## Quick reference

- **Maintained by**:
  the Docker Community

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

## Supported tags and respective `Dockerfile` links

- `3.4.0-preview1-bookworm`, `3.4-rc-bookworm`, `3.4.0-preview1`, `3.4-rc`
- `3.4.0-preview1-slim-bookworm`, `3.4-rc-slim-bookworm`, `3.4.0-preview1-slim`, `3.4-rc-slim`
- `3.4.0-preview1-bullseye`, `3.4-rc-bullseye`
- `3.4.0-preview1-slim-bullseye`, `3.4-rc-slim-bullseye`
- `3.4.0-preview1-alpine3.20`, `3.4-rc-alpine3.20`, `3.4.0-preview1-alpine`, `3.4-rc-alpine`
- `3.4.0-preview1-alpine3.19`, `3.4-rc-alpine3.19`
- `3.3.1-bookworm`, `3.3-bookworm`, `3-bookworm`, `bookworm`, `3.3.1`, `3.3`, `3`, `latest`
- `3.3.1-slim-bookworm`, `3.3-slim-bookworm`, `3-slim-bookworm`, `slim-bookworm`, `3.3.1-slim`, `3.3-slim`, `3-slim`, `slim`
- `3.3.1-bullseye`, `3.3-bullseye`, `3-bullseye`, `bullseye`
- `3.3.1-slim-bullseye`, `3.3-slim-bullseye`, `3-slim-bullseye`, `slim-bullseye`
- `3.3.1-alpine3.20`, `3.3-alpine3.20`, `3-alpine3.20`, `alpine3.20`, `3.3.1-alpine`, `3.3-alpine`, `3-alpine`, `alpine`
- `3.3.1-alpine3.19`, `3.3-alpine3.19`, `3-alpine3.19`, `alpine3.19`
- `3.2.4-bookworm`, `3.2-bookworm`, `3.2.4`, `3.2`
- `3.2.4-slim-bookworm`, `3.2-slim-bookworm`, `3.2.4-slim`, `3.2-slim`

- `3.2.4-bullseye`, `3.2-bullseye`
- `3.2.4-slim-bullseye`, `3.2-slim-bullseye`
- `3.2.4-alpine3.20`, `3.2-alpine3.20`, `3.2.4-alpine`, `3.2-alpine`
- `3.2.4-alpine3.19`, `3.2-alpine3.19`
- `3.1.5-bookworm`, `3.1-bookworm`, `3.1.5`, `3.1`
- `3.1.5-slim-bookworm`, `3.1-slim-bookworm`, `3.1.5-slim`, `3.1-slim`
- `3.1.5-bullseye`, `3.1-bullseye`
- `3.1.5-slim-bullseye`, `3.1-slim-bullseye`
- `3.1.5-alpine3.20`, `3.1-alpine3.20`, `3.1.5-alpine`, `3.1-alpine`
- `3.1.5-alpine3.19`, `3.1-alpine3.19`

## Quick reference (cont.)

- **Where to file issues**:
  https://github.com/docker-library/ruby/issues
- **Supported architectures**: (more info)
  `amd64`, `arm32v5`, `arm32v6`, `arm32v7`, `arm64v8`, `i386`, `mips64le`, `ppc64le`, `riscv64`, `s390x`
- **Published image artifact details**:
  repo-info repo's `repos/ruby/` directory (history)
  (image metadata, transfer size, etc)
- **Image updates**:
  official-images repo's `library/ruby` label
  official-images repo's `library/ruby` file (history)
- **Source of this description**:
  docs repo's `ruby/` directory (history)

## What is Ruby?

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language. According to its authors, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, and Lisp. It supports multiple programming paradigms, including functional, object-oriented, and imperative. It also has a dynamic type system and automatic memory management.

> wikipedia.org/wiki/Ruby_(programming_language)

## How to use this image

### Create a `Dockerfile` in your Ruby app project

```
FROM ruby:3.0

# throw errors if Gemfile has been modified since Gemfile.lock
RUN bundle config --global frozen 1

WORKDIR /usr/src/app

COPY Gemfile Gemfile.lock ./
RUN bundle install

COPY . .

CMD ["./your-daemon-or-script.rb"]
```

Put this file in the root of your app, next to the `Gemfile`.

You can then build and run the Ruby image:

```
$ docker build -t my-ruby-app .
$ docker run -it --name my-running-script my-ruby-app
```

#### Generate a `Gemfile.lock`

The above example `Dockerfile` expects a `Gemfile.lock` in your app directory. This `docker run` will help you generate one. Run it in the root of your app, next to the `Gemfile`:

```
$ docker run --rm -v "$PWD":/usr/src/app -w /usr/src/app ruby:3.0 bundle install
```

## Run a single Ruby script

For many simple, single file projects, you may find it inconvenient to write a complete `Dockerfile`. In such cases, you can run a Ruby script by using the Ruby Docker image directly:

```
$ docker run -it --rm --name my-running-script -v "$PWD":/usr/src/myapp -w /usr/src/my
```

## Encoding

By default, Ruby inherits the locale of the environment in which it is run. For most users running Ruby on their desktop systems, that means it's likely using some variation of `*.UTF-8` (`en_US.UTF-8`, etc). In Docker however, the default locale is `C`, which can have unexpected results. If your application needs to interact with UTF-8, it is recommended that you explicitly adjust the locale of your image/container via `-e LANG=C.UTF-8` or `ENV LANG C.UTF-8`.

## Image assumptions

This image sets several environment variables which change the behavior of Bundler and Gem for running a single application within a container (especially in such a way that the development sources of the application can be bind-mounted inside a container and not have `.bundle` from the host interfere with the proper functionality of the container).

The environment variables we set are canonically listed in the above-linked `Dockerfiles`, but some of them include `GEM_HOME`, `BUNDLE_PATH`, `BUNDLE_BIN`, `BUNDLE_SILENCE_ROOT_WARNING`, and `BUNDLE_APP_CONFIG`.

If these cause issues for your use case (running multiple Ruby applications in a single container, for example), setting them to the empty string *should* be sufficient for undoing their behavior.

# Image Variants

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

### `ruby:<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 or bullseye 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.

This tag is based off of `buildpack-deps`. `buildpack-deps` is designed for the average user of Docker who has many images on their system. It, by design, has a large number of extremely common Debian packages. This reduces the number of packages that images that derive from it need to install, thus reducing the overall size of all images on your system.

### `ruby:<version>-slim`

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

### `ruby:<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).

## 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 `ruby/` 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.