Title: Running mvn install from Dockerfile gives error but, running it from inside the container is successful

Post Body:

I have a project at https://github.com/picklu13/dockerbuildtest. When I run `docker-compose up --build` from the `builders` directory, the build errors with '[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (/app). Please verify you invoked Maven from the correct directory. -> [Help 1]'

Now if I comment out the last line of the Dockerfile `# RUN mvn clean package` and run `docker-compose up --build`, the container starts up. Then I go into the container with `docker exec -it <id> bash` and run `mvn clean install` which succeeds.

My question is, why did the first build fail with no pom.xml error although it was already present.

Accepted Answer:

In your first case, you're running `build` from `builders` directory but the context passed to Docker does not contains `pom.xml` (it contains what is in your current directory as per `docker-compose` instruction `context: .`)

In your second case, up will mount your project's directory in `app` with:

```
 volumes:        - ~/dockerbuildtest/:/app/
```

Meaning that `~/dockerbuildtest/pom.xml` will appear in container as `/app/pom.xml`. Running `mvn` from `app` now works.

> why did the first build fail with no pom.xml error although it was already present

It was *not* present: in first build, there is no `pom.xml` copied into your image (see `COPY` instruction).

You could ensure your context contains `pom.xml` by running your build command from your project's root directory and specifying Dockerfile such as:

```
    build:       # will use current directory as build context       # by running from your project's root dir, context will c
```

And copy your `pom.xml` in your Dockerfile such as:

```
# [...] WORKDIR /app  # copy pom.xml from context into image COPY pom.xml /app/pom.xml  # run from /app directory which now co
```

Highest Rated Answer:

You mount a volume and I think by the time the container boots up, the specified resource is not at said location, due to the volume not yet being loaded.

You could try to not use a volume and instead use a COPY command in your dockerfile to make sure the resource is actually in /app before the container starts.

[EDIT] Even if you scenario does not allow you to do so, I recommend still testing with the COPY to see if this indeed is a volume problem.