# **WordPress**

**WordPress** (**WordPress.org**) is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source" \o "Free and open-source) [content management system](https://en.wikipedia.org/wiki/Content_management_system" \o "Content management system) (CMS) written in [PHP](https://en.wikipedia.org/wiki/PHP" \o "PHP)[[4]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-4) and paired with a [MySQL](https://en.wikipedia.org/wiki/MySQL" \o "MySQL) or [MariaDB](https://en.wikipedia.org/wiki/MariaDB" \o "MariaDB) database. Features include a [plugin architecture](https://en.wikipedia.org/wiki/Plug-in_(computing)" \o "Plug-in (computing)) and a [template system](https://en.wikipedia.org/wiki/Web_template_system" \o "Web template system), referred to within WordPress as Themes. WordPress was originally created as a [blog-publishing system](https://en.wikipedia.org/wiki/Blog" \o "Blog) but has evolved to support other types of web content including more traditional [mailing lists](https://en.wikipedia.org/wiki/Electronic_mailing_list" \o "Electronic mailing list) and [forums](https://en.wikipedia.org/wiki/Internet_forum" \o "Internet forum), media galleries, membership sites, [learning management systems](https://en.wikipedia.org/wiki/Learning_management_system" \o "Learning management system) (LMS) and [online stores](https://en.wikipedia.org/wiki/Shopping_cart_software" \o "Shopping cart software). WordPress is used by more than 60 million websites,[[5]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-5) including 33.6% of the top 10 million websites as of April 2019,[[6]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-Usage_of_content_management_systems_for_websites-6)[[7]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-7) WordPress is one of the most popular content management system solutions in use.[[8]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-8) WordPress has also been used for other application domains such as [pervasive display systems](https://en.wikipedia.org/wiki/Pervasive_display_systems" \o "Pervasive display systems) (PDS).[[9]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-9)

WordPress was released on May 27, 2003, by its founders, American developer [Matt Mullenweg](https://en.wikipedia.org/wiki/Matt_Mullenweg" \o "Matt Mullenweg)[[1]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-release-1) and English developer [Mike Little](https://en.wikipedia.org/wiki/Mike_Little" \o "Mike Little),[[10]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-release_commit-10)[[11]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-11) as a [fork](https://en.wikipedia.org/wiki/Fork_(software_development)" \o "Fork (software development)) of *b2/cafelog*. The software is released under the [GPLv2](https://en.wikipedia.org/wiki/GNU_General_Public_License" \l "Version_2" \o "GNU General Public License) (or later) license.[[12]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-12)

To function, WordPress has to be installed on a [web server](https://en.wikipedia.org/wiki/Web_server" \o "Web server), either part of an [Internet hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service" \o "Internet hosting service) like [WordPress.com](https://en.wikipedia.org/wiki/WordPress.com" \o "WordPress.com) or a computer running the software package WordPress.org in order to serve as a [network host](https://en.wikipedia.org/wiki/Host_(network)" \o "Host (network)) in its own right.[[13]](https://en.wikipedia.org/wiki/WordPress" \l "cite_note-13) A local computer may be used for



# How to use this image

$ docker run --name some-wordpress --network some-network -d wordpress

The following environment variables are also honored for configuring your WordPress instance:

* -e WORDPRESS\_DB\_HOST=...
* -e WORDPRESS\_DB\_USER=...
* -e WORDPRESS\_DB\_PASSWORD=...
* -e WORDPRESS\_DB\_NAME=...
* -e WORDPRESS\_TABLE\_PREFIX=...
* -e WORDPRESS\_AUTH\_KEY=..., -e WORDPRESS\_SECURE\_AUTH\_KEY=..., -e WORDPRESS\_LOGGED\_IN\_KEY=..., -e WORDPRESS\_NONCE\_KEY=..., -e WORDPRESS\_AUTH\_SALT=..., -e WORDPRESS\_SECURE\_AUTH\_SALT=..., -e WORDPRESS\_LOGGED\_IN\_SALT=..., -e WORDPRESS\_NONCE\_SALT=... (default to unique random SHA1s, but only if other environment variable configuration is provided)
* -e WORDPRESS\_DEBUG=1 (defaults to disabled, non-empty value will enable WP\_DEBUG in wp-config.php)
* -e WORDPRESS\_CONFIG\_EXTRA=... (defaults to nothing, non-empty value will be embedded verbatim inside wp-config.php -- especially useful for applying extra configuration values this image does not provide by default such as WP\_ALLOW\_MULTISITE; see [docker-library/wordpress#142](https://github.com/docker-library/wordpress/pull/142) for more details)

If the WORDPRESS\_DB\_NAME specified does not already exist on the given MySQL server, it will be created automatically upon startup of the wordpress container, provided that the WORDPRESS\_DB\_USER specified has the necessary permissions to create it.

If you'd like to be able to access the instance from the host without the container's IP, standard port mappings can be used:

$ docker run --name some-wordpress -p 8080:80 -d wordpress

Then, access it via http://localhost:8080 or http://host-ip:8080 in a browser.

If you'd like to use an external database instead of a mysql container, specify the hostname and port with WORDPRESS\_DB\_HOST along with the password in WORDPRESS\_DB\_PASSWORD and the username in WORDPRESS\_DB\_USER (if it is something other than root):

$ docker run --name some-wordpress -e WORDPRESS\_DB\_HOST=10.1.2.3:3306 \

-e WORDPRESS\_DB\_USER=... -e WORDPRESS\_DB\_PASSWORD=... -d wordpress

When running WordPress with TLS behind a reverse proxy such as NGINX which is responsible for doing TLS termination, be sure to set X-Forwarded-Proto appropriately (see ["Using a Reverse Proxy" in "Administration Over SSL" in upstream's documentation](https://wordpress.org/support/article/administration-over-ssl/" \l "using-a-reverse-proxy)). No additional environment variables or configuration should be necessary (this image automatically adds the noted HTTP\_X\_FORWARDED\_PROTO code to wp-config.php if any of the above-noted environment variables are specified).

If your database requires SSL, [WordPress ticket #28625](https://core.trac.wordpress.org/ticket/28625) has the relevant details regarding support for that with WordPress upstream. As a workaround, [the "Secure DB Connection" plugin](https://wordpress.org/plugins/secure-db-connection/) can be extracted into the WordPress directory and the appropriate values described in the configuration of that plugin added in wp-config.php.

## Docker Secrets

As an alternative to passing sensitive information via environment variables, \_FILE may be appended to the previously listed environment variables, causing the initialization script to load the values for those variables from files present in the container. In particular, this can be used to load passwords from Docker secrets stored in /run/secrets/<secret\_name> files. For example:

$ docker run --name some-wordpress -e WORDPRESS\_DB\_PASSWORD\_FILE=/run/secrets/mysql-root ... -d wordpress:tag

Currently, this is supported for WORDPRESS\_DB\_HOST, WORDPRESS\_DB\_USER, WORDPRESS\_DB\_PASSWORD, WORDPRESS\_DB\_NAME, WORDPRESS\_AUTH\_KEY, WORDPRESS\_SECURE\_AUTH\_KEY, WORDPRESS\_LOGGED\_IN\_KEY, WORDPRESS\_NONCE\_KEY, WORDPRESS\_AUTH\_SALT, WORDPRESS\_SECURE\_AUTH\_SALT, WORDPRESS\_LOGGED\_IN\_SALT, WORDPRESS\_NONCE\_SALT, WORDPRESS\_TABLE\_PREFIX, and WORDPRESS\_DEBUG.

Run docker stack deploy -c stack.yml wordpress (or docker-compose -f stack.yml up), wait for it to initialize completely, and visit http://swarm-ip:8080, http://localhost:8080, or http://host-ip:8080 (as appropriate).

## Adding additional libraries / extensions

This image does not provide any additional PHP extensions or other libraries, even if they are required by popular plugins (e.g. [it cannot send e-mails](https://github.com/docker-library/wordpress/issues/30)). There are an infinite number of possible plugins, and they potentially require any extension PHP supports. Including every PHP extension that exists would dramatically increase the image size.

If you need additional PHP extensions, you'll need to create your own image FROM this one. The [documentation of the php image](https://github.com/docker-library/docs/blob/master/php/README.md" \l "how-to-install-more-php-extensions) explains how to compile additional extensions. Additionally, the [wordpress Dockerfile](https://github.com/docker-library/wordpress/blob/618490d4bdff6c5774b84b717979bfe3d6ba8ad1/apache/Dockerfile" \l "L5-L9) has an example of doing this.

The following Docker Hub features can help with the task of keeping your dependent images up-to-date:

* [Automated Builds](https://docs.docker.com/docker-hub/builds/) let Docker Hub automatically build your Dockerfile each time you push changes to it.

## Include pre-installed themes / plugins

Mount the volume containing your themes or plugins to the proper directory; and then apply them through the wp-admin webui. Ensure read/write/execute permissions are in place for the user.

* Themes go in a subdirectory in /var/www/html/wp-content/themes/
* Plugins go in a subdirectory in /var/www/html/wp-content/plugins/

## Running as an arbitrary user

See [the "Running as an arbitrary user" section of the php image documentation](https://hub.docker.com/_/php/).

When running WP-CLI via the cli variants of this image, it is important to note that they're based on Alpine, and have a default USER of Alpine's www-data, whose UID is 82 (compared to the Debian-based WordPress variants whose default effective UID is 33), so when running wordpress:cli against an existing Debian-based WordPress install, something like --user 33:33 is likely going to be necessary (possibly also something like -e HOME=/tmp depending on the wp command invoked and whether it tries to use ~/.wp-cli). See [docker-library/wordpress#256](https://github.com/docker-library/wordpress/issues/256) for more discussion around this.

# Image Variants

The wordpress images come in many flavors, each designed for a specific use case.

## wordpress:<version>

This is the defacto image. If you are unsure about what your needs are, you probably want to use this one. It is designed to be used both as a throw away container (mount your source code and start the container to start your app), as well as the base to build other images off of.

## wordpress:<version>-alpine

This image is based on the popular [Alpine Linux project](http://alpinelinux.org/), available in [the alpine official image](https://hub.docker.com/_/alpine). Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

This variant is highly recommended when final image size being as small as possible is desired. The main caveat to note is that it does use [musl libc](http://www.musl-libc.org/) instead of [glibc and friends](http://www.etalabs.net/compare_libcs.html), so certain software might run into issues depending on the depth of their libc requirements. However, most software doesn't have an issue with this, so this variant is usually a very safe choice. See [this Hacker News comment thread](https://news.ycombinator.com/item?id=10782897) for more discussion of the issues that might arise and some pro/con comparisons of using Alpine-based images.

To minimize image size, it's uncommon for additional related tools (such as git or bash) to be included in Alpine-based images. Using this image as a base, add the things you need in your own Dockerfile (see the [alpine image description](https://hub.docker.com/_/alpine/) for examples of how to install packages if you are unfamiliar).

## wordpress:cli

This image variant does not contain WordPress itself, but instead contains [WP-CLI](https://wp-cli.org/).

The simplest way to use it with an existing WordPress container would be something similar to the following:

$ docker run -it --rm \

--volumes-from some-wordpress \

--network container:some-wordpress \

wordpress:cli user list

Generally speaking, for WP-CLI to interact with a WordPress install, it needs access to the on-disk files of the WordPress install, and access to the database (and the easiest way to accomplish that such that wp-config.php does not require changes is to simply join the networking context of the existing and presumably working WordPress container, but there are many other ways to accomplish that which will be left as an exercise for the reader).