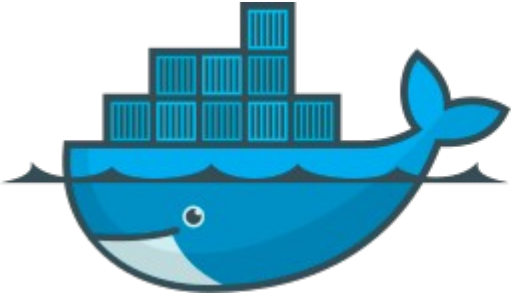
BGD2

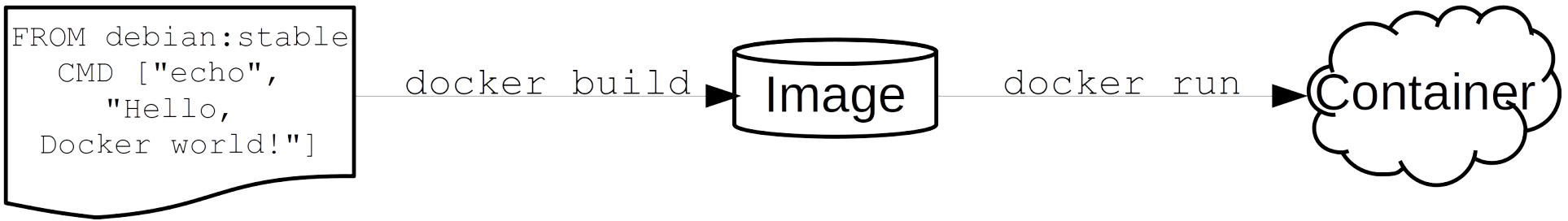
Dockerfiles



Andreas Scheibenpflug

Dockerfiles

- Dockerfiles enthalten Instruktionen zum Erstellen von Docker Images
- Der `docker build` Befehl baut aus einem Dockerfile ein Image



FROM

- Definiert das Basisimage für das neu zu erstellende Image
- `FROM <image[:tag]> [AS <name>]`
- Beispiel:
  - `FROM ubuntu:latest`
- Name wird für Multi-Stage builds benötigt, z.B. für
  - `COPY –from=<name>…`
- Multi-Stage builds (> v17.05)
  - Mehrere FROMs in einem Dockerfile möglich
  - Befehle ab dem letzten FROM ergeben das finale Image

RUN

- Führt einen Befehl im Image aus
- Konfiguration des Images
  - Installation von Software
  - Konfiguration von Software
  - ...
- Beispiel
  - `RUN apt-get install nginx`
  - `RUN dotnet build`

WORKDIR

- `WORKDIR <path>`
- Wird bei relativen Pfadangaben von `COPY, RUN, CMD, ADD, ENTRYPOINT` verwendet
- Beispiel
  - `COPY /root/file.txt .`
- Existiert das Verzeichnis nicht, wird es angelegt

COPY

- `COPY <src> <dest>`
- Kopiert Dateien/Verzeichnisse vom Hostsystem in das Image
- Dateien sind im Container verfügbar
- Erstellt Zielverzeichnis falls dieses nicht existiert
- Unterscheidung ob `<dest>` eine Datei oder Verzeichnis ist aufgrund des ‚/‘
- Beispiele
  - `COPY ./notes.txt /root/`
  - `COPY ./NotesAPI/ .`
  - `COPY ./NotesAPI/ /root/NotesAPI/`

ADD

- Wie COPY, aber
  - Wenn <src> eine URL ist, wird die Datei heruntergeladen und in <dest> gespeichert
  - Wenn <src> ein tar Archive ist, wird dieses in <dest> entpackt
- Beispiele
  - `ADD https://download.geofabrik.de/europe/albania-latest.osm.pbf /root/`
  - `ADD rootfs.tar.xz /`

ENTRYPOINT

- Legt die auszuführende Anwendung fest
- Beim Starten des Containers wird die in ENTRYPOINT definierte Anwendung ausgeführt
- Format
  - Shell form: `ENTRYPOINT command param1 …`
  - Exec form: `ENTRYPOINT ["executable","param1", ...]`

CMD

- Zwei Aufgaben
  - Definiert das auszuführende Programm
  - Definiert Default Parameter für ENTRYPOINT
- Format:
  - Exec form: `CMD ["executable","param1","param2"]`
  - Shell form: `CMD command param1 param2`
  - Parameter für ENTRYPOINT: `CMD ["param1","param2"]`
- Da bei der Exec Form keine Shell gestartet wird, gibt es keine Substitutionen → z.B. werden keine Shell Variablen ausgewertet

ENTRYPOINT, docker run und CMD

- ENTRYPOINT in exec form:
  - Parameter von CMD werden an ENTRYPOINT angehängt
  - Parameter von docker run werden an ENTRYPOINT angehängt
- ENTRYPOINT in shell form:
  - CMD / docker run Argumente werden nicht berücksichtigt
  - ENTRYPOINT wird als Subprozess von /bin/sh gestartet → Anwendung erhält kein SIGTERM bei docker stop

ENTRYPOINT - Beispiel

- `FROM debian:stable`
- `ENTRYPOINT ["/bin/echo"]`
- `CMD ["Hello, CMD"]`

- `docker run ... ["Hello, docker"]`

ENV

- Setzen von Umgebungsvariablen
- `ENV <key>[=]<value>`
- Sind zur Laufzeit im Container verfügbar
- `docker run --env <key>=<value>`
- Beispiele
  - `ENV DOTNET_SDK_VERSION 1.0.4`
  - `ENV DOTNET_SDK_VERSION=1.0.4`

ARG

- Setzen von Variablen zur Buildzeit
- `ARG <name>[=<default value>]`
- Sind zur Laufzeit im Container nicht verfügbar
- Müssen bei Multi-Stage Builds in jeder Stage definiert werden um verwendet werden zu können
- `docker run --build-arg <key>=<value>`

EXPOSE

- `EXPOSE <port> [<port>/<protocol>...]`
- Dient als Information für den Benutzer des Dockerfiles an welchen Ports ein Dienst hört
- Muss trotzdem mit `docker run -p` freigegeben werden

Beispiele

- https://hub.docker.com/r/microsoft/dotnet/~/dockerfile/
- https://github.com/docker-library/openjdk/blob/89851f0abc3a83cfad5248102f379d6a0bd3951a/6-jre/Dockerfile