hub

Explore / Official Images / nginx

nginx

Docker Official Image • 1B+ • ☆10K+

Official build of Nginx.

WEB SERVERS

```
docker pull nginx
```

Copy

Overview Tags

Quick reference

- **Maintained by**:
  the NGINX Docker Maintainers
- **Where to get help**:
  the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

Supported tags and respective `Dockerfile` links

- `1.27.0`, `mainline`, `1`, `1.27`, `latest`, `1.27.0-bookworm`, `mainline-bookworm`, `1-bookworm`, `1.27-bookworm`, `bookworm`
- `1.27.0-perl`, `mainline-perl`, `1-perl`, `1.27-perl`, `perl`, `1.27.0-bookworm-perl`, `mainline-bookworm-perl`, `1-bookworm-perl`, `1.27-bookworm-perl`, `bookworm-perl`

- `1.27.0-otel`, `mainline-otel`, `1-otel`, `1.27-otel`, `otel`, `1.27.0-bookworm-otel`, `mainline-bookworm-otel`, `1-bookworm-otel`, `1.27-bookworm-otel`, `bookworm-otel`
- `1.27.0-alpine`, `mainline-alpine`, `1-alpine`, `1.27-alpine`, `alpine`, `1.27.0-alpine3.19`, `mainline-alpine3.19`, `1-alpine3.19`, `1.27-alpine3.19`, `alpine3.19`
- `1.27.0-alpine-perl`, `mainline-alpine-perl`, `1-alpine-perl`, `1.27-alpine-perl`, `alpine-perl`, `1.27.0-alpine3.19-perl`, `mainline-alpine3.19-perl`, `1-alpine3.19-perl`, `1.27-alpine3.19-perl`, `alpine3.19-perl`
- `1.27.0-alpine-slim`, `mainline-alpine-slim`, `1-alpine-slim`, `1.27-alpine-slim`, `alpine-slim`, `1.27.0-alpine3.19-slim`, `mainline-alpine3.19-slim`, `1-alpine3.19-slim`, `1.27-alpine3.19-slim`, `alpine3.19-slim`
- `1.27.0-alpine-otel`, `mainline-alpine-otel`, `1-alpine-otel`, `1.27-alpine-otel`, `alpine-otel`, `1.27.0-alpine3.19-otel`, `mainline-alpine3.19-otel`, `1-alpine3.19-otel`, `1.27-alpine3.19-otel`, `alpine3.19-otel`
- `1.26.1`, `stable`, `1.26`, `1.26.1-bookworm`, `stable-bookworm`, `1.26-bookworm`
- `1.26.1-perl`, `stable-perl`, `1.26-perl`, `1.26.1-bookworm-perl`, `stable-bookworm-perl`, `1.26-bookworm-perl`
- `1.26.1-otel`, `stable-otel`, `1.26-otel`, `1.26.1-bookworm-otel`, `stable-bookworm-otel`, `1.26-bookworm-otel`
- `1.26.1-alpine`, `stable-alpine`, `1.26-alpine`, `1.26.1-alpine3.19`, `stable-alpine3.19`, `1.26-alpine3.19`
- `1.26.1-alpine-perl`, `stable-alpine-perl`, `1.26-alpine-perl`, `1.26.1-alpine3.19-perl`, `stable-alpine3.19-perl`, `1.26-alpine3.19-perl`
- `1.26.1-alpine-slim`, `stable-alpine-slim`, `1.26-alpine-slim`, `1.26.1-alpine3.19-slim`, `stable-alpine3.19-slim`, `1.26-alpine3.19-slim`
- `1.26.1-alpine-otel`, `stable-alpine-otel`, `1.26-alpine-otel`, `1.26.1-alpine3.19-otel`, `stable-alpine3.19-otel`, `1.26-alpine3.19-otel`

Quick reference (cont.)

- **Where to file issues**:
  https://github.com/nginxinc/docker-nginx/issues
- **Supported architectures**: (more info)
  `amd64`, `arm32v5`, `arm32v6`, `arm32v7`, `arm64v8`, `i386`, `mips64le`, `ppc64le`, `s390x`

- **Published image artifact details**:
  repo-info repo's `repos/nginx/` directory (history)
  (image metadata, transfer size, etc)

- **Image updates**:
  official-images repo's `library/nginx` label
  official-images repo's `library/nginx` file (history)

- **Source of this description**:
  docs repo's `nginx/` directory (history)

What is nginx?

Nginx (pronounced "engine-x") is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, as well as a load balancer, HTTP cache, and a web server (origin server). The nginx project started with a strong focus on high concurrency, high performance and low memory usage. It is licensed under the 2-clause BSD-like license and it runs on Linux, BSD variants, Mac OS X, Solaris, AIX, HP-UX, as well as on other *nix flavors. It also has a proof of concept port for Microsoft Windows.

wikipedia.org/wiki/Nginx

How to use this image

Hosting some simple static content

```
$ docker run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d nginx
```

Alternatively, a simple `Dockerfile` can be used to generate a new image that includes the necessary content (which is a much cleaner solution than the bind mount above):

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

Place this file in the same directory as your directory of content ("static-html-directory"), run `docker build -t some-content-nginx .`, then start your container:

```
$ docker run --name some-nginx -d some-content-nginx
```

Exposing external port

```
$ docker run --name some-nginx -d -p 8080:80 some-content-nginx
```

Then you can hit `http://localhost:8080` or `http://host-ip:8080` in your browser.

Customize configuration

You can mount your configuration file, or build a new image with it.

If you wish to adapt the default configuration, use something like the following to get it from a running nginx container:

```
$ docker run --rm --entrypoint=cat nginx /etc/nginx/nginx.conf > /host/path/nginx
```

And then edit `/host/path/nginx.conf` in your host file system.

For information on the syntax of the nginx configuration files, see the official documentation (specifically the Beginner's Guide).

Mount your configuration file

```
$ docker run --name my-custom-nginx-container -v /host/path/nginx.conf:/etc/nginx/
```

Build a new image with your configuration file

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

If you add a custom `CMD` in the Dockerfile, be sure to include `-g daemon off;` in the `CMD` in order for nginx to stay in the foreground, so that Docker can track the process properly (otherwise your container will stop immediately after starting)!

Then build the image with `docker build -t custom-nginx .` and run it as follows:

```
$ docker run --name my-custom-nginx-container -d custom-nginx
```

Using environment variables in nginx configuration (new in 1.19)

Out-of-the-box, nginx doesn't support environment variables inside most configuration blocks. But this image has a function, which will extract environment variables before nginx starts.

Here is an example using docker-compose.yml:

```
web:
  image: nginx
  volumes:
   - ./templates:/etc/nginx/templates
  ports:
   - "8080:80"
  environment:
   - NGINX_HOST=foobar.com
   - NGINX_PORT=80
```

By default, this function reads template files in `/etc/nginx/templates/*.template` and outputs the result of executing `envsubst` to `/etc/nginx/conf.d`.

So if you place `templates/default.conf.template` file, which contains variable references like this:

```
listen       ${NGINX_PORT};
```

outputs to `/etc/nginx/conf.d/default.conf` like this:

```
listen       80;
```

This behavior can be changed via the following environment variables:

- `NGINX_ENVSUBST_TEMPLATE_DIR`
  - A directory which contains template files (default: `/etc/nginx/templates`)
  - When this directory doesn't exist, this function will do nothing about template processing.
- `NGINX_ENVSUBST_TEMPLATE_SUFFIX`
  - A suffix of template files (default: `.template`)
  - This function only processes the files whose name ends with this suffix.
- `NGINX_ENVSUBST_OUTPUT_DIR`
  - A directory where the result of executing envsubst is output (default: `/etc/nginx/conf.d`)

- The output filename is the template filename with the suffix removed.
  - ex.) `/etc/nginx/templates/default.conf.template` will be output with the filename `/etc/nginx/conf.d/default.conf`.
- This directory must be writable by the user running a container.

Running nginx in read-only mode

To run nginx in read-only mode, you will need to mount a Docker volume to every location where nginx writes information. The default nginx configuration requires write access to `/var/cache/nginx` and `/var/run`. This can be easily accomplished by running nginx as follows:

```
$ docker run -d -p 80:80 --read-only -v $(pwd)/nginx-cache:/var/cache/nginx -v $(
```

If you have a more advanced configuration that requires nginx to write to other locations, simply add more volume mounts to those locations.

Running nginx in debug mode

Images since version 1.9.8 come with `nginx-debug` binary that produces verbose output when using higher log levels. It can be used with simple CMD substitution:

```
$ docker run --name my-nginx -v /host/path/nginx.conf:/etc/nginx/nginx.conf:ro -d
```

Similar configuration in docker-compose.yml may look like this:

```
web:
  image: nginx
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
  command: [nginx-debug, '-g', 'daemon off;']
```

Entrypoint quiet logs

Since version 1.19.0, a verbose entrypoint was added. It provides information on what's happening during container startup. You can silence this output by setting environment variable `NGINX_ENTRYPOINT_QUIET_LOGS`:

```
$ docker run -d -e NGINX_ENTRYPOINT_QUIET_LOGS=1 nginx
```

User and group id

Since 1.17.0, both alpine- and debian-based images variants use the same user and group ids to drop the privileges for worker processes:

```
$ id
uid=101(nginx) gid=101(nginx) groups=101(nginx)
```

Running nginx as a non-root user

It is possible to run the image as a less privileged arbitrary UID/GID. This, however, requires modification of nginx configuration to use directories writeable by that specific UID/GID pair:

```
$ docker run -d -v $PWD/nginx.conf:/etc/nginx/nginx.conf nginx
```

where nginx.conf in the current directory should have the following directives re-defined:

```
pid        /tmp/nginx.pid;
```

And in the http context:

```
http {
    client_body_temp_path /tmp/client_temp;
    proxy_temp_path       /tmp/proxy_temp_path;
    fastcgi_temp_path     /tmp/fastcgi_temp;
    uwsgi_temp_path       /tmp/uwsgi_temp;
    scgi_temp_path        /tmp/scgi_temp;
...
}
```

Alternatively, check out the official Docker NGINX unprivileged image.

Image Variants

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

`nginx:<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.

Some of these tags may have names like bookworm in them. These are the suite code names for releases of Debian and indicate which release the image is based on. If your image needs to install any additional packages beyond what comes with the image, you'll likely want to specify one of these explicitly to minimize breakage when there are new releases of Debian.

`nginx:<version>-perl` / `nginx:<version>-alpine-perl`

Starting with nginx:1.13.0 / mainline and nginx:1.12.0 / stable, the perl module has been removed from the default images. A separate `-perl` tag variant is available if you wish to use the perl module.

`nginx:<version>-alpine`

This image is based on the popular Alpine Linux project, available in the `alpine` official image. Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

This variant is useful when final image size being as small as possible is your primary concern. The main caveat to note is that it does use musl libc instead of glibc and friends, so software will often run into issues depending on the depth of their libc requirements/assumptions. See this Hacker News comment thread 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 for examples of how to install packages if you are unfamiliar).

`nginx:<version>-slim`

This image does not contain the common packages contained in the default tag and only contains the minimal packages needed to run `nginx`. Unless you are working in an environment where *only* the `nginx` image will be deployed and you have space constraints, we highly recommend using the default image of this repository.

License

View license information for the software contained in this image.

As with all Docker images, these likely also contain other software which may be under other licenses (such as Bash, etc from the base distribution, along with any direct or indirect dependencies of the primary software being contained).

Some additional license information which was able to be auto-detected might be found in the `repo-info` repository's `nginx/` directory.

As for any pre-built image usage, it is the image user's responsibility to ensure that any use of this image complies with any relevant licenses for all software contained within.

Recent Tags

stable-perl stable-otel stable-bookworm-perl stable-bookworm-otel stable-bookworm stable-alpine3.19-slim stable-alpine3.19-perl stable-alpine3.19-otel stable-alpine3.19 stable-alpine-slim

About Official Images

Docker Official Images are a curated set of Docker open source and drop-in solution repositories.

Why Official Images?

These images have clear documentation, promote best practices, and are designed for the most common use cases.

Docker Hub Documentation

Features
Container Runtime
Developer Tools
Docker App
Kubernetes

Company

About Us
Resources
Blog
Customers
Partners
Newsroom
Events and Webinars
Careers
Contact Us
System Status