

Dockerfile Best Practices



Tibor Vass
Docker, Inc.
@tiborvass



Sebastiaan van Stijn
Docker, Inc.
@thajztah



Dockerfile

Blueprint to build Docker images

Popular: 1+ million Dockerfiles on GitHub



BuildKit: builder v2

Concurrency, lazy context upload,
better caching, new Dockerfile features, ...

*Windows support
coming soon*



Use latest Docker, enable BuildKit today!

Docker client:

```
export DOCKER_BUILDKIT=1
```

Or, Docker daemon config:

```
{  
  "features": {"buildkit": true}  
}
```

<https://docs.docker.com/engine/reference/builder/>



docker docs



Search the docs

Guides

Product manuals

Glossary

Reference

Samples

File formats

Dockerfile reference

Compose file reference

Cloud stack file reference

Command-Line Interfaces (CLIs)

Application Programming Interfaces (APIs)

Drivers and specifications

Compliance control references

Estimated reading time: 69 minutes

Dockerfile reference

Docker can build images automatically by reading the instructions from a `Dockerfile`. A `Dockerfile` is a text document that contains all the commands a user could call on the command line to assemble an image. Using `docker build` users can create an automated build that executes several command-line instructions in succession.

This page describes the commands you can use in a `Dockerfile`. When you are done reading this page, refer to the `Dockerfile Best Practices` for a tip-oriented guide.

Usage

The `docker build` command builds an image from a `Dockerfile` and a `context`. The build's context is the set of files at a specified location `PATH` or `URL`. The `PATH` is a directory on your local filesystem. The `URL` is a Git repository location.

Improving Dockerfiles

Areas of improvements

- (Incremental) build time
- Image size
- Maintainability
- Security
- Consistency/Repeatability

Example project

Basic Java Spring Hello world web app

```
-rw-r--r-- 1    656 Dec  4 12:20 Dockerfile
drwxr-xr-x 2   6.1M Dec  4 09:44 docs/
-rw-r--r-- 1   1.7K Dec  3 09:48 pom.xml
-rw-r--r-- 1   1.0K Dec  4 10:12 README.md
drwxr-xr-x 4    44K Dec  3 09:48 src/
drwxr-xr-x 2   17M Dec  4 09:50 target/
```


Let's improve this Dockerfile

```
FROM debian
COPY . /app
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk ssh emacs
CMD ["java", "-jar", "/app/target/app.jar"]
```

Let's improve this Dockerfile

```
FROM debian
COPY . /app
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk ssh emacs vim
CMD ["java", "-jar", "/app/target/app.jar"]
```



Incremental build time

Make build cache your friend

Order matters for caching

```
FROM debian
```

```
COPY . /app
```

```
RUN apt-get update
```

```
RUN apt-get -y install openjdk-8-jdk ssh vim
```

```
COPY . /app
```

```
CMD ["java", "-jar", "/app/target/app.jar"]
```

Order from least to most frequently changing content.

More specific COPY to limit cache busts

```
FROM debian
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk ssh vim
COPY . /app
COPY target/app.jar /app
CMD ["java", "-jar", "/app/target/app.jar"]
```

Only copy what's needed. Avoid "COPY ." if possible

Line buddies: apt-get update & install

```
FROM debian
```

```
RUN apt-get update
```

```
RUN apt-get -y install openjdk-8-jdk ssh vim
```

```
RUN apt-get update \  
&& apt-get -y install \  
openjdk-8-jdk ssh vim
```

```
COPY target/app.jar /app
```

```
CMD ["java", "-jar", "/app/app.jar"]
```

Prevents using an outdated package cache

Reduce image size

Faster deploys, smaller attack surface

Remove unnecessary dependencies

```
FROM debian
RUN apt-get update \
    && apt-get -y install --no-install-recommends \
        openjdk-8-jdk ssh vim
COPY target/app.jar /app
CMD ["java", "-jar", "/app/app.jar"]
```


Remove package manager cache

```
FROM debian
RUN apt-get update \
    && apt-get -y install --no-install-recommends \
        openjdk-8-jdk \
    && rm -rf /var/lib/apt/lists/*
COPY target/app.jar /app
CMD ["java", "-jar", "/app/app.jar"]
```

Remove package manager cache

```
FROM debian
RUN apt-get update \
    && apt-get -y install --no-install-recommends \
        openjdk-8-jdk \
    && rm -rf /var/lib/apt/lists/*
COPY target/app.jar /app
CMD ["java", "-jar", "/app/app.jar"]
```

Maintainability

Use official images where possible

- Reduce time spent on maintenance
(frequently updated with fixes)
- Reduce size (shared layers between images)
- Pre-configured for container use
- Built by smart people 🤖

Use official images when possible

```
FROM debian  
RUN apt-get update \  
&& apt-get -y install --no-install-recommends \  
openjdk-8-jdk \  
&& rm -rf /var/lib/apt/lists/*  
FROM openjdk  
COPY target/app.jar /app  
CMD ["java", "-jar", "/app/app.jar"]
```

Use more specific tags

```
FROM openjdk:latest
```

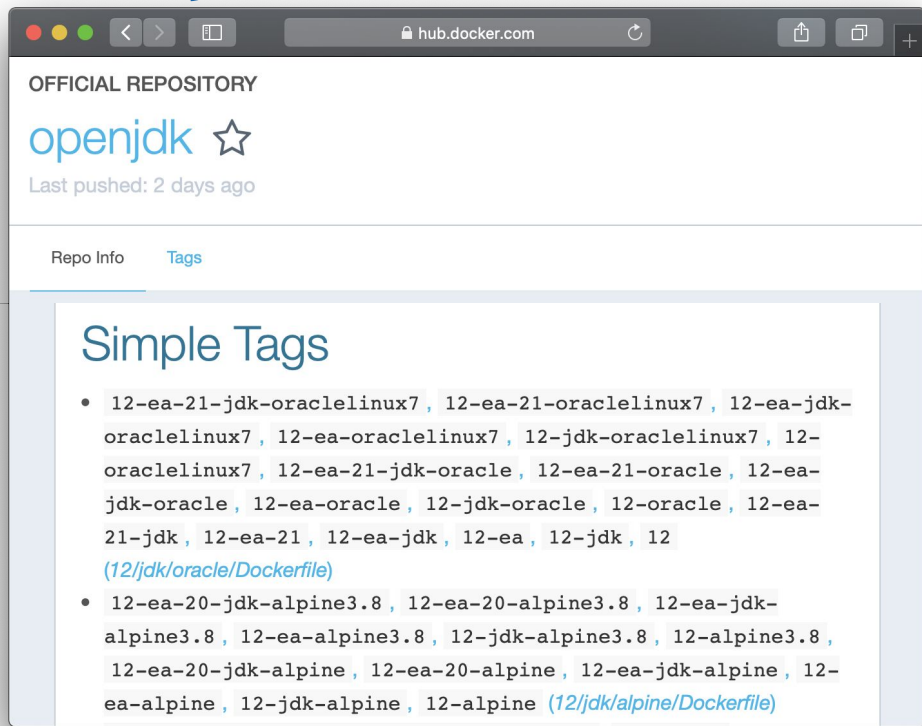
```
FROM openjdk:8
```

```
COPY target/app.jar /app
```

```
CMD ["java", "-jar", "/app/app.jar"]
```

The "latest" tag is a rolling tag. Be specific, to prevent unexpected changes in your base image.

Pick your variant



Read the image's documentation on Docker Hub

https://hub.docker.com/_/openjdk

Look for minimal flavors

REPOSITORY	TAG	SIZE
openjdk	8	624MB
openjdk	8-jre	443MB
openjdk	8-jre-slim	204MB
openjdk	8-jre-alpine	83MB

Just using a different base image reduced the image size by 540 MB

Reproducibility

The Dockerfile as blueprint,
source code the source of truth

Build from source in a consistent environment

Make the Dockerfile your blueprint:

- It describes the build environment
- Correct versions of build tools installed
- Prevent inconsistencies between environments
- There may be system dependencies
- *The "source of truth" is the source code, not the build artifact*

Build from source in a consistent environment

~~FROM openjdk:8-jre-alpine~~

FROM maven:3.6-jdk-8-alpine

WORKDIR /app

~~COPY app.jar /app~~

COPY pom.xml .

COPY src ./src

RUN mvn -e -B package

CMD ["java", "-jar", "/app/app.jar"]

Build from source in a consistent environment

```
FROM maven:3.6-jdk-8-alpine
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```

Fetch dependencies in a separate step

```
FROM maven:3.6-jdk-8-alpine
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```

Identify build dependencies

```
FROM maven:3.6-jdk-8-alpine
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```

Multi-stage builds to remove build deps

```
FROM maven:3.6-jdk-8-alpine AS builder  
WORKDIR /app  
COPY pom.xml .  
RUN mvn -e -B dependency:resolve  
COPY src ./src  
RUN mvn -e -B package  
CMD ["java", "-jar", "/app/app.jar"]
```

```
FROM openjdk:8-jre-alpine  
COPY --from=builder /app/target/app.jar /  
CMD ["java", "-jar", "/app.jar"]
```

Multi-stage builds to remove build deps

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
WORKDIR /app
```

```
COPY pom.xml .
```

```
RUN mvn -e -B dependency:resolve
```

```
COPY src ./src
```

```
RUN mvn -e -B package
```

```
FROM openjdk:8-jre-alpine
```

```
COPY --from=builder /app/target/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```


Multi-stage Dockerfiles

Not just for reducing image size

Projects with many stages

- Moby: 16 stages

<https://github.com/moby/moby/blob/master/Dockerfile>

- BuildKit: 44 stages

<https://github.com/moby/buildkit/blob/master/hack/dockerfiles/test.buildkit.Dockerfile>

Multi-stage use cases

- Separate build from runtime environment (shrinking image size)
- Slight variations on images (DRY)
- Build/dev/test/lint/... specific environments
- Delinearizing your dependencies (concurrency)
- Platform-specific stages

Building specific stages with --target

```
FROM image_or_stage AS stage_name
```

...

```
$ docker build --target stage_name
```

Different image flavors

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-jessie AS release-jessie  
COPY --from=builder /app/target/app.jar /  
CMD ["java", "-jar", "/app.jar"]
```

```
FROM openjdk:8-jre-alpine AS release-alpine  
COPY --from=builder /app/target/app.jar /  
CMD ["java", "-jar", "/app.jar"]
```

```
$ docker build --target release-jessie .
```

Different image flavors

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-jessie AS release-jessie
```

```
COPY --from=builder /app/target/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

```
FROM openjdk:8-jre-alpine AS release-alpine
```

```
COPY --from=builder /app/target/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

```
$ docker build --target release-jessie .
```

Different image flavors

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-jessie AS release-jessie
```

```
COPY --from=builder /app/target/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

```
FROM openjdk:8-jre-alpine AS release-alpine
```

```
COPY --from=builder /app/target/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

```
$ docker build --target release-jessie .
```

Different image flavors (DRY / global ARG)

```
ARG flavor=alpine
```

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-$flavor AS release
```

```
COPY --from=builder /app/target/app.jar /  
CMD ["java", "-jar", "/app.jar"]
```

```
$ docker build --target release  
--build-arg flavor=jessie .
```


Various environments: build, dev, test, lint, ...

Examples of possible stage layout:

- builder: all build dependencies
- build (or binary): builder + built artifacts
- cross: same as build but for multiple platforms
- dev: build(er) + dev/debug tools
- lint: minimal lint dependencies
- test: all test dependencies + build artifacts to be tested
- release: final minimal image with build artifacts

Various environments: build, dev, test, lint, ...

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-alpine AS lint
```

```
RUN wget https://github.com/checkstyle/checkstyle/releases/download/checkstyle-8.15/checkstyle-8.15-all.jar
```

```
COPY checks.xml .
```

```
COPY src /src
```

```
RUN java -jar checkstyle-8.15-all.jar -c checks.xml /src
```

Various environments: build, dev, test, lint, ...

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM openjdk:8-jre-alpine AS release  
COPY --from=builder /app/target/app.jar /  
CMD ["java", "-jar", "/app.jar"]
```

```
FROM builder AS dev  
RUN apk add --no-cache strace vim tcpdump  
ENTRYPOINT ["ash"]
```

Various environments: build, dev, test, lint, ...

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
RUN mvn -e -B package -DskipTests
```

```
FROM builder AS unit-test
```

```
RUN mvn -e -B test
```

```
FROM release AS integration-test
```

```
RUN apk add --no-cache curl
```

```
RUN ./test/run.sh
```

BRACE YOURSELVES

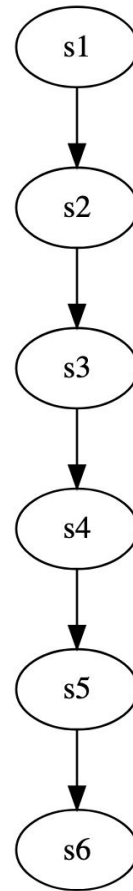


CONCURRENCY IS COMING



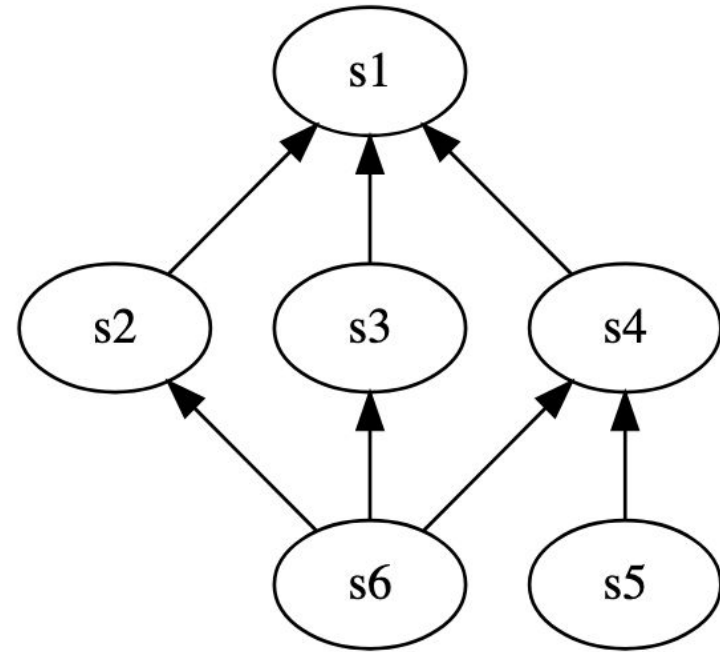
From linear Dockerfile stages ...

- all stages are executed in sequence
- without BuildKit, unneeded stages are unnecessarily executed but discarded



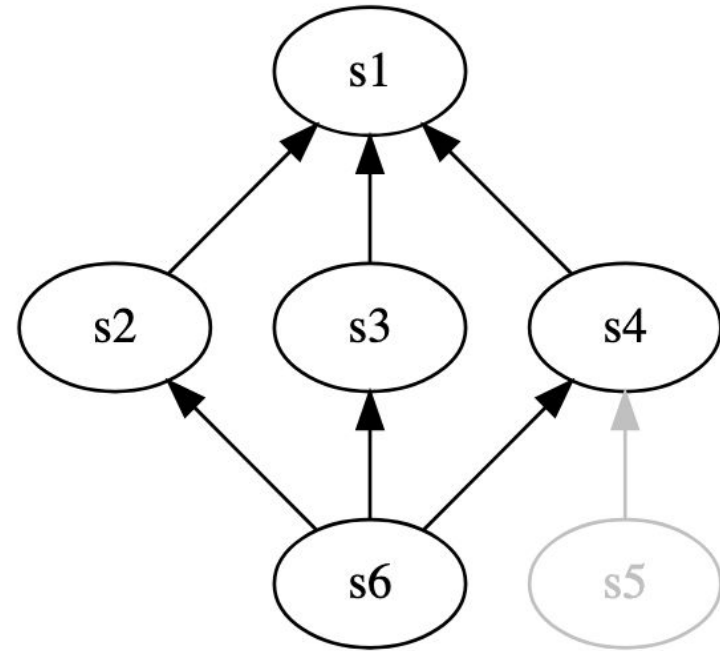
... to multi-stage graphs with BuildKit

- BuildKit traverses from bottom (stage name from `--target`) to top



... to multi-stage graphs with BuildKit

- BuildKit traverses from bottom (stage name from `--target`) to top
- Unneeded stages are not even considered



Multi-stage: build concurrently

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
...
```

```
FROM tiborvass/whalesay AS assets
```

```
RUN whalesay "Hello DockerCon!" > /out/assets.html
```

```
FROM openjdk:8-jre-alpine AS release
```

```
COPY --from=builder /app/app.jar /
```

```
COPY --from=assets /out /assets
```

```
CMD ["java", "-jar", "/app.jar"]
```

Multi-stage: build concurrently

```
FROM maven:3.6-jdk-8-alpine AS builder-base
...
FROM gcc:8-alpine AS builder-someClib
...
RUN git clone ... \
    ./configure --prefix=/out && make && make install
FROM g++:8-alpine AS builder-someCPPlib
...
RUN git clone ... \
    cmake ...

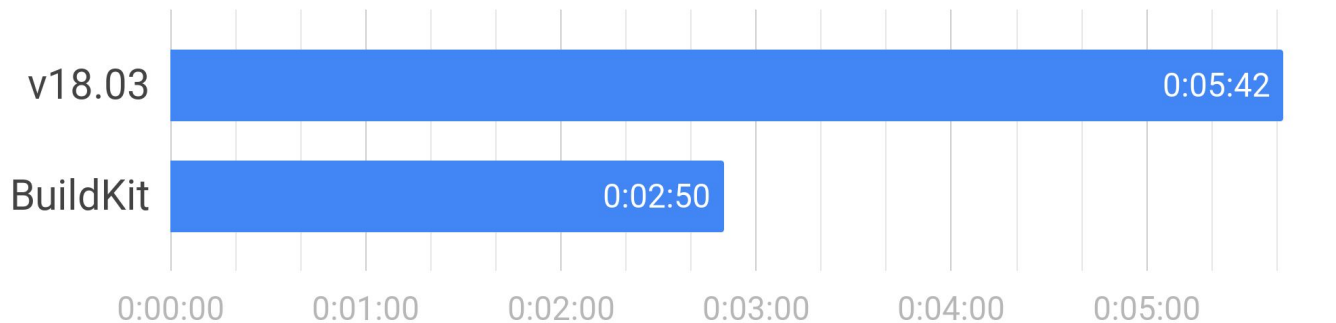
FROM builder-base AS builder
COPY --from=builder-someClib /out /
COPY --from=builder-someCPPlib /out /
...
```

Concurrency pattern:
multiple
COPY --from ...
COPY --from ...

Benchmarks

Based on github.com/moby/moby Dockerfile, master branch. **Smaller** is better.

Time for full build from empty state

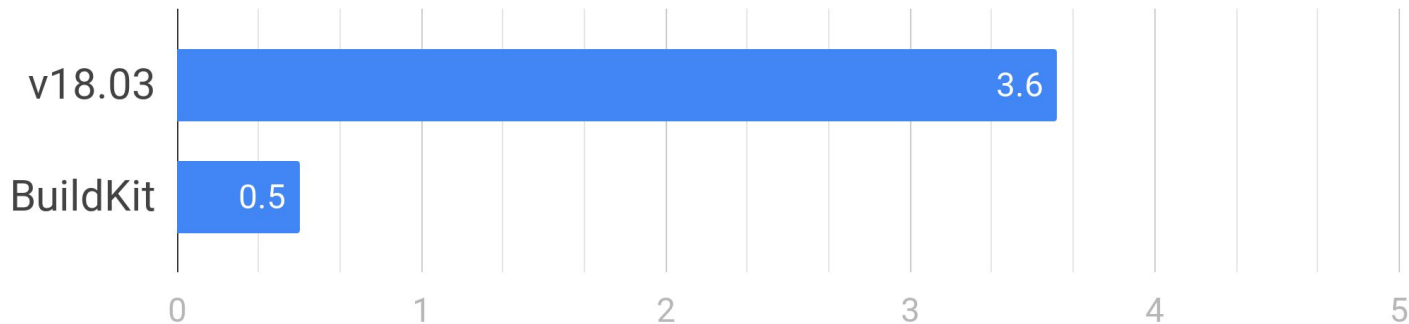


**2.0x
faster**

Benchmarks

Based on github.com/moby/moby Dockerfile, master branch. **Smaller** is better.

Repeated build with matching cache

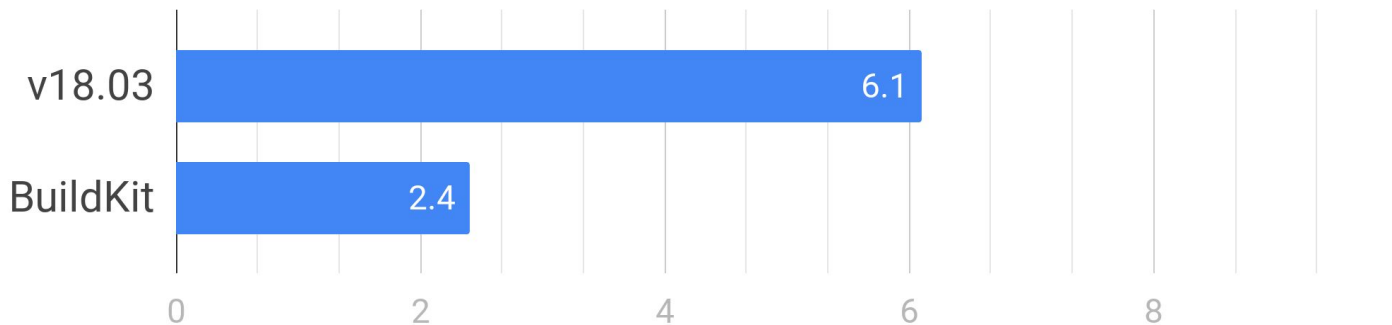


**7.2x
faster**

Benchmarks

Based on github.com/moby/moby Dockerfile, master branch. **Smaller** is better.

Repeated build with new source code



**2.5x
faster**

New Dockerfile features

Enabling new features

```
# syntax=docker/dockerfile:1.0-experimental
```

```
FROM maven:3.6-jdk-8-alpine AS builder
WORKDIR /app
COPY . /app
RUN mvn -e -B package
```


```
FROM openjdk:8-jre-alpine
COPY --from=builder /app/app.jar /
CMD ["java", "-jar", "/app.jar"]
```

Experimental as in, not in mainline Dockerfile syntax.

The 1.0-experimental image will not break in the future.

For more details: [docs/experimental-syntaxes.md](https://github.com/moby/buildkit/blob/master/frontend/dockerfile/docs/experimental.md)

<https://github.com/moby/buildkit/blob/master/frontend/dockerfile/docs/experimental.md>

 moby / **buildkit**

Watch 61

Unstar 1,353

Fork 133

<> Code

Issues 98

Pull requests 8


Actions


Projects 0

Insights

Settings

Branch: master **buildkit / frontend / dockerfile / docs / experimental.md** Find file Copy path

 **chendave** doc: make sure the intended message is redirected to file successfully 63771e6 15 days ago

3 contributors 

138 lines (100 sloc) 5.69 KB

Raw Blame History

Dockerfile frontend experimental syntaxes

Note for Docker users

If you are using Docker v18.06 or later, BuildKit mode can be enabled by setting `export DOCKER_BUILDKIT=1` on the client side. Docker v18.06 also requires the daemon to be [running in experimental mode](#).

You need to use `docker build` CLI instead of `buildctl` CLI mentioned in this document. See [the docker build document](#) for the usage.

Context mounts (v18.09+ w/ BuildKit)

```
# syntax=docker/dockerfile:1.0-experimental
```

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
WORKDIR /app
```

```
COPY . /app
```

```
RUN --mount=target=. mvn -e -B package -DoutputDirectory=/
```

```
FROM openjdk:8-jre-alpine
```

```
COPY --from=builder /app/app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

Context mounts (v18.09+ w/ BuildKit)

```
# syntax=docker/dockerfile:1.0-experimental
```

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
WORKDIR /app
```

```
RUN --mount=target=. mvn -e -B package -DoutputDirectory=/
```

```
FROM openjdk:8-jre-alpine
```

```
COPY --from=builder /app.jar /
```

```
CMD ["java", "-jar", "/app.jar"]
```

Cache dependencies (before BuildKit)

```
FROM maven:3.6-jdk-8-alpine
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```

Application cache (v18.09+ w/ BuildKit)

```
# syntax=docker/dockerfile:1.0-experimental
```

```
FROM maven:3.6-jdk-8-alpine AS builder
```

```
WORKDIR /app
```

```
RUN --mount=target=. --mount=type=cache,target=/root/.m2 \  
    && mvn package -DoutputDirectory=
```

```
FROM openjdk:8-jre-alpine
```

```
COPY --from=builder /app.jar /
```

```
CMD ["java", "/app.jar"]
```

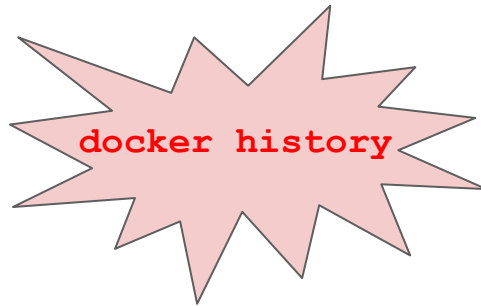
apt: /var/lib/apt/lists
go: ~/.cache/go-build
go-modules: \$GOPATH/pkg/mod
npm: ~/.npm
pip: ~/.cache/pip

Secrets (DON'T DO THIS)

```
FROM baseimage
RUN ...
ENV AWS_ACCESS_KEY_ID=...
ENV AWS_SECRET_ACCESS_KEY=...
RUN ./fetch-assets-from-s3.sh
RUN ./build-scripts.sh
```

Secrets (DON'T DO THIS EITHER)

```
FROM baseimage
RUN ...
ARG AWS_ACCESS_KEY_ID
ARG AWS_SECRET_ACCESS_KEY
RUN ./fetch-assets-from-s3.sh
RUN ./build-scripts.sh
```



```
$ docker build --build-arg \
AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID ... .
```

Secrets (DO THIS, v18.09+ w/ BuildKit)

```
# syntax=docker/dockerfile:1-experimental
FROM baseimage
RUN ...
RUN --mount=type=secret,id=aws,target=/root/.aws/credentials
,required ./fetch-assets-from-s3.sh
RUN ./build-scripts.sh
```

```
$ docker build --secret id=aws,src=~/.aws/credentials .
```


Private git repos (DON'T DO THIS)

```
FROM baseimage
```

```
COPY ./keys/private.pem /root/.ssh/private.pem
```

```
ARG REPO_REF=19ba7bcd9976ef8a9bd086187df19ba7bcd997f2
```

```
RUN git clone git@github.com:org/repo /work && cd /work \  
    && git checkout -b $REPO_REF
```

Private git repos (DO THIS, v18.09+ w/ BuildKit)

```
FROM alpine
RUN apk add --no-cache openssh-client
RUN mkdir -p -m 0700 ~/.ssh && ssh-keyscan github.com >>
  ~/.ssh/known_hosts
ARG REPO_REF=19ba7bcd9976ef8a9bd086187df19ba7bcd997f2
RUN --mount=type=ssh,required \
  git clone git@github.com:org/repo /work && cd /work \
  && git checkout -b $REPO_REF

$ eval $(ssh-agent)
$ ssh-add ~/.ssh/id_rsa
$ docker build --ssh=default .
```

Improvements recap

We went from:

- inconsistent build/dev/test environments
- bloated image
- slow build and incremental build times (cache busts)
- building insecurely

To:

- consistent build/dev/test environments
- minimal image
- very fast build and incremental build times
- building more securely

Read more on blog posts



Tõnis Tiigi [Follow](#)
Jun 4 · 6 min read

Advanced multi-stage build patterns

```
FROM buildkit-export AS buildkit-buildkitd.oci_only~
COPY --from=buildkitd.oci_only /usr/bin/buildkitd.oci_only /usr/bin/~
COPY --from=buildctl /usr/bin/buildctl /usr/bin/~
ENTRYPOINT ["buildkitd.oci_only"]~

# Copy together all binaries for containerd worker mode~
FROM buildkit-export AS buildkit-buildkitd.containerd_only~
COPY --from=runc /usr/bin/runc /usr/bin/~
COPY --from=buildkitd.containerd_only /usr/bin/buildkitd.containerd_only /usr/bin/~
COPY --from=buildctl /usr/bin/buildctl /usr/bin/~
ENTRYPOINT ["buildkitd.containerd_only"]~

FROM alpine AS containerd-runtime~
COPY --from=runc /usr/bin/runc /usr/bin/~
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/containerd* /usr/bin/~
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/ctr /usr/bin/~
VOLUME /var/lib/containerd~
VOLUME /run/containerd~
ENTRYPOINT ["containerd"]~

FROM buildkit-${BUILDKIT_TARGET}~
```

<https://medium.com/@tonistiigi/advanced-multi-stage-build-patterns-6f741b852fae>

<https://medium.com/@tonistiigi/build-secrets-and-ssh-forwarding-in-docker-18-09-ae8161d066>

Build secrets and SSH forwarding in Docker 18.09



Tõnis Tiigi [Follow](#)
Nov 8 · 6 min read

```
3. docker build --ssh=default . (docker)

# docker build --ssh=default .
[+] Building 9.0s (8/9)
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 386B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> resolve image config for docker.io/docker/dockerfile-upstream:experimental 1.0s
=> CACHED docker-image://docker.io/docker/dockerfile-upstream:experimental@ 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 0.3s
=> [1/4] FROM docker.io/library/alpine@sha256:621c2f39f8133acb8e64023a94dbdf 0.0s
=> resolve docker.io/library/alpine@sha256:621c2f39f8133acb8e64023a94dbdf 0.0s
=> sha256:621c2f39f8133acb8e64023a94dbdf0d5ca81896102b9e5 2.03kB / 2.03kB 0.0s
=> sha256:02892826401a9d18f0ea01f8a2f35d328ef039db4e1edcc45c6 528B / 528B 0.0s
=> sha256:196d12cf6ab19273823e700516e98eb1910b03b17840f9d 1.51kB / 1.51kB 0.0s
=> [2/4] RUN apk add --no-cache openssh-client git 2.6s
=> [3/4] RUN mkdir -p -m 0600 ~/.ssh && ssh-keyscan github.com >> ~/.ssh/kno 2.7s
=> [4/4] RUN --mount=type=ssh ssh git@github.com 2>&1 | grep "Hi tonistiigi" 1.3s
```



Thank you!

- Multi-stage, multi-stage, multi-stage
- `DOCKER_BUILDKIT=1`

OSS Summit: Advanced BuildKit
sessions on Thursday, May 2 at 12:30pm
in room 2020

Follow us @tiborvass @thaJeztah