

# **Best practices when writing a Dockerfile (Ruby)**

# Hello

This is Florin.

**Docker > Dockerfile**

# Dockerfile?

- FROM: Specifies the parent image.
- RUN: Runs a command during the build - e.g. to install dependencies.
- USER: Sets the user that will run the instructions.
- COPY: Copies a file to the container during the build.
- WORKDIR: Sets the working directory for all future instructions.
- CMD: Sets the default command and/or parameters.

# Example

```
FROM ruby
```

```
COPY . ./
```

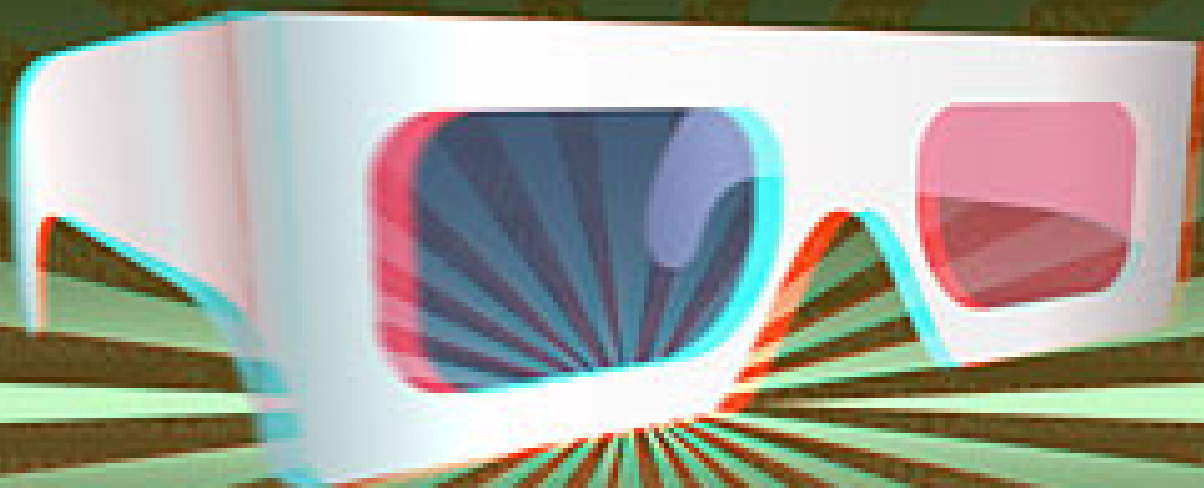
```
RUN bundle install
```

```
RUN apt-get update
```

```
RUN apt-get install -y build-essential
```

```
RUN apt-get install -y ruby-dev
```

```
CMD bundle exec rackup
```



**PUT ON YOUR 3D GLASSES NOW**

1.

```
# Hmmmm...
```

```
FROM ruby
```

```
CMD ruby -e "puts 1 + 2"
```

## 1. Pin your base image version

```
FROM ruby:2.5.5
```

```
CMD ruby -e "puts 1 + 2"
```



2.

```
# Hmmmm . . .
```

```
FROM random-dude-on-the-internet/ruby:2.5.5
```

```
CMD ruby -e "puts 1 + 2"
```

## 2. Use only trusted or official base images

```
FROM ruby:2.5.5
```

```
CMD ruby -e "puts 1 + 2"
```



ruby ☆

**Docker Official Images**

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

```
FROM your-own-registry.com/ruby:2.5.5
```

```
CMD ruby -e "puts 1 + 2"
```

**3.**

```
FROM ruby:2.5.5
```

```
# Hmmm...
```

```
RUN gem install sinatra
```

### 3. Pin your application dependencies

```
FROM ruby:2.5.5
```

```
RUN gem install sinatra -v 2.0.5
```

Gemfile.lock , package-lock.json , yarn.lock , Pipfile.lock , ...

4.

```
# .gitignore  
  
.env  
.bundle/  
node_modules/
```

```
# Dockerfile  
  
FROM ruby:2.5.5  
  
# Hmmm...  
COPY . ./
```

## 4. Add a .dockerignore file to your repository

```
# .gitignore  
  
.env  
.bundle/  
node_modules/
```

```
# .dockerignore  
  
.env  
.bundle/  
node_modules/  
.git/  
.gitignore
```

5.

```
FROM ruby:2.5.5
```

```
# Hmmm...
```

```
RUN apt update
```

```
RUN apt install -y mysql-client
```

```
RUN apt install -y postgresql-client
```

```
RUN apt install -y nginx
```

```
RUN bundle install
```

```
CMD ruby -e "puts 1 + 2"
```

## 5. Group commands by how likely they are to change individually

```
FROM ruby:2.5.5
```

```
# We usually only need to run this once
```

```
RUN apt update && \  
    apt install -y mysql-client postgresql-client nginx
```

```
# We usually run this every time we add a new dependency
```

```
RUN bundle install
```

```
CMD ruby -e "puts 1 + 2"
```



6.

```
FROM ruby:2.5.5
```

```
# Source code
```

```
COPY my-code/ /srv/
```

```
# Application dependencies
```

```
COPY Gemfile Gemfile.lock ./
```

```
RUN bundle install
```

```
# Operating system dependencies
```

```
RUN apt update && \
```

```
    apt install -y mysql-client postgresql-client nginx
```

## 6. Place the least likely to change commands at the top

```
FROM ruby:2.5.5
```

```
# Operating system dependencies
```

```
RUN apt update && \  
    apt install -y mysql-client postgresql-client nginx
```

```
# Application dependencies
```

```
COPY Gemfile Gemfile.lock ./
```

```
RUN bundle install
```

```
# Source code
```

```
COPY my-source-code /srv/
```

7.

```
FROM ruby:2.5.5
```

```
RUN gem install sinatra -v 2.0.5
```

```
RUN echo 'require "sinatra"; run Sinatra::Application.run!' > config.ru
```

```
# Hmmm... Which user will run this?
```

```
CMD rackup
```

## 7. Avoid running your application as root

```
FROM ruby:2.5.5
```

```
RUN gem install sinatra -v 2.0.5
```

```
# Create a dedicated user for running the application
```

```
RUN adduser -D my-sinatra-user
```

```
# Set the user for RUN, CMD or ENTRYPOINT calls from now on
```

```
# Note that this doesn't apply to COPY or ADD, which use a --chown argument instead
```

```
USER my-sinatra-user
```

```
# Set the base directory that will be used from now on
```

```
WORKDIR /home/my-sinatra-user
```

```
RUN echo 'require "sinatra"; run Sinatra::Application.run!' > config.ru
```

```
# This is run under the my-sinatra-user user
```

```
CMD rackup
```

8.

```
FROM ruby:2.5.5-alpine
```

```
RUN adduser -D my-sinatra-user
```

```
USER my-sinatra-user
```

```
WORKDIR /home/my-sinatra-user
```

```
# Hmm... Which user owns these files?
```

```
COPY Gemfile Gemfile.lock ./
```

```
RUN bundle install
```

```
CMD rackup
```

## 8. When running COPY or ADD (as a different user) use --chown

```
FROM ruby:2.5.5-alpine
```

```
RUN adduser -D my-sinatra-user
```

```
USER my-sinatra-user
```

```
WORKDIR /home/my-sinatra-user
```

```
# The files will be owned by my-sinatra-user
```

```
COPY --chown=my-sinatra-user Gemfile Gemfile.lock ./
```

```
RUN bundle install
```

```
CMD rackup
```

## 9. Avoid leaking secrets inside your image

```
FROM ruby:2.5.5
```

```
ENV DB_PASSWORD "secret stuff"
```

Secrets should never appear inside your *Dockerfile* in plain text!

## 9. Avoid leaking secrets inside your image

Alternatives:

- Build-time arguments: the `ARG` command and `--build-arg` Docker argument.
- Environment variables: the `-e` or `--env-file` Docker arguments.
- Kubernetes secrets or similar.



10.

```
FROM ruby:2.5.5
```

```
ARG PRIVATE_SSH_KEY
```

```
# Hmmm...
```

```
RUN echo "${PRIVATE_SSH_KEY}" > /root/.ssh/id_rsa
```

```
# More hmmm...
```

```
RUN bundle install
```

```
# Cleaning up the private key
```

```
RUN rm /root/.ssh/id_rsa
```

## 10. Always clean up injected secrets within the same build step

```
FROM ruby:2.5.5
```

```
ARG PRIVATE_SSH_KEY
```

```
RUN echo "${PRIVATE_SSH_KEY}" > /root/.ssh/id_rsa && \  
    bundle install && \  
    rm /root/.ssh/id_rsa
```

## 11. Fetching private dependencies via a Github token injected through the gitconfig

```
FROM ruby:2.5.5

ARG GITHUB_TOKEN

# This is a private gem that GITHUB_TOKEN has access to
RUN echo 'source "https://rubygems.org"; gem "some-private-gem", git: "git@github.com:myorg/some-private-gem"' > Gemfile

# First insteadOf is for Gemfile, second is for package.json
RUN git config --global url."https://${GITHUB_TOKEN}:x-oauth-basic@github.com:myorg".insteadOf git@github.com:myorg && \
  git config --global --add url."https://${GITHUB_TOKEN}:x-oauth-basic@github.com:myorg".insteadOf ssh://git@github && \
  bundle install && \
  rm ~/.gitconfig
```

```
docker build --build-arg GITHUB_TOKEN=xxx .
```

12.

```
# Hmmmm . . .
```

```
FROM ruby:2.5.5
```

```
CMD ruby -e "puts 1 + 2"
```

```
> docker images -a | grep base-image
```

```
normal-base-image    869MB
```

## 12. Minimize image size by opting for small base images when possible

```
FROM ruby:2.5.5-alpine
```

```
CMD ruby -e "puts 1 + 2"
```

```
> docker images -a | grep base-image
```

normal-base-image	869MB
small-base-image	45.3MB

## 12. Minimize image size by opting for small base images when possible

Pay special attention to:

- The **package manager** and the available packages
- The **choice of shell** (some base images might not even provide a shell)
- **Security** and **stability**

## 13.

```
FROM ruby:2.5.5-alpine

# Nokogiri's build dependencies
RUN apk add --update \
    build-base \
    libxml2-dev \
    libxslt-dev

# Nokogiri, yikes
RUN echo 'source "https://rubygems.org"; gem "nokogiri"' > Gemfile

RUN bundle install

CMD /bin/sh
```

## 13. Use multi-stage builds to reduce the size of your image

```
# The "builder" image will build nokogiri
FROM ruby:2.5.5-alpine AS builder

RUN apk add --update \
    build-base \
    libxml2-dev \
    libxslt-dev
RUN echo 'source "https://rubygems.org"; gem "nokogiri"' > Gemfile
RUN bundle install

# The final image: we start clean and copy over the built bundle
FROM ruby:2.5.5-alpine

COPY --from=builder /usr/local/bundle/ /usr/local/bundle/

CMD /bin/sh
```



## 13. Use multi-stage builds to reduce the size of your image

```
> docker images | grep nokogiri
```

nokogiri-simple	251MB
nokogiri-multistage	70.1MB

## 14.

```
FROM ruby:2.5.5-alpine
```

```
# This is a secret
```

```
ARG PRIVATE_SSH_KEY
```

```
# Our Gemfile contains a private dependency
```

```
RUN echo 'source "https://rubygems.org"; gem "a-private-gem"' > Gemfile
```

```
# We require the secret for installing the private dependencies
```

```
RUN mkdir -p /root/.ssh/ && \  
  echo "${PRIVATE_SSH_KEY}" > /root/.ssh/id_rsa && \  
  bundle install && \  
  rm /root/.ssh/id_rsa
```

```
CMD ruby -e "puts 1 + 2"
```

```
docker build --build-arg PRIVATE_SSH_KEY=xxx .
```

## 14.

```
> docker history my-fancy-image
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
67e60c0853ab	19 seconds ago	/bin/sh -c #(nop) CMD ["/bin/sh" "-c" "ruby...	0B	
94dd778c4b5d	20 seconds ago	1 PRIVATE_SSH_KEY=xxx /bin/sh -c mkdir -p /...	30.9MB	
32a993af7bfb	About a minute ago	1 PRIVATE_SSH_KEY=xxx /bin/sh -c echo 'sour...	45B	
2be964ad91c7	About a minute ago	/bin/sh -c #(nop) ARG PRIVATE_SSH_KEY	0B	
44723f3ab2bd	4 months ago	/bin/sh -c #(nop) CMD ["irb"]	0B	
<missing>	4 months ago	/bin/sh -c mkdir -p "\$GEM_HOME" && chmod 777...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV PATH=/usr/local/bundl...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV BUNDLE_PATH=/usr/loca...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV GEM_HOME=/usr/local/b...	0B	
<missing>	4 months ago	/bin/sh -c set -ex && apk add --no-cache -...	45.5MB	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBYGEMS_VERSION=3.0.3	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_DOWNLOAD_SHA256=...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_VERSION=2.5.5	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_MAJOR=2.5	0B	
<missing>	4 months ago	/bin/sh -c mkdir -p /usr/local/etc && { e...	45B	
<missing>	4 months ago	/bin/sh -c apk add --no-cache gmp-dev	3.4MB	
<missing>	4 months ago	/bin/sh -c #(nop) CMD ["/bin/sh"]	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ADD file:a86aea1f3a7d68f6a...	5.53MB	

## 14. Use multi-stage builds to avoid leaking secrets in your docker history

```
# The builder image makes use of the secret
FROM ruby:2.5.5-alpine AS builder

ARG PRIVATE_SSH_KEY

RUN echo 'source "https://rubygems.org"; gem "a-private-gem"' > Gemfile

RUN mkdir -p /root/.ssh/ && \
    echo "${PRIVATE_SSH_KEY}" > /root/.ssh/id_rsa && \
    bundle install && \
    rm /root/.ssh/id_rsa

# The final image doesn't need the secret
FROM ruby:2.5.5-alpine

COPY --from=builder /usr/local/bundle/ /usr/local/bundle/

CMD ruby -e "puts 1 + 2"
```

## 14. Use multi-stage builds to avoid leaking secrets in your docker history

```
> docker history my-fancy-image
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
2706a2f47816	8 seconds ago	/bin/sh -c #(nop) CMD ["/bin/sh" "-c" "ruby...	0B	
86509dba3bd9	9 seconds ago	/bin/sh -c #(nop) COPY dir:e110956912ddb292a...	3.16MB	
44723f3ab2bd	4 months ago	/bin/sh -c #(nop) CMD ["irb"]	0B	
<missing>	4 months ago	/bin/sh -c mkdir -p "\$GEM_HOME" && chmod 777...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV PATH=/usr/local/bundl...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV BUNDLE_PATH=/usr/loca...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV GEM_HOME=/usr/local/b...	0B	
<missing>	4 months ago	/bin/sh -c set -ex && apk add --no-cache -...	45.5MB	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBYGEMS_VERSION=3.0.3	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_DOWNLOAD_SHA256=...	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_VERSION=2.5.5	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ENV RUBY_MAJOR=2.5	0B	
<missing>	4 months ago	/bin/sh -c mkdir -p /usr/local/etc && { e...	45B	
<missing>	4 months ago	/bin/sh -c apk add --no-cache gmp-dev	3.4MB	
<missing>	4 months ago	/bin/sh -c #(nop) CMD ["/bin/sh"]	0B	
<missing>	4 months ago	/bin/sh -c #(nop) ADD file:a86aea1f3a7d68f6a...	5.53MB	

## 15.

FROM ruby:2.5.5

RUN echo 'source "https://rubygems.org"; gem "sinatra"' > Gemfile

RUN bundle install

# A simple Sinatra app which prints out HUUUUUP when the process receives the HUP signal.

RUN echo 'require "sinatra"; set bind: "0.0.0.0"; Signal.trap("HUP") { puts "HUUUUUP" }; run Sinatra::Application.run!' > config.ru

CMD bundle exec rackup

```
> docker exec $(docker ps -q) ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.7	0.0	2388	756	pts/0	Ss+	14:36	0:00	/bin/sh -c bundle exec rackup
root	6	0.8	0.2	43948	25556	pts/0	Sl+	14:36	0:00	/usr/local/bundle/bin/rackup
root	13	0.0	0.0	7640	2588	?	Rs	14:37	0:00	ps aux

```
> docker kill --signal=HUP $(docker ps -q)
```

<crickets>

## 15. When setting the CMD instruction, prefer the exec format over the shell format

FROM ruby:2.5.5

RUN echo 'source "https://rubygems.org"; gem "sinatra"' > Gemfile

RUN bundle install

# A simple Sinatra app which prints out HUUUUUP when the process receives the HUP signal.

RUN echo 'require "sinatra"; set bind: "0.0.0.0"; Signal.trap("HUP") { puts "HUUUUUP" }; run Sinatra::Application.run!' > config.ru

CMD ["bundle", "exec", "rackup"]

```
> docker exec $(docker ps -q) ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	29.0	0.2	43988	25632	pts/0	Ssl+	14:47	0:00	/usr/local/bundle/bin/rackup
root	8	0.0	0.0	7640	2668	?	Rs	14:47	0:00	ps aux

```
> docker kill --signal=HUP $(docker ps -q)
```

HUUUUUP

**Bonus!**



# 16.

```
# Dockerfile

FROM ruby:2.5.5-alpine

# A Gemfile that contains a test dependency (minitest)
RUN echo 'source "https://rubygems.org"; gem "sinatra"; group(:test) { gem "minitest" }' > Gemfile
RUN echo 'require "sinatra"; run Sinatra::Application.run!' > config.ru
RUN echo 'require "minitest/spec"; require "minitest/autorun"; class TestIndex < MiniTest::Test; def test_it_passes; assert(true); end; end' > test.rb

# By default we don't install development or test dependencies
RUN bundle install --without development test

CMD ["rackup"]
```

```
# development.Dockerfile

FROM ruby:2.5.5-alpine

# A Gemfile that contains a test dependency (minitest)
RUN echo 'source "https://rubygems.org"; gem "sinatra"; group(:test) { gem "minitest" }' > Gemfile
RUN echo 'require "sinatra"; run Sinatra::Application.run!' > config.ru
RUN echo 'require "minitest/spec"; require "minitest/autorun"; class TestIndex < MiniTest::Test; def test_it_passes; assert(true); end; end' > test.rb

# Here we want the development dependencies
RUN bundle install

CMD ["rackup"]
```

16.

```
docker build -f development.Dockerfile -t development:v1 .
```

```
docker run -it --rm development:v1 ruby test.rb
```

```
# Running:
```

```
.
```

```
Finished in 0.000725s, 1379.8413 runs/s, 1379.8413 assertions/s.
```

```
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

```
docker build -t production:v1 .
```

```
docker run -it --rm production:v1
```

## 16. Combine production, test and development build processes into a single Dockerfile by using multi-stage builds

```
FROM ruby:2.5.5-alpine AS builder

# A Gemfile that contains a test dependency (minitest)
RUN echo 'source "https://rubygems.org"; gem "sinatra"; group(:test) { gem "minitest" }' > Gemfile
RUN echo 'require "sinatra"; run Sinatra::Application.run!' > config.ru

# By default we don't install development or test dependencies
RUN bundle install --without development test

# A separate build stage installs test dependencies and runs your tests
FROM builder AS test
RUN bundle install --with test
RUN echo 'require "minitest/spec"; require "minitest/autorun"; class TestIndex < MiniTest::Test; def test_it_passes; assert(true); end; end' > test.rb
RUN bundle exec ruby test.rb

# The production artifact doesn't contain any test dependencies
FROM ruby:2.5.5-alpine

COPY --from=builder /usr/local/bundle/ /usr/local/bundle/
COPY --from=builder /config.ru ./

CMD ["rackup"]
```

## 16. Combine production, test and development build processes into a single Dockerfile by using multi-stage builds

```
DOCKER_BUILDKIT=1 docker build .
```

```
DOCKER_BUILDKIT=1 docker build --target=test .
```

**Let's wrap things up**

```
# .dockerignore
```

```
.env
```

```
.bundle/
```

```
.git/
```

```
FROM ruby:2.5.5-alpine AS builder
```

```
ARG GITHUB_TOKEN
```

```
RUN apk add --update \  
    build-base \  
    libxml2-dev \  
    libxslt-dev \  
    git
```

```
COPY Gemfile Gemfile.lock ./
```

```
RUN git config --global url."https://${GITHUB_TOKEN}:x-oauth-basic@github.com/myorg".insteadOf git@github.com:myorg && \  
    bundle install --without development test && \  
    rm ~/.gitconfig
```

```
FROM ruby:2.5.5-alpine
```

```
COPY --from=builder /usr/local/bundle/ /usr/local/bundle/
```

```
RUN adduser -D myuser
```

```
USER myuser
```

```
WORKDIR /home/myuser
```

```
COPY --chown=myuser . ./
```

```
CMD ["bundle", "exec", "rackup"]
```

# Questions?

<https://lipanski.com>