1. Work In Progress should not be kept in unnamed, stopped containers — keep in mind that unnamed containers are potential victims to be deleted when memory is issue. So try to keep your undone work not in this kind of vulnerable containers.
2. Clean-up your images regularly — images can build up quickly if you forget to clean them up costing big memory. Although Docker is memory friendly (common parts from containers won’t be duplicated), redundant images can cause extra space.
3. Fetch dependencies that you trust — don’t fetch any unstable or unpopular dependencies that you don’t trust the owner. If that dependency is removed or changed, it can cause your image big problems.
4. Put the parts, that change a lot, to the end of the DockerFile — Docker caches each step as a separate container. So every time repeated actions are not done again and again since it checks for each step its cache. So putting unstable parts to the end of the DockerFile makes it performant since stable parts are won’t be repeated.
5. Don’t make your containers fetch dependencies when starting — Fetching dependencies should be separate step which needs to be open to extension.
6. Don’t mount shared folders in your DockerFile that points to your host machine — works perfectly on your machine, but not on colleague’s.
7. No “God” Images — try to keep your images simple and small. DockerFiles are not java classes. One image should not build environment for multiple functionalities.
8. Build everything from scratch — don’t rely on somebodies’ ready image. Build everything from scratch to have minimal dependencies to others.
9. Tag your build — Builds should have reasonable tags. This helps you to sort the builds later. Good practice is to tag them with relevant git hash.
10. Use small base images — try to find the base image that is minimal such as -slim or -alpine.
11. Root from official images — official images are maintained by the owner. If you build your image upon an unofficial image, there is no security that this base image will not disappear some day in future causing downtime to you.
12. [Create ephemeral containers](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#create-ephemeral-containers)— The image defined by your DockerFile should generate containers that are as ephemeral as possible. By “ephemeral”, we mean that the container can be stopped and destroyed, then rebuilt and replaced with an absolute minimum set up and configuration ©.
13. Use multi-stage builds — Docker allows you to have multiple stages with AS command. This helps you to have for dev, test, prod different image building flows. This allows minimises the images size.
14. Sort multi-line arguments — instead of having multiple RUN commands, try to combine them. This makes your DockerFile more readable.
15. Use .dockerignore file — you can exclude unnecessary files from building. It is very similar to .gitignore File.