Compiling and Installing

This document covers compilation and installation of the Apache HTTP Server on Unix and Unix-like systems only. For compiling and installation on Windows, see Using Apache HTTP Server with Microsoft Windows and Compiling Apache for Microsoft Windows. For other platforms, see the platform documentation.

Apache httpd uses `libtool` and `autoconf` to create a build environment that looks like many other Open Source projects.

If you are upgrading from one minor version to the next (for example, 2.4.8 to 2.4.9), please skip down to the upgrading section.

Overview for the impatient

Installing on Fedora/CentOS/Red Hat Enterprise Linux

```
sudo dnf install httpd
sudo service httpd start
```

Older releases of these distros use `yum` rather than `dnf`. See the Fedora project's documentation for platform-specific notes.

Installing on Ubuntu/Debian

```
sudo apt install apache2
sudo service apache2 start
```

See Ubuntu's documentation for platform-specific notes.

Installing from source

Download	Download the latest release from http://httpd.apache.org/download.cgi

Extract	`$ gzip -d httpd-NN.tar.gz`
	`$ tar xvf httpd-NN.tar`
	`$ cd httpd-NN`

Configure	`$ ./configure --prefix=PREFIX`
Compile	`$ make`
Install	`$ make install`
Customize	`$ vi PREFIX/conf/httpd.conf`
Test	`$ PREFIX/bin/apachectl -k start`

NN must be replaced with the current version number, and *PREFIX* must be replaced with the filesystem path under which the server should be installed. If *PREFIX* is not specified, it defaults to `/usr/local/apache2`.

Each section of the compilation and installation process is described in more detail below, beginning with the requirements for compiling and installing Apache httpd.

Don't see your favorite platform mentioned here? Come help us improve this doc.

Requirements

The following requirements exist for building Apache httpd:

APR and APR-Util

Make sure you have APR and APR-Util already installed on your system. If you don't, or prefer to not use the system-provided versions, download the latest versions of both APR and APR-Util from Apache APR, unpack them into `/httpd_source_tree_root/srclib/apr` and `/httpd_source_tree_root/srclib/apr-util` (be sure the directory names do not have version numbers; for example, the APR distribution must be under /httpd_source_tree_root/srclib/apr/) and use `./configure`'s `--with-included-apr` option. On some platforms, you may have to install the corresponding `-dev` packages to allow httpd to build against your installed copy of APR and APR-Util.

Perl-Compatible Regular Expressions Library (PCRE)

This library is required but not longer bundled with httpd. Download the source code from http://www.pcre.org, or install a Port or Package. If your build system can't find the pcre-config script installed by the PCRE build, point to it using the `--with-pcre` parameter. On some platforms, you may have to install the corresponding `-dev` package to allow httpd to build against your installed copy of PCRE.

Disk Space

Make sure you have at least 50 MB of temporary free disk space available. After installation the server occupies approximately 10 MB of disk space. The actual disk space requirements will vary considerably based on your chosen configuration options, any third-party modules, and, of course, the size of the web site or sites that you have on the server.

ANSI-C Compiler and Build System

Make sure you have an ANSI-C compiler installed. The GNU C compiler (GCC) from the Free Software Foundation (FSF) is recommended. If you don't have GCC then at least make sure your vendor's compiler is ANSI compliant. In addition, your `PATH` must contain basic build tools such as `make`.

Accurate time keeping

Elements of the HTTP protocol are expressed as the time of day. So, it's time to investigate setting some time synchronization facility on your system. Usually the `ntpdate` or `xntpd` programs are used for this purpose which are based on the Network Time Protocol (NTP). See the NTP homepage for more details about NTP software and public time servers.

Perl 5 [OPTIONAL]

For some of the support scripts like `apxs` or `dbmmanage` (which are written in Perl) the Perl 5 interpreter is required (versions 5.003 or newer are sufficient). If no Perl 5 interpreter is found by the `configure` script, you will not be able to use the affected support scripts. Of course, you will still be able to build and use Apache httpd.

Download

The Apache HTTP Server can be downloaded from the Apache HTTP Server download site, which lists several mirrors. Most users of Apache on unix-like systems will be better off downloading and compiling a source version. The build process (described below) is easy, and it allows you to customize your server to suit your needs. In addition, binary releases are often not up to date with the latest source releases. If you do download a binary, follow the instructions in the `INSTALL.bindist` file inside the distribution.

After downloading, it is important to verify that you have a complete and unmodified version of the Apache HTTP Server. This can be accomplished by testing the downloaded tarball against the PGP signature. Details on how to do this are available on the download page and an extended example is available describing the use of PGP.

Extract

Extracting the source from the Apache HTTP Server tarball is a simple matter of uncompressing, and then untarring:

```
$ gzip -d httpd-NN.tar.gz
$ tar xvf httpd-NN.tar
```

This will create a new directory under the current directory containing the source code for the distribution. You should `cd` into that directory before proceeding with compiling the server.

Configuring the source tree

The next step is to configure the Apache source tree for your particular platform and personal requirements. This is done using the script `configure` included in the root directory of the distribution. (Developers downloading an unreleased version of the Apache source tree will need to have `autoconf` and `libtool` installed and will need to run `buildconf` before proceeding with the next steps. This is not necessary for official releases.)

To configure the source tree using all the default options, simply type `./configure`. To change the default options, `configure` accepts a variety of variables and command line options.

The most important option is the location `--prefix` where Apache is to be installed later, because Apache has to be configured for this location to work correctly. More fine-tuned control of the location of files is possible with additional configure options.

Also at this point, you can specify which features you want included in Apache by enabling and disabling modules. Apache comes with a wide range of modules included by default. They will be compiled as shared objects (DSOs) which can be loaded or unloaded at runtime. You can also choose to compile modules statically by using the option `--enable-module=static`.

Additional modules are enabled using the `--enable-module` option, where *module* is the name of the module with the `mod_` string removed and with any underscore converted to a dash. Similarly, you can disable modules with the `--disable-module` option. Be careful when using these options, since `configure` cannot warn you if the module you specify does not exist; it will simply ignore the option.

In addition, it is sometimes necessary to provide the `configure` script with extra information about the location of your compiler, libraries, or header files. This is done by passing either environment variables or command line options to `configure`. For more information, see the `configure` manual page. Or invoke `configure` using the `--help` option.

For a short impression of what possibilities you have, here is a typical example which compiles Apache for the installation tree `/sw/pkg/apache` with a particular compiler and flags plus the two additional modules `mod_ldap` and `mod_lua`:

```
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-ldap=shared \
--enable-lua=shared
```

When `configure` is run it will take several minutes to test for the availability of features on your system and build Makefiles which will later be used to compile the server.

Details on all the different `configure` options are available on the `configure` manual page.

Build

Now you can build the various parts which form the Apache package by simply running the command:

```
$ make
```

Please be patient here, since a base configuration takes several minutes to compile and the time will vary widely depending on your hardware and the number of modules that you have enabled.

Install

Now it's time to install the package under the configured installation *PREFIX* (see `--prefix` option above) by running:

```
$ make install
```

This step will typically require root privileges, since *PREFIX* is usually a directory with restricted write permissions.

If you are upgrading, the installation will not overwrite your configuration files or documents.

Customize

Next, you can customize your Apache HTTP server by editing the configuration files under *PREFIX*`/conf/`.

```
$ vi PREFIX/conf/httpd.conf
```

Have a look at the Apache manual under *PREFIX*`/docs/manual/` or consult http://httpd.apache.org/docs/trunk/ for the most recent version of this manual and a complete reference of available configuration directives.

Test

Now you can start your Apache HTTP server by immediately running:

```
$ PREFIX/bin/apachectl -k start
```

You should then be able to request your first document via the URL `http://localhost/`. The web page you see is located under the `DocumentRoot`, which will usually be *PREFIX*`/htdocs/`. Then stop the server again by running:

```
$ PREFIX/bin/apachectl -k stop
```

Upgrading

The first step in upgrading is to read the release announcement and the file `CHANGES` in the source distribution to find any changes that may affect your site. When changing between major releases (for example, from 2.0 to 2.2 or from 2.2 to 2.4), there will likely be major differences in the compile-time and run-time configuration that will require manual adjustments. All modules will also need to be upgraded to accommodate changes in the module API.

Upgrading from one minor version to the next (for example, from 2.2.55 to 2.2.57) is easier. The `make install` process will not overwrite any of your existing documents, log files, or configuration files. In addition, the developers make every effort to avoid incompatible changes in the `configure` options, run-time configuration, or the module API between minor versions. In most cases you should be able to use an identical `configure` command line, an identical configuration file, and all of your modules should continue to work.

To upgrade across minor versions, start by finding the file `config.nice` in the `build` directory of your installed server or at the root of the source tree for your old install. This will contain the exact `configure` command line that you used to configure the source tree. Then to upgrade from one version to the next, you need only copy the `config.nice` file to the source tree of the new version, edit it to make any desired changes, and then run:

```
$ ./config.nice
$ make
$ make install
$ PREFIX/bin/apachectl -k graceful-stop
$ PREFIX/bin/apachectl -k start
```

You should always test any new version in your environment before putting it into production. For example, you can install and run the new version along side the old one by using a different `--prefix` and a different port (by adjusting the `Listen` directive) to test for any incompatibilities before doing the final upgrade.

You can pass additional arguments to `config.nice`, which will be appended to your original `configure` options:

```
$ ./config.nice --prefix=/home/test/apache --with-port=90
```

Third-party packages

A large number of third parties provide their own packaged distributions of the Apache HTTP Server for installation on particular platforms. This includes the various Linux distributions, various third-party Windows packages, Mac OS X, Solaris, and many more.

Our software license not only permits, but encourages, this kind of redistribution. However, it does result in a situation where the configuration layout and defaults on your installation of the server may differ from what is stated in the documentation. While unfortunate, this situation is not likely to change any time soon.

A description of these third-party distributions is maintained in the HTTP Server wiki, and should reflect the current state of these third-party distributions. However, you will need to familiarize yourself with your particular platform's package management and installation procedures.