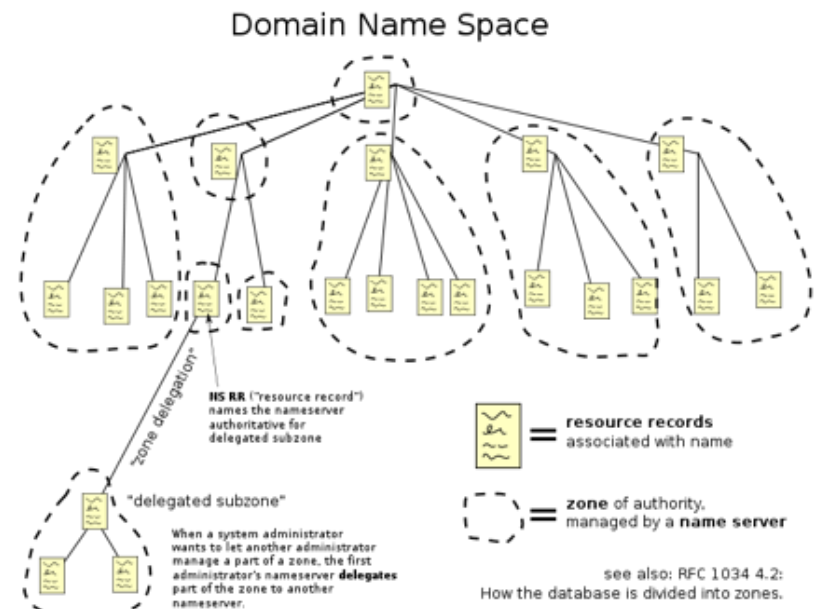
If you are using a virtual machine and working in a bridged networking mode, your machine is available to other machines on your network. Make sure you disable your DNS server once you have completed these exercises.

If you are working on a stand-alone Linux installation, again your DNS server will be available to the rest of the network. Make sure you disable it when you have complete these exercises. You have been warned!!!

DNS Zones

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more *resource records*, which hold information associated with the domain name. The tree sub-divides into *zones* beginning at the root zone. A DNS zone may consist of only one domain or may comprise many domains and sub-domains depending on the administrative authority delegated to the manager.

Administrative responsibility over any zone may be divided by creating additional zones. Authority is said to be *delegated* for a portion of the old space, usually in form of sub-domains, to another nameserver and administrative entity. The old zone ceases to be authoritative for the new zone.



Recommended Reading

- [DNS Zone Files](#)

[Hierarchical domain](#) name system (Credit: Wikipedia)

DNS Levels of Authority: Example

For example:

lms.lincs.pscit.org consists of a number of different levels of authority. Start with the right of the address and work your way to the left.

.org = ICANN

.pscit = Polk State College

.lincs = A division within Polk State College Information Technology

lms = The machine name found within the Polk State College Information Technology authority

In order to make changes to the domain record for any machine or level of authority, the machine or authority in charge must authorize that change. This process helps prevent malicious activity such as redirecting an entire bank's transactions to a server not owned or controlled by the bank.

Recommended Reading

- [DNS Guide](#)

Structure of Domain Names

The definitive descriptions of the rules for forming domain names appear in [RFC1035](#), [RFC1123](#), and [RFC2181](#). A domain name consists of one or more parts, technically called *labels*, that are conventionally concatenated, and delimited by dots, such as *example.com*.

- ❖ The right-most label conveys the top-level domain (TLD); for example, the domain name *www.example.com* belongs to the top-level domain of *com*.
- ❖ The hierarchy of domains descends from right to left; each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label *example* specifies a subdomain of the *com* domain, and *www* is a subdomain of *example.com*. This tree of subdivisions may have up to 127 levels.
- ❖ Each label may contain up to 63 characters. The full domain name may not exceed a total length of 253 characters in its external dotted-label specification. In the internal binary representation of the DNS, the maximum length requires 255 octets of storage. In practice, some domain registries may have shorter limits.
- ❖ DNS names may technically consist of any character representable in an octet. However, the allowed formulation of domain names in the DNS root zone and most other sub domains, uses a preferred format and character set. The characters allowed in a label are a subset of the ASCII character set, and includes the characters *a* through *z*, *A* through *Z*, digits *0* through *9*, and the hyphen. This rule is known as the *LDH rule* (letters, digits, hyphen). Domain names are interpreted in case-independent manner. Labels may not start or end with a hyphen.
- ❖ A hostname is a domain name that has at least one IP address associated. For example, the domain names *www.example.com* and *example.com* are also hostnames, whereas the *com* domain is not.

Distribution of DNS Records

The Domain Name System is maintained by a distributed database system, which uses the client-server model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root nameservers—the servers to query when looking up (*resolving*) a top level domain (TLD), such as .com or .org.

An *authoritative* name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server returns answers to queries about domain names that have been specifically configured by the administrator.

An authoritative name server can either be a *master* server or a *slave* server. A master server is a server that stores the original (*master*) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records.

As you configure our example name server in this lesson, you are going to setup a *fake.com* domain in which your nameserver will be the authoritative nameserver for *fake.com*.

Suggested Reading

- [Common DNS Errors](#)

Caching Name Servers for Efficiency

In principle, authoritative name servers are sufficient for the operation of the Internet. However, with only authoritative name servers operating, every DNS query must start with recursive queries at the root zone of the Domain Name System and each user's system must implement resolver software capable of recursive operation.

To improve efficiency, reduce DNS traffic across the Internet, and increase performance in end-user applications, the Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question. Typically, such *caching* DNS servers, also called *DNS caches*, also implement the recursive algorithm necessary to resolve a given name starting with the DNS root through to the authoritative name servers of the queried domain. With this function implemented in the name server, user applications gain efficiency in design and operation.

The combination of DNS caching and recursive functions in a name server is not mandatory; the functions can be implemented independently in servers for special purposes. Internet service providers (ISPs) typically provide recursive and caching name servers for their customers. In addition, many home networking routers implement DNS caches and recursors to improve efficiency in the local network.

DNS from the Client Side

The client-side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.

A DNS query may be either a non-recursive query or a recursive query:

- ❖ A *non-recursive query* is one in which the DNS server provides a record for a domain for which it is authoritative itself, or it provides a partial result without querying other servers.
- ❖ A *recursive query* is one for which the DNS server will fully answer the query (or give an error) by querying other nameservers as needed. DNS servers are not required to support recursive queries.

The resolver, or another DNS server acting recursively on behalf of the resolver, negotiates use of recursive service using bits in the query headers.

Resolving usually entails iterating through several name servers to find the needed information. However, some resolvers function simplistically and can communicate only with a single name server. These simple resolvers (called *stub resolvers*) rely on a recursive name server to perform the work of finding information for them.

Resolving Domain Names

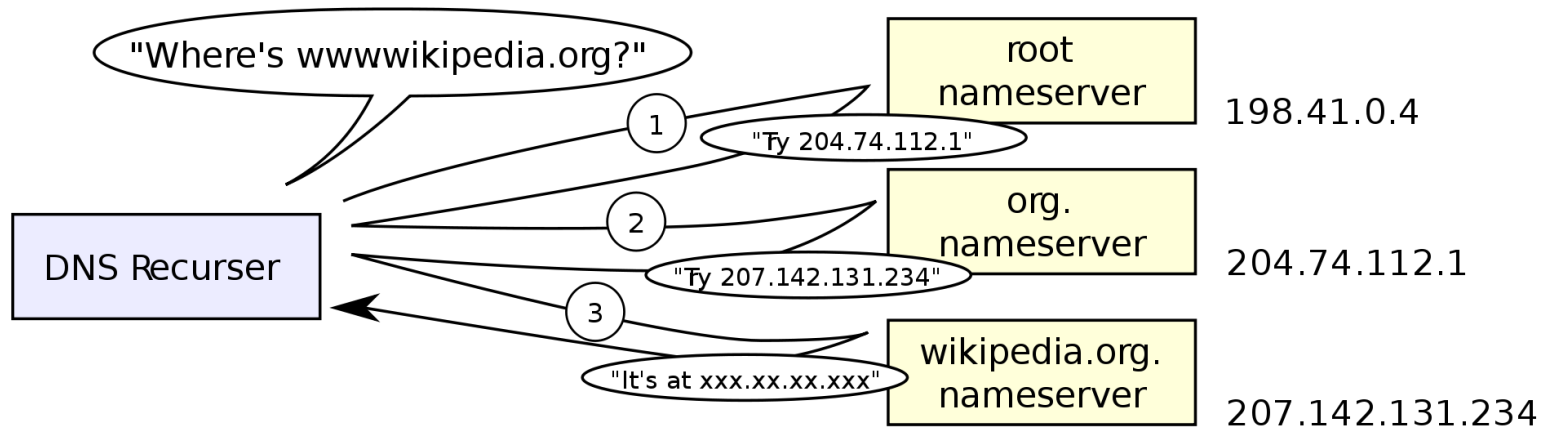
Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label. A DNS recursor consults three nameservers to resolve the address `www.wikipedia.org`.

The process entails:

- ❖ A network host is configured with an initial cache (so called *hints*) of the known addresses of the root nameservers. Such a *hint file* is updated periodically by an administrator from a reliable source.
- ❖ A query to one of the root servers to find the server authoritative for the top-level domain.
- ❖ A query to the obtained TLD server for the address of a DNS server authoritative for the second-level domain.
- ❖ Repetition of the previous step to process each domain name label in sequence until the final step which returns the IP address of the host sought.
- ❖ The mechanism in this simple form would place a large operating burden on the root servers, with every search for an address starting by querying one of them. Such heavy use of root servers would create an insurmountable bottleneck for trillions of queries placed every day. In practice, caching is used in DNS servers to overcome this problem, and as a result, root nameservers actually are involved with very little of the total traffic.

View the next screen for a visual representation of the resolving process.

Resolving Domain Names (Contd)



DNS Recurser communicates with three nameservers before resolving the address for `www.wikipedia.com` (Credit: [Wikipedia](#))

Reverse DNS Lookup

There are times when a client (or server) may want to verify that traffic is coming from a machine associated with the domain name and not being “spoofed” or copied by another source. This is important for email traffic as well as secure communications.

A reverse lookup is a query of the DNS for domain names when the IP address is known. Multiple domain names may be associated with an IP address. The DNS stores IP addresses in the form of domain names as specially formatted names in pointer (PTR) records within the infrastructure top-level domain [ARPA](#). For IPv4, the domain is in-addr.arpa. For IPv6, the reverse lookup domain is ip6.arpa. The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.

When performing a reverse lookup, the DNS client converts the address into these formats, and then queries the name for a PTR record following the delegation chain as for any DNS query. For example, the IPv4 address 208.80.152.2 is represented as a DNS name in the form *2.152.80.208.in-addr.arpa*. The DNS resolver begins by querying the root servers, which point to ARIN's servers for the 208.in-addr.arpa zone. From there the Wikimedia servers are assigned for 152.80.208.in-addr.arpa, and the PTR lookup completes by querying the wikimedia nameserver for 2.152.80.208.in-addr.arpa, which results in an authoritative response.

If the server does not return an authoritative response that matches that domain, we would suspect a problem had occurred and the traffic was not valid.

Contents of a DNS Record

A Resource Record (RR) is the basic data element in the domain name system. Each record has a type (A, MX, etc.), an expiration time limit, a class, and some type-specific data. Resource records of the same type define a resource record set. The order of resource records in a set (returned by a resolver to an application) is undefined, but servers often implement round-robin balancing to achieve load balancing. DNSSEC, however, works on complete resource record sets in a canonical order.

When sent over an IP network, all records use the common format specified in RFC1035.

Field	Description
NAME	Name of the node to which this record pertains
TYPE	Type of RR in numeric form (e.g. 15 for MX RRs)
CLASS	Class code
TTL	Count of seconds that the RR stays valid (The maximum is 231-1, which is about 68 years.)
RDLENGTH	Length of RDATA field
RDATA	Additional RR-specific data

Review the explanation of each field on next page

Contents of a DNS Record (Contd)

NAME is the fully qualified domain name of the node in the tree. The name may be shortened using label compression where the ending letters of domain names may be substituted for the ending letters of the current domain name.

TYPE is the record type. It indicates the format of the data and it gives a hint of its intended use. For example, the *A* record is used to translate from a domain name to an IPv4 Address, the *NS* record lists which nameservers can answer lookups on a DNS zone, and the *MX* record specifies the mail server used to handle mail for a domain specified in an email address.

RDATA is data of type-specific relevance, such as the IP address for address records or the priority and hostname for *MX* records. Well-known record types may use label compression in the *RDATA* field, but "unknown" record types must not (RFC3957).

The *CLASS* of a record is set to *IN* (for *Internet*) for common DNS records involving Internet hostnames, servers, or IP addresses.

Suggested Reading

- [DNS Zone Records](#)

Installing the DNS Server



DNS Server: Installation

Having discussed the basis of Domain Name Services, we will now setup a domain server on your test machine. For these exercises, you will use the Red Hat-based Fedora distribution.

Step 1: Install and Update Server

1. Install Fedora per prior instructions.
2. Log in and open the terminal or console.
3. Type **su** to log in as root user and enter your root password when prompted. (It is not good practice to work as root, however, root access will save you time and typing in this case. Alternatively, you may enter the following commands using **sudo**.)
4. Type: **yum upgrade**
5. Accept any software upgrades that may be waiting to be installed.

Review Picture A on the next screen for additional reference

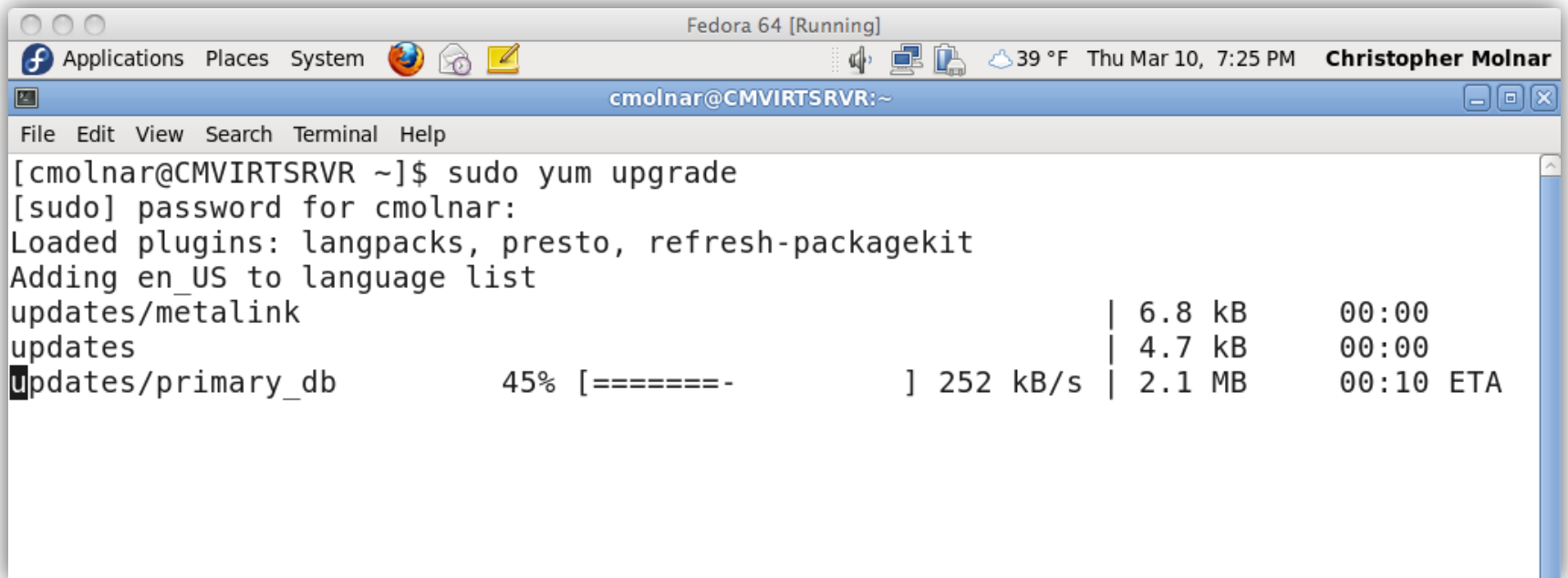
Required Reading

- [Installing BIND](#)

Suggested Review

- [DNS for Scientists](#)

Yum Upgrade



The screenshot shows a terminal window titled "Fedora 64 [Running]" with the user "Christopher Molnar". The terminal prompt is "cmolnar@CMVIRTSRVR:~". The user has entered the command "sudo yum upgrade". The terminal output shows the command being executed, the password prompt, and the progress of the upgrade. The progress bar shows that the "updates/primary_db" is 45% complete, with a speed of 252 kB/s and an estimated time to complete of 00:10.

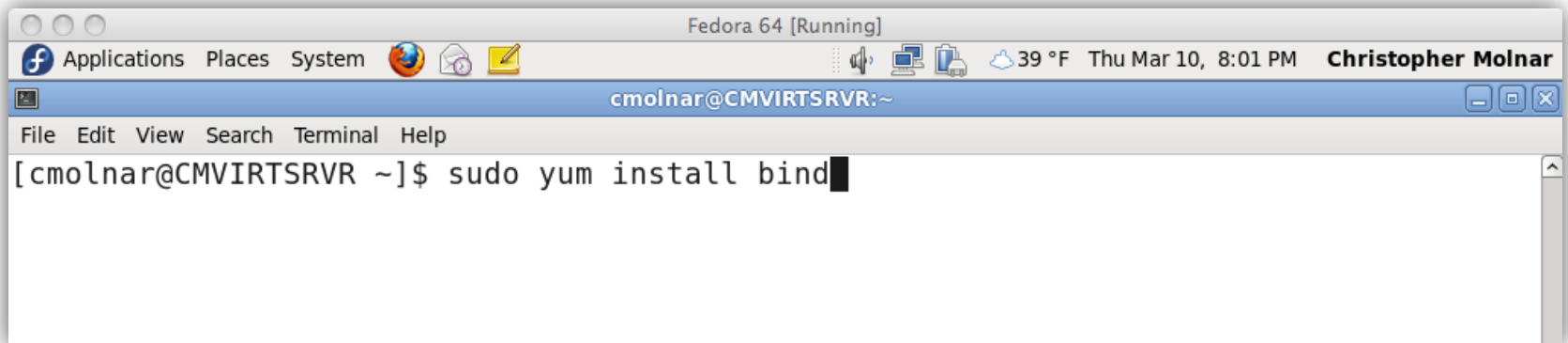
```
[cmolnar@CMVIRTSRVR ~]$ sudo yum upgrade
[sudo] password for cmolnar:
Loaded plugins: langpacks, presto, refresh-packagekit
Adding en_US to language list
updates/metalink | 6.8 kB | 00:00
updates | 4.7 kB | 00:00
updates/primary_db 45% [===== ] 252 kB/s | 2.1 MB | 00:10 ETA
```

Picture A: Root login and Yum upgrade

Step 2: Install Name Server

The *Named* server is installed in most Linux distributions through the “bind” package. To install BIND follow these steps:

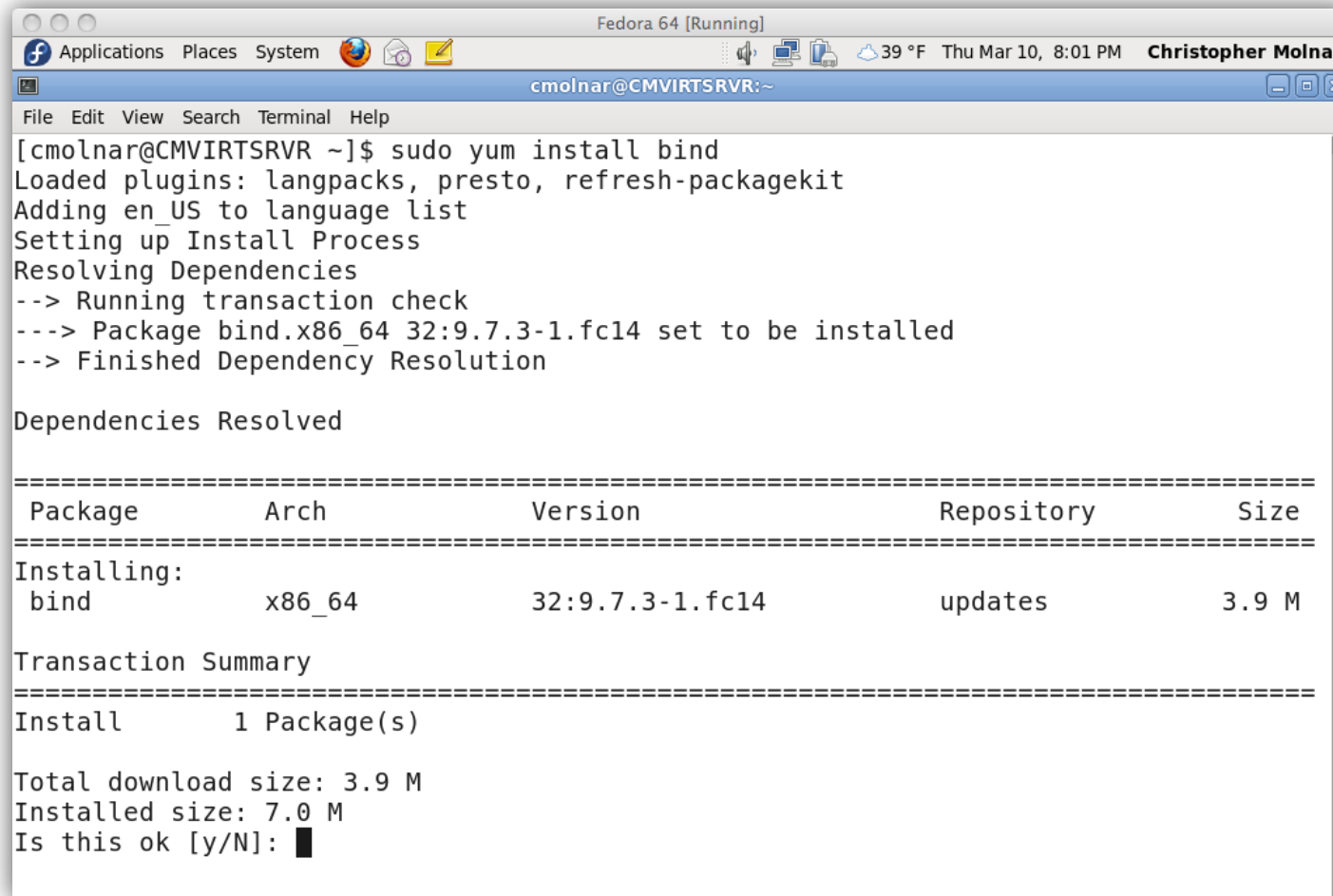
1. Type: **yum install bind** (At the command prompt while logged in through *su*, or by using the *sudo* command.)
2. Accept any dependencies the server wishes to install along with bind.

A screenshot of a Fedora 64 terminal window. The window title is "Fedora 64 [Running]". The top bar shows the user "Christopher Molnar" and the time "Thu Mar 10, 8:01 PM". The terminal prompt is "cmlnar@CMVIRTSRVR:~". The command "sudo yum install bind" is entered at the prompt, with a cursor at the end of the line. The terminal window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help".

```
[cmlnar@CMVIRTSRVR ~]$ sudo yum install bind
```

Picture B: Command to install BIND (See Pic D on next screen for additional details.)

Yum Upgrade



```
[cmolnar@CMVIRTSVR ~]$ sudo yum install bind
Loaded plugins: langpacks, presto, refresh-packagekit
Adding en_US to language list
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package bind.x86_64 32:9.7.3-1.fc14 set to be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package            Arch             Version          Repository      Size
=====
Installing:
bind               x86_64          32:9.7.3-1.fc14 updates         3.9 M

Transaction Summary
=====
Install            1 Package(s)

Total download size: 3.9 M
Installed size: 7.0 M
Is this ok [y/N]: █
```

Picture D: Installing BIND

Step 3: Locate the BIND Config Files

The named (or bind) configuration files are located in the /etc directory.

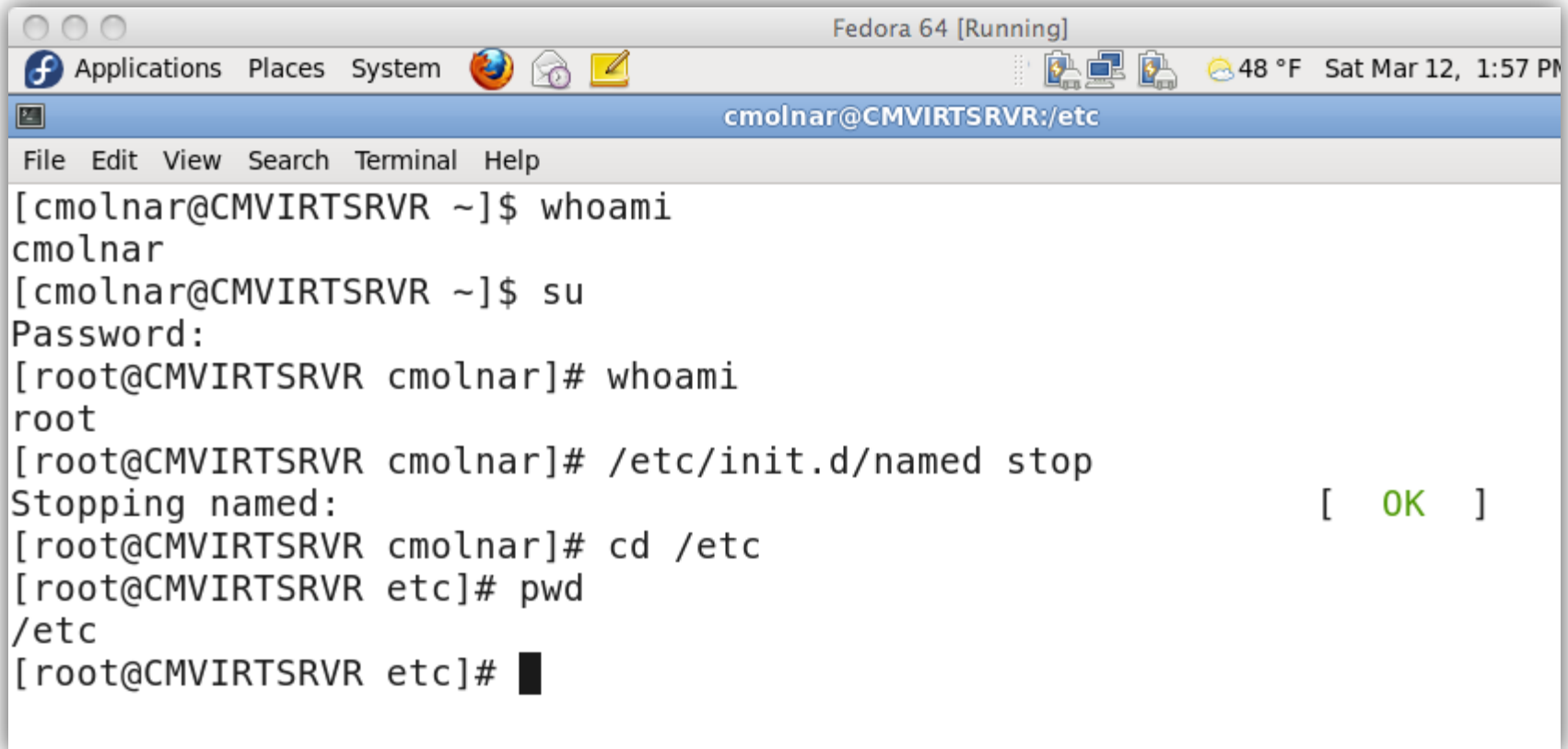
To view the named configuration files, type: **ls /etc/named***

Once you have verified the files are present, make sure the bind server is shut down and change to the /etc directory. To do this, follow these steps:

1. Type: **whoami** (to verify you are logged in as root – if not, put the word **sudo** in front of the following commands)
2. Type: **/etc/init.d/named stop**
3. Type: **cd /etc**
4. Type: **pwd** (to verify your location)

Review picture E on the next page for a screen shot of the above commands.

Step 3: Locate the BIND Config Files (Contd)



A terminal window titled "Fedora 64 [Running]" with a blue header bar showing "cmolnar@CMVIRTSRVR:/etc". The window contains the following text:

```
File Edit View Search Terminal Help
[cmolnar@CMVIRTSRVR ~]$ whoami
cmolnar
[cmolnar@CMVIRTSRVR ~]$ su
Password:
[root@CMVIRTSRVR cmolnar]# whoami
root
[root@CMVIRTSRVR cmolnar]# /etc/init.d/named stop
Stopping named: [ OK ]
[root@CMVIRTSRVR cmolnar]# cd /etc
[root@CMVIRTSRVR etc]# pwd
/etc
[root@CMVIRTSRVR etc]# █
```

Picture E: Commands to locate BIND Config files

Step 3b: Record IP and DNS Settings

The main configuration file is `/etc/named.conf`. Most DNS server problems are usually caused when this file is misconfigured.

To ensure a problem-free installation, verify your IP settings by typing **ifconfig** in your terminal window.

Write down the IP address of your “eth0” or “eth1” device.

Then verify your current DNS settings by typing:
nslookup www.psc.edu

Write down the address of the “Server” that appears in the first line. The address displayed is your current DNS server address.

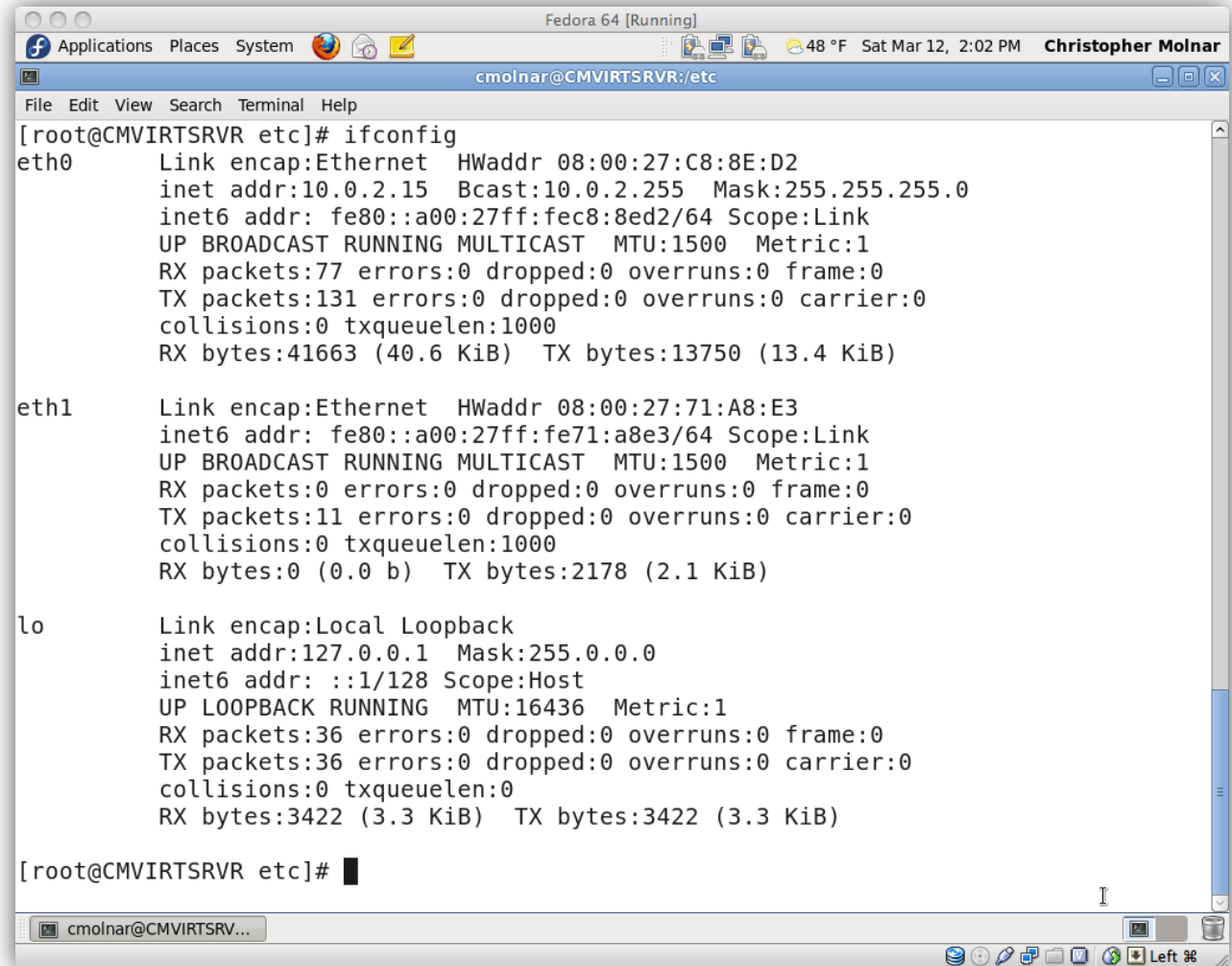
Review pictures F and G on the next two screens for more information.

Recommended Reading

- ❖ [Ifconfig Manpage](#)
- ❖ [Nslookup manpage](#)

Step 3b: Record IP and DNS Settings (Contd)

Picture F: Screenshot showing the results of the ifconfig command



```
Fedora 64 [Running]
Applications Places System 48 °F Sat Mar 12, 2:02 PM Christopher Molnar
cmolnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
[root@CMVIRTSRVR etc]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:C8:8E:D2
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec8:8ed2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:131 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41663 (40.6 KiB)  TX bytes:13750 (13.4 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:71:A8:E3
          inet6 addr: fe80::a00:27ff:fe71:a8e3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:2178 (2.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3422 (3.3 KiB)  TX bytes:3422 (3.3 KiB)

[root@CMVIRTSRVR etc]#
```

Step 3b: Record IP and DNS Settings (Contd)

```
[root@CMVIRTSRVR etc]# nslookup www.psc.edu
Server:                137.99.25.14
Address:               137.99.25.14#53

Non-authoritative answer:
Name:   www.psc.edu
Address: 128.182.65.57

[root@CMVIRTSRVR etc]# █
```

Picture G: Screenshot showing DNS Server address

Step 3c: Verify Proper DNS Operations

Fedora contains a utility called “dig” that allows you to verify proper operation of the DNS system prior to making any changes. DIG will search for a DNS record and return all records and fields associated with that DNS name. To verify proper operation:

Type **dig www.psc.edu** at the command prompt. (You should receive a number of DNS fields from your ISP’s name server. Try to match them to the DNS record descriptions on slide C4L1S25 of this lesson.)

Type **dig testfake.com** at the command prompt. (You should NOT receive results from the DNS records because *testfake.com* does not exist. The records you are specifically looking for is the IP address; if no IP addresses are returned, the domain name does not exist.

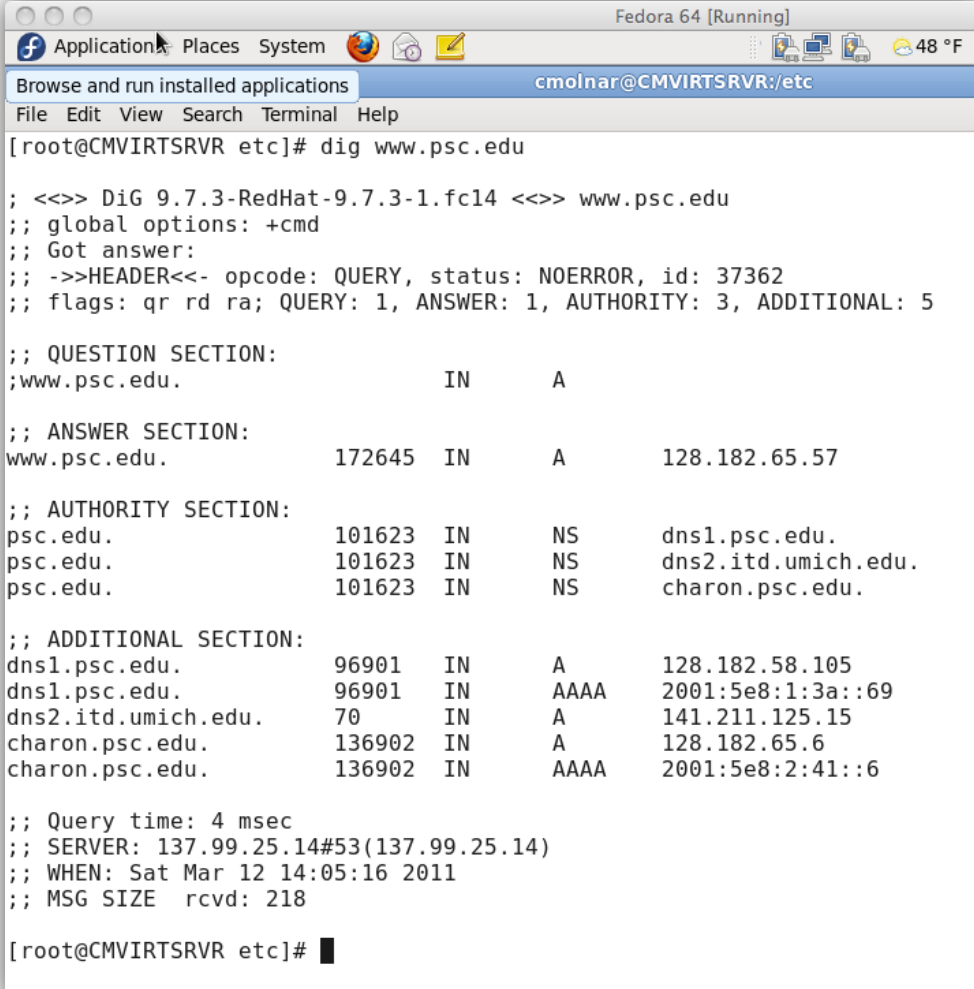
Now that we have our original settings, we can move forward and configure a domain name and server. Again, you must be careful making these changes. If you are working on a network without using your virtual machine in NAT mode, you can cause chaos in your workplace or educational institute. BE CAREFUL – Your network administrator will NOT be happy if you cause chaos on the network!

Suggested Review

- [Dig manpage](#)

Step 3c: Verify Proper DNS Operations

Picture H: Screenshot showing the results of the DIG command for psc.edu domain



```
Fedora 64 [Running]
Application Places System
Browse and run installed applications cmolnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
[root@CMVIRTSRVR etc]# dig www.psc.edu

; <<>> DiG 9.7.3-RedHat-9.7.3-1.fc14 <<>> www.psc.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37362
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;www.psc.edu.                IN      A

;; ANSWER SECTION:
www.psc.edu.                172645  IN      A      128.182.65.57

;; AUTHORITY SECTION:
psc.edu.                    101623  IN      NS     dns1.psc.edu.
psc.edu.                    101623  IN      NS     dns2.itd.umich.edu.
psc.edu.                    101623  IN      NS     charon.psc.edu.

;; ADDITIONAL SECTION:
dns1.psc.edu.              96901  IN      A      128.182.58.105
dns1.psc.edu.              96901  IN      AAAA   2001:5e8:1:3a::69
dns2.itd.umich.edu.       70     IN      A      141.211.125.15
charon.psc.edu.           136902  IN      A      128.182.65.6
charon.psc.edu.           136902  IN      AAAA   2001:5e8:2:41::6

;; Query time: 4 msec
;; SERVER: 137.99.25.14#53(137.99.25.14)
;; WHEN: Sat Mar 12 14:05:16 2011
;; MSG SIZE rcvd: 218

[root@CMVIRTSRVR etc]#
```

Step 4: Adjust Named.Conf to Listen

The first thing we need to do is to tell *named.conf* to listen to a valid network adapter and define who can contact named. If you are setting up your server to serve DNS to your local network, you will want to add the IP address of your network adapter from the *ifconfig* command in slide C4L1S34. In order to set the correct listening port, follow the directions below: (Do not exit vi when finished).

1. From the terminal window, log in as root, and in the */etc* directory type:
vi named.conf
2. Locate the section labeled “*options*” and find the “*listen*” stance.
3. Put your cursor at the end of the line and type: **A** (to go into insert mode).
4. Press **ENTER**
5. Type: **Listen-on port 53 { 10.0.2.15 };**
6. Press **ENTER** and **ESC** to exit insert mode.

Now the server will know to listen for a DNS query on the appropriate port.

Next we have to tell the server what machines may contact it.

Required Reading

- [Named.Conf](#)

Step 4: Adjust Named.Conf to Listen

```
options {  
    listen-on port 53 { 127.0.0.1; };  
    listen-on port 53 { 10.0.2.15; };  
    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";  
    allow-query    { localhost; };  
    recursion yes;  
  
    dnssec-enable yes;
```

Picture 1: Screenshot showing configuration of named.conf to listen on a specific port

Step 5: Set Authorized Clients

We want the local host, as well as all devices on the 10.0.2.* network to be able to use the DNS server we just configured. Follow the directions below to make the following changes:

1. Locate the “*Allow-query*” stance in the open vi window.
2. Place your cursor after the “localhost;” and press “**a**” to enter insert mode at the location of the cursor.
3. Type: **10.0.2.0/24**; (to allow your current network to access your DNS server. If your IP address is different, use the network broadcast address instead—It usually ends with a zero.)
4. Press **ESC**, then **:w** to save the file but DO NOT exit vi.

The above changes allow our localhost (the current machine, and the machines on the local network) to access your DNS server for queries.

Next, we need to make sure that the DNS server knows where to forward unknown requests.

Step 5: Set Authorized Clients

```
listen-on port 53 { 10.0.2.15, };  
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.tx  
allow-query { localhost; 10.0.2.0/24; };  
recursion yes;
```

Picture J: Screenshot showing changes to localhost

Step 6: Set DNS Forwarding Address

Our local DNS server cannot store all the information about all DNS addresses on the Internet. Therefore, we need to be able to direct DNS requests to a higher level DNS server to find addresses that may not be contained locally. In order to do this, we need to set forwarder lines.

1. From your current vi window, scroll down to the line just before the closing bracket } of the *Options* section.
2. Place your cursor on the following blank line and press **I** for insert.
3. Press **ENTER** once to create a blank line.
4. Type the following: **forwarders { [dnsserver]; }**
5. Press **ESC** after the last line. Replace “[dnsserver]” with the IP address of the DNS server from your NSLOOKUP command in Step 3b (C4L1S35)
6. Type **:w** to save, but do NOT exit vi.

Review the screen shot of the process on the next screen.

Now, we have our basic settings complete. Next we are going to define a zone DB file.


Step 6: Set DNS Forwarding Address

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on port 53 { 10.0.2.15; };
    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";
    allow-query    { localhost; 10.0.2.0/24; };
    recursion yes;

    dnssec-enable yes;
    dnssec-validation yes;
    dnssec-lookaside auto;

    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    managed-keys-directory "/var/named/dynamic";
    forwarders { 139.99.25.14; };
};
```



Picture K: Screenshot showing forwarding DNS address

Step 7: Define a Zone DB File

In this step, we need to tell Named where all the data for testfake.com will be found. Follow these directions to define a zone db file for the testfake.com domain we are going to create.

1. Use your cursor to go down to the line before the “insert” statements.
2. Enter insert mode by typing: **A**
3. Press **ENTER** to go to a new line.
4. Enter the following code and press escape when done.

```
zone “testfake.com” {  
    type master;  
    file “/etc/named/testfake.com.db;  
};
```

5. Save your file by typing **:w**
6. Exit vi by typing **:q**

What the above code is telling Named is that all the information for the *testfake.com* domain will be found in a db file named *testfake.com.db* in the */etc/named* directory. The remainder of our configuration will be done in that file.

Review the screen shot of the process on the next screen.

Required Reading

- ❖ [Zone File Format](#)

Step 7: Define a Zone DB File (Contd)



```
zone "testfake.com" {  
    type master;  
    file "/etc/named/testfake.com.db";  
};  
  
include "/etc/named.rfc1912.zones";  
include "/etc/named.root.key";
```

Picture L: Screenshot showing configuration of a DB zone for testfake.com domain

Step 8: Configure Zone DB File

Follow these directions to create the zone db file for the testfake.com domain.

1. From the command line type:
vi /etc/named/testfake.com.db (this name must match exactly the name you used in the named.conf file of the prior slide).
1. Enter insert mode by typing: **I**
2. Press **ENTER** to go to a new line.
3. Enter code on right and press **ESC** when done.
4. Save your file by typing **:w**
5. Exit vi by typing **:q**



```
$TTL 3D
@ IN SOA ns.testfake.com. admin.testfake.com. (
    2007062001
    28800
    3600
    604800
    38400
);
testfake.com. IN NS ns.testfake.com.
ns IN A 127.0.0.1
@ IN A 127.0.0.1
www IN A 127.0.0.1
mail IN A 127.0.0.1
gw IN A 127.0.0.1
TXT "OUR FAKE ZONE"
~
```

Completing this exercise creates a zone record for your fake zone. Remember, the IP address of 127.0.0.1 is always your own local machine. If you would like to experiment further, you can use an IP address of another machine on your network or a remote machine but using 127.0.0.1 is the safest approach.

Step 8: Define a Zone DB File (Contd)



```
Fedora 64 [Running]
cmolnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
$TTL 3D
@ IN SOA ns.testfake.com. admin.testfake.com. (
  2007062001
  28800
  3600
  604800
  38400
);
testfake.com. IN      NS      ns.testfake.com.
ns             IN      A      127.0.0.1
@             IN      A      127.0.0.1
www           IN      A      127.0.0.1
mail          IN      A      127.0.0.1
gw            IN      A      127.0.0.1
              TXT     "Our Fake Zone"
```

Picture M: Screenshot showing configuration of a DB zone for testfake.com domain

Step 9: Start DNS Server

Follow these directions to start the DNS server:

From the command prompt (still as root user) type:

```
/etc/init.d/named start
```

This command will start the named server. If you get any errors, go back and check your entries in the configuration file and ensure they match the entries in this lesson.

Next we need to add your local machine to the client's DNS resolver settings.

A screenshot of a Linux terminal window. The window title is 'cmlnar@CMVIRTSRVR:/etc'. The terminal shows the command '/etc/init.d/named start' being executed, followed by the output 'Starting named:' and '[OK]'. The terminal prompt is '[root@CMVIRTSRVR etc]#'. The window's top bar shows system information: '52 °F Sat Mar 12, 3:15 PM Chris'.

```
f Applications Places System 52 °F Sat Mar 12, 3:15 PM Chris
cmlnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
[root@CMVIRTSRVR etc]# /etc/init.d/named start
Starting named: [ OK ]
[root@CMVIRTSRVR etc]#
```

Picture N: Screenshot showing command to start the DNS server

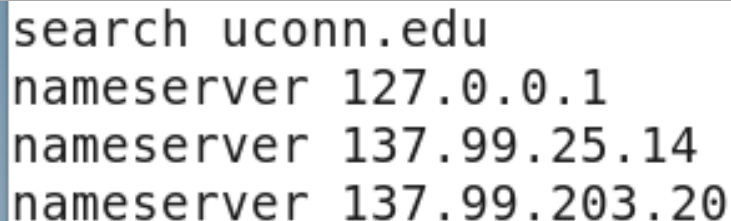
Step 10: Change the resolv.conf File

In order to test the new domain name system server, we need to adjust the `/etc/resolv.conf` file to include your local DNS server. Follow these directions:

1. Type: **vi /etc/resolv.conf**
2. Place your cursor right before the first “`nameserver`” line and press **I** to enter insert mode.
3. Type: **nameserver 127.0.0.1** and press **ENTER**.
4. Press **ESC**.
5. Type **:x** to save and exit vi.

Now, your DNS server at your local machine `127.0.0.1` will be first on the client’s list of name servers to contact.

Next, we need to test what we have accomplished.



```
search uconn.edu
nameserver 127.0.0.1
nameserver 137.99.25.14
nameserver 137.99.203.20
```

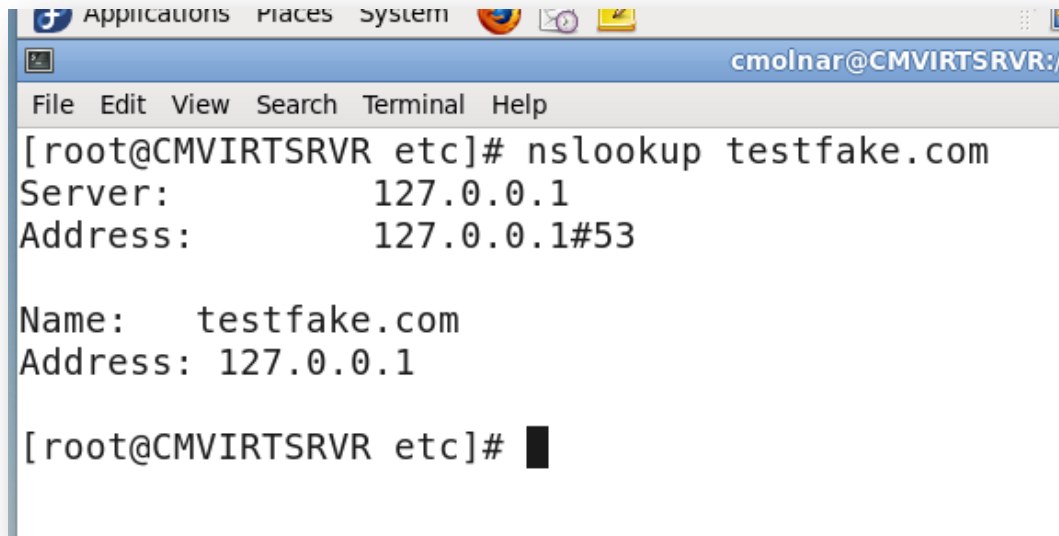
Picture 0: Screenshot showing the three name-servers contacted after a domain name search

Step 11: Test the Server Using Nslookup

At the command prompt type:

nslookup testfake.com

If everything is working correctly, the nslookup command will return 127.0.0.1 as an IP address from the server 127.0.0.1.

A screenshot of a terminal window titled 'cmolnar@CMVIRTSRVR:/' with a menu bar containing 'File Edit View Search Terminal Help'. The terminal shows the command '[root@CMVIRTSRVR etc]# nslookup testfake.com' and its output: 'Server: 127.0.0.1', 'Address: 127.0.0.1#53', 'Name: testfake.com', and 'Address: 127.0.0.1'. The prompt '[root@CMVIRTSRVR etc]#' is followed by a cursor.

```
cmolnar@CMVIRTSRVR:/  
File Edit View Search Terminal Help  
[root@CMVIRTSRVR etc]# nslookup testfake.com  
Server:          127.0.0.1  
Address:         127.0.0.1#53  
  
Name:   testfake.com  
Address: 127.0.0.1  
  
[root@CMVIRTSRVR etc]# █
```

Picture P: Screenshot showing server testing using nslookup

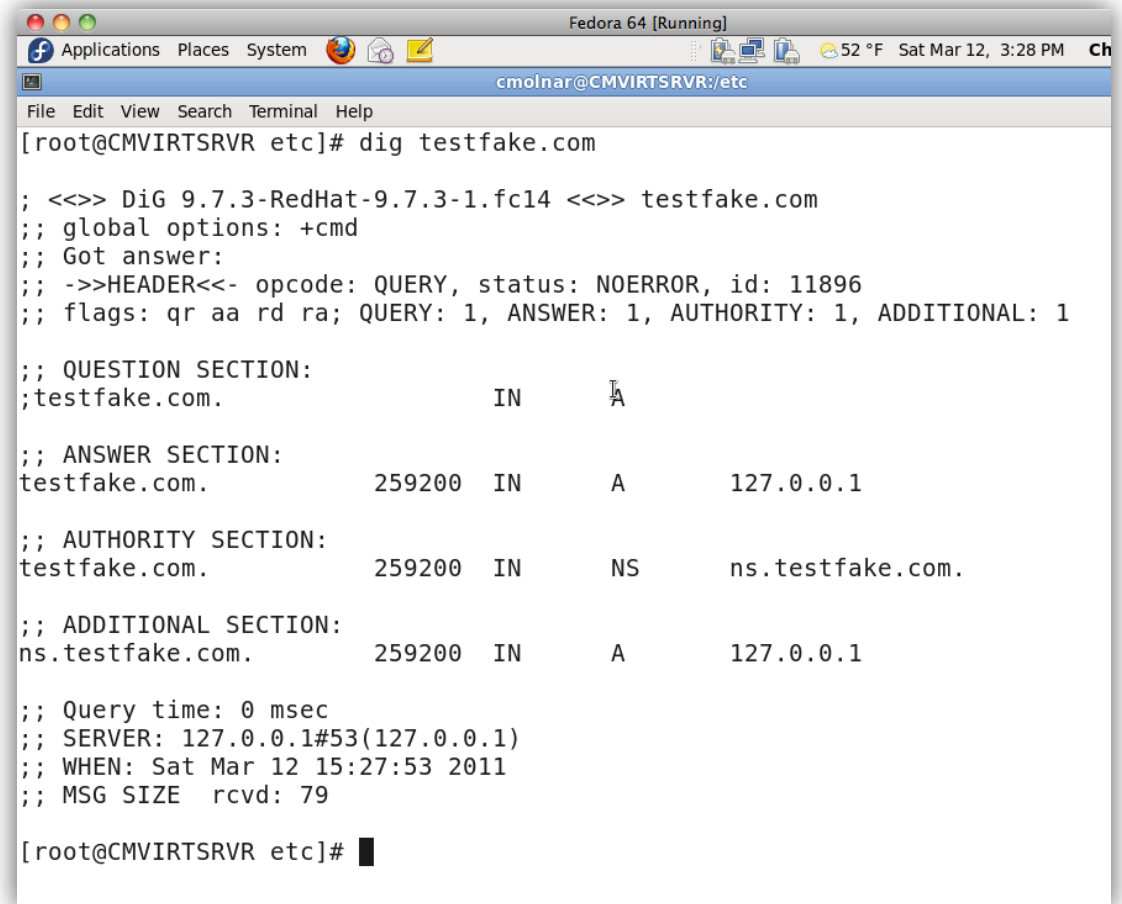
Step 12: Test the Server Using DIG

Next we are going to use the DIG command to test our DNS file settings. At the command prompt type:

dig testfake.com

The DIG command should return settings which match those that you entered into the file on slide (C4L1S37).

Picture Q: Screenshot showing server testing using DIG



```
Fedora 64 [Running]
Applications Places System 52 °F Sat Mar 12, 3:28 PM
cmolnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
[root@CMVIRTSRVR etc]# dig testfake.com

; <<>> DiG 9.7.3-RedHat-9.7.3-1.fc14 <<>> testfake.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 11896
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
testfake.com.                IN      A

;; ANSWER SECTION:
testfake.com.                259200  IN      A      127.0.0.1

;; AUTHORITY SECTION:
testfake.com.                259200  IN      NS     ns.testfake.com.

;; ADDITIONAL SECTION:
ns.testfake.com.            259200  IN      A      127.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 12 15:27:53 2011
;; MSG SIZE rcvd: 79

[root@CMVIRTSRVR etc]#
```

Step 13: Test the Subdomains

In our zone.db file, we had a number of sub-domains including: *www*, *mail*, and *gw*. Let us use DIG to test these subdomains as well.

At the command line enter:

dig www.testfake.com

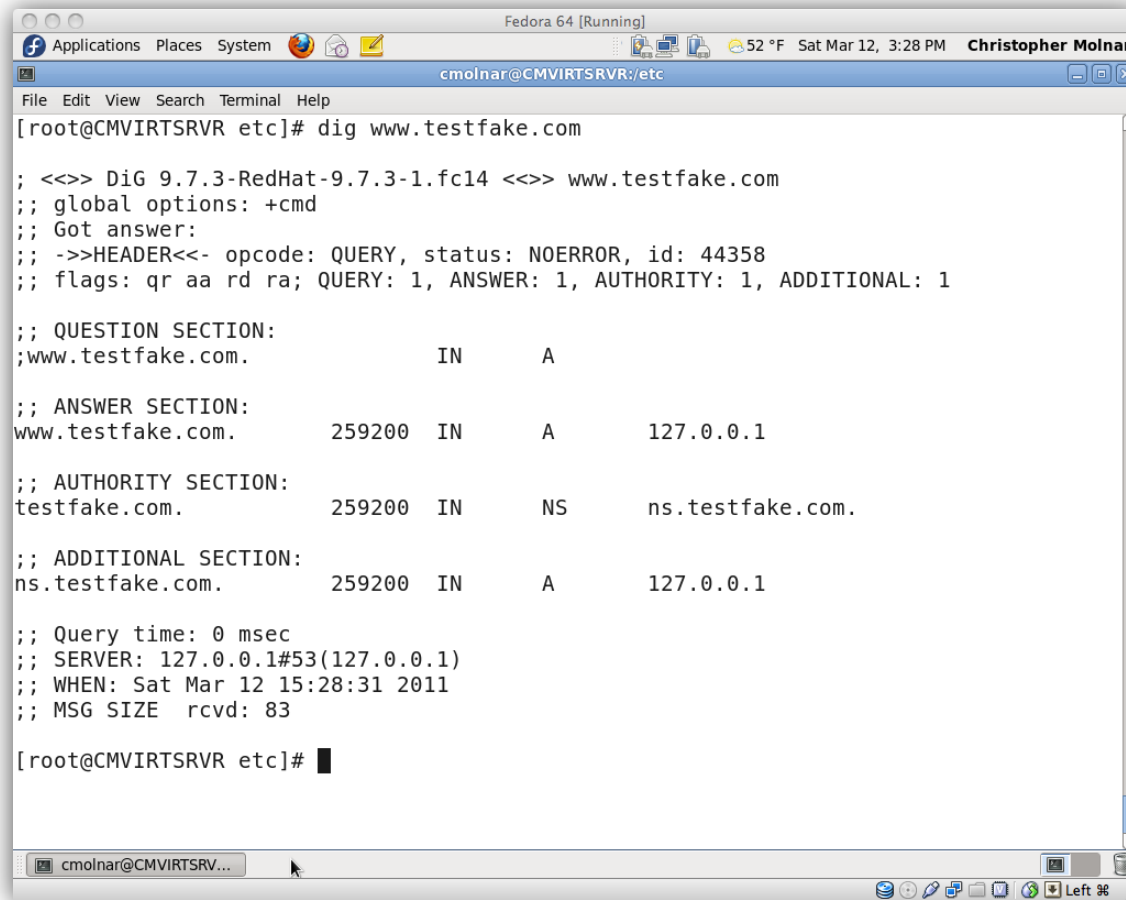
The IP address should display as 127.0.0.1

Finally type: **dig www.psc.edu**

Make sure the DNS server forwards the request to another name server per your configuration of the forwarder line.

Review screenshots on the next two slides for reference

Step 13: Test Subdomains (Contd)



```
Fedora 64 [Running]
Applications Places System 52°F Sat Mar 12, 3:28 PM Christopher Molnar
cmolnar@CMVIRTSRVR:/etc
File Edit View Search Terminal Help
[root@CMVIRTSRVR etc]# dig www.testfake.com

; <<>> DiG 9.7.3-RedHat-9.7.3-1.fc14 <<>> www.testfake.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44358
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.testfake.com.                IN      A

;; ANSWER SECTION:
www.testfake.com.                259200  IN      A      127.0.0.1

;; AUTHORITY SECTION:
testfake.com.                    259200  IN      NS     ns.testfake.com.

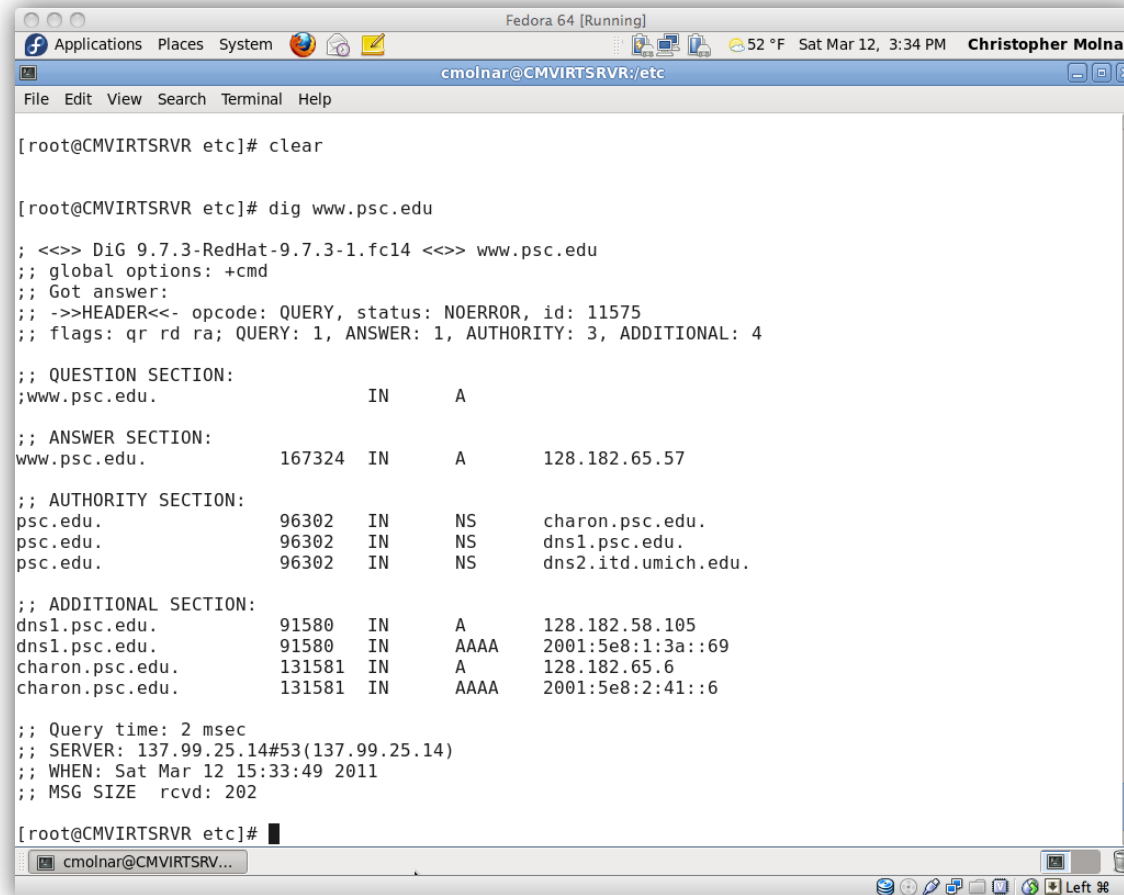
;; ADDITIONAL SECTION:
ns.testfake.com.                 259200  IN      A      127.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 12 15:28:31 2011
;; MSG SIZE rcvd: 83

[root@CMVIRTSRVR etc]#
```

Picture R: Screenshot showing testing of subdomain using DIG command

Step 13: Test Subdomains (Contd)



```
[root@CMVIRTSRVR etc]# clear

[root@CMVIRTSRVR etc]# dig www.psc.edu

; <<>> DiG 9.7.3-RedHat-9.7.3-1.fc14 <<>> www.psc.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11575
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 4

;; QUESTION SECTION:
;www.psc.edu.                IN      A

;; ANSWER SECTION:
www.psc.edu.                167324 IN      A      128.182.65.57

;; AUTHORITY SECTION:
psc.edu.                    96302  IN      NS     charon.psc.edu.
psc.edu.                    96302  IN      NS     dns1.psc.edu.
psc.edu.                    96302  IN      NS     dns2.itd.umich.edu.

;; ADDITIONAL SECTION:
dns1.psc.edu.              91580  IN      A      128.182.58.105
dns1.psc.edu.              91580  IN      AAAA   2001:5e8:1:3a::69
charon.psc.edu.            131581 IN      A      128.182.65.6
charon.psc.edu.            131581 IN      AAAA   2001:5e8:2:41::6

;; Query time: 2 msec
;; SERVER: 137.99.25.14#53(137.99.25.14)
;; WHEN: Sat Mar 12 15:33:49 2011
;; MSG SIZE rcvd: 202

[root@CMVIRTSRVR etc]#
```

Picture S: Screenshot showing testing of subdomain using DIG command

Lesson Summary

In this lesson, we discussed the DNS system, which is basically a phone book for the Internet.

We then learned to install and configure a basic DNS server, as well as configure a client to point to the newly created server. To minimize security risks, we used our local client for testing so that your test machine would not be broadcasting incorrect DNS information across a large network and possibly get you in trouble at work or school.

What would happen if you created a record for google.com on your school's network and sent everyone to an IP address of 127.0.0.1?

We also used the *nslookup* and *dig* commands to check your DNS settings and the record information found on your DNS server. These commands are required when testing a new DNS server.

If you wish to setup a true DNS server on your local network, you should change all the 127.0.0.1 IP addresses to actual IP addresses, and the line in your *resolv.conf* file would be the external (adapter) IP address of the DNS server.

Recommended Reading

- ❖ [Linux DNS Services](#)
- ❖ [Configuring DNS](#)
- ❖ [BIND Config](#)