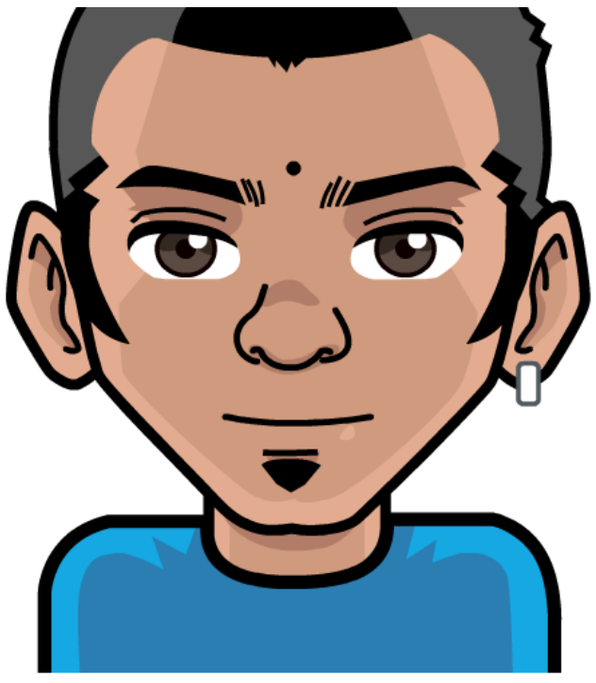
RAJU GANDHI

DEEP DIVE INTO DOCKERFILES



RAJU GANDHI

@LOOSELYTYPED

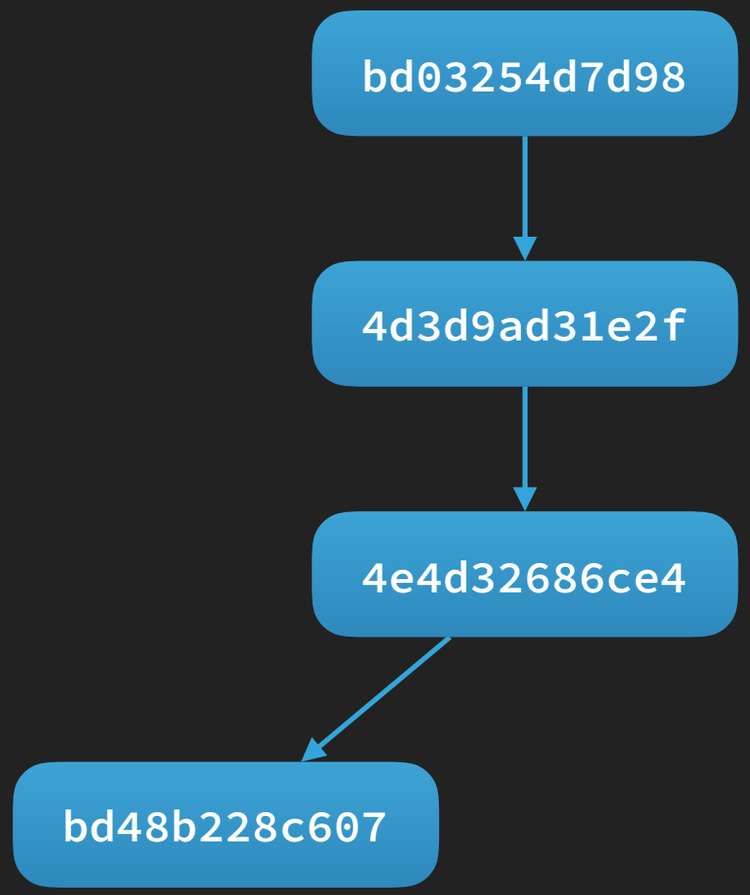
CTO - INTEGRALLIS SOFTWARE

DOCKERFILES?

THE WHAT AND WHY

- A set of instructions to build a Docker image
- Plain text, version controlled
- Provides insight into the image needs/capabilities/intents

UNION FILE SYSTEM

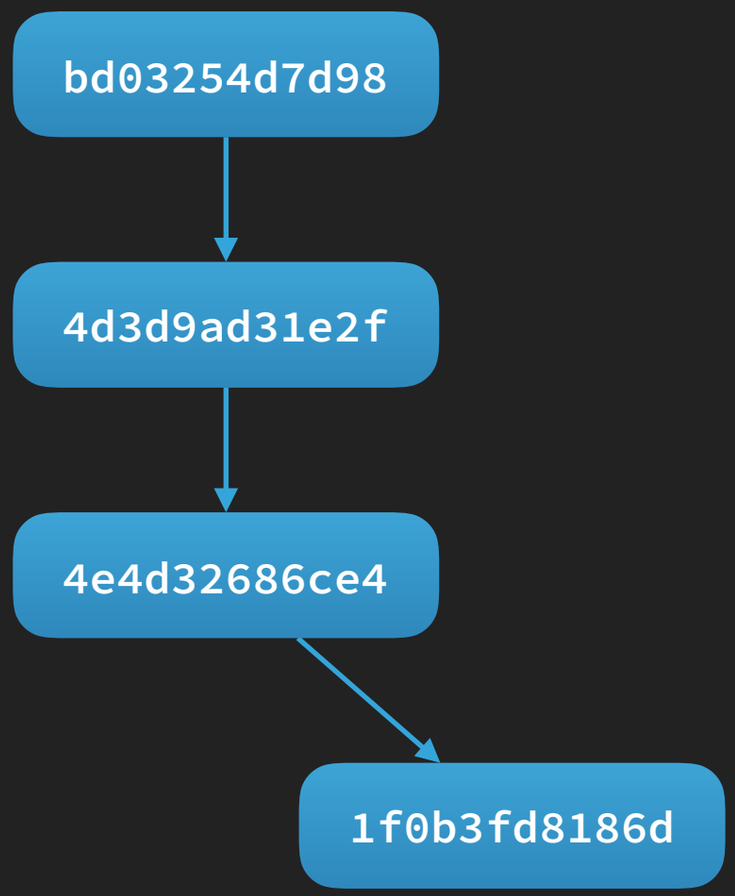



```
FROM openjdk:8u131-jre

RUN apt-get update && apt-get install -y netcat

COPY build/libs/app-fat.jar /var/app.jar

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

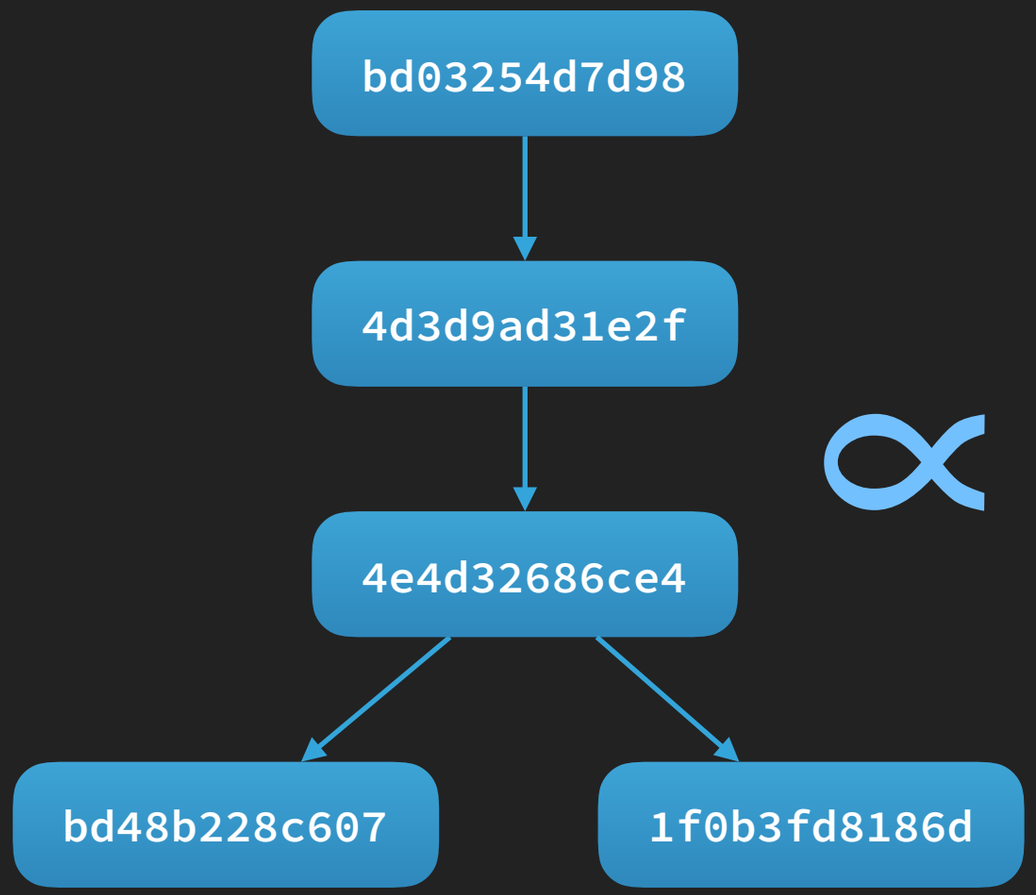



```
FROM openjdk:8u131-jre

RUN apt-get update && apt-get install -y netcat

COPY build/libs/app-fat.jar /var/app.jar

CMD ["ls", “-al"]
```




```
FROM openjdk:8u131-jre

RUN apt-get update && apt-get install -y netcat

COPY build/libs/app-fat.jar /var/app.jar

CMD [“...”]
```

THE CODEZ

- Each instruction adds a “write-only” layer to the UFS
- Layers are cached for future builds
- Deletes *increase* the image size

DON'TS

- Don’t sacrifice readability/maintainability!
- Don’t add unnecessary instructions
- Beware that whitespace changes can invalidate cache
  - Think of formatting upfront

DO'S

- Be cognizant of layers and image sizes
- Order matters!
  - Keep the “moving” bits lower in the Dockerfile
  - Keep “informational” bits lower in the Dockerfile

ONE CONCERN PER CONTAINER . . . FOR THE MOST PART

@LOOSELYTYPED

FROM

NOTES

- Implies “ancestry”
- Has to be the first line
- Has implications on `WORKDIR`,`USER`, `ENTRYPOINT` (and `CMD`), and `ONBUILD`, `EXPOSE` and other commands
- Create a base image with `FROM scratch`

DON'TS

- Multiple `FROM`s are allowed. Do **NOT** do this

DO'S

- Pin down the exact tag
  - Do not use “latest” tag
- Inspect ancestor images for `USER`s, `PORT`s, `ENV`s, `VOLUME`s, `LABEL`s and anything that can be inherited

RUN

DON'TS

- Be cognizant of the effects (and drawbacks) of caching
- Do not do OS level upgrades (eg. `RUN dist-upgrade`)

DOS

- Group common operations
  - Clean up as well (reduces image sizes)
- Use multiline (\) to make PR / auditing easier

ADD/COPY

DON'TS

- Avoid ADD
- Do not leave “residual” artifacts

DO'S

- Instead of ADD
  - Combine COPY and RUN
    - OR RUN with wget/curl/tar/unzip
  - See DO'S under RUN
- Be mindful of what you put in the `.dockerignore` file

USER

DON'TS

- Do not switch USER often
- Avoid using root

DO'S

- Create a user (if you can) for your service
  - `RUN groupadd -r myuser && useradd -r -g myapp myuser`
- `USER myuser`

LABELS

DONT'S

- Define individual labels separately

DO'S

- Use them liberally
- Labels can see ENV variables. Use this!
  - BUILD_NUMBER, GIT_SHA
- Use both image (compile) and container (run) LABELS
- Apply a standard convention
- Build tooling on top of the conventions

ENTRYPOINT/CMD

DONT'S

- Avoid the “shell” form

DO'S

- Use the “exec” form
  - Shell expansion will **not** happen!
- Use `ENTRYPOINT` and `CMD` together
- Use a “entrypoint-script”
  - Always “exec” (or “gosu”)

EXPOSE

DON'TS

- Avoid "`docker run -P`"

DO'S

- Do document the ports your application needs exposed

ENV

DO'S

- Use them for documentation/refactoring
- Use `docker run <image-name> env`
  - Or `docker inspect`
- Use them in conjunction with ARG
  - And default values

```
ARG PROJECT_VERSION
ENV PROJECT_VERSION ${PROJECT_VERSION:-2.3}

# ENVs are available in the container
# docker run -itP --rm <image-name> env
# RUN supports ENV expansion
# RUN wget -O ${PROJECT_URL}
```

CLOSING THOUGHTS

DO'S

- Do create your own "ancestry" of images
- Consider using multi-stage builds

```
FROM openjdk:8u141-jdk as builder

ENV VERSION=3.2
ENV GRADLE_HOME=/gradle-$VERSION
ENV PATH=$PATH:$GRADLE_HOME/bin

RUN apt-get update \
    && apt-get install -y \
       unzip \
       curl \
    && curl -L https://someurl/gradle-$VERSION-bin.zip -o gradle-$VERSION-bin.zip \
    && unzip gradle-$VERSION-bin.zip \
    && echo "export GRADLE_HOME=$GRADLE_HOME" >> $HOME/.bashrc \
    && echo "export PATH=$PATH:$GRADLE_HOME/bin" >> $HOME/.bashrc \
    && /bin/bash -c 'source $HOME/.bashrc' \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /code
ADD . ./

RUN ["./gradlew", "test"]
RUN ["./gradlew", "jar"]


FROM openjdk:8u131-jre
WORKDIR /code
COPY --from=builder /code/build/libs/code.jar .
CMD ["java", "-jar", "code.jar"]
```

THANKS!!

RAJU GANDHI

RESOURCES

- Best practices for writing Dockerfiles
- Guidance for Docker Image Authors
- Dockerfiles for reference
  - redis
  - Jenkins
  - Postgresql