A closer look at Docker BuildKit and Buildx .

CNCF Meetup Linz

Public Cloud Group

Agenda.

01. Introduction
02. Docker Build
03. BuildKit
04. Mounts
05. Buildx
06. Docker Build Cloud
07. Questions

About Me

- Software Engineering Master @ **FH Hagenberg**
- Software Engineer and Platform Engineer @ **ENGEL Austria GmbH**
- Cloud Consultant Azure @ **Public Cloud Group GmbH**

Sophia Zehethofer
sophia.zehethofer@pcg.io

https://github.com/sophher/cncf-docker

Goal of the Talk

- Containerized Angular App
- Lint, unit tests and build completely with Docker
- Hosted on Github
- Available on Github Codespaces
- Cloud only
- Fast and reproducible

Docker Build

- **Build, Ship and Run Any App, Anywhere**

- **Dockerfile describes the container image**
  - Base Image is picked with `FROM`
  - Application files are added with `COPY`
  - Installations and other commands can be executed with `RUN`
  - Arguments and environment variables can be defined with `ARG` and `ENV`

- **Executed with the Docker Build CLI**
  - `docker build --file Dockerfile --tag app .`

- **Build context must be given**
  - Folder, where docker gets its files from (most often ".")
  - Files can be excluded with `.dockerignore` (hashes)

DEMO 1

Docker Build

- Angular demo app
- Local build, test and lint
- Container image build with Docker

Docker BuildKit

- New build Engine of the Docker CLI
- Standalone version available
- Default since Docker Engine v23.0
- In older versions available with
  - `DOCKER_BUILDKIT=1 docker build .`
  - `/etc/docker/daemon.json`

```
{
  "features": {
    "buildkit": true
  }
}
```

- Docker daemon must be running
- Runs a build container in the background

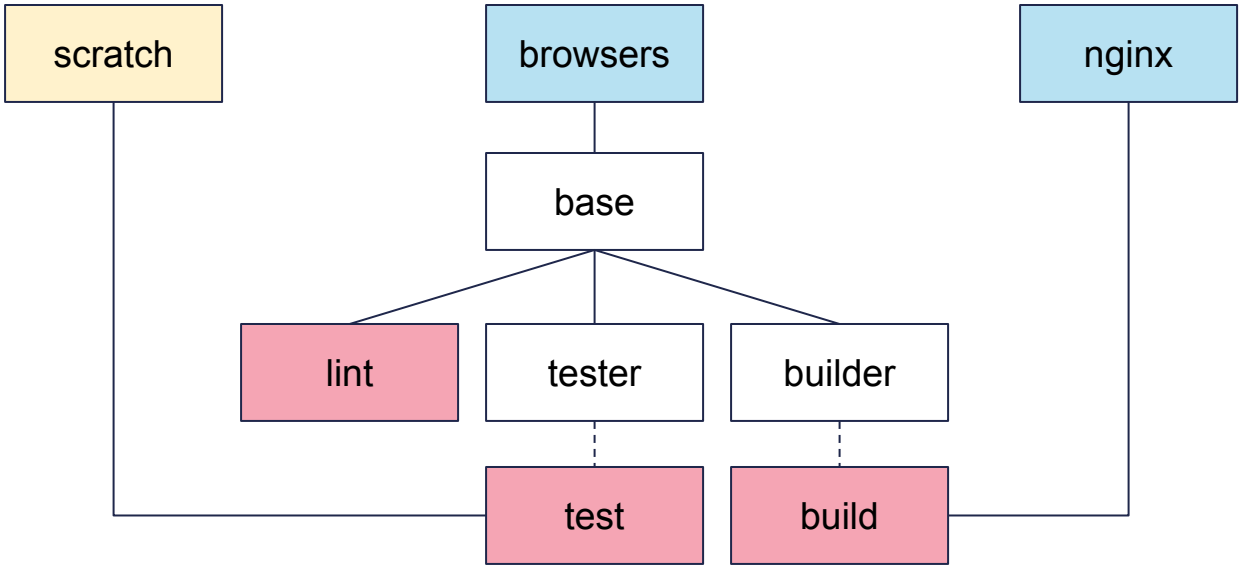
Multi-Stage Builds

- **Multiple Stages**
  - `FROM <image/stage> AS <stage>`
- **Copy files from previous stage**
  - `COPY --from=<stage>`
- **Build stage can be targeted**
  - `docker build --target <stage>`
- **Dependency Tree**
  - Enables build and download parallelization of images and stages
- **Build Layer Cache**
  - Each command (`RUN`, `COPY`, …) is cached in a layer
  - Disable the whole cache with `--no-cache`
  - Bust a cache layer and its dependents with an `ARG CACHE_BUST` and timestamp
  - `docker build --build-arg CACHE_BUST=$(date +%s) .`

Output Types

- **Docker image (default)**
- **OCI image layout**
  - `docker buildx build --output type=oci,dest=<path/to/output> .`
  - Outputs the image layers locally in a tarball
- **Cacheonly**
  - `docker buildx build --output type=cacheonly .`
  - No output
  - Useful for tests and deployments, that only have exit codes
- **Local**
  - `docker buildx build --output type=local,dest=<path/to/output> .`
  - Outputs the whole file system as folders and files
  - To pick files use `FROM scratch` with `COPY --from`
- **Tar**
  - Same as local, but the file system is compressed in a tarball
  - `docker buildx build --output type=tar,dest=<path/to/output> .`

Dependency Tree



DEMO 2

BuildKit

- Lint
- Test
- Build

Bind Mounts

- **Mount local folders into the build container**
- **Two options: read-only and read-write**
  - `RUN --mount=type=bind,source=.,target=/usr/src,ro`
  - `RUN --mount=type=bind,source=.,target=/usr/src,rw`
- **Scoped to a RUN command**
- **Not written to disk (!!!)**
  - Changes are lost when the `RUN` block is finished
  - Files to extract must be copied to an unmounted destination

Cache Mounts

- **Specify a folder, which is persistent for multiple builds or stages**
  - `RUN --mount=type=cache,id=pnpm,target=/root/.local/share/pnpm`
  - When used in multiple `RUN` commands, use an `id` as the identifier
- **Saved in the Docker cache, not in a local folder**
- **Clean with prune**
  - `docker builder prune --filter type=exec.cachemount`

Secret Mounts

- **Secret Mounts are not saved in a build layer**
- **Secret Environment Variable**
  - `RUN --mount=type=secret,id=kube`
  - `docker build --secret id=kube,env=KUBECONFIG .`
- **Secret File**
  - `RUN --mount=type=secret,id=aws`
  - `docker build --secret id=aws,src=$HOME/.aws/credentials .`
- **Secrets are mounted as files (both ENV and File)**
  - `/run/secrets/<id>`

Devcontainer

- **Development environment in a container**
- **Visual Studio Code connects remotely to this container**
- **Configured via committable JSON file in the workspace**
  - `./devcontainer/devcontainer.json`
- **Image or Dockerfile can be used as the dev environment**
  - Target can be specified
  - The same stage as in the CI/CD system can be used
- **Visual Studio Code extensions and settings are configurable**
- **Ports can be forwarded**
- **Post build steps are available**
  - e.g. `pnpm install`

GitHub Codespaces

- Online Visual Studio Code for GitHub
- Executed on a Cloud Agent
- Capable of automatic Devcontainer boot
- Secure development agents, that can be audited
- Fast onboarding

DEMO 3

Mounts

- Bind Mount
- Cache Mount
- Devcontainer
- GitHub Codespaces

Docker Buildx

- **Extended Capabilities for BuildKit**
- **Mostly experimental**
  - Continuous integration in the default build CLI
- **Multiple Platforms**
  - `docker buildx build --platform linux/amd64,linux/arm64 .`
  - QUEMU support
- **Local and Cloud Builders**
  - Multiple builders for multiple platforms and parallelization
- **Build multiple targets with bake**
  - `docker buildx bake`
  - Bakefile
  - Hashicorp Configuration Language (HCL) possible
  - Variables and Functions
  - Target Definitions
  - Group Definitions

DEMO 4

Buildx

- Docker bake
- Local Builder

Docker Build Cloud

- **Buildx Agents in the Cloud**
  - Available for multiple platforms out of the box
  - High parallelization
  - No agent setup needed
- **Docker Cache in the Cloud**
  - CI agents and local clients can use the cache
- **Tightly integrated in the Docker CLI**
  - GUI Docker Desktop
  - CLI Plugin https://github.com/docker/buildx-desktop (!!!)

GitHub Actions

- **CI/CD System for GitHub**
- **Actions for Docker Build Cloud are available**
  - `docker/login-action@v3`
  - `docker/setup-buildx-action@v3`
  - `docker/build-push-action@v6`
  - `docker/bake-action@v6`
- **Configurable with workspace yaml file**
  - `./.github/workflows/<name>.yml`

DEMO 5+6

Docker Build Cloud

- Docker Cloud build on dev system
- Docker Cloud build with GitHub Actions

Why build with Docker? – Pros

- **Build Infrastructure as Code**
  - Tools and versions are defined in the Dockerfile
  - Local and CI tooling are exactly the same
  - Pull Requests are easy when tool versions have to change
  - Builds are reproducible
- **Build agent images can be kept minimal**
  - Only Git and Docker must be installed
  - Can be run on different systems
- **Caching can be leveraged for faster builds**
  - Cloud caches for even more cache hits locally and on CI
- **Fast onboarding**
  - No local tool installation necessary
  - No version mismatches
  - No setup needed when using Codespaces + Devcontainer

Why not build with Docker? – Cons

- **Verbosity and awkward workarounds**
  - Output Types and FROM scratch
  - Cachebust
  - Mounts
- **Monorepo build tools and Docker daemon**
  - Docker in Docker
  - Daemonless (podman/buildah)
- **CI/CD only easy with Docker Build Cloud**
  - Build Agent Cleanup necessary (cache)
  - CPU and memory for Docker must be limited
  - Parallelization is difficult
- **Dockerfile maintenance**
  - Use Renovate or Dependabot
- **Dockerfile duplications in multi-repos**
  - No include statement
  - Base Image needed

Questions?

https://github.com/sophher/cncf-docker

GET IN TOUCH WITH US

Let's work together.

PCG
Public Cloud Group

Sophia Zehethofer
Cloud Consultant Azure
sophia.zehethofer@pcg.io

PUBLIC CLOUD GROUP GMBH
Peter-Behrens-Platz 10
4020 Linz

VISIT OUR WEBSITE

With a product portfolio designed to accompany organizations of all sizes in their cloud journey and competence that is a synonym for highly qualified staff that customers and partners like to work with, PCG is positioned as a reliable and trustworthy partner for the hyperscalers, relevant and with repeatedly validated competence and credibility.

We have the highest partnership status with the three relevant hyperscalers. As experienced providers, we advise our customers independently.