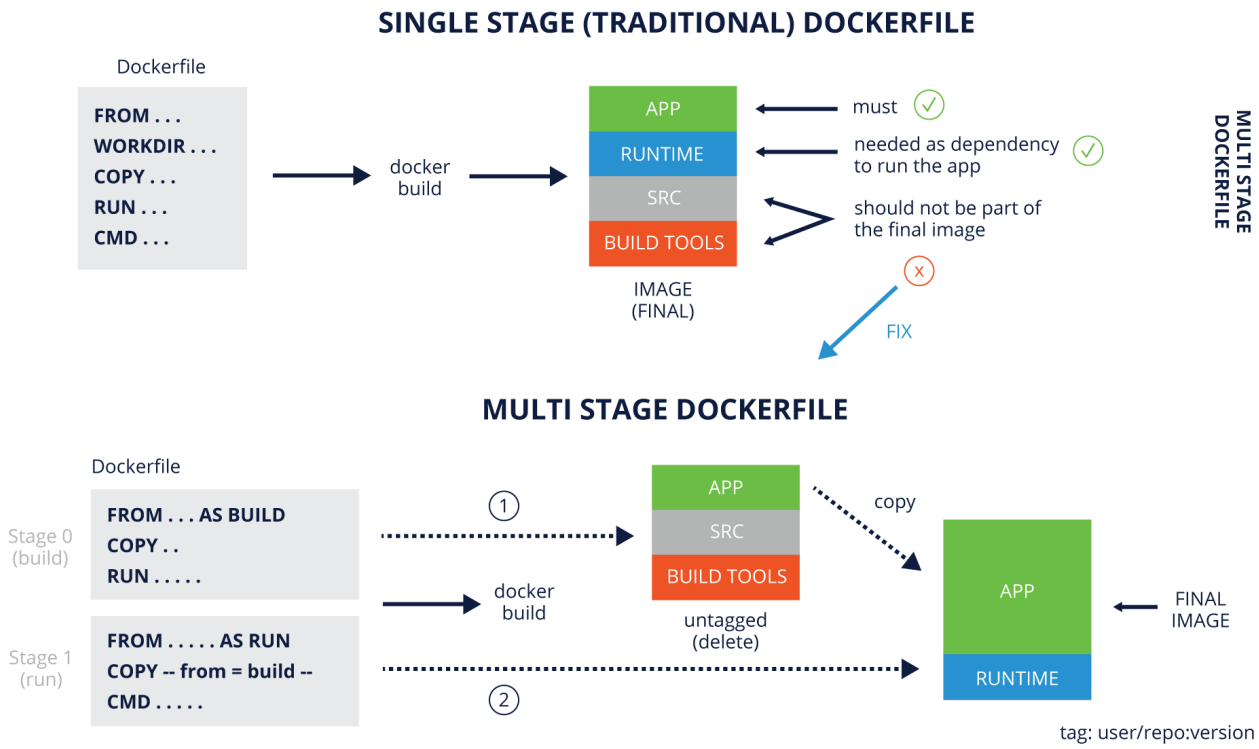
Lab 5. Multi-Stage Docker Build

So far, you have learned how to build an image with Dockerfile. However, the one you have built has at least two issues:

1. The final image also contains the build tools (e.g. Maven, JDK) and dependencies (~/.m2) which are not necessary to run the application. A minimal runtime would have been sufficient. Packaging tools along with an application increase the size of the image.
2. Final image also contains the source code, which is definitely not desired. Only the final application and its runtime should be part of what you distribute to the client/deploy to production.

How do we fix this? Is there a way to reduce the size of this image significantly? Yes! This is where a multi-stage Dockerfile comes in handy and solves the problem. Following diagram depicts the difference between a single stage (traditional) Dockerfile and a multi-stage Dockerfile:



The above diagram shows two stages but you can add more, if necessary. In this example with a two-stage Dockerfile this is what happens:

- The first stage, BUILD (stage #0), uses an image with all the build tools including Maven, JDK, etc. It creates an intermediate container to run the build.
- An intermediate container created for the build stage is where the source gets copied. This is also where it is compiled to create the application artifacts (e.g. JAR file), which are ready to be deployed.
- The second stage, RUN (stage #1), launches another container. This time it uses a minimalistic image with runtime, and only with the components needed to run the application.
- A copy logic is added to this stage to fetch the deploy ready artifact from the build stage by referencing the `--from` option to the COPY instruction. This ensures that only the application binaries and not the source code are added to the final image.
- The second stage is also the final stage in this file. An image is created by packaging the runtime and application alone, and is then tagged using the option that you provide with the `docker build` command.

Refactoring Spring Boot Application with a Multi-stage Dockerfile

Study this multi-stage Dockerfile provided on GitHub: initcron/sysfoo, Sample Java Web Application with Maven which Prints System Information, and then refactor the Dockerfile that

you have created for the PetClinic Spring Boot application to create two stages, BUILD and RUN, similar to the example above. Do not look at the solution before you attempt it.

Summary

In this lab, you learned how to create small, secure and optimized container images using multi-stage Dockerfiles, and how to avoid getting unnecessary files such as source code, build tools, etc., packaged into the container image.