NonRootPortBinding

What you need

You need two things:

- Linux with “POSIX capabilities”;
- utilities to manipulate these capabilities.

Here, we will use:

- Linux - any distribution will do, as long as it has a recent kernel compiled with POSIX capabilities (so beware if you use a custom kernel);
- the setcap(8) and getcap(8) utilities.

We suppose that you have Apache installed, and that the httpd binary is `/usr/sbin/httpd`. Some distributions put it in another location (Debian, for instance, uses `/usr/sbin/apache2`).

First step: add capabilities to the httpd binary

The capabilities are added per file. This is why we need to modify the httpd binary itself. The capability we need to add is `CAP_NET_BIND_SERVICE`, which is explicitly defined as the capacity for an executable to bind to a port less than 1024.

You need to be root to do that, so first, be root. Then, add the capability to the httpd binary:

```
root@myhost # setcap cap_net_bind_service=+ep /usr/sbin/httpd
```

Check that the capability is added:

```
root@myhost # getcap /usr/sbin/httpd
/usr/sbin/httpd = cap_net_bind_service+ep
```

Second step: preparing your environment

NOTE: this section assumes a Fedora, Red Hat or derivate distribution. Adapt instructions below accordingly.

Login as root. Choose a user with which you want to run Apache (create one if needed). It can be a system only user (ie, no shell), but for testing purposes:

- we will use a regular user, named `test`,
- we will use the bundled package configuration.

Start by ensuring that httpd is NOT currently running (`service httpd status`). Then copy over the system apache configuration to the user's home (we use it since it binds to port 80 by default, which is what we want to test):

```
root@myhost # cp -a /etc/httpd ~test/
root@myhost # chown -R test ~test/httpd
```

Then login as `test`. Modify the environment:

```
# logs and run are symlinks to directories the test user cannot write to, remove the symlinks and create
directories instead
test@myhost $ cd httpd
test@myhost $ rm -f logs run
test@myhost $ mkdir logs run
```

Now, test that you can run it:

```
test@myhost $ pwd
/home/test/httpd
test@myhost $ httpd -d $(pwd) -DNO_DETACH
```

Apache will then run in the foreground. If it quits immediately, check the console output, or `logs/error_log`: fix errors (very probably a permission problem) and try again.

You're done!

Reverting

Use `setcap` again, to remove the capability:

```
root@myhost # setcap cap_net_bind_service=-ep /usr/sbin/httpd
```

Caveats

- with this setup, *any* nonprivileged user can now run Apache on privileged ports. So, be very careful about what you do. Additionally, you can further restrict execution of the httpd binary, either using standard credentials (`chmod`, `chown` et al) or, even better, ACLs;
- if you upgrade Apache, changes you have made to httpd will be lost, you'll need to do them again...

Alternative method (iptables/linux): NAT

You can use nat based method to redirect traffic from port 80 to 8080.

```
root@myhost # iptables -t nat -A PREROUTING -d <ip> -p tcp --dport 80 -m addrtype --dst-type LOCAL -j DNAT --to-
destination <ip>:8080
root@myhost # iptables -t nat -A OUTPUT     -d <ip> -p tcp --dport 80 -m addrtype --dst-type LOCAL -j DNAT --to-
destination <ip>:8080
```

Obviously the configuration of your apache server to listen on port 8080

NB. POSIX draft 1003.1e specifies a concept of permissions called "capabilities", but this draft has been withdrawn. However, that's the term used in most of the Linux documentation for this kernel feature.