

Apache Cheatsheet

By Dejan Panovski • Updated on Oct 10, 2026

Apache HTTP Server at a glance: service commands, file layout on Ubuntu and RHEL, virtual hosts, modules, .htaccess, redirects, TLS, reverse proxy, and log formats.

Apache is packaged as `apache2` on Ubuntu and Debian and as `httpd` on Fedora and RHEL, with different file layouts and helper tools. This cheatsheet covers service and `apachectl` commands, configuration paths on both families, virtual hosts, modules, directory access and `.htaccess`, redirects, TLS, reverse proxy settings, access control, caching, logging, and common startup errors.

Service and CLI Commands

Use `apache2` on Ubuntu and Debian, `httpd` on Fedora and RHEL.

<code>sudo systemctl reload apache2</code>	Apply a new configuration with a graceful restart
<code>sudo systemctl restart apache2</code>	Stop and start the service
<code>sudo systemctl start apache2</code>	Start the service
<code>sudo systemctl stop apache2</code>	Stop the service
<code>sudo systemctl status apache2</code>	Show the service state and recent log lines
<code>sudo systemctl enable --now httpd</code>	Start now and at every boot on Fedora and RHEL
<code>sudo apachectl configtest</code>	Test the configuration for syntax errors
<code>sudo apachectl -t -D DUMP_INCLUDES</code>	List every configuration file that is loaded
<code>sudo apachectl graceful</code>	Restart workers after current requests finish
<code>sudo apachectl graceful-stop</code>	Stop after current requests finish
<code>apachectl -v</code>	Print the version
<code>apachectl -V</code>	Print the version, MPM, and build settings

Run `configtest` before every reload. On Ubuntu and Debian, call `apache2ctl` or `apachectl` rather than the `apache2` binary, which needs the variables from `/etc/apache2/envvars`. On Fedora and RHEL, use `sudo httpd -t -D DUMP_INCLUDES`, `httpd -v`, and `httpd -V` for these diagnostic options; the packaged `apachectl` does not forward arbitrary arguments to `httpd`.

Configuration Layout

The two package families lay out the configuration differently.

<code>/etc/apache2/apache2.conf</code>	Main file on Ubuntu and Debian
<code>/etc/apache2/ports.conf</code>	Listen directives on Ubuntu and Debian
<code>/etc/apache2/sites-available/</code>	Virtual host files on Ubuntu and Debian
<code>/etc/apache2/mods-available/</code>	Module load and config snippets
<code>/etc/apache2/conf-available/</code>	Global config snippets
<code>/etc/httpd/conf/httpd.conf</code>	Main file on Fedora and RHEL
<code>/etc/httpd/conf.d/*.conf</code>	Virtual hosts and snippets on Fedora and RHEL
<code>/etc/httpd/conf.modules.d/</code>	Module loading on Fedora and RHEL
<code>/var/www/html</code>	Default document root on both
<code>/var/log/apache2/</code>	Logs on Ubuntu and Debian
<code>/var/log/httpd/</code>	Logs on Fedora and RHEL

Apache runs as `www-data` on Ubuntu and Debian and as `apache` on Fedora and RHEL. Give that user read access to the document root.

Sites, Modules, and Config Snippets

Ubuntu and Debian helpers that manage the `*-enabled` symlinks. Before switching MPMs, migrate to PHP-FPM and disable `mod_php` if it is enabled.

<code>sudo a2ensite example.com</code>	Enable <code>example.com.conf</code>
<code>sudo a2dissite 000-default</code>	Disable the default site
<code>sudo a2enmod rewrite</code>	Enable a module
<code>sudo a2dismod autoindex</code>	Disable a module
<code>sudo a2enconf servername</code>	Enable <code>servername.conf</code>
<code>sudo a2disconf serve-cgi-bin</code>	Disable a config snippet
<code>sudo apachectl -M</code>	List loaded modules
<code>sudo apachectl -S</code>	List virtual hosts and which one is the default
<code>sudo a2dismod mpm_prefork && sudo a2enmod mpm_event</code>	Switch to event; test and restart afterward

Site files must end in `.conf`. After an MPM change, run `configtest` and restart the service. Fedora and RHEL use `conf.d/` for snippets and `LoadModule` lines in `conf.modules.d/`. Inspect modules with `sudo httpd -M` and virtual hosts with `sudo httpd -S`.

Virtual Hosts

Directives that decide which site answers a request.

<code><VirtualHost *:80> ... </VirtualHost></code>	Define a site on port 80
<code>ServerName example.com</code>	Primary host name for the site
<code>ServerAlias www.example.com *.example.com</code>	Additional host names
<code>DocumentRoot /var/www/example.com/public_html</code>	Directory served for the site
<code>DirectoryIndex index.html index.php</code>	File served when a directory is requested
<code>ServerAdmin webmaster@example.com</code>	Address shown on some error pages
<code>ErrorLog \${APACHE_LOG_DIR}/example.com-error.log</code>	Per-site error log on Ubuntu and Debian
<code>CustomLog \${APACHE_LOG_DIR}/example.com-access.log combined</code>	Per-site access log
<code>Listen 8080</code>	Accept connections on another port

When no `ServerName` or `ServerAlias` matches, Apache serves the first virtual host defined for that address and port. On Ubuntu and Debian that is usually `000-default.conf`, because files load in alphabetical order.

Directory Access and .htaccess

Control what Apache may serve from a directory and what `.htaccess` may change.

<code><Directory /var/www/example.com> ... </Directory></code>	Apply settings to a directory tree
<code>Require all granted</code>	Allow access
<code>Require all denied</code>	Deny access
<code>Options -Indexes</code>	Turn off directory listings
<code>Options +FollowSymLinks</code>	Follow symbolic links
<code>AllowOverride None</code>	Ignore <code>.htaccess</code> files
<code>AllowOverride All</code>	Let <code>.htaccess</code> override any directive
<code>AllowOverride FileInfo AuthConfig</code>	Allow rewrites and auth only
<code><FilesMatch "\.(env ini log)\$"> Require all denied </FilesMatch></code>	Block sensitive file types

`.htaccess` is read on every request in every parent directory, which costs performance. Put rules in the virtual host when you control the server, and keep `AllowOverride None` where `.htaccess` is not needed.

Redirects and Rewrites

`Redirect` comes from `mod_alias`. The `Rewrite*` directives need `a2enmod rewrite`.

<code>Redirect permanent / https://example.com/</code>	301 redirect that keeps the path
<code>Redirect 302 /old /new</code>	Temporary redirect for one path
<code>RedirectMatch 301 ^/blog/(.*)\$ /news/\$1</code>	Regex redirect that keeps the tail
<code>RewriteEngine On</code>	Enable the rewrite engine for this context
<code>RewriteCond %{HTTPS} off</code>	Match only plain HTTP requests
<code>RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]</code>	Redirect to HTTPS
<code>RewriteCond %{HTTP_HOST} ^www\.(.+)\$ [NC]</code>	Match the www host, case-insensitive
<code>RewriteRule ^ https://%1%{REQUEST_URI} [L,R=301]</code>	Redirect to the bare domain
<code>RewriteCond %{REQUEST_FILENAME} !-f</code>	Continue only when no file exists
<code>RewriteRule ^ index.php [L]</code>	Front controller in <code>.htaccess</code> or <code><Directory></code>

Prefer `Redirect` for plain redirects. In `.htaccess`, patterns omit the leading slash: `^old/` instead of `^/old/`. Use the final two rows together only in `.htaccess` or `<Directory>` context, where `REQUEST_FILENAME` is a filesystem path.

HTTPS and TLS

Terminate TLS on port 443. Enable mod_ssl with `sudo a2enmod ssl` on Ubuntu and Debian, or install the `mod_ssl` package on Fedora and RHEL.

<code><VirtualHost *:443> ... # # </VirtualHost></code>	Define the TLS site
<code>SSLEngine on</code>	Turn on TLS for the virtual host
<code>SSLCertificateFile /etc/letsencrypt/live/example.com/fullchain.pem</code>	Certificate and intermediate chain
<code>SSLCertificateKeyFile /etc/letsencrypt/live/example.com/privkey.pem</code>	Private key
<code>SSLProtocol -all +TLSv1.2 +TLSv1.3</code>	Allow only modern protocol versions
<code>Protocols h2 http/1.1</code>	Enable HTTP/2, needs mod_http2
<code>Header always set Strict-Transport-Security "max-age=63072000"</code>	Send HSTS, needs mod_headers
<code>sudo certbot --apache -d example.com -d www.example.com</code>	Issue and install a certificate
<code>sudo a2ensite default-ssl</code>	Enable the bundled self-signed TLS site

The prefork MPM severely restricts HTTP/2 concurrency. Prefer `mpm_event` ; if mod_php is enabled, migrate PHP to PHP-FPM and disable mod_php before switching MPMS.

Reverse Proxy

Forward requests to an application. Enable the modules with `sudo a2enmod proxy proxy_http headers` .

<code>ProxyPass / http://127.0.0.1:3000/</code>	Forward requests to a local application
<code>ProxyPassReverse / http://127.0.0.1:3000/</code>	Rewrite Location headers in redirects
<code>ProxyPreserveHost On</code>	Pass the original Host header
<code>RequestHeader set X-Forwarded-Proto "https"</code>	Tell the application TLS was used
<code>ProxyPass /ws ws://127.0.0.1:3000/ws</code>	Proxy WebSockets, needs mod_proxy_wstunnel
<code>ProxyPass / http://127.0.0.1:3000/ upgrade=websocket</code>	WebSocket upgrade in mod_proxy_http on 2.4.47+
<code>ProxyPass /static !</code>	Exclude a path from proxying, list it first
<code>ProxyTimeout 300</code>	Wait longer for a slow backend
<code>ProxyRequests Off</code>	Keep forward proxying disabled

mod_proxy adds `X-Forwarded-For` , `X-Forwarded-Host` ,and `X-Forwarded-Server` automatically. Order matters: Apache uses the first `ProxyPass` that matches, so put specific paths before `/` .

Access Control and Hardening

Limit who reaches a path and what the server reveals. Use `htpasswd -c` only for a new file, because it overwrites an existing one.

<code>Require ip 10.0.0.0/8</code>	Allow a network
<code>Require not ip 203.0.113.5</code>	Block an address inside <RequireAll>
<code>AuthType Basic</code>	Turn on HTTP basic authentication
<code>AuthName "Restricted"</code>	Realm shown in the login prompt
<code>AuthUserFile /etc/apache2/.htpasswd</code>	Point to the password file
<code>Require valid-user</code>	Accept any user in the file
<code>sudo htpasswd -c /etc/apache2/.htpasswd admin</code>	Create a new password file
<code>sudo htpasswd /etc/apache2/.htpasswd editor</code>	Add or update a user
<code>LimitRequestBody 67108864</code>	Cap request bodies at 64 MB
<code>ServerTokens Prod</code>	Send only Apache in the Server header
<code>ServerSignature Off</code>	Remove the version footer from error pages
<code>TraceEnable Off</code>	Disable the TRACE method

On Ubuntu and Debian, `ServerTokens`, `ServerSignature`, and `TraceEnable` live in `/etc/apache2/conf-available/security.conf`.

Compression and Caching

Cut response size with `mod_deflate` and set browser caching with `mod_expires` and `mod_headers`.

<code>sudo a2enmod deflate expires headers</code>	Enable the modules on Ubuntu and Debian
<code>AddOutputFilterByType DEFLATE text/html text/css application/javascript</code>	Compress these types
<code>AddOutputFilterByType DEFLATE application/json image/svg+xml</code>	Compress JSON and SVG
<code>ExpiresActive On</code>	Turn on expiry headers
<code>ExpiresByType image/webp "access plus 1 month"</code>	Cache images for a month
<code>ExpiresByType text/css "access plus 1 year"</code>	Cache CSS for a year inside the asset block below
<code><Directory /var/www/assets> Header set Cache-Control "public, max-age=31536000, immutable" </Directory></code>	Cache fingerprinted public assets in this directory
<code>Header unset ETag</code>	Drop ETags when a CDN or long expiry handles caching
<code>FileETag None</code>	Stop generating ETags

The directory must contain only fingerprinted public assets. Keep the one-year CSS expiry and ETag settings in this block, away from HTML and application responses. Do not compress images, archives, or video that are already compressed.

Logging

Choose what gets recorded and where to watch it.

<code>ErrorLog</code> <code>#{APACHE_LOG_DIR}/error.log</code>	Set the error log file
<code>LogLevel warn</code>	Set the error log level
<code>LogLevel info ssl:warn</code> <code>rewrite:trace3</code>	Set levels per module
<code>CustomLog</code> <code>#{APACHE_LOG_DIR}/access.log combined</code>	Write access logs in the combined format
<code>LogFormat "%h %t \"%r\" %>s %b %D" timed</code>	Define a custom format named <code>timed</code>
<code>SetEnvIf Request_URI</code> <code>"/health\$" nolog</code>	Mark health checks
<code>CustomLog</code> <code>#{APACHE_LOG_DIR}/access.log combined</code> <code>env=!nolog</code>	Skip marked requests
<code>sudo tail -f</code> <code>/var/log/apache2/error.log</code>	Follow errors live
<code>sudo journalctl -u</code> <code>apache2 -f</code>	Follow service-level messages

Error log levels run `emerg` , `alert` , `crit` , `error` , `warn` , `notice` , `info` , `debug` ,and `trace1` to `trace8` .Use `rewrite:trace3` to debug rewrite rules, and turn it off afterward because it logs every request.

Log Format Strings

Placeholders for `LogFormat` , the Apache counterpart to nginx variables.

<code>%h</code>	Client address
<code>%a</code>	Client IP, after <code>mod_remoteip</code> when it is enabled
<code>%u</code>	Authenticated user
<code>%t</code>	Time the request was received
<code>%r</code>	First line of the request
<code>%>s</code>	Final status code
<code>%b</code>	Response size in bytes, – for zero
<code>%D</code>	Time to serve the request in microseconds
<code>%T</code>	Time to serve the request in seconds
<code>%v</code>	Canonical <code>ServerName</code> of the virtual host
<code>#{Referer}i</code>	Referer request header
<code>#{User-Agent}i</code>	User-Agent request header
<code>#{X-Forwarded-For}i</code>	Forwarding chain from a proxy

The combined format is `%h %l %u %t \"%r\" %>s %b \"#{Referer}i\" \"#{User-Agent}i\"` .

Common Errors

Startup and request errors you will see in the error log or `systemctl status` .

<code>AH00558: Could not reliably determine the server's fully qualified domain name</code>	Add <code>ServerName localhost</code> in a config snippet
<code>AH00072: make_sock: could not bind to address [::]:80</code>	Another process holds the port; find it with <code>sudo ss -tlnp 'sport = :80'</code>
<code>Invalid command 'RewriteEngine'</code>	Enable the module with <code>sudo a2enmod rewrite</code>
<code>Invalid command 'ProxyPass'</code>	Enable proxy and <code>proxy_http</code>
<code>AH01630: client denied by server configuration</code>	Add <code>Require all granted</code> to the <code><Directory></code> block
<code>403 Forbidden with correct Require</code>	Fix filesystem permissions; Apache needs execute on every parent directory
<code>.htaccess rules ignored</code>	Set <code>AllowOverride All</code> for that directory
<code>500 after editing .htaccess</code>	Check the error log for the bad directive

AH00558 is only a warning. Apache still starts, but the message appears on every reload until `ServerName` is set.

Firewall

Open HTTP and HTTPS after installing Apache.

<code>sudo ufw allow 'Apache Full'</code>	Open ports 80 and 443 with UFW
<code>sudo ufw allow 'Apache'</code>	Open port 80 only
<code>sudo ufw allow 'Apache Secure'</code>	Open port 443 only
<code>sudo ufw app info 'Apache Full'</code>	Show the ports in a profile
<code>sudo firewall-cmd --permanent --add-service={http,https}</code>	Open both ports with firewalld
<code>sudo firewall-cmd --reload</code>	Apply the firewalld change

Related Guides

Use these guides for the longer explanations behind these commands.

Apache Commands: Manage and Troubleshoot Your Web Server	Service, module, and diagnostic commands in detail
How to Start, Stop, and Restart Apache on Linux	Reload versus restart on each distribution
How to Install Apache on Ubuntu 26.04	Installation, firewall rules, and first steps
How to Set Up Apache Virtual Hosts on Ubuntu	Hosting several sites on one server
Redirect HTTP to HTTPS in Apache	Redirect and rewrite patterns
How to Force HTTPS using .htaccess	HTTPS redirects when you cannot edit the virtual host
Secure Apache with Let's Encrypt	Certbot setup and renewal
Configuring the Apache Error and Access Logs	Log formats, levels, and rotation
Nginx Cheatsheet	The same reference for nginx