**Docker Best Practices**

**References**

* [Docker Guide- Dockerfile Best Practices](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)
* [Docker Guide- Docker Best Practices](https://docs.docker.com/develop/dev-best-practices/)

**Goals**

* "ephemeral" containers-- containers should be as easy as possible to tear down and build up, requiring minimal configuration
* lightweight containers and images
  + "ephemeral" and lightweight go hand in hand, but additionally lightweight relies on a couple of other best practices

**Build Context**

The working directory (including subdirectories) is sent over to the Docker daemon when the image is created i.e. the docker build command is run.

Thus, it's important to be mindful of what is in that directory. The less that's there the faster the process and lighter weight the image.

**Leverage multi-stage builds and image cache**

Previous images are cached, since images are layered this can be leveraged to dramatically speed up building an image. It reduces the amount of times that you must pull from remote storage or rebuild image layers.

\*\* Be mindful of the cache though as some layers maybe cached and not properly updated. For instance, RUN apt-get update will be read as the same string every time and will not be run again if the cached image was built in exactly the same way with no other changes.

**Additional Best Practices:**

* Least number of ultimate layers for an image possible
  + These will be defined by the different commands in the dockerfile
* Each Container should ideally serve one sole purpose and applications should be decoupled as much as possible
* Make commands in dockerfile readable by separating them on to different lines with \ (the escape character) and by keeping those lines organized in some fashion
* Use volumes for persistent data
* Use secrets for sensitive data and config files for configurations that are not sensitive