Title: How can I optimize docker builds of multiple, related maven java projects? (caching)

Post Body:

I manage a large propriety system that's compromised of about a dozen services in java. We have a core set of java libs that these all share ), and all the components/apps are built using maven. Outside of the core SDK jars though each app has its own unique set of dependencies. I can't figure out what the best approach is to both building and deploying inside docker. Ideally I want the entire lifecycle in docker, using a multi-stage build approach. But, I can't see how to optimize this with the huge number of dependencies.

It looks like I can do 2 approaches.

> Build as we have before, using maven and a common cache on the CI server (jenkins) so that dependencies are fetched once and cached, and accessible to all the apps. Then have a dockerfile for each app that just copies the product jar and it's dependencies (or a fat jar) into the container, and set it up to execute. Downside of this approach is that the build itself is something that could differ between developers and the CI server. Potentially use a local maven cache like nexus just to avoid pulling deps from the internet everytime? But that still doesn't solve the problem that a dev build won't necessarily match the CI build environment.
>
> Use multi-stage dockerfile for each project. I've tried this, and it does work and I managed to get the maven dependencies layer to cache so that it doesn't fetch too often. Unfortunately that intermediate build layer was hitting 1-2gb per application, and I can't remove the 'dangling' intermediates from the daemon or all the caching is blowing away. It also means there's a tremendous amount of duplication in the jars that have to be downloaded for each application if something changes in the poms. (ie they all use junit and log4j and many other similarities)

Is there a way to solve this optimally that I'm not seeing? All the blogs I've found basically focus on the 2 approaches above (with some that focus on running maven itself in a container, which really doesn't solve anything for me). I'll probably need to end up going with Option 1 if there aren't any other good solutions.

I've checked around on stackoverflow and blogs, and everything I can find seems to assume that you're really just building a single app and not a suite of them, where it becomes important to not repeat the dependency downloads.

Accepted Answer:

I think it is OK to use the .m2/repository filesystem cache as long as you set the `--update-snapshots` option in your maven build. It scales better, because you cache each .jar only once per build environment and not once per application. Additionally a change in a single dependency does not invalidate the entire cache, which would be the case if you use docker-layer-caching.

Unfortunately that cannot be combined well with multi-stage builds at the moment, but you are not the only one asking for it. This issue requests adding a `--volume` option to the docker build command. This one asks for allowing instructions like this in the Dockerfile: `RUN --mount=m2repo=/var/mvn/repo mvn install`.

Both features would allow you to use the local maven filesystem cache during your multistage build.

For the moment I would advise to keep your option 1 as solution, unless you are facing many issues which are due to differing build environments.

Highest Rated Answer: None