docker docs

Docker development best practices

https://docs.docker.com/develop/dev-best-practices/

Docker development best practices

How to keep your images small

Where and how to persist application data

Use CI/CD for testing and deployment

Differences in development and production environments

How to keep your images small

- Start with an appropriate base image.

How to keep your images small

- Start with an appropriate base image.

  For instance, if you need a JDK, consider basing your image on the official `openjdk` image, rather than starting with a generic `ubuntu` image and installing `openjdk` as part of the Dockerfile.

docker docs

How to keep your images small

- Use multistage builds.

https://docs.docker.com/build/building/multi-stage/

docker docs

How to keep your images small

- Use multistage builds.

https://docs.docker.com/build/building/multi-stage/

For instance, you can use the `maven` image to build your Java application, then reset to the `tomcat` image and copy the Java artifacts into the correct location to deploy your app, all in the same Dockerfile. This means that your final image doesn't include all of the libraries and dependencies pulled in by the build, but only the artifacts and the environment needed to run them.

How to keep your images small

- If you need to use a version of Docker that does not include multistage builds, try to reduce the number of layers in your image by minimizing the number of separate `RUN` commands

```
RUN apt-get -y update
RUN apt-get install -y python
```

```
RUN apt-get -y update && apt-get install -y python
```

How to keep your images small

- If you have multiple images with a lot in common, consider creating your own base image with the shared components, and basing your unique images on that. Docker only needs to load the common layers once, and they are cached. This means that your derivative images use memory on the Docker host more efficiently and load more quickly.

How to keep your images small

- To keep your production image lean but allow for debugging, consider using the production image as the base image for the debug image. Additional testing or debugging tooling can be added on top of the production image.

How to keep your images small

- When building images, always tag them with useful tags which codify version information, intended destination (`prod` or `test` , for instance), stability, or other information that is useful when deploying the application in different environments. Do not rely on the automatically-created `latest` tag.

docker docs

How to keep your images small

* Start with an appropriate base image.
* Use multistage builds | Try to reduce the number of layers
* Consider creating your own base image with the shared components.
* Consider using the production image as the base debug image.
* Always tag images with tags which codify version and destination.