Build a DNS server on Linux using BIND

Aug 24, 2001
Thomas Nooning CCNA, CCDA

Although Linux has yet to make a noticeable dent in the world of desktop computing, it continues to make quite a name for itself as a network server. Because it is extremely reliable, Linux can be counted on to run important services that are required in today's Internet era. Thus, some of the most popular uses for Linux include mail servers, Web servers, and DNS servers.

We're going to examine how to create a DNS server out of a Red Hat Linux machine using the magic of BIND. BIND, or Berkeley Internet Name Domain, is a software package that implements a DNS server on UNIX/Linux systems. We will review the installation procedures, initial configuration, and system settings required to get a DNS server up and running on Linux.

Downloading and installing BIND

You can download source packages for BIND from www.isc.org/products/BIND/. Here, you will find the latest version, 9.1.3 (at the time of this writing), as well as support information and FAQs.

Since we're working with Red Hat in this example, you may want to use an RPM version of BIND, which you can locate by doing a search on www.rpmfind.net. For distributions that use RPMs, this is the easiest way to install the BIND package. After you download the RPM for BIND, *bind-9.1.0-10.i386.rpm*, for example, you need to run the command *rpm –ivh bind-9.1.0-10.i386.rpm* as root. As the RPM attempts to install BIND, be sure to watch out for any missing dependencies that might cause the installation to fail.

If you prefer to install from the binary source package, after you have downloaded *bind-9.1.3.tar.gz*, for example, you may uncompress it by running *tar xvfz bind-9.1.3.tar.gz* as root. In the case of the tarball, a *bind-9.1.3* directory is created, which has everything needed for a new installation. Next, move into the newly formed directory and execute the ./configure command. After the script goes through the configuration steps, run *make* and then *make install* to complete the process. BIND should now be installed on your system with a basic configuration set. Now it's time to tweak it according to your needs.

Configuring BIND

Once BIND is installed on your system, you can configure it a number of ways. The two most common configurations are those used in an ISP-type setting, where the DNS server will accept and resolve requests from anyone (or a predefined subset of users), and for Web hosting, where the server will resolve requests for hosted domains only. As the use of the server changes, you can also change the type of configuration.

DNS servers are known as either a Master or a Slave. The Master server, also known as the *primary* server, is the definitive source of information regarding a domain. The Master domain server is also the source of zone transfers directed to the Slave server. The Slave server, while

also being authoritative, receives all of its zone information from the Master. A common mistake is to attempt to modify zone files on the Slave server and not on the Master. Why two servers? Redundancy, a good idea in any configuration, is a native part of BIND and DNS. Fortunately, it will usually not break your budget to put a second Linux box in your server room to serve as a secondary DNS server.

Keep in mind that while there are two main types of servers, the Master can also act as a Slave for other domains. This will be seen most often in an ISP environment where customers could have their own Master server and would like to use the ISP as a Slave for backup purposes. Configuration for this, and most everything else in BIND, is done through *named.conf*. This file stores server and zone information in a clear-text format. It will look something like **Listing A**.

Many options are available for use on your DNS server. I recommend looking through the included documentation if you require a special installation. In this case, the *notify-source* directive tells the server where to send NOTIFY messages, which the Master sends the Slave when it detects that a change has been made to a zone file. The *pid-file* option merely tells the daemon the pathname to which the server writes its process ID information. This is usually */var/run/named.pid*, but you may need to change it if you have reconfigured your directory layout.

The first zone entry shown above is needed to tell BIND where it can locate root server information. This is how your server will send and receive information about not only your domains but also about all domains on the Internet. Not every server will have an entry for every domain name, but every server (except for those simply caching) will know how to get the information. Of course, this list changes periodically, so make a note to update it regularly.

The second zone listed in our sample named.conf file—example.com—is a "master" domain entry, which means that this DNS server holds the authoritative information for example.com that all of the other DNS servers on the Internet will use to reference anything related to this domain. The example.com entry references the file /var/named/sample.com.zone, which is a plain-text file that tells the DNS server everything about the example.com domain, including the serial number, the refresh rate, all DNS records, and many other options. **Listing B** shows an example of this zone file.

SOA is short for Start of Authority, which is the preamble necessary for all zone files. The serial number lets the server keep track of updates. As long as this number increases incrementally from the last time the daemon started, it will reread the information into its database. For instance, you could start at 0 and add a digit after every modification, or you could use a date entry (as in the example) like 200101111. This is helpful because it lets you see when the last update was made and whether it was updated more than once in a single day. The next four lines deal with refresh and time-out periods in seconds. If no manual or server-wide refresh of the BIND database occurs, the server will automatically attempt to reread the information. Changing from the values listed here is not usually necessary. Only domains that for one reason or another change their information extremely often will need to make modifications. The nameservers are then listed so BIND will know who has control over this domain.

The MX record is listed next. This lets the server know what information to give when mail

information is requested for sample.com. In this case, it is mail.sample.com, which has a priority of 10. You may list multiple MX records as backup mail servers in case one fails. The lowest number is considered the primary. Notice there is also a corresponding A record that gives the IP address for mail.sample.com. This is required so that the DNS servers know specifically where to direct mail requests for the domain.

An A record is merely the way in which IP address are assigned to subdomain entries, such as www, mail, ftp, or ns. These should always be entered in the format above and will always correspond to an IP address. For example, when a user requests www.sample.com, he or she will be directed to the domain's Web server at IP address 212.204.219.71.

There is also a *CNAME* entry in the example above. CNAME, which stands for "canonical name," is a way of using aliases as opposed to IP addresses. Standard practice dictates these should refer back to A records already in use.

Now that we've examined a Master entry in *named.conf*, let's look at one for a Slave:

```
zone "sample2.com" {
type slave;
file "/var/named/sample2.com.zone";
master { 10.0.0.1; };
```

The two main differences are the *type*, which is either *master* or *slave*, and the entry, which gives the IP address of the Master DNS server. Everything else will be the same as in a Master entry.

Starting BIND

The program that needs to run in order to start DNS services is *named*, pronounced "name D." You can run this program with the command */etc/rc.d/init.d/named start*. If the server is already running, you can use *restart*. This script, which should be placed into the proper directory upon installation, will run at startup to initiate the server. As always, you should check to make sure *named* is running by executing a *ps aux*, which will give you a full list of all current processes.

Conclusion

You now have a good idea of how to get a DNS server started with BIND on Linux. From downloading the latest version to configuring the basics, you should be able to turn a Red Hat Linux machine into a full-fledged DNS server. DNS and BIND are topics that could easily fill the pages of a book—and have. I recommend the documentation that comes with the program, as well as O'Reilly's *DNS and BIND*. The resolving of domain names is a requirement for the Internet to function and is an excellent use for Linux.

Do you have tips for configuring BIND on Linux?

We look forward to getting your input and hearing about your experiences regarding this topic. Join the discussion below or send the editor an e-mail.

ý