Docker Multi-Stage Builds

Ryan Blunden

Developer, DevOps Coach, Instructor.

Aspiring Docker Certified Trainer.

Focussing on Docker, DevOps and DevSecOps.

What is good code?

- Explicit and simple operations.
- Easy to read.
- Easy to understand.
- Easy to maintain.

Optimise code for humans while enabling computers to compile/execute.

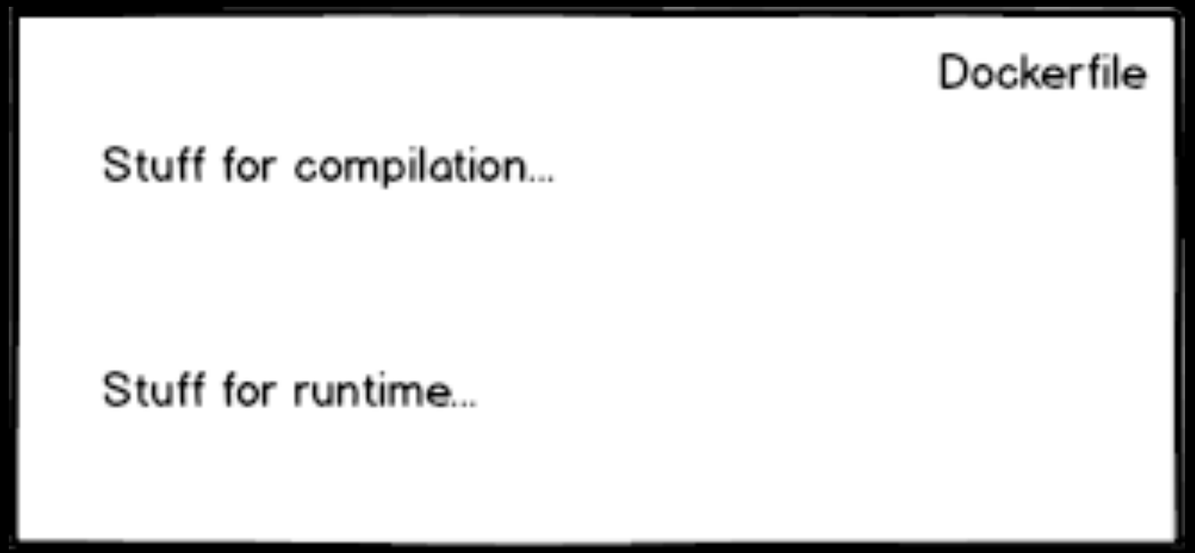
Current Dockerfile requires sacrificing readability in order to reduce the number of layers and image size.

```
# add gosu for easy step-down from root
ENV GOSU_VERSION 1.7
RUN set -x \
    & apt-get update & apt-get install y -no-install-recommends ca-certificates wget && rm -rf /var/lib/apt/lists/* \
    && wget O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu$(dpkg -printarchitecture)" \
    && wget O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu$(dpkg -printarchitecture).asc" \
    && export GNUPGHOME="$(mktemp -d)" \
    && gpg -keyserver ha.pool.skskeyservers.net -recvkeys B42F6819007F00F88E364FD4036A9C25BF357DD4 \
    && gpg -batch -verify /usr/local/bin/gosu.asc /usr/local/bin/gosu \
    && rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc \
    && chmod +x /usr/local/bin/gosu \
    && gosu nobody true \
    && apt-get purge y -auto-remove ca-certificates wget
```

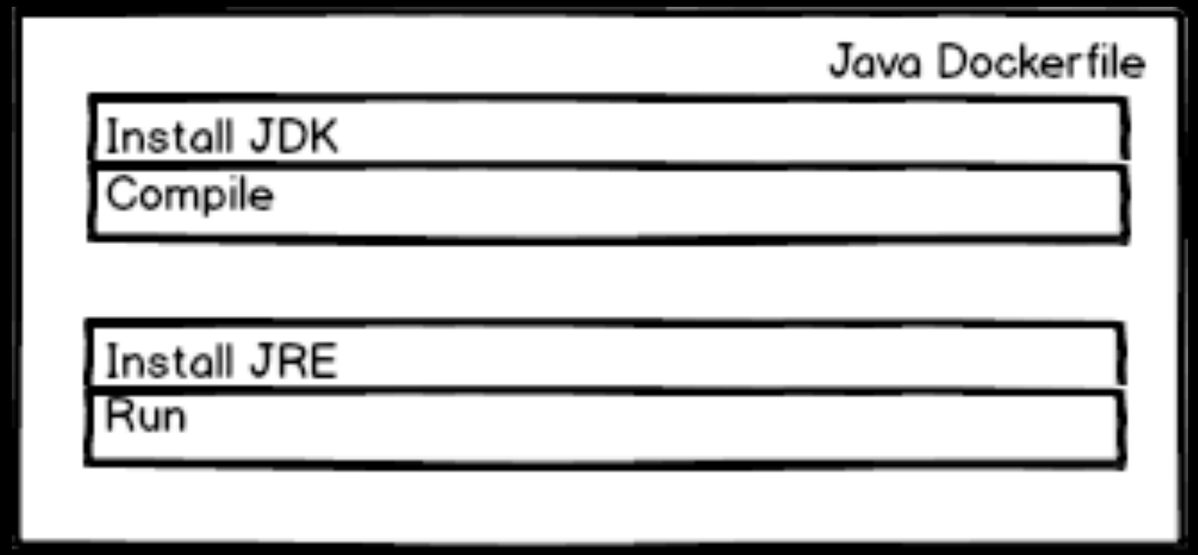
So we want the smallest possible images while writing the best Dockerfile code for humans.

A Dockerfile could be thought of as a recipe for baking a cake

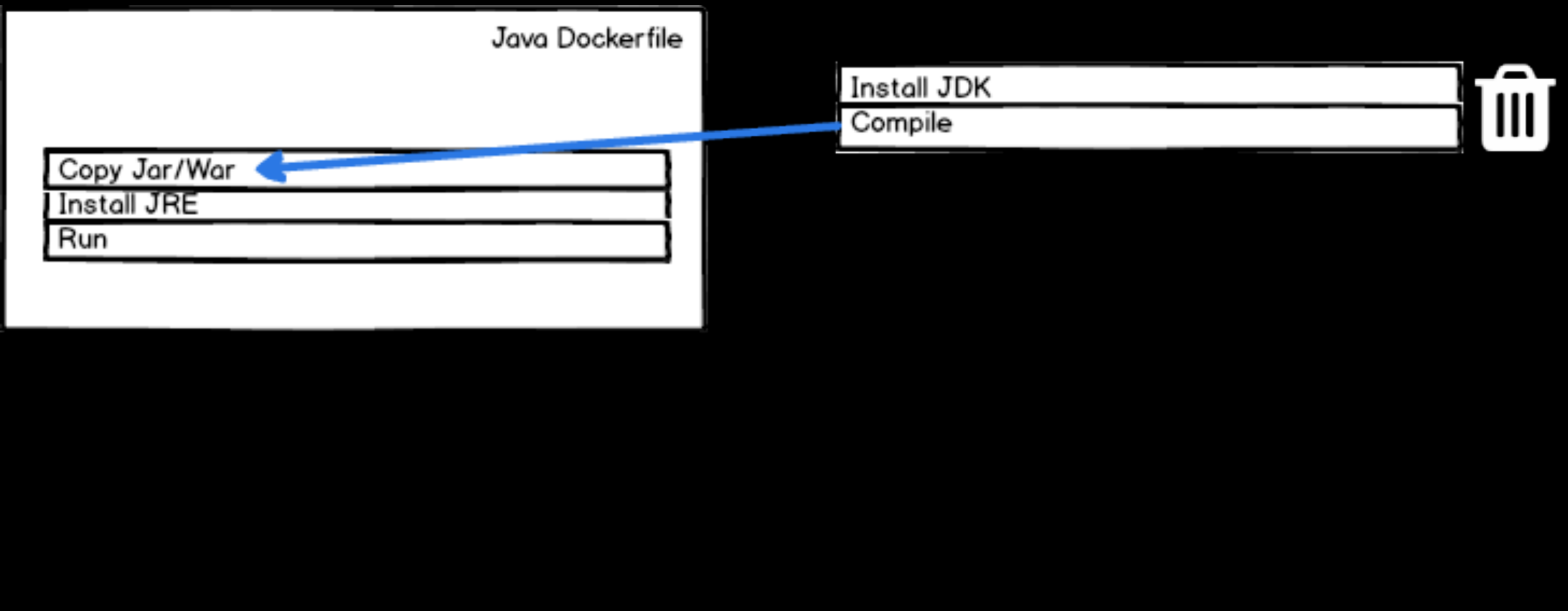
Except unlike a cake, a Dockerfile often contains ingredients I don’t want to consume once baked.


Dockerfile
Stuff for compilation...
Stuff for runtime...

Take a Java image.

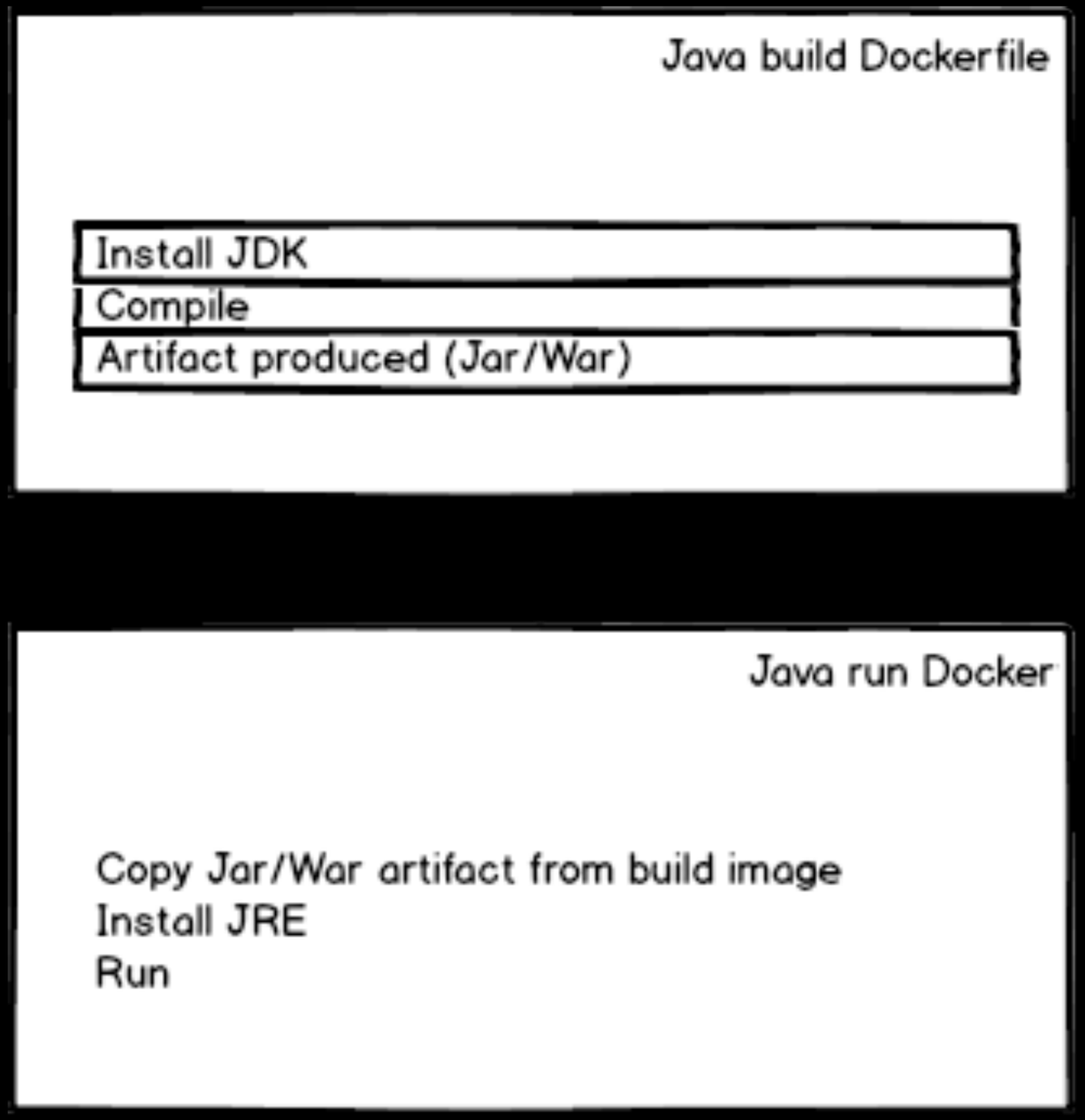

Java Dockerfile
Install JDK
Compile
Install JRE
Run

We don’t need or want the JDK at the end. We just needed it for compilation.


Java Dockerfile
Copy Jar/War
Install JRE
Run
Install JDK
Compile

Enter the “Builder Pattern”.

https://medium.com/@alexeiled/docker-pattern-the-build-container-b0d0e86ad601


Java build Dockerfile
Install JDK
Compile
Artifact produced (Jar/War)
1. Compile and build artifact.
Java run Docker
Copy Jar/War artifact from build image
Install JRE
Run
2. Copy artifact and run

It helped us create lean images, but readability suffered.

Docker needed to solve this problem within the Docker file.

Hello Docker Multi-Stage Builds!

Currently CE Edge Docker 17.05

https://docs.docker.com/engine/userguide/eng-image/multistage-build/

Docker Multi-Stage Builds mean:

- Dockerfiles now allow multiple **FROM** statements.
- Each **FROM** statement marks the start of a new **build context** (accessed by number or name).
- Each new context (FROM) is like starting a new Dockerfile.
- You then copy what you want from any previous context into the new context.
- Whichever is the last FROM statement is the final base image

Lets look at some code.

https://github.com/arun-gupta/docker-java-multistage

Single stage Dockerfile (816MB)

```
FROM maven:3.5-jdk-8

COPY src /usr/src/myapp/src
COPY pom.xml /usr/src/myapp
RUN mvn -f /usr/src/myapp/pom.xml clean package

ENV WILDFLY_VERSION 10.1.0.Final
ENV WILDFLY_HOME /usr

RUN cd $WILDFLY_HOME & curl http://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-$WILDFLY_VERSION.tar.gz | tar zx
& mv $WILDFLY_HOME/wildfly-$WILDFLY_VERSION $WILDFLY_HOME/wildfly

RUN cp /usr/src/myapp/target/people-1.0-SNAPSHOT.war $WILDFLY_HOME/wildfly/standalone/deployments/people.war

EXPOSE 8080

CMD "/usr/wildfly/bin/standalone.sh", "-b", "0.0.0.0"
```

More complex, can't use the Wildfly image as it needs the JDK.

More layers. Larger image size.

Multi-Stage Dockerfile (584MB, almost 30% smaller)

```
FROM maven:3.5-jdk-8 as BUILD

COPY src /usr/src/myapp/src
COPY pom.xml /usr/src/myapp
RUN mvn -f /usr/src/myapp/pom.xml clean package

FROM jboss/wildfly:10.1.0.Final

COPY -from=BUILD /usr/src/myapp/target/people1.0-SNAPSHOT.war /opt/jboss/wildfly/standalone/deployments/people.war
```

More readable. Easier to understand. Explicit. Simple.

Docker Multi-Stage Builds give us:

- Smallest possible image sizes - reduced storage and network costs
- Better code - improved readability, maintenance.
- Everything in the one Dockerfile - Less complexity.

Credit: http://knowyourmeme.com/photos/68108-win-epic-win-for-the-win

Thank-you!

Ryan Blunden

- Email: Ryan.Blunden@gmail.com
- LinkedIn: https://au.linkedin.com/in/ryanblunden
- Twitter: @ryan_blunden