Docker - What is it?

Hampus Londögård

hampus.londogard@afry.com

Docker

> *an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of* ***OS-level virtualization*** *on Linux.*

https://docker-curriculum.com/#

Containers

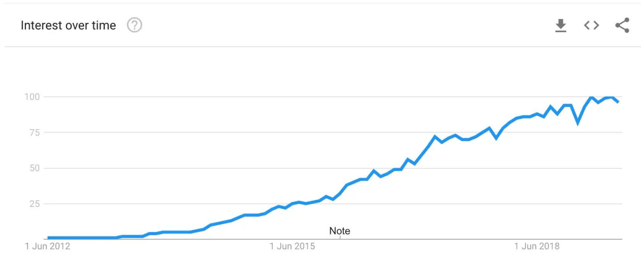
- Industry **Standard**
- Allows to **isolate a environment** in the OS
- Compared to VM it has **lower overhead**
  - Doesn’t virtualize hardware

Containers

Why use them?

The why

- Allows us to have **reproducible** environments
- **Decouple** Hardware and Code
- Decouple different Apps by **isolation**
- Easier to **deploy** and hence **scale up/down**



Containers

How to use them

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

Dockerfile

This can be your development environment!
No more dependency issues!

1. Define the container environment
2. (opt) Define how to run app/server

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

This is your server/inference deployment

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt  /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update  &&  apt-get install build-essential  -y
RUN pip install  --no-cache-dir  -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update && apt-get install build-essential -y
RUN pip install --no-cache-dir -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```dockerfile
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update &&  apt-get install build-essential -y
RUN pip install --no-cache-dir -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update && apt-get install build-essential -y
RUN pip install --no-cache-dir -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt  /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update  &&  apt-get install build-essential  -y
RUN pip install  --no-cache-dir  -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt  /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update  &&  apt-get install build-essential  -y
RUN pip install  --no-cache-dir  -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

1. Define the container environment
2. (opt) Define how to run app/server

```
FROM python:3.10-slim-bullseye

COPY . /app
COPY requirements.txt /app
WORKDIR /app
COPY .streamlit/* ~/.streamlit/

RUN apt-get update &&  apt-get install build-essential -y
RUN pip install --no-cache-dir -r requirements.txt
# EXPOSE 80 - not required, azure solve themself

ENV AZ_STORAGE_KEY=$AZ_STORAGE_KEY

ENTRYPOINT ["streamlit", "run"]
CMD ["1_📈_PIM_Tool.py", "--server.headless", "true",
"--client.caching", "true", "--server.runOnSave", "false"]
```

Dockerfile

The “Gotchas”

- Dockerfiles are “solid state”
    - Ones built it’s built
- Think long about dynamic parts
    - Or be ready to rebuild ;)

What to do now?

Build, Run & Push!

- `docker build -t <tag> <folder>`
- `docker run -p {from}:{to} <tag>`
    - (opt) Add volume, `-v /dev/local:/dev/docker`
- `docker push <tag>`

I have a hard time to remember

I have multiple containers

...

How do I make my life simple?

Docker Registry

- Pre-built images exists
  - `hub.docker.com/r/`**`nvidia/cuda/`**
  - `hub.docker.com/r/`**`fastai/fastai`**
  - `nvcr.io/`**`nvidia/pytorch`**`:22.08-py3`
  - ...etc

Aha!

Docker Compose

- A way to automatically re-run and compose multiple containers
  - `docker compose (up|down)`
- Makes it easy to
  - run multiple images together
  - automatically re-run images
  - “save” configurations

```
services:
 nc:
   image: nextcloud:apache
   environment:
     - POSTGRES_HOST=db
     - POSTGRES_PASSWORD=nextcloud
   ports:
     - 80:80
   restart: always
   volumes:
     - nc_data:/var/www/html
 db:
   image: postgres:alpine
   environment:
     - POSTGRES_PASSWORD=nextcloud
   restart: always
   volumes:
     - db_data:/var/lib/postgresql/data
   expose:
     - 5432
volumes:
 db_data:
 nc_data:
```

Kubernetes

How the “bigcorps” do it, including Cloud

> Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

https://kubernetes.io/

Kubernetes

How the “bigcorps” do it, including Cloud

- We’ll use it indirectly through Azure, AWS & GCP
- We’re surely not large enough to use it locally, Docker Compose suffice

Conclusions

- Isolated environment for server/development
- Horizontal scaling
- Standard