

# Docker

7

18 Feb 2022

## The most Basic Dockerfile

- Put this in a file named Dockerfile

```
FROM busybox
RUN echo "building simple docker image"
CMD echo "Hello Container"
```

```
docker build -t hello
```

← current Directory

## Installing Program with Docker build

- Put this in a Dockerfile

```
FROM debian:sid
RUN apt-get -y update
RUN apt-get install nano
CMD "nano" "/tmp/notes"
```

## Adding a File through Docker build

- Put this in a Dockerfile

```
FROM example/nanoor
ADD notes.txt /notes.txt
CMD "nano" "/notes.txt"
```

← Local Image

# Dockerfile Syntax

## The FROM Statement

- Which image to download and start from
- Must be the first command in your Dockerfile

~~FROM~~

```
FROM java:8
```

## The MAINTAINER Statement

- Defines the author of this Dockerfile

```
MAINTAINER firstname lastname <email@example.com>
```

## The RUN Statement

- Runs the command line, waits for it to finish, and saves the result

```
RUN unzip install.zip /opt/install
```

## The ADD Statement

- Adds local files local

```
ADD run.sh /run.sh
```

*image*

- Adds the contents of tar archives

```
ADD project.tar.gz /install/
```



• Works with URLs as well

(11)

ADD `https://projod.example.com/download/1.0/projod.rpm /projod`

## The ENV statement

- Sets environment variables
- Both during the build and when running the result

```
ENV DB_HOST = db.production.example.com  
ENV DB_PORT = 5432
```

## The ENTRYPOINT and CMD statements

- ENTRYPOINT specifies the start of the command to run
- CMD specifies the whole command to run
- If you have both ENTRYPOINT and CMD, they are combined together
- If your container acts like a command line program, you can use ENTRYPOINT
- If you are unsure, you can use CMD

## Shell Form vs. Exec Form

- ~~Entry~~ ENTRYPOINT RUN and CMD can use either form

- Shell form looks like this:

```
nano notes.txt
```

- Exec form looks like this:

```
["/bin/nano", "notes.txt"]
```

### The EXPOSE statement

- Maps a port into the container

```
EXPOSE 8080
```

### The VOLUME statement

- Defines shared or ephemeral volumes

```
VOLUME ["/host/path/" : "/container/path/"]  
VOLUME ["/shared-data"]
```

- Avoid defining shared folder in Dockerfiles

### The WORKDIR statement

- Sets the directory the container starts in

```
WORKDIR /install/
```

### The USER statement

- Sets which user the container will run as

```
USER arthur  
USER 1000
```



14, Feb  
2022

13

## Multi-project Docker files

FROM ubuntu:16.04

RUN apt-get update

RUN apt-get <sup>install</sup> -y curl

RUN ~~cat~~ curl https://google.com | wc -c  
> google-size

ENTRYPOINT echo google is this big;  
cat google-size;

FROM ubuntu:16.04 as builder

RUN apt-get update

RUN apt-get -y install curl

RUN curl https://google.com | wc -c  
> google-size

FROM alpine

COPY --from=builder /google-size /google-size

ENTRYPOINT echo google is this big;  
cat google-size

## What Kernels Do



- Respond to messages from the hardware
- Start and schedule programs
- Control and Organize Storage

## Node

FROM node:latest

COPY 

WORKDIR /home/node

COPY  local file  To Docker container

RUN npm install

CMD ["nano", "index.js"]

## Inspect

> docker inspect container\_id( or ) container\_name

> docker network inspect network\_name

## File System

/var/lib/docker  
├── aufs  
├── containers  
├── images  
└── volumes

## Volumes

> docker run -v volume create data volume

> docker run -v data volume : /var/lib/mysql mysql

> docker run -v /data/mysql : /var/lib/mysql mysql



> docker run \

--mount type=bind, source=/data/mysql, target=/var/lib/mysql mysql

## Network

Default

Bridge

> docker run ubuntu

None

> docker run ubuntu  
--network=none

host

> docker run ub  
--network=host

## Bridge

> docker network create \

--driver bridge \

--subnet 182.18.0.0/16

custom-isolated network

## Private Registry

> docker run -d -p 5000:5000 \

--name registry registry:2

> docker image tag my-image \

localhost:5000/my-image

> docker push localhost:5000/my-image

> docker pull localhost:5000/my-image

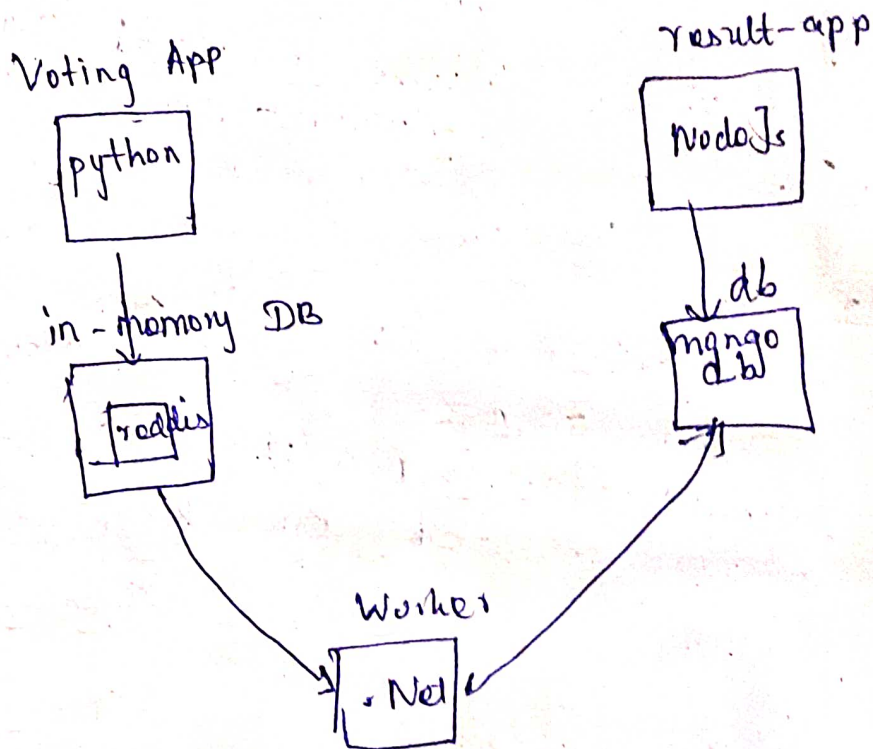
> docker pull 192.168.56.1100:5000/my-image

Remove All images

> docker image prune -a

Docker compose - build

docker-compose.yml



docker run

> docker run -d --name=redis redis

> docker run -d --name=db postgres

> docker run -d --name=vote -p 5000:80  
--link redis:redis voting-app

> docker run -d --name=result -p 5000:80  
--link db:db result-app

> docker run -d --name=worker --link db:db  
--link redis:redis worker



redis:

image: redis

db:

image: postgres:9.4

vote:

image: voting-app

ports:

- 5000:80

links:

- redis

result:

image: result

ports:

- 5000:80

links:

- db

worker:

image: worker

links:

- db

- redis

redis:

image: <sup>redis</sup>~~postgres~~:9.4

db:

image: postgres:9.4

vote:

build: <sup>dockerfile</sup>  
./vote

ports:

- 5000:80

links:

- redis

result:

build: <sup>Dockerfile</sup>  
./result

ports:

- 5000:80

links:

- db

worker:

build: ./worker

links:

- db

- redis

### 3 Different Versions

redis:  
image: redis  
db:  
image: postgres: 9.4  
vote:  
image: voting-app  
ports: - 5000:80  
links: - redis

Redis is a network  
data structure

Version: 2  
Services:  
redis:  
image: redis  
db:  
image: postgres: 9.4  
vote:  
image: voting-app  
ports: - 5000:80

With Redis separate  
network

Version: 3  
Services:  
redis:  
image: redis  
db:  
image: postgres  
vote:  
image: voting-  
ports: - 5000:80



## Dockerfile Docker Flask

19

```
FROM python: alpine3.7
```

```
WORKDIR /python-docker
```

```
COPY requirement.txt requirement.txt
```

```
RUN pip3 install -r requirement.txt
```

```
COPY . .
```

```
CMD ["python3", "-m", "flask", "run",  
     "--host = 0.0.0.0"]
```

> docker build -t flask-app-image .

> docker run -d -p 5005:5000  
--name = flask-app-container flask-app-image

> docker exec -ti flask-app-container sh

Check the running status

> curl -i localhost:49160

## Node Js

```
FROM node: 16
```

```
WORKDIR /usr/src/app
```

```
COPY package*.json .
```

```
RUN npm install
```

```
COPY . .
```

```
EXPOSE 8080
```

```
CMD ["node", "server.js"]
```

## Tomcat

```
FROM tomcat: 8.0-alpine
ADD sample.war /usr/local/tomcat/webapps
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

## Nginx

```
FROM nginx: latest
WORKDIR /usr/share/nginx/html
ADD homepage.html .
EXPOSE 80/tcp
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

index.html file present under the WORKDIR