5. Connect your devices to the wireless network you just created with the following settings:
    - IP address: 10.99.66.56 (and so on)
    - Subnet mask: 255.255.0.0

To make this more convenient, you might want to install a DHCP and DNS server on your machine, so it's not necessary to configure IPs on devices manually. A handy tool for this is `dnsmasq` which you can use for performing both DHCP and DNS operations.

Basic firewall using iptables

A firewall is a network service which is used to filter network traffic for unwanted traffic, block it, and allow the desired traffic to pass. The most powerful tool on Linux is `iptables`, which has kernel integration in recent versions of the kernels.

How to do it...

`iptables` is present, by default, on all modern Linux distributions. We will see how to configure `iptables` for common scenarios.

1. Block traffic to a specific IP address:

```
#iptables -A OUTPUT -d 8.8.8.8 -j DROP
```

   If you run `PING 8.8.8.8` in another terminal before running the iptables command, you will see this:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=56 time=221 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=56 time=221 ms
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

   Here, the ping fails the third time because we used the `iptables` command to drop all traffic to `8.8.8.8`.

2. Block traffic to a specific port:

```
#iptables -A OUTPUT -p tcp -dport 21 -j DROP
$ ftp ftp.kde.org
ftp: connect: Connection timed out
```

How it works...

`iptables` is the standard command used for firewall on Linux. The first argument in `iptables` is `-A` which instructs `iptables` to append a new rule to the **chain** specified as the next parameter. A chain is simply a collection of rules, and in this recipe we have used the `OUTPUT` chain which runs on all the outgoing traffic.

In the first step, the `-d` parameter specifies the destination to match with the packet being sent. After that, we use the parameter `-j` to instruct `iptables` to `DROP` the packet.

Similarly, in the second one, we use the `-p` parameter to specify that this rule should match only `TCP` on the port specified with `-dport`. Using this we can block all the outbound `FTP` traffic.

There's more...

While playing with `iptables` commands, you might want to clear the changes made to the `iptables` chains. To do this, just use:

```
#iptables --flush
```