ONE PAGE SUMMARY

IT & Software › Docker › Multi Stage Builds

Introduction to Multi Stage Builds

Streamline Docker images with Multi Stage Builds for efficiency.

DEFINITION

> Multi Stage Builds in Docker allow you to create smaller, more efficient images by separating the build environment from the final product. This reduces the size of the final image and improves performance.

KEY CONCEPTS

Build Stages

Each stage in a Multi Stage Build can have its own base image and dependencies.

Final Image

Only the necessary artifacts from the last stage are included in the final image.

Layer Caching

Docker caches layers, making subsequent builds faster by reusing unchanged layers.

Reducing Size

By using Multi Stage Builds, you can significantly reduce the size of your Docker images.

Build Context

The build context is the set of files available to the Docker build process.

EXAMPLES

→ FROM node:14 AS builder; RUN npm install; COPY . .; RUN npm run build; FROM nginx; COPY --from=builder /app/build /usr/share/nginx/html;

→ FROM golang:1.16 AS builder; WORKDIR /app; COPY . .; RUN go build -o myapp; FROM alpine; COPY --from=builder /app/myapp /usr/local/bin/myapp;

MEMORY TIPS

★ Think of stages as layers of a cake, each adding a unique flavor.

★ Remember 'Build, Copy, Run' to recall the main steps in Multi Stage Builds.

★ Use the acronym 'SIMPLE' - Separate, Image, Multi, Performance, Layer, Efficiency.

COMMON MISTAKES

✗ Forgetting to copy necessary files from earlier stages to the final image.

✗ Neglecting to clean up unnecessary dependencies in earlier stages.

✗ Using too many stages without clear purpose, complicating the build process.

QUICK RECAP

Multi Stage Builds in Docker optimize image size and performance by separating build and runtime environments. This method enhances efficiency and reduces the final image footprint.