The /etc/hosts file

Every system will have to keep its copy of the table of the hostnames and their IP addresses. This file is responsible for IP addresses.

On Linux systems, this table is the **/etc/hosts** file.

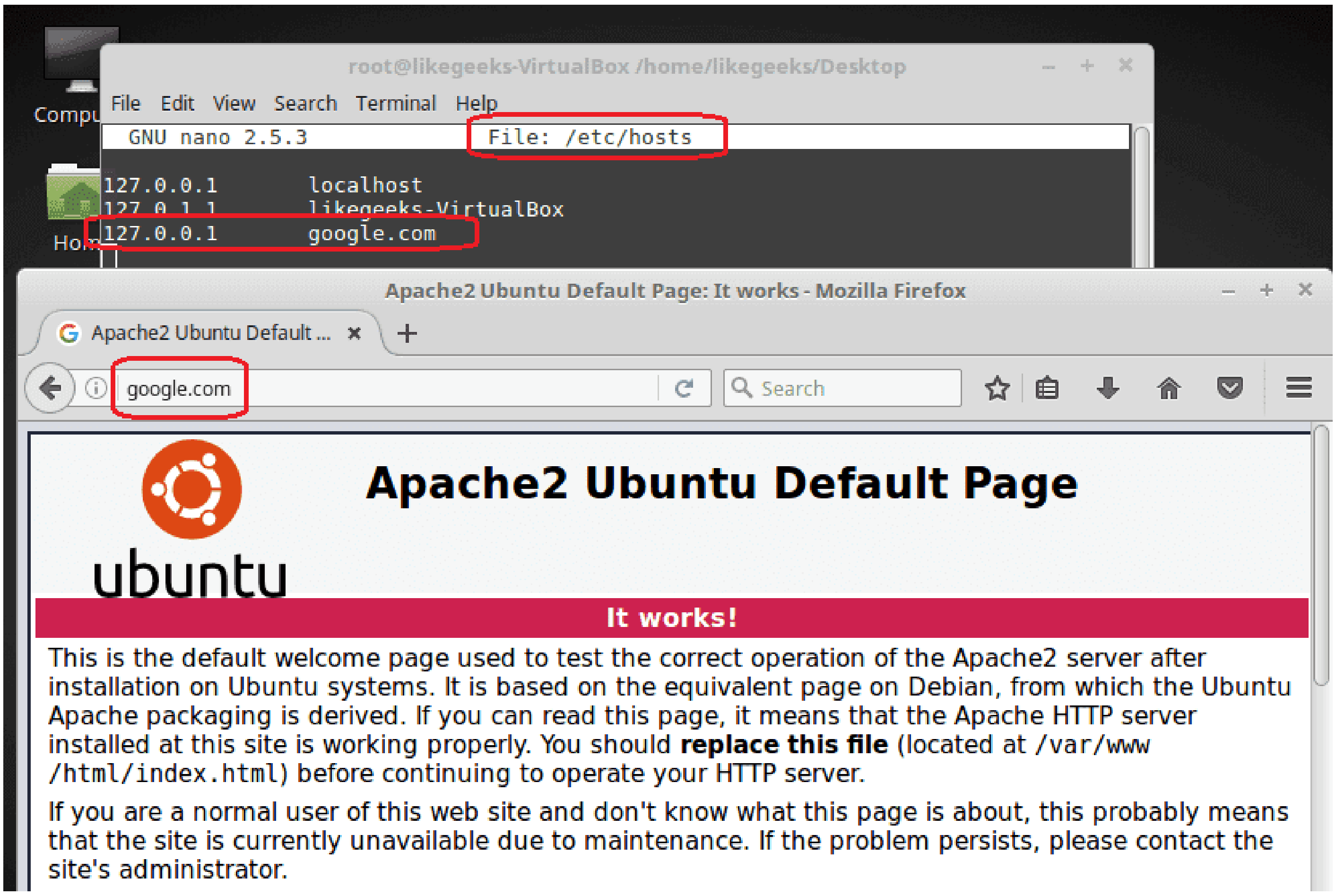
So even if you don't have a DNS server or DNS server is unavailable, this file can translate IP addresses to names using /etc/hosts file.

That means the system query this file first before going to the DNS server, and if it finds the domain, it will translate it without going to any DNS servers.

Try to edit /etc/hosts and type the following:

```
127.0.0.1                google.com
```

Then go to your browser and type google.com and see the results. If you have **Apache server** installed on your system and your localhost is running, it will show the index page of the localhost instead of the google page.


root@likegeeks-VirtualBox /home/likegeeks/Desktop
File Edit View Search Terminal Help
Apache2 Ubuntu Default Page: It works - Mozilla Firefox
Apache2 Ubuntu Default ...
google.com
Search
ubuntu

```
  GNU nano 2.5.3                File: /etc/hosts

127.0.0.1       localhost
127.0.1.1       likegeeks-VirtualBox
127.0.0.1       google.com
```

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

You can translate google.com to any other IP address of any site and see the result to ensure that.

So what this file is doing is translating IP addresses to names, but this for the same connected network. So what about the outside networks and how to maintain all those records for all systems?

Will everybody manages his own /etc/hosts file and update it himself? Of course not.

Domain names

When you visit a website, you type the FQDN (Fully Qualified Domain Name) or the domain name like this: likegeeks.com or www.google.com

Each domain consists of domain components; the dot separates these components.

The text **com** is the **top-level domain component,** and **google** is the **second-level domain component,** and **www** is the **third-level domain component**

When you visit any website, the browser silently adds a dot at the end, but not visible to you, so the domain will be like www.google.com. Notice the dot after .com; this **dot** is called the **root domain.**

But why we added this root domain or the dot?

Because this dot is served by the **root name servers, at** the time of this post, there are 13 root name servers in the world, you can think of them as the brain of the internet, if they go OFF the world will be without the internet.

And why 13?

Because maybe an earthquake or a natural disaster happens in one place in the world may destroy a root server, so the others serve until the damaged server returns online.

Those root name servers have names like this: a.root-server.net, b.root-server.net, and so on.

Top Level domain names (TLDs)

We saw a top-level domain component, such as com domains.

Top-level domains (TLDs) are divided into categories based on geographical or functional aspects.

There are more than 800 top-level domains on the web at the time of writing this post.

The top-level domains categories are:

- Generic top-level domain like (.org, .com, .net, .gov, .edu and so on).
- Country-code top-level domains like (.us, .ca, and so on) corresponding to the country codes for the United States and Canada, respectively.
- New branded top-level domains like (.linux, .microsoft, .companyname and so on).
- Infrastructure top-level domains like the .arpa domain.

Subdomains

When you visit a website like mail.google.com, the mail here is a subdomain of google.com.

Only the name servers for mail.google.com know all the hosts existing beneath it, so google answers if there is mail subdomain or not, the root name servers have no clue about that.

Types of DNS servers

There are three types of DNS servers:

- **Primary DNS servers**: They contain the domain's configuration files, and they respond to the DNS queries.
- **Secondary DNS server**: They work as a backup and load distribution. Primary servers know the existence of the secondary name servers and send updates to them.
- **Caching DNS server**: All they do is caching the DNS responses, so you don't need to ask the primary or secondary DNS server again. You can make your system work as a caching server easily, as we will see later on this post.

Setting up Linux DNS server

There are many packages on Linux that implement DNS functionality, but we will focus on the **BIND DNS server**. Many servers around the world use it.

If you are using Red Hat based distro like CentOS, you can install it like this:

```
$ dnf -y install bind
```

Or on Debian based systems like Ubuntu:

```
$ apt-get install bind9
```

Once the installation completed, you can start it and enable it to run at boot time.

```
$ systemctl start named

$ systemctl enable named
```

Configuring BIND

The service configuration is **/etc/named.conf** file.

There are some statements that BIND uses in that file like:

options	used for global BIND configuration.
logging	what can be logged and what can be ignored. I recommend you review the **Linux syslog server**.
zone	define DNS zone.
include	to include another file in named.conf.

From the options statement, you can see that the working directory for BIND is /var/named directory.

The zone statement enables you to define a DNS zone.

Like the domain google.com which also has subdomains like mail.google.com and analytics.google.com and other subdomains.

Every one of these three (the domain and subdomains) has a zone defined by the zone statement.

Defining a primary zone

We know from the DNS server types that there are primary, secondary, and cache DNS servers.

Primary and secondary are equally authoritative in their answers, unlike the caching server.

To define a primary zone, you can use the following syntax:

```
/etc/named.conf



zone      "likegeeks.com" {

type master;

file likegeeks.com.db

};
```

The file that contains the zone information is located in **/var/named** directory since this is the working directory, as we know from the options.

Note that the server software or the hosting panel you’re using creates this file with this name automatically for you, so if your domain is example.org, the file will be **/var/named/example.org.db**.

The type is master, which means this is a primary zone.

Defining a secondary zone

The same as the primary zone definition with little change.

```
zone      "likegeeks.com" {

type slave;

masters Primary Nameserver IP Address Here; ;

file likegeeks.com.db

};
```

In the secondary zone, the domain is the same as the primary zone, and the type **slave** here means this is a **secondary zone**, and the masters option to list the IP addresses of the primary nameserver and finally, the file is the path of the primary's zone files.

Defining a caching zone

It is necessary to have a caching zone, so you decrease the queries on the DNS server.

To define a caching zone, you need to define three-zone sections the first one:

```
zone      "." IN {

type hint;

file "root.hint";

};
```

The first line contains a dot, which is the root name servers. The type hint, which means a caching zone entry, and the file "root.hints"**;** specifies the file that contains the root servers (the 13 root name server). You can get the latest root name server from **http://www.internic.net/zones/named.root**

The second zone defined in the **/etc/named.rfc1912.zones** file and included in /etc/named.conf via include directive, which is already included by default.

```
zone      "localhost" IN {

type master;

file "localhost.db";

};
```

The third zone defines the reverse lookup for the localhost.

```
zone      "0.0.127.in-addr.arpa" IN {

type master;

file "127.0.0.rev";

};
```

Putting these three zones on /etc/named.conf will make your system work as a caching DNS server. Now you should type the content of the files referenced like likegeeks.com.db, localhost.db, and 127.0.0.rev.

These files contain the DNS record types for each zone with some options. So what are those DNS record types and how to write them?

DNS records types

The database files consist of record types like **SOA, NS, A, PTR, MX, CNAME, and TXT**.

So let’s start with each record type and see how we can configure it.

SOA: Start of Authority Record

The SOA record describes the site’s DNS entries with the following format:

```
example.com.      86400      IN      SOA      ns1.example.com.

2017012604 ;serial

86400 ;refresh, seconds

7200 ;retry, seconds

3600000 ;expire, seconds

86400 ;minimum, seconds

)
```

The first line starts with the domain example.com. and ends with a period. Which is the same as the zone definition in /etc/named.conf file.

Keep in mind that DNS configuration files are extremely picky.

The **IN** word means Internet record.

The **SOA** word means Start of Authority record.

The ns1. example.com**.** is the domain’s name server.

The mail.host.com. is the domain administrator email. You may notice there is no @ sign, and we replaced it with the period, and there is a trailing period.

Line 2 is the serial number, we use it to tell the name server about the file update time, so if you make a change to the zone data, you have to increment this number. The serial number has the format YYYYMMDDxx where xx is starting from 00.

Line 3 is the refresh rate in seconds. How often secondary DNS servers should query the primary server to check for updates.

Line 4 is the retry rate in seconds. This is the time that the secondary DNS server takes for waiting after trying to connect to the primary DNS server and cannot reach it. The specified number of retry seconds.

Line 5 is the expire directive. If the secondary server cannot connect to the primary server for an update, it should discard the value after the specified number of seconds.

Line 6 tells the caching servers can't connect to the primary DNS server; they wait before expiring an entry, this line defines the wait time.

NS: Name Server records

You can use the NS record to specify the name servers for a zone. The NS records are like this:

```
IN          NS          ns1.example.com.

IN          NS          ns2.example.com.
```

You don't have to create two NS records, but we prefer to have backup name servers.

A and AAAA: address records

The A record maps the hostname to an IP address:

```
support IN         A            192.168.1.5
```

If you have a host at support.example.com on address 192.168.1.5, you can type like the above example.

Note: we wrote the host without a period.

PTR: pointer records

The PTR record is for doing the reverse name resolution, you give an IP address, and it returns the hostname.

This is the opposite of what A record does.

```
192.168.1.5       IN            PTR        support.example.com.
```

Here we type the full hostname with the trailing period.

MX: Mail exchange records

The MX record tells about the **mail server** records.

```
example.com.    IN          MX        10          mail
```

The domain ends with a period; the number 10 is the importance of the mail server, if you have multiple mail servers, the lower number is the less important.

CNAME: Canonical Name Records

CNAME records are like shortcuts for hostnames.

Suppose you have a site that has a hostname of whatever-bignameis.example.com, and since the system is a web server, an alias of www or CNAME record can be created for the host.

So you can create a CNAME record to make the name www.example.com:

```
whatever-bignameis      IN          A                192.168.

www                     IN          CNAME            whatever
```

The first line tells the DNS server about the location of the alias; the second line creates the alias that points to www.

TXT records

You can put any text on TXT records like your contact information or any other information you want the people to know when they query your DNS server.

You can write TXT records like this:

```
example.com.    IN          TXT        " YOUR INFO GOES HERE"
```

Also, you can use the **RP record** to put the contact information.

```
example.com.    IN          RP          mail.example.com.
```

DNS TTL value

In **/etc/named.conf** on the top there is **$TTL** entry.

This entry informs BIND about the time to live value for each individual record.

It takes a value in seconds like **14400 seconds (4 hours)**, so the DNS servers will cache your zone up to four hours then will query your DNS server again.

You can lower the value, but the default value is fair unless you know what you are doing.

Catching configuration errors

When you write your zone files, maybe you forget a period or space or any other error.

You can diagnose your Linux DNS server errors from the log. The BIND service through errors in /var/log/messages, you can use the **tail command** to view real-time error log using -f option.

```
$tail -f /var/log/messages
```

So when you write a zone file or modify /etc/named.config and restart your service and it shows an error, you can easily identify the error from the log.

Host command

After you have successfully added or modified your records, you can use the host command to see if your host if resolved correctly.

If you give it a hostname, it will answer with the corresponding IP addresses.

```
$ host example.com
```

Also, you can perform reverse lookups.

```
$ host 192.168.1.5
```

You can check the **host and dig command**.

Whois command

You can use the whois command to get the domain owner's details.

Also, the owner's email addresses, and contact phone numbers.

```
$ whois example.com
```

The rndc command

You can use the rndc tool to manage the name server securely.

You can check the status of the Linux DNS server like this:

```
$ rndc status
```

Also, if you make a change to any of the zone files, you can reload the service without restart the named service.

```
$ rndc reload example.com
```

Here we reload the example.com zone file.

You can reload all zones like this:

```
$ rndc reload
```

Or maybe you add new zones or change the configuration of the service; you can reload the configuration like this:

```
$ rndc reconfig
```

Linux DNS resolver

We’ve seen how a Linux DNS server works and how to configure it. The other part is the client who is contacting the DNS server.

The client is the resolver; you can check the configuration file **/etc/resolv.conf**

On Debian based distros, you can check **/etc/resolvconf/resolv.conf.d/** directory.

The **/etc/resolv.conf** file contains the local DNS servers that the system uses.

The first line is for the default search domain, and the second line indicates the IP address of the name server.

You can use your own DNS server once your BIND service running, just type them in the resolver.conf file.

Working with the Linux DNS server is pretty easy. I hope you find the post useful and easy.