Several applications that make use of a network operate by opening and connecting to something called ports, which denote services such as data transfer, remote shell login, and so on. Several interesting management tasks can be performed on a network consisting of many machines. Shell scripts can be used to configure the nodes in a network, test the availability of machines, automate execution of commands at remote hosts, and so on. This chapter focuses on different recipes that introduce interesting tools or commands related to networking, and also how they can be used for solving different problems.

Setting up the network

Before digging through recipes based on networking, it is essential to have a basic understanding of setting up a network, terminologies, and commands for assigning IP address, adding routes, and so on. This recipe will give an overview of different commands used in GNU/Linux for networking and their usages from the basics.

Getting ready

A network interface is used to connect a machine to a network. Usually, Linux denotes network interfaces using names like eth0, eth1 (referring to Ethernet interfaces). Other interfaces, such as usb0, wlan0, and so on are available for USB network interfaces, wireless LAN respectively.

In this recipe, we will use these commands: `ifconfig`, `route`, `nslookup`, and `host`.

`ifconfig` is the command that is used to configure and display details about network interfaces, subnet mask, and so on. On a typical system, it should be available at `/sbin/ifconfig`.

How to do it...

1. List the current network interface configuration:

```
$ ifconfig
lo        Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6078 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6078 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
          RX bytes:634520 (634.5 KB)  TX bytes:634520 (634.5 KB)
```

```
wlan0     Link encap:EthernetHWaddr 00:1c:bf:87:25:d2
inet addr:192.168.0.82  Bcast:192.168.3.255  Mask:255.255.252.0
inet6addr: fe80::21c:bfff:fe87:25d2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:420917 errors:0 dropped:0 overruns:0 frame:0
          TX packets:86820 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
          RX bytes:98027420 (98.0 MB)  TX bytes:22602672 (22.6 MB)
```

The left-most column in the `ifconfig` output lists the names of network interfaces, and the right-hand columns show the details related to the corresponding network interface.

2. In order to manually set the IP address for a network interface, use:

   ```
   # ifconfig wlan0 192.168.0.80
   ```

   You will need to run the preceding command .as root. `192.168.0.80` is the address to be set,

   Set the subnet mask along with the IP address as follows:

   ```
   # ifconfig wlan0 192.168.0.80  netmask 255.255.252.0
   ```

3. Automatically configure network interfaces. If you are connecting to, let's say a wired network which supports automatically assigning IPs, just use this to configure the network interface:

   ```
   # dhclient eth0
   ```

There's more...

Let's go through a few more essential commands and their usage.

Printing the list of network interfaces

Here is a one-line command sequence to print the list of network interfaces available on a system:

```
$ ifconfig | cut -c-10 | tr -d ' ' | tr -s '\n'
lo
wlan0
```

The first 10 characters of each line in `ifconfig` output is reserved for writing names of network interfaces. Hence, we use `cut` to extract the first 10 characters of each line. `tr -d ' '` deletes every space character in each line. Now, the `\n` newline character is squeezed using `tr -s '\n'` to produce a list of interface names.