Configure options

There are usually three steps when building an application from source: the configuration, the compilation, and the installation. The configuration step allows you to select a number of options that will not be **editable** after the program is built, as it has a direct impact on the project binaries. Consequently, it is a very important stage that you need to follow carefully if you want to avoid surprises later, such as the lack of a specific module or having configuration files located in a random folder.

The process consists of appending certain switches to the `configure` command that comes with the source code. The three types of switches that you can activate will be covered later, but let's first study the easiest way to proceed.

The easy way

If, for some reason, you do not want to bother with the configuration step, such as for testing purposes or simply because you will be recompiling the application in the future, you may simply use the `configure` command with no switches. Execute the following three commands to build and install a working version of Nginx, starting with the `configure` command:

Copy

```
[alex@example.com nginx-1.8.0]# ./configure
```

Running this command should initiate a long procedure of verifications to ensure that your system contains all of the necessary components. If the configuration process fails, check the prerequisites section again, as it is the most common cause of errors. For information about why the command failed, you may also refer to the `objs/autoconf.err` file, which provides a more detailed report. The `make` command will compile the application. This step should not cause any errors as long as the configuration went fine.

Copy

```
[alex@example.com nginx-1.8.0]# make
[root@example.com nginx-1.8.0]# make install
```

This last step will copy the compiled files as well as other resources to the installation directory, by default, `/usr/local/nginx`. You may need to be logged in as root to perform this operation, depending on permissions granted to the `/usr/local` directory.

Again, if you build the application without configuring it, you take the risk of missing out on a lot of features, such as the optional modules and others that we are about to discover.

Path options

When running the `configure` command, you are offered the possibility to enable some switches that let you specify the directory or file paths for a variety of elements. Note that the options offered by the configuration switches may change according to the version you downloaded. The options listed below are valid with the stable version, as of release 1.8.0. If you use another version, run the `./configure --help` command to list the available switches for your setup.

Using a switch typically consists of appending some text to the command line, for instance, using the `--conf-path` switch:

Copy

```
[alex@example.com nginx-1.8.0]# ./configure --conf-path=/etc/nginx/nginx.conf
```

Here is an exhaustive list of the configuration switches for configuring paths:

Switch	Usage	Default Value

`--prefix=…`	The base folder in which Nginx will be installed.	`/usr/local/nginx` Note: If you configure other switches using relative paths, they will connect to the base folder. For example, specifying `--conf-path=conf/nginx.conf` will result in your configuration file being found at `/usr/local/nginx/conf/nginx.conf`.
`--sbin-path=…`	The path where the Nginx binary file should be installed.	`<prefix>/sbin/nginx`.
`--conf-path=…`	The path of the main configuration file.	`<prefix>/conf/nginx.conf`.
`--error-log-path=…`	The location of your error log. Error logs can be configured very accurately in the configuration files. This path only applies in case you do not specify any error logging directive in your configuration.	`<prefix>/logs/error.log`.
`--pid-path=…`	The path of the Nginx `pid` file. You can specify the `pid` file path in the configuration file. If that's not the case, the value you specify for this switch will be used.	`<prefix>/logs/nginx.pid`. Note: The `pid` file is a simple text file containing the process identifier. It is placed in a predefined location so that other applications can easily find the `pid` of a running program.
`--lock-path=…`	The location of the lock file. Again, it can be specified in the configuration file, but if it isn't, this value will be used.	`<prefix>/logs/nginx.lock`. Note: The lock file allows other applications to determine whether or not the program is running. In the case of Nginx, it is used to make sure that the process is not started twice.
`--with-perl_modules_path=…`	Defines the path to the Perl modules. This switch must be defined if you want to include additional Perl modules.	
`--with-perl=…`	Path to the Perl binary file; used to execute Perl scripts. This path must be set if you want to allow execution of Perl scripts.	
`--http-log-path=…`	Defines the location of the access logs. This path is used only if the access log directive is unspecified in the configuration files.	`<prefix>/logs/access.log`.
`--http-client-body-temp-path=…`	Directory used for storing temporary files generated by client requests.	`<prefix>/client_body_temp`.
`--http-proxy-temp-path=…`	Location of the temporary files used by the proxy.	`<prefix>/proxy_temp`.

`--http-fastcgi-temp-path=…` `--http-uwsgi-temp-path=…` `--http-scgi-temp-path=…`	Location of the temporary files used by the HTTP FastCGI, uWSGI, and SCGI modules.	Respectively `<prefix>/fastcgi_temp` , `<prefix>/uwsgi_temp` , and `<prefix>/scgi_temp` .
`--builddir=…`	Location of the application build.	

Prerequisites options

Prerequisites come in the form of libraries and binaries. You should, by now, have them all installed on your system. However, even though they are present on your system, there may be occasions where the configuration script is unable to locate them. The reasons might be diverse, for example, if they were installed in non-standard directories.

In order to solve such problems, you are given the option to specify the path of prerequisites using the following switches (miscellaneous prerequisite-related options have been grouped together):

Compiler options	**Description**
`--with-cc=…`	Specifies an alternate location for the C compiler.
`--with-cpp=…`	Specifies an alternate location for the C preprocessor.
`--with-cc-opt=…`	Defines additional options to be passed to the C compiler command line.
`--with-ld-opt=…`	Defines additional options to be passed to the C linker command line.
`--with-cpu-opt=…`	Specifies a different target processor architecture among the following values: `pentium` , `pentiumpro` , `pentium3` , `pentium4` , `athlon` , `opteron` , `sparc32` , `sparc64` , and `ppc64` .
PCRE options	**Description**
`--without-pcre`	Disables usage of the PCRE library. This setting is not recommended, as it will remove support for regular expressions, consequently disabling the Rewrite module.
`--with-pcre`	Forces usage of the PCRE library.

`--with-pcre=…`	Allows you to specify the path of the PCRE library source code.
`--with-pcre-opt=…`	Additional options to build the PCRE library.
`--with-pcre-jit=…`	Build PCRE with JIT compilation support.
MD5 options	**Description**
`--with-md5=…`	Specifies the path to the MD5 library sources.
`--with-md5-opt=…`	Additional options to build the MD5 library.
`--with-md5-asm`	Uses assembler sources for the MD5 library.
SHA1 options	**Description**
`--with-sha1=…`	Specifies the path to the SHA1 library sources.
`--with-sha1-opt=…`	Additional options to build the SHA1 library.
`--with-sha1-asm`	Uses assembler sources for the SHA1 library.
zlib options	**Description**
`--with-zlib=…`	Specifies the path to the `zlib` library sources.
`--with-zlib-opt=…`	Additional options to build the `zlib` library.
`--with-zlib-asm=…`	Uses assembler optimizations for the target architectures `pentium` and `pentiumpro` .
OpenSSL options	**Description**

`--with-openssl=…`	Specifies the path of the OpenSSL library sources.
`--with-openssl-opt=…`	Additional options to build the OpenSSL library.
Libatomic	**Description**
`--with-libatomic=…`	Forces usage of the `libatomic_ops` library on systems other than `x86`, `amd64`, and `sparc`. This library allows Nginx to perform atomic operations directly instead of resorting to lock files. Depending on your system, it may result in a decrease in `SEGFAULT` errors and possibly higher request serving rate.
`--with-libatomic=…`	Specifies the path of the `Libatomic` library sources.

Module options

Modules, which will be detailed in Chapter 4, **Module Configuration**, need to be selected before compiling the application. Some are enabled by default and some need to be enabled manually, as you will see in the following table.

Modules enabled by default

The following switches allow you to disable modules that are enabled by default:

Modules enabled by default	Description
`--without-http_charset_module`	Disables the `Charset` module to re-encode web pages.
`--without-http_gzip_module`	Disables the `Gzip` compression module.
`--without-http_ssi_module`	Disables the `Server Side Include` module.
`--without-http_userid_module`	Disables the `User ID` module providing user identification via cookies.
`--without-http_access_module`	Disables the `Access` module allowing access configuration for IP address ranges.
`--without-http_auth_basic_module`	Disables the `Basic Authentication` module.
`--without-http_autoindex_module`	Disables the `Automatic Index` module.
`--without-http_geo_module`	Disables the `Geo` module allowing you to define variables depending on IP address ranges.
`--without-http_map_module`	Disables the `Map` module that allows you to declare map blocks.
`--without-http_referer_module`	Disables the `Referer control` module.
`--without-http_rewrite_module`	Disables the `Rewrite` module.
`--without-http_proxy_module`	Disables the `Proxy` module to transfer requests to other servers.

`--without-http_fastcgi_module`
`--without-http_uwsgi_module`
`--without-http_scgi_module`
	Disables the FastCGI, uWSGI, or SCGI modules to interact with FastCGI, uWSGI, or SCGI processes respectively.

`--without-http_memcached_module`	Disables the `Memcached` module to interact with the **memcache daemon**.
`--without-http_limit_conn_module`	Disables the `Limit Connections` module to restrict resource usage according to defined zones.

Modules enabled by default	Description
`--without-http_limit_req_module`	Disables the `Limit Requests` module allowing you to limit the number of requests per user.
`--without-http_empty_gif_module`	Disables the `Empty GIF` module to serve a blank GIF image from memory.
`--without-http_browser_module`	Disables the `Browser` module to interpret the `User Agent` string.
`--without-http_upstream_ip_hash_module`	Disables the `Upstream IP Hash` module providing the `ip_hash` directive in upstream blocks.
`--without-http_upstream_least_conn_module`	Disables the `Upstream Least Conn` module providing the `least_conn` directive in upstream blocks.
`--without-http_split_clients_module`	Disables the `Split Clients` module

Modules disabled by default

The following switches allow you to enable modules that are disabled by default:

Modules disabled by default	Description
`--with-http_ssl_module`	Enables the `SSL` module to serve pages over HTTPS.
`--with-http_realip_module`	Enables the `Real IP` module to read the real IP address from the request header data.
`--with-http_addition_module`	Enables the `Addition` module, which lets you append or prepend data to the response body.
`--with-http_xslt_module`	Enables the `XSLT` module to apply XSL transformations to XML documents. Note: You will need to install the `libxml2` and `libxslt` libraries on your system if you wish to compile these modules.
`--with-http_image_filter_module`	Enables the `Image Filter` module that lets you apply modification to images. Note: You will need to install the `libgd` library on your system if you wish to compile this module.
`--with-http_geoip_module`	Enables the `GeoIP` module to achieve geographic localization using MaxMind's GeoIP binary database. Note: You will need to install the `libgeoip` library on your system if you wish to compile this module.
`--with-http_sub_module`	Enables the `Substitution` module to replace text in web pages.
`--with-http_dav_module`	Enables the `WebDAV` module (Distributed Authoring and Versioning via Web).
`--with-http_flv_module`	Enables the `FLV` module for special handling of `.flv` (Flash video) files.
`--with-http_mp4_module`	Enables the `MP4` module for special handling of `.mp4` video files.
`--with-http_gzip_static_module`	Enables the `Gzip Static` module to send pre-compressed files.

Modules disabled by default	Description
`--with-http_random_index_module`	Enables the `Random Index` module to pick a random file as the directory index.
`--with-http_secure_link_module`	Enables the `Secure Link` module to check the presence of a keyword in the URL.
`--with-http_stub_status_module`	Enables the `Stub Status` module, which generates a server statistics and information page.
`--with-google_perftools_module`	Enables the `Google Performance Tools` module.
`--with-http_degradation_module`	Enables the `Degradation` module that controls the behavior of your server depending on current resource usage.
`--with-http_perl_module`	Enables the `Perl` module, allowing you to insert Perl code directly into your Nginx configuration files and to make Perl calls from SSI.
`--with-http_spdy_module`	Enables the `SPDY` module allowing clients to communicate with Nginx over the SPDY protocol.
`--with-http_gunzip_module`	Enables the `Gunzip` module, which offers to decompress a gzip-encoded response from a backend server before forwarding it to the client.
`--with-http_auth_request_module`	Enables the `Auth Reques` t module. This module allows you to delegate the HTTP authentication mechanism to a backend server via a subrequest. The status code of the response can be stored in a variable.

Miscellaneous options

Other options are available in the configuration script, for example, regarding the mail server proxy feature or event management. These have been enlisted as follows:

Mail server proxy options	Description
`--with-mail`	Enables `mail server proxy` module. Supports POP3, IMAP4, SMTP. It is disabled by default.
`--with-mail_ssl_module`	Enables SSL support for the mail server proxy. It is disabled by default.
`--without-mail_pop3_module`	Disables the `POP3` module for the mail server proxy. It is enabled by default when the mail server proxy module is enabled.
`--without-mail_imap_module`	Disables the `IMAP4` module for the mail server proxy. It is enabled by default when the mail server proxy module is enabled.
`--without-mail_smtp_module`	Disables the `SMTP` module for the mail server proxy. It is enabled by default when the mail server proxy module is enabled.
Event management:	Allows you to select the event notification system for the Nginx sequencer. For advanced users only.
`--with-rtsig_module`	Enables the `rtsig` module to use `rtsig` as event notification mechanism.

Mail server proxy options	Description
`--with-select_module`	Enables the `select` module to use select as event notification mechanism. By default, this module is enabled unless a better method is found on the system— `kqueue` , `epoll` , `rtsig` ,or `poll` .
`--without-select_module`	Disables the `select` module.
`--with-poll_module`	Enables the `poll` module to use poll as event notification mechanism. By default, this module is enabled if available, unless a better method is found on the system— `kqueue` , `epoll` ,or `rtsig` .
`--without-poll_module`	Disables the `poll` module.

User and group options	Description
`--user=...`	Default user account to start the Nginx worker processes. This setting is used only if you do not specify the user directive in the configuration file.
`--group=...`	Default user group to start the Nginx worker processes. This setting is used only if you do not specify the group directive in the configuration file.

Other options	Description
`--with-ipv6`	Enables IPv6 support.
`--without-http`	Disables the HTTP server.
`--without-http-cache`	Disables HTTP caching features.
`--add-module=PATH`	Adds a third-party module to the compile process by specifying its path. This switch can be repeated indefinitely if you wish to compile multiple modules.
`--with-debug`	Enables additional debugging information to be logged.
`--with-file-aio`	Enables support for AIO (Asynchronous IO disk operations).

Configuration examples

Here are a few examples of configuration commands that may be used for various cases. In these examples, the path switches have been omitted, as they are specific to each system and leaving the default values may simply function correctly.

Note

Be aware that these configurations do not include additional third-party modules. Please refer to Chapter 4, **Module Configuration**, for more information about installing add-ons.

About the prefix switch

During the configuration, you should take particular care of the `--prefix` switch. Many of the future configuration directives (that will be approached in further chapters) will be based on the path you select at this point. While it is not a definitive problem since absolute paths can still be employed, you should know that the prefix cannot be changed once the binaries have been compiled.

There is also another issue that you may run into if you plan to keep up with the times and update Nginx as new versions are released. The default prefix (if you do not override the setting by using the `--prefix` switch) is `/usr/local/nginx`. This is a path that does not include the version number. Consequently, when you upgrade Nginx, if you do not specify a different prefix, the new install files will override the previous ones, which among other problems, could potentially erase your currently running binaries.

It is thus recommended to use a different prefix for each version you will be using. Use the following command to specify a prefix that is specific to version 1.8.0:

Copy

```
./configure --prefix=/usr/local/nginx-1.8.0
```

Additionally, to make future changes simpler, you may create a symbolic link `/usr/local/nginx` pointing to `/usr/local/nginx-1.8.0`. Once you upgrade, you can update the link to make it point to `/usr/local/nginx-newer.version`. This will allow the `init` script to always make use of the latest installed version of Nginx.

Regular HTTP and HTTPS servers

The first example describes a situation where the most important features and modules to serve HTTP and HTTPS content are enabled, and the mail-related options are disabled:

Copy

```
./configure --user=www-data --group=www-data --with-http_ssl_module --with-http_realip_module
```

As you can see, the command is rather simple and most switches have been left out. The reason for this is that the default configuration is rather efficient, and most of the important modules are enabled. You will only need to include the `http_ssl` module to serve HTTPS content, and optionally, the **real IP** module to retrieve your visitors' IP addresses in case you are running Nginx as backend server.

All modules enabled

The next situation includes all available modules, and it is up to you whether you want to use them or not at runtime:

Copy

```
./configure --user=www-data --group=www-data --with-http_ssl_module --with-http_realip_module --with-http_addition_module --with-h
```

This configuration opens up a wide range of possible configuration options. Chapter 3, **HTTP Configuration**, to Chapter 6, **Apache and Nginx Together**, provide more detailed information on module configuration.

With this setup, all optional modules are enabled, thus requiring additional libraries to be installed— `libgeoip` for the Geo IP module, `libgd` for the Image Filter module, `libxml2`, and `libxslt` for the XSLT module. You may install those prerequisites using your system package manager, for example, by running `yum install libxml2` or `apt-get install libxml2`.

Mail server proxy

This last build configuration is somewhat special, as it is dedicated to enabling mail server proxy features—a darker and less documented side of Nginx. You can enable all mail related features through the following command:

Copy

```
./configure --user=www-data --group=www-data --with-mail --with-mail_ssl_module
```

If you wish to completely disable the HTTP serving features and only dedicate Nginx to mail proxying, you may add the `--without-http` switch.

> **Note**
>
> Note that in the preceding commands, the user and group used to run the Nginx worker processes will be `www-data`, which implies that this user and group must exist on your system.

Build configuration issues

In some cases, the `configure` command may fail—after a long list of checks, you may receive a few error messages on your terminal. In most (if not all) cases, these errors are related to missing prerequisites or unspecified paths.

In such cases, proceed with the following verifications carefully to make sure you have all it takes to compile the application, and optionally, consult the `objs/autoconf.err` file for more details about the compilation problem. This file is generated during the configure process and will tell you exactly which part of the process failed.

Make sure you installed the prerequisites

There are basically four main prerequisites: GCC, PCRE, zlib, and OpenSSL. The last three are libraries that must be installed in two packages: the library itself and its development sources. Make sure you have installed both for each of them. Refer to the prerequisites section at the beginning of this chapter for additional information. Note that other prerequisites such as LibXML2 or LibXSLT may be required to enable extra modules (for example, in the case of the HTTP XSLT module).

If you are positive that all of the prerequisites were installed correctly, perhaps the issue comes from the fact that the `configure` script is unable to locate the prerequisite files. In that case, make sure that you include the configuration switches related to file paths, as described earlier.

For example, the following switch allows you to specify the location of the OpenSSL library files:

Copy

```
./configure [...] --with-openssl=/usr/lib64
```

The OpenSSL library file will be looked for in the specified folder.

Directories exist and must be writable

Always remember to check the obvious; everyone makes even the simplest of mistakes sooner or later. Make sure that the directory you placed the Nginx files in has **read and write** permissions for the user running the configuration and compilation scripts. Also ensure that all paths specified in the `configure` script switches are existing, valid paths.

Compiling and installing the program

The configuration process is of utmost importance—it generates a makefile for the application depending on the selected switches and performs a long list of requirement checks on your system. Once the `configure` script is successfully executed, you can proceed with compiling Nginx.

Compiling the project equates to executing the `make` command in the project source directory:

Copy

```
[alex@example.com nginx-1.8.0]$ make
```

A successful build should result in the appearance of a final message: `make[1]: leaving directory`. This should be followed by the project source path.

Again, problems might occur at compile time. Most of these problems can originate in missing prerequisites or the specification of invalid paths. If this occurs, run the `configure` command again and triple-check the switches and all of the prerequisite options. It may also so happen that you downloaded an overly recent version of the prerequisites, which may not be backwards compatible. In such cases, the

best option is to visit the official website of the missing component and download an older version.

If the compilation process was successful, you are ready for the next step: installing the application. The following command must be executed with root privileges:

Copy

```
[root@example.com nginx-1.8.0]# make install
```

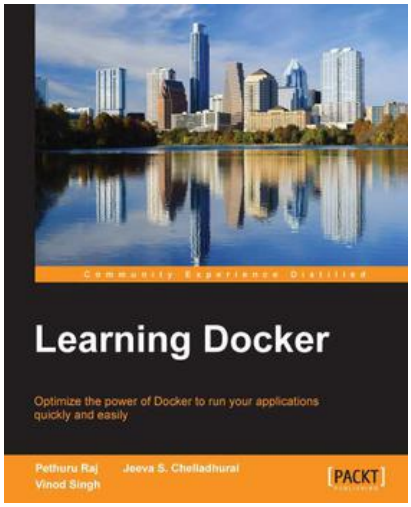
The `make install` command executes the `install` section of the `make` file. In other words, it performs a few simple operations, such as copying binaries and configuration files to the specified `install` folder. It also creates directories to store log and HTML files if these do not already exist. The `make install` step is not generally a source of problems, unless your system encounters an exceptional error, such as a lack of storage space or memory.

Note

You may require root privileges to install the application in the `/usr/local/` folder, depending on the folder permissions.

Next Section (/mapt/book/networking-and-servers/9781785280337/1/ch01lvl1sec11/controlling-the-nginx-service)

Related titles



(/mapt/book/virtualization_and_cloud/9781784397937)

(/mapt/book/application_development/9781782168454)