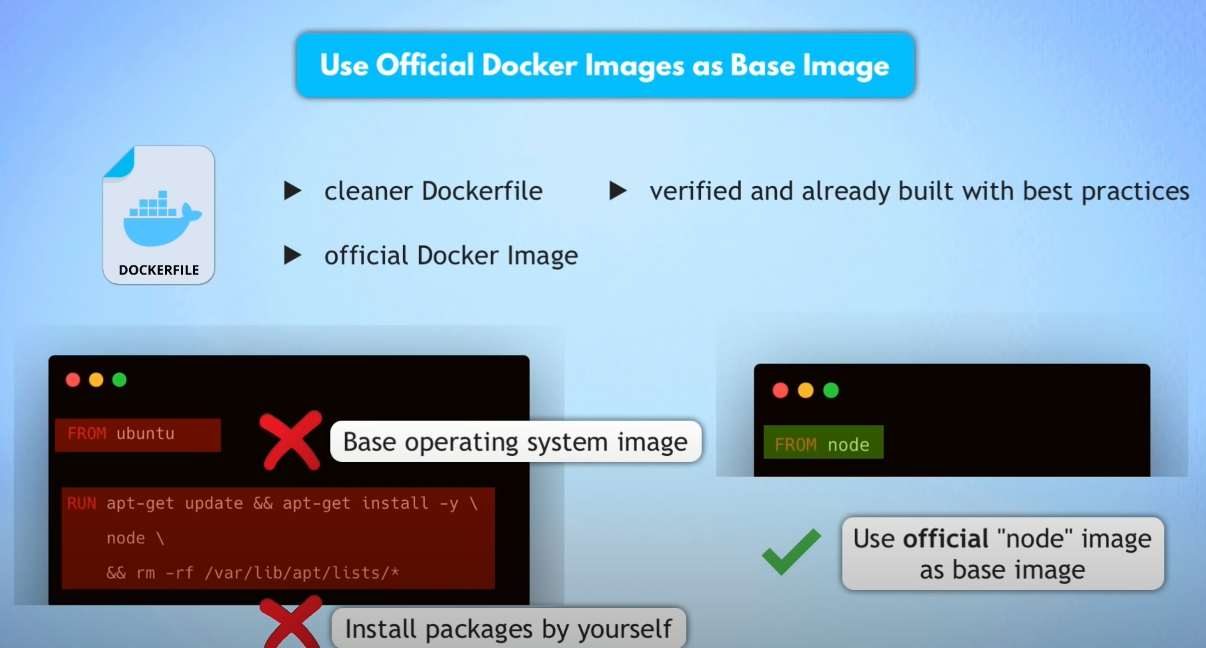
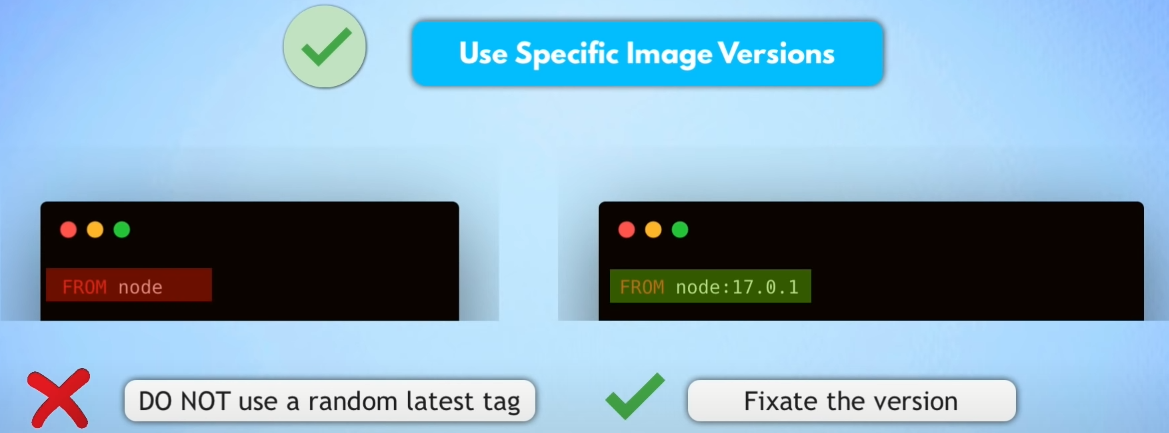
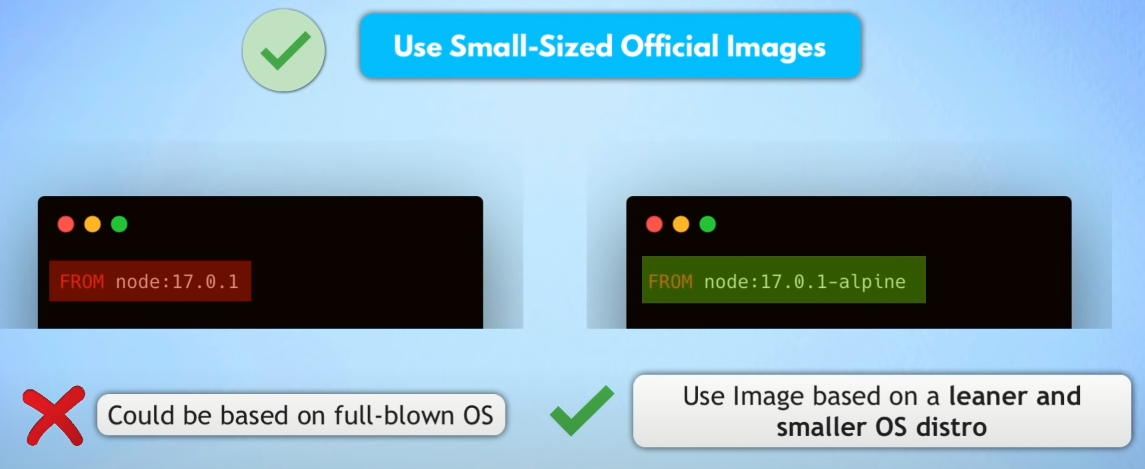
1. Always use official image. Instead of downloading Ubuntu and then installing node.js and other components, it’s better to download node.js image in the first place



1. Use specific docker image version, because if not specified – latest version will be applied. And it might have bugs or can break something

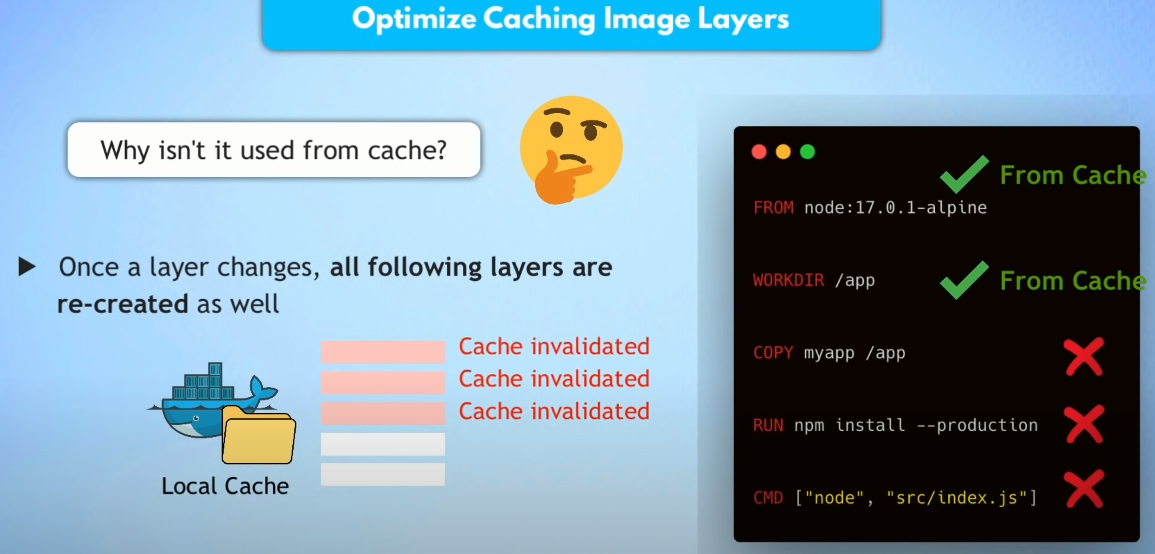


1. Use small sized official images, like alpine. Because big sized images are based on heavy OS, like full Ubuntu, and it might include features that are not needed. So in order to save space it’s better to use small sized images. Also number of vulnerabilities increases with heavier OS. So if it’s not required to have specific utilities, it’s better to stick with light weight distros

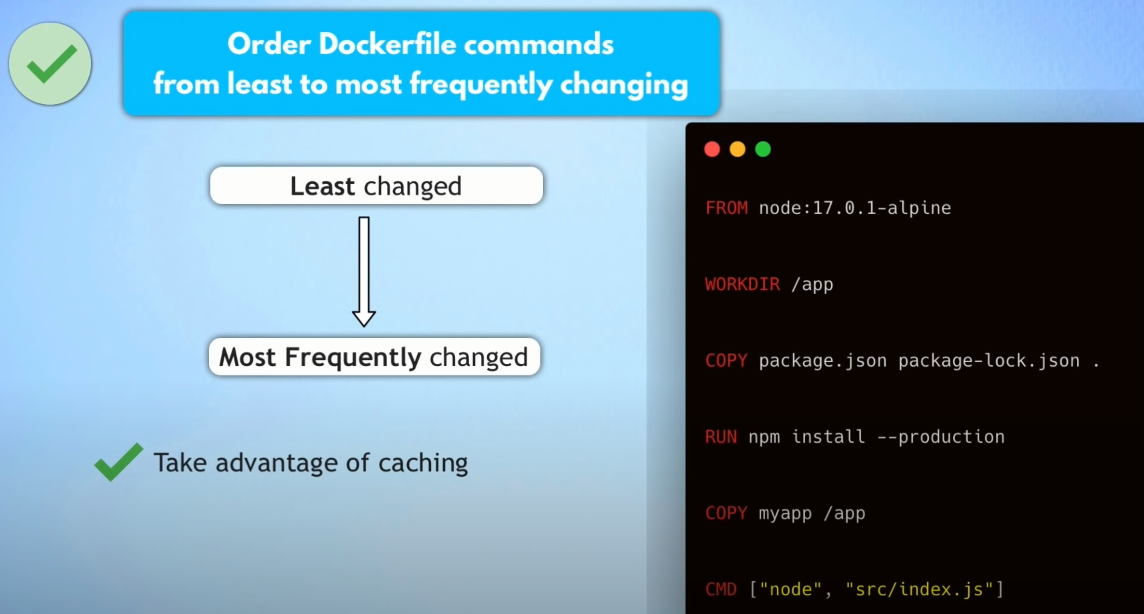


1. Optimize caching image layers. Each commands creates an image layer. Docker caches each layer, saved on filesystem. If nothing has changed in a layer (or layers preceding it), it will be reused from the cache – increases building an image and downloading an image

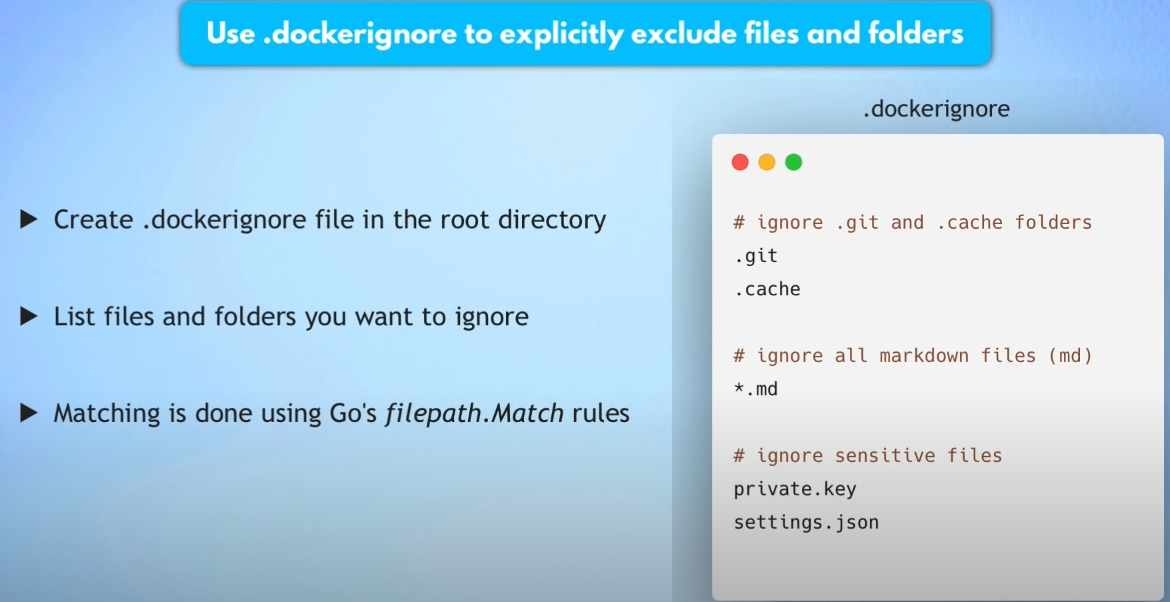
If we change some files in our project folder, then **COPY** command layer won’t be used from cache (obvious), but then other layers won’t be used as well



That’s why we have to restructure dockerfile, so command (layers) that won’t be necessary to execute every time, has to be on top

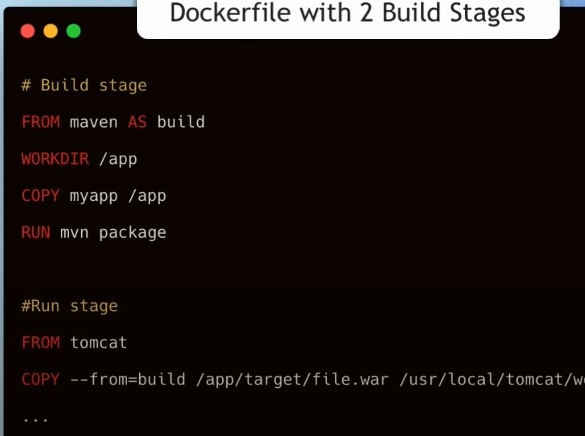


1. Ignore files, that we don’t want to include in an image

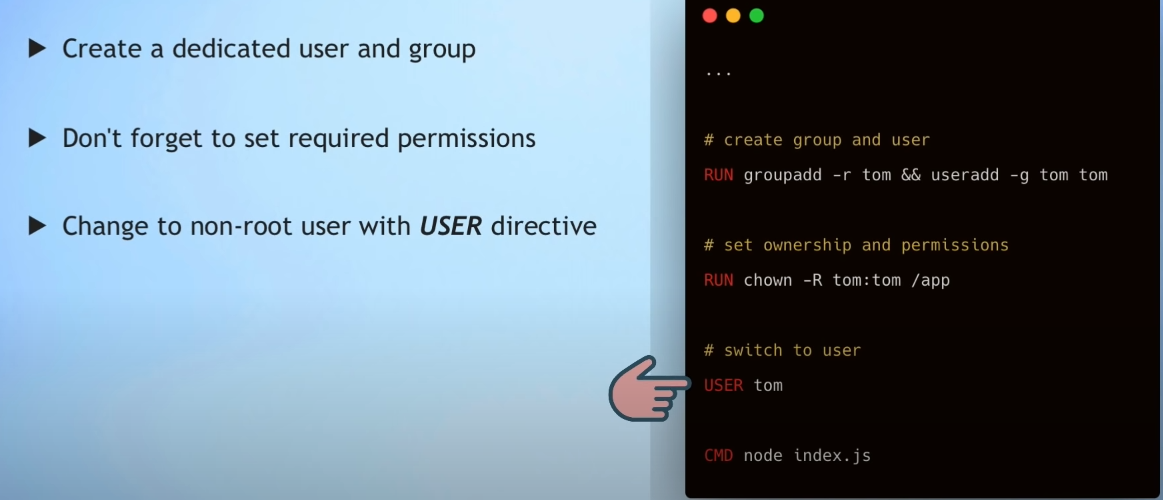


1. Make use of multi-stage builds. If during building an image we need to have other image, that will be used only for creating our image. Like libraries, dependencies, … If we keep that artifacts in final image, it will increase image size and also increase attack surface.

For example, when building java application, we need JDK, but JDK is not needed for running the application



1. Use the least privileged user. By default when container is created, it’s created from root user. That can lead to security issues. Some official images already create least privileged user. Like nodejs creates user - node



1. Scan images for vulnerabilities. Use **docker scan image:tag**. Login to DockerHub is necessary. Docker uses **snyk** service for scanning

Scan shows type of vulnerability, and also in which version of image this vulnerability has been fixed.

