Title: Docker with Maven and Tomcat

Post Body:

i am trying to create a docket container with Java,Maven and Tomcat.

So far, i'm able to create a container with Maven and compile my java code and create the final WAR file.

But i want to **copy this WAR file to a particular folder**(webapp folder of tomcat) of another container which is of Tomcat.

Approach one, Dockerfile

```
FROM maven:3.6.1-jdk-8 as maven_builder  ENV HOME=/app  WORKDIR $HOME  ADD pom.xml $HOME  RUN ['/usr/local/bin/mvn-entrypoint.
```

i tried to simply copy it, but the intermediate container is deleted before copying.

Approach 2, i am trying to write a YAML for docker-compose and use shared volume concept.

```
version: '3' services:     maven-build:         build: .         volumes:             - 'myshare:/shared'      tomcat-build:
```

i'm not able to figure out, how do i copy a particular file from one container(i.e maven_build) to another container (i.e tomcat's webapp folder).

Accepted Answer:

Your first (multi-stage build) approach is better practice.

In the multi-stage build, every time you have a new `FROM` command to start a new image, the execution environment resets. In particular, when the first stage sets a `HOME` environment variable, that gets reset in the second stage, so the final `COPY` command is copying out of `/wc_admin/...` and not the directory where the application got built.

It's typical to treat Docker filesystem layouts as fixed, and to not set variables like `HOME` that are meaningful in other contexts but not really in Docker. I'd just hard-code `/app` throughout:

```
FROM maven:3.6.1-jdk-8 as maven_builder WORKDIR /app ADD pom.xml . ... FROM tomcat:8.5.43-jdk8 COPY --from=maven_builder /app/
```

The volumes approach has two big problems. People try to take advantage of Docker populating a named volume with content from the image, but this only works the first time you run it. If you rebuild the image, Docker refuses to touch the volume (it has important user data that must be preserved, that's what volumes are for) and so your volume will keep your old `.war` file. If you'll ever run your application in Kubernetes, it will not auto-populate volumes from images and you'll need to do more work to make a volumes-for-sharing setup work.

Highest Rated Answer:

Your first attempt is almost correct, minus one issue. Environmental variables are local for the context of the container. **They are not inherited among build stages.**

In this snippet

```
FROM tomcat:8.5.43-jdk8  COPY --from=maven_builder $HOME/wc_admin/target/wc-admin.war /usr/local/tomcat/webapps
```

`tomcat` image knows nothing about `$HOME` environmental variable of maven image

```
FROM maven:3.6.1-jdk-8 as maven_builder  ENV HOME=/app
```

thus it uses its own `$HOME` envar.

Try this:

```
COPY --from=maven_builder /app/wc_admin/target/wc-admin.war /usr/local/tomcat/webapps
```