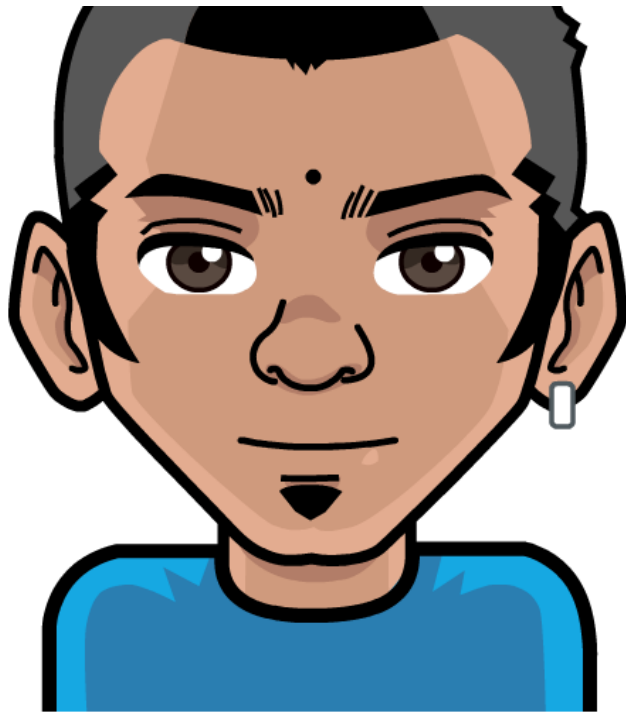RAJU GANDHI

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

```
bd03254d7d98          FROM looselytyped/java-on-ubuntu

4d3d9ad31e2f          ARG VERSION

4e4d32686ce4          COPY build/libs/pid-demo.jar pid-demo.jar

bd48b228c607          CMD ["java", "-jar", "pid-demo.jar"]
```
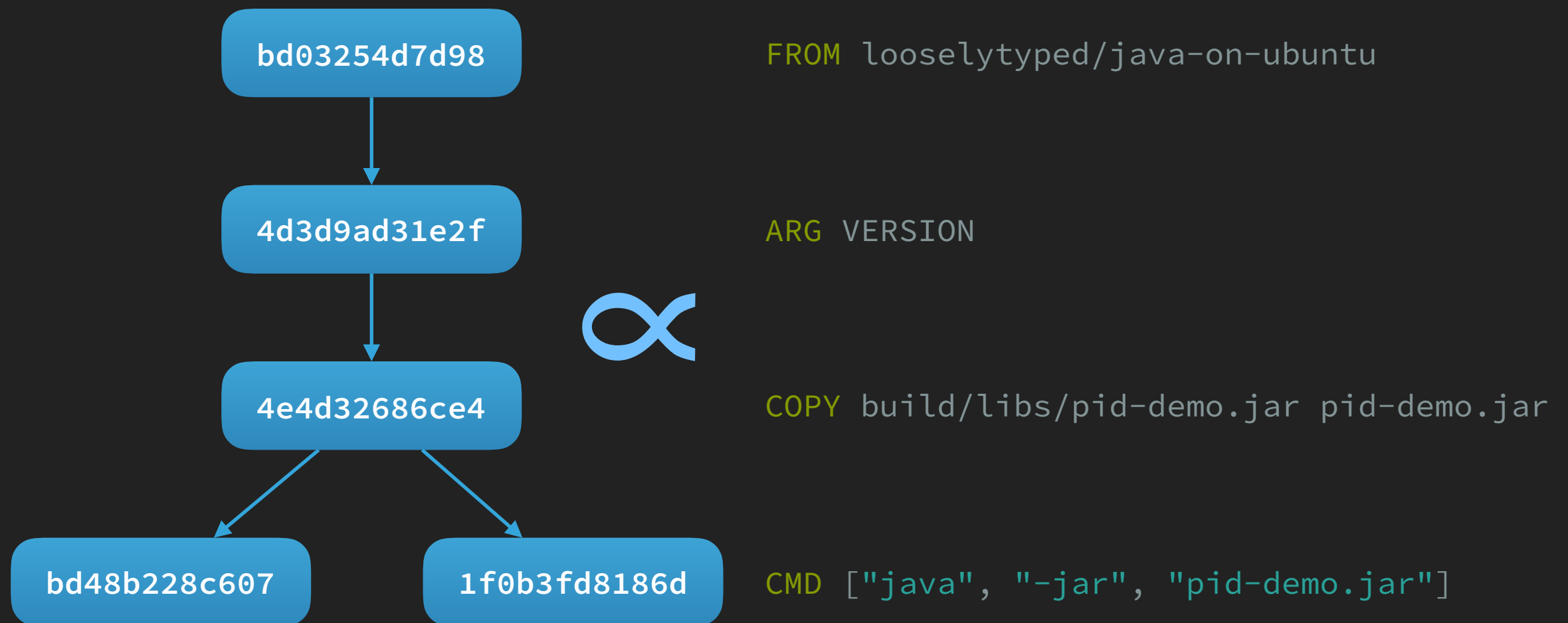
bd03254d7d98   FROM looselytyped/java-on-ubuntu

4d3d9ad31e2f   ARG VERSION

4e4d32686ce4   COPY build/libs/pid-demo.jar pid-demo.jar

1f0b3fd8186d   CMD ["ls", "-al"]

```
bd03254d7d98          FROM looselytyped/java-on-ubuntu

4d3d9ad31e2f          ARG VERSION

4e4d32686ce4          COPY build/libs/pid-demo.jar pid-demo.jar

bd48b228c607    1f0b3fd8186d    CMD ["java", "-jar", "pid-demo.jar"]
```

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 FROMs are allowed. Do **NOT** do <u>this</u>

DO'S

- ▸ Pin down the exact tag

    - ▸ Do not use "latest" tag

- ▸ Inspect ancestor images for USERs, PORTs, ENVs, VOLUMEs, LABELs 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

  - ▸ 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 <u>convention</u>

▸ 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}
```
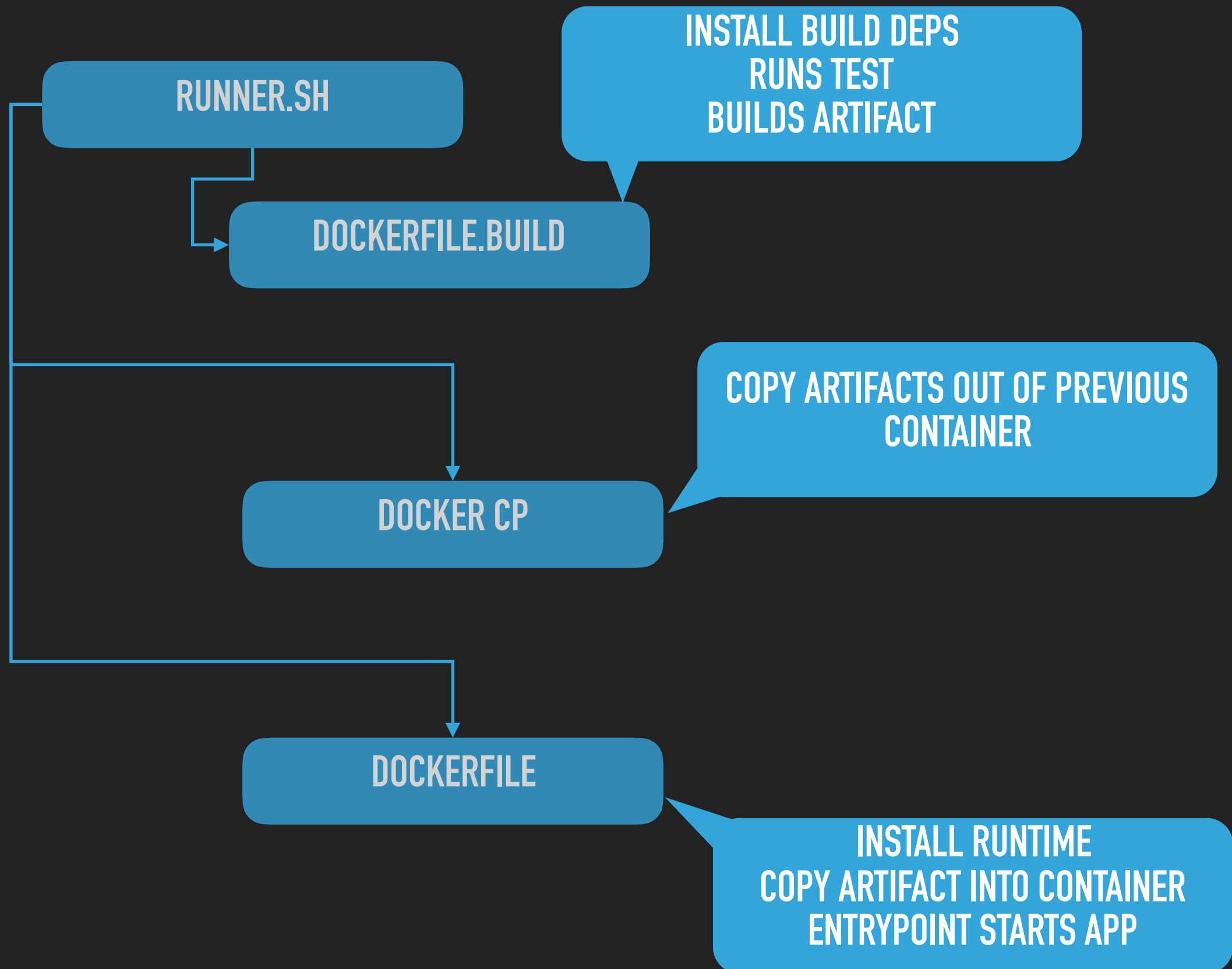
CLOSING THOUGHTS

DO'S

- ▸ Do create your own "ancestry" of images

- ▸ Do create *multiple* Dockerfiles per project (if necessary)

  - ▸ Dockerfile.build (installs JDK, build dependencies, builds war)

  - ▸ Dockerfile (installs JRE, and runtime e.g tomcat) with artifact

  - ▸ A wrapper script to tie it altogether

▸ Best practices for writing Dockerfiles

▸ Guidance for Docker Image Authors

▸ Dockerfiles for reference

  ▸ redis

  ▸ Jenkins

  ▸ Postgresql