Dockerfile

Build Image

- Docker image ????: dive

Basic build command

```
cd /path/to/Dockerfile
docker build -t your-tag/your-name .
```

Set the version

More details: docker-bookstack/Dockerfile at master · linuxserver/docker-bookstack · GitHub

```
FROM ghcr.io/linuxserver/baseimage-alpine-nginx:3.17

# set version label
ARG BUILD_DATE
ARG VERSION
ARG BOOKSTACK_RELEASE
LABEL build_version="Linuxserver.io version:- ${VERSION} Build-date:- ${BUILD_DATE}"
LABEL maintainer="homerr"
```

????:

Build the image

```
docker build \
  --build-arg BUILD_DATE=$( date +"%FT%T%z" ) \
  --build-arg VERSION=v22.07.3-ls36 \
  --no-cache \
  --pull \
  -t lscr.io/linuxserver/bookstack:latest .
```

Get the version of container

```
docker inspect -f '{{ index .Config.Labels "build_version" }}' <container-name>
```

Examples

Chroma DB

```
FROM python:3.11-slim-bookworm AS builder
ARG REBUILD_HNSWLIB
RUN apt-get update --fix-missing && apt-get install -y --fix-missing \
    build-essential \
    gcc \
    g++ \
    cmake \
    autoconf && \
    rm -rf /var/lib/apt/lists/* && \
    mkdir /install

WORKDIR /install

COPY ./requirements.txt requirements.txt

RUN pip install --no-cache-dir --upgrade --prefix="/install" -r requirements.txt
RUN if [ "$REBUILD_HNSWLIB" = "true" ]; then pip install --no-binary :all: --force-reinstall
--no-cache-dir --prefix="/install" chroma-hnswlib; fi

FROM python:3.11-slim-bookworm AS final

RUN mkdir /chroma
WORKDIR /chroma

COPY --from=builder /install /usr/local
COPY ./bin/docker_entrypoint.sh /docker_entrypoint.sh
COPY ./ /chroma

RUN apt-get update --fix-missing && apt-get install -y curl && \
    chmod +x /docker_entrypoint.sh && \
    rm -rf /var/lib/apt/lists/*

ENV CHROMA_HOST_ADDR "0.0.0.0"
ENV CHROMA_HOST_PORT 8000
```

```
ENV CHROMA_WORKERS 1
ENV CHROMA_LOG_CONFIG "chromadb/log_config.yml"
ENV CHROMA_TIMEOUT_KEEP_ALIVE 30

EXPOSE 8000

ENTRYPOINT ["/docker_entrypoint.sh"]
CMD [ "--workers ${CHROMA_WORKERS} --host ${CHROMA_HOST_ADDR} --port ${CHROMA_HOST_PORT} --
proxy-headers --log-config ${CHROMA_LOG_CONFIG} --timeout-keep-alive
${CHROMA_TIMEOUT_KEEP_ALIVE}"]
```

Python App

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
CMD ["python", "app.py"]
```

Learning

- Writing An Optimized Dockerfile
- What is the Difference Between COPY and ADD Instructions in Dockerfile
- Best practices for writing Dockerfiles
- How to Build Slim and Fast Docker Images with Multi-Stage Builds
- Docker Build Tutorial: Learn Contexts, Architecture, and Performance Optimization Techniques
- Node.js
  - How to Containerize a Node.js Application Using Docker – A Beginner's Guide

---

Revision #16
Created 2020-10-15 20:12:02 CST by A-Lang (Admin)
Updated 2026-03-09 11:53:28 CST by A-Lang (Admin)