Dockerfile Best Practices

Bhushan Lodha

Senior Software Consultant

Tuesday 7th July 2020, 6pm

Why should I care about writing good dockerfile?

- ➔ Image size
  - ◆ Smaller image size improves security, performance, efficiency, and maintainability of your containers
- ➔ Build time
  - ◆ Reducing build time can significantly reduce your app’s development cost
- ➔ Security
  - ◆ Important to protect your app from external attacks
- ➔ Consistency
  - ◆ Consistent images are easier to maintain

Reduce Image Size

Faster deploys, smaller attack surface

Image Variants

```
FROM ruby:2.7.0

COPY Gemfile* ./
RUN bundle install

COPY . .

CMD ["bundle", "exec", "rails", "server", "-b", "0.0.0.0"]
```

Image	Size
ruby:2.7	842mb
ruby:2.7-slim	149mb
ruby:2.7-alpine	52mb

Base Image

Ubuntu/Debian	**Alpine**
Built around GNU C library	Built around musl libc
More mature	Less mature
More stable	Less stable
More 3rd party packages	Fewer 3rd party packages
Less memory efficient	More memory efficient
Bigger base image footprint	Smaller base image footprint

Premature optimization is root of (all) evil

Remove unnecessary dependencies

```
RUN apt-get install --no-install-recommends \
    build-essential \
    libpq-dev \
    ssh
```

before	726mb
after	692mb

Remove package manager cache, logs & docs

```
RUN apt-get install build-essential && \
    rm -rf /var/lib/apt/lists/* \
           /var/lib/dpkg \
           /var/lib/cache \
           /var/lib/log \
           /usr/share/doc/ \
           /usr/share/man/ \
           /var/log/* \
           /var/cache/*
```

before	692mb
after	674mb

Remove unnecessary application files

- ➔ Test files
- ➔ Asset files
- ➔ Documentation
- ➔ Application dependency test files
- ➔ Application cache
- ➔ Dependency cache

Use multistage builds to separate build & runtime env

```
FROM ruby:2.7.1-slim as builder
RUN apt-get install build-essential \
    tzdata libpq-dev libgeos-dev
COPY Gemfile Gemfile.lock ./
RUN bundle install

FROM ruby:2.7.1-slim as release
RUN apt-get install tzdata
WORKDIR /code
COPY --from=builder /usr/local/bundle /usr/local/bundle
COPY . /code
CMD ["bundle", "exec", "rails", "server"]
```

before	674mb
after	301mb

Reduce Build Time

Leverage build cache

Order matters!

```
FROM ruby:2.7.1-slim
WORKDIR /code

COPY . /code

RUN apt-get update && apt-get install \
    build-essential tzdata

COPY . /code
```

Order from least to most frequently changing content

Copy specific file to limit cache busts

```
FROM ruby:2.7
WORKDIR /code

COPY . /code
RUN bundle install

COPY Gemfile Gemfile.lock /code
RUN bundle install
```

Identify cacheable units

```
FROM ruby:2.7

RUN apt-get update
RUN apt-get install libpq-dev libgeos-dev

RUN apt-get update && \
    apt-get install libpq-dev libgeos-dev
```

Use Buildkit

- ➔ Second generation image builder
- ➔ Parallel build execution
- ➔ Much faster and more accurate caching mechanism

Enable Buildkit

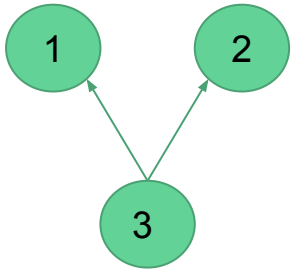
```
> DOCKER_BUILDKIT=1 docker build .
```

```
# demon configuration
{ "features": { "buildkit": true } }
```

Buildkit: Parallel Build Execution

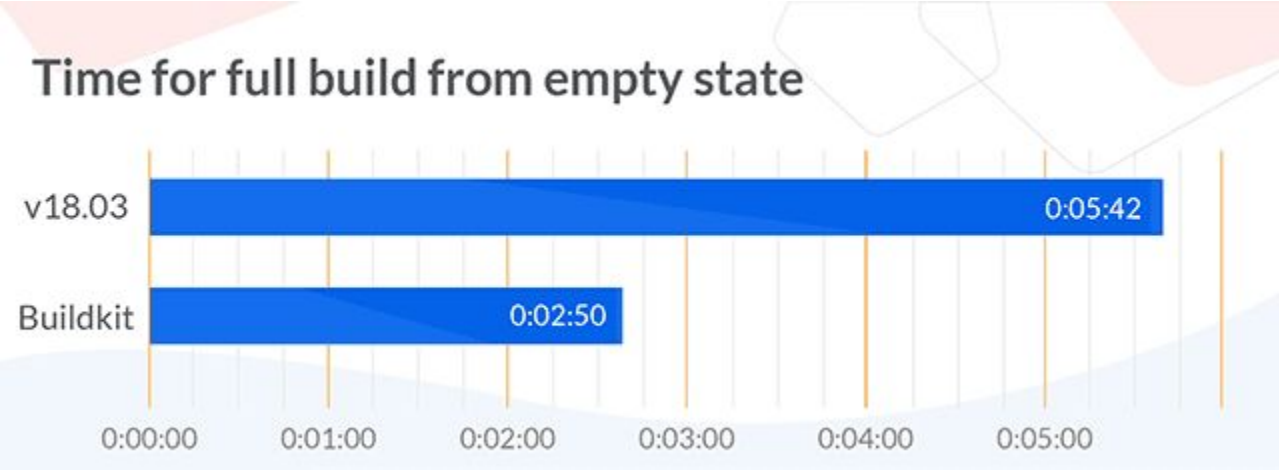
```
FROM debian as stage1
RUN echo "Hello stage1" > stage1.html

FROM debian as stage2
RUN echo "Hello stage2" > stage2.html

FROM debian as stage3
COPY --from=stage1 stage1.html /out
COPY --from=stage2 stage2.html /out
```

with buildkit



without buildkit

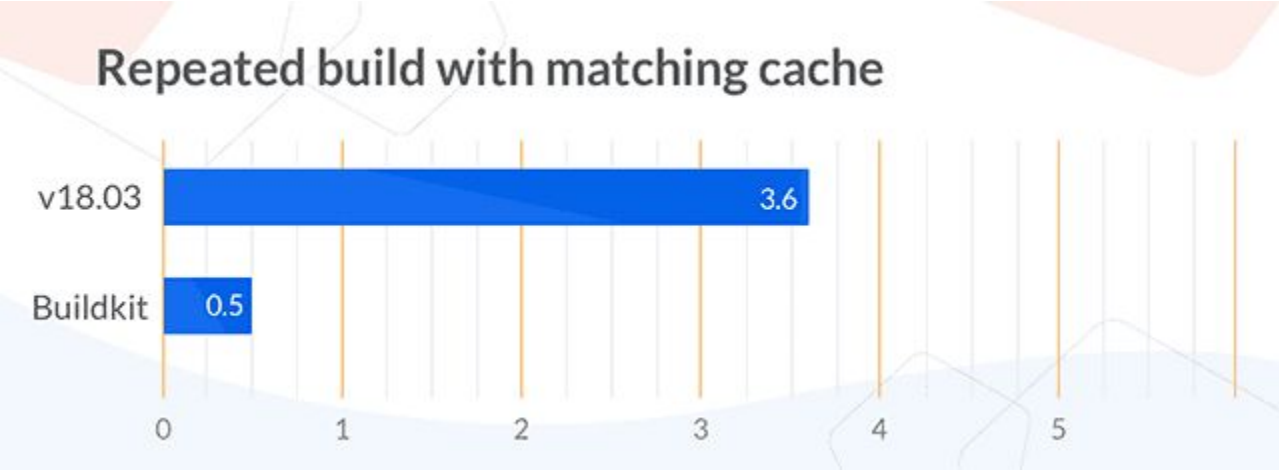
Buildkit performance benchmarking



Based on the docker build from scratch, the results are 2.5x faster

source: https://www.xenonstack.com/blog/docker-buildkit/

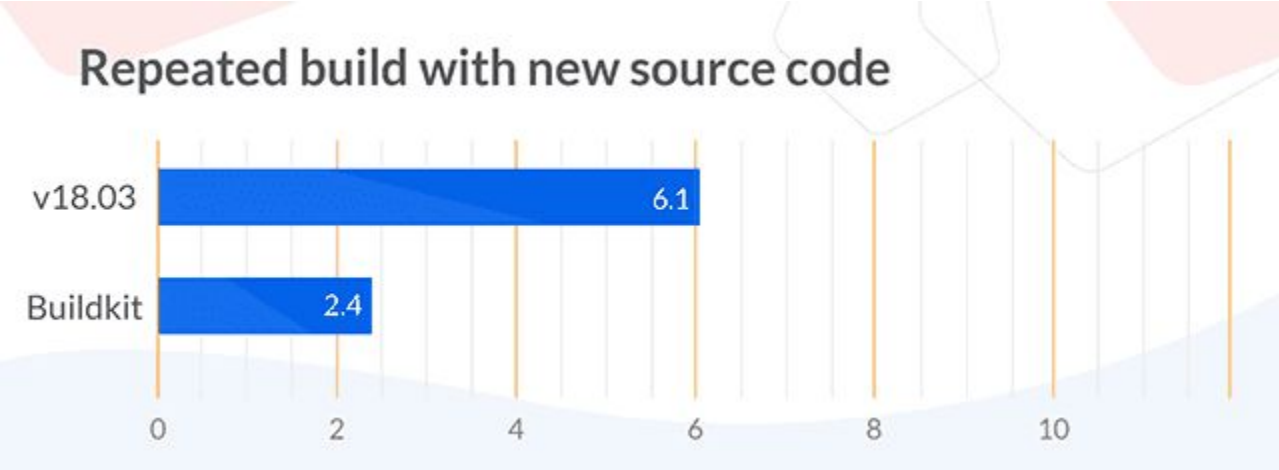
Buildkit performance benchmarking



Rerunning the same build with local cache the speed is 7x faster.

source: https://www.xenonstack.com/blog/docker-buildkit/

Buildkit performance benchmarking



Repeatable docker builds with new source code leads to 2.5 fx faster build speed.

source: https://www.xenonstack.com/blog/docker-buildkit/

Security

Avoid running your application as a root user

```
FROM debian
RUN useradd -ms /bin/bash  app
USER app
WORKDIR /app
```

Use --chown when running ADD or COPY

```
FROM ruby:2.7
RUN useradd -ms /bin/bash  app
USER app
WORKDIR /app

COPY --chown=app Gemfile Gemfile.lock ./
RUN bundle install

COPY --chown . ./
```

Consistency

Consistent image is easier to maintain

Pin base image version

```
FROM ruby

CMD ["ruby", "-e", "3", "+", "4"]

FROM ruby:2.7

CMD ["ruby", "-e", "3", "+", "4"]
```

Add .dockerignore file

```
# Ignore bundler config.
/.bundle
.bundleignore

# Ignore Byebug command history file.
/.byebug_history

# Ignore .git as it's not needed with the docker built.
/.git
/.gitignore
/.cache

yarn-error.log
/coverage/

config/master.key
yarn-debug.log
.yarn-integrity

# Ignore the default SQLite database.
/db/*.sqlite3
/db/*.sqlite3-journal

# Ignore all logfiles and tempfiles.
/log/*
/tmp/*
!/log/.keep
!/tmp/.keep

# Ignore uploaded files in development.
/storage/*
!/storage/.keep

*.md
```

Use official images when possible

```
FROM some-repo/ruby

FROM ruby:2.7
```

Set Locale

```
FROM ruby:2.7

RUN sed -i -e 's/# en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen && \
    locale-gen
ENV LANG=en_US.UTF-8 \
  LANGUAGE=en_US:en \
  LC_ALL=en_US.UTF-8
```

Often overlooked but very important

Questions?

Thank you!

bhushanlodha@gmail.com