DOCKER

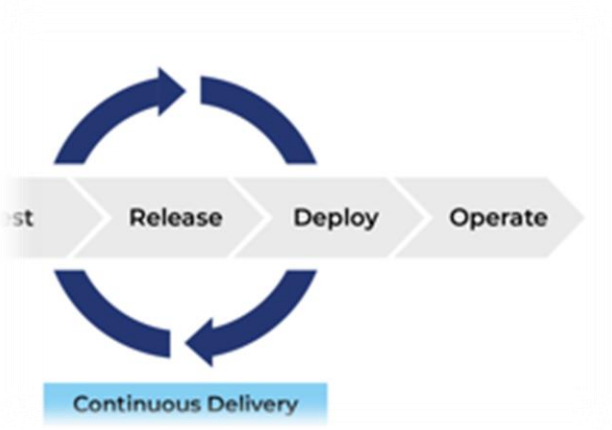
SELEZNEV ARTEM
SENIOR DE @ SBER

НА СЕГОДНЯ

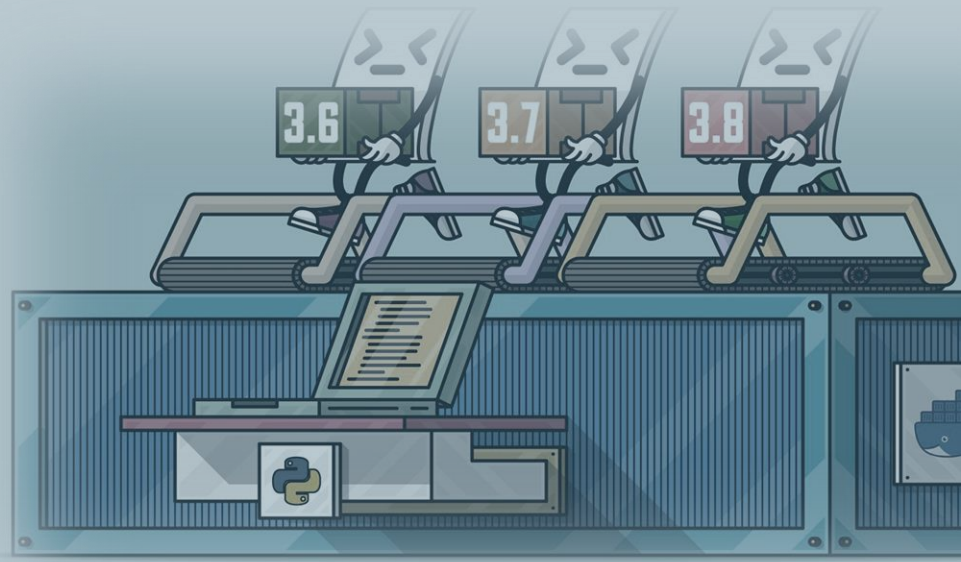
- **Docker**

НА СЕГОДНЯ

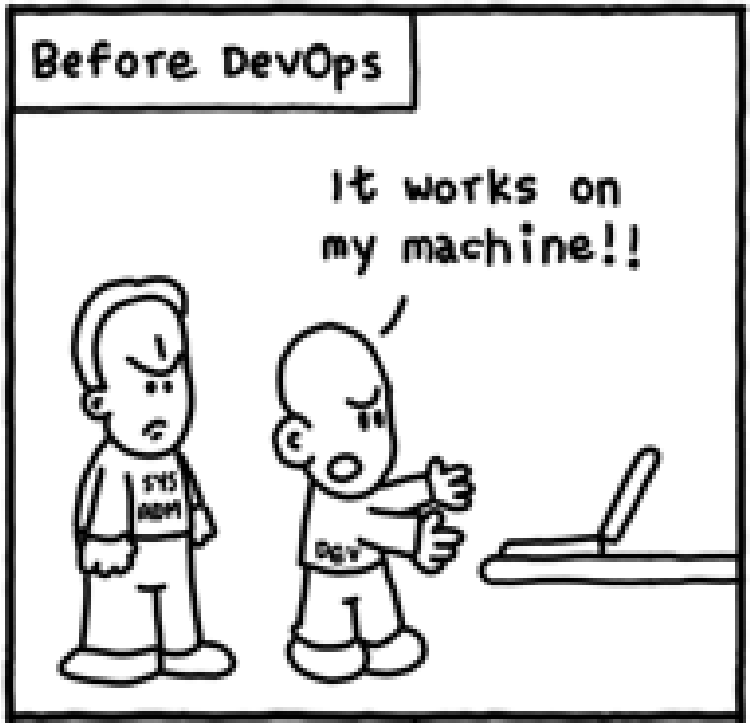
- **Docker**
- **CI/CD & Docker**



DOCKER?



DOCKER

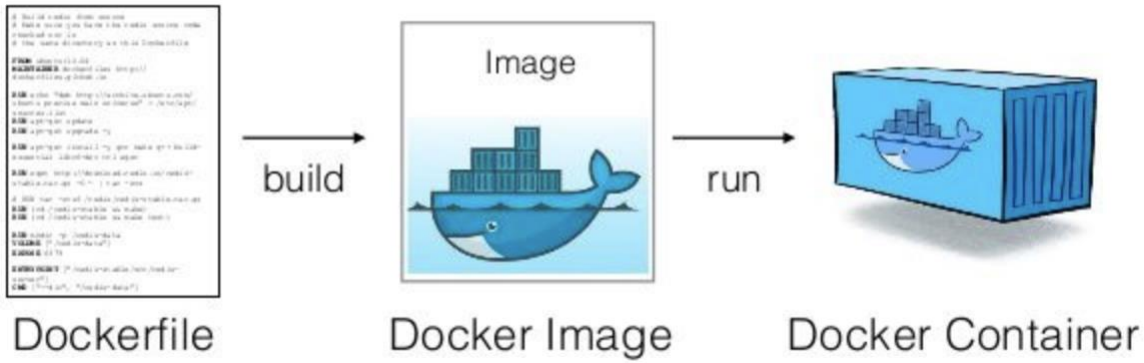


ЗАПОМНИМ О DOCKER

- **Docker != Virtual Machine**
- **Процесс на определенном порту**
- **Ваш финальный результат (модель + docker)**
- **DS кейсы:**
  - передача моделей и !рабочих результатов!
  - тестирование новых инструментов
  - АБ + тесты / Многорукие бандиты
  - Реактивная смена моделей (от обстоятельств)

DOCKER - ПРОСТО

- **Image** – скрипт базирующийся на образе, то что будем собирать (Class)
- **Layer** – часть последовательности для сборки Image (Inheritance)
- **Container** – собранный Image (Object)



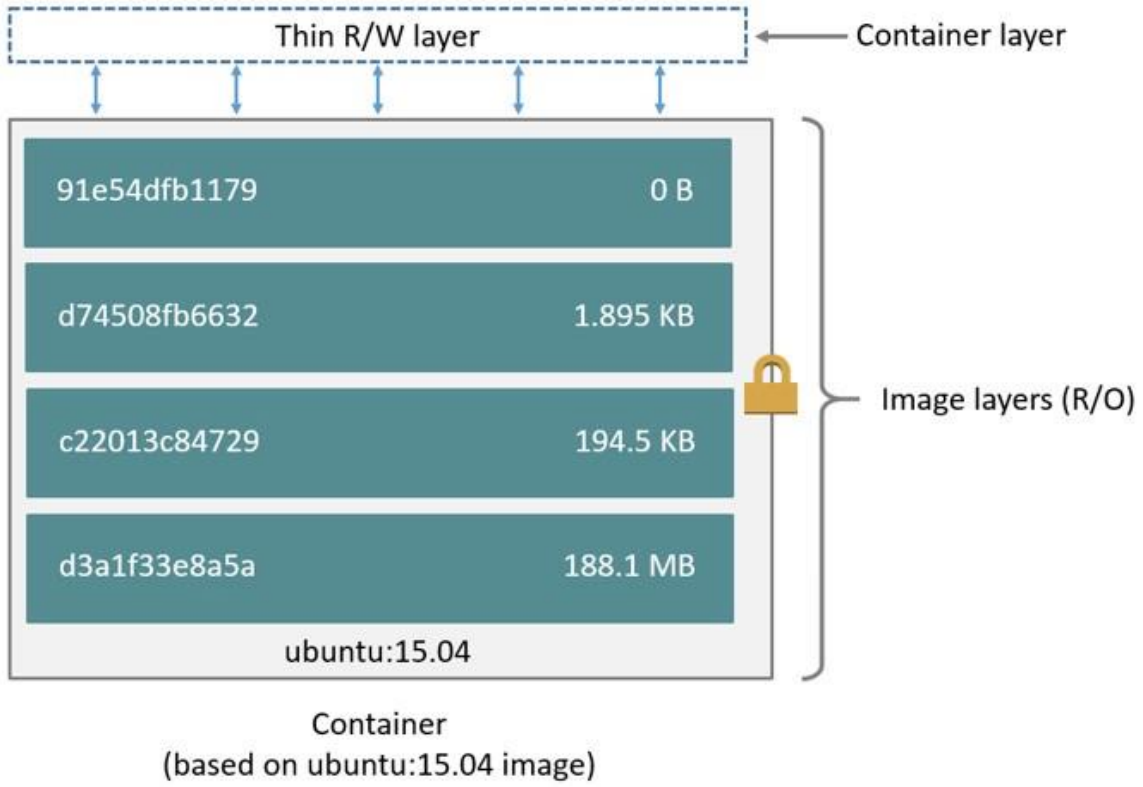
DOCKER - ПРОСТО

- **Image** – скрипт базирующийся на образе,
  то что будем собирать
- **Layer** – часть последовательности для
  сборки Image
- **Container** – собранный Image (процесс)
- **Dockerfile** – набор инструкций для
  формирования и сборки Image

DOCKER - ПРОСТО

Dockerfile

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```



DOCKER - ПРОСТО

- **Image** – скрипт базирующийся на образе,
  то что будем собирать
- **Layer** – часть последовательности для
  сборки Image
- **Container** – собранный Image (процесс)
- **Dockerfile** – набор инструкций для
  формирования и сборки Image
- **DockerHub/Container Registry** – репозиторий
  для контейнеров

УПАКОВАТЬ МОДЕЛЬ

DOCKERFILE

- **FROM** - на какой базе
- **ENV** - установка окружения
- **WORKDIR** – создание директории для работы
- **COPY / ADD**– копируем с «машины» в Docker контейнер
- **RUN** - первые команды в Shell | Bash
  (выполнено / установлено)
- **EXPOSE** - указание портов
- **CMD / ENTRYPOIN** - что запустить в докере
  [команда, параметры,,,,]

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)

```
#RUN nohup python serv/serv_app.py > /dev/null 2>&1&
```

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный
- Не забываем о .dockerignore

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный
- Не забываем о .dockerignore
- Установка ! только ! нужных библиотек

```
numpy==1.19.2
pandas==1.1.1
scikit-learn==0.23.2
xgboost==1.2.1
lightgbm==3.0.0
joblib>=0.14.1
pytest==6.1.2
pytest-html==2.1.1
pytest-cov==2.10.1
hypothesis==5.41.1
pytest-xdist==2.1.0
python-coveralls==2.9.3
flake8==3.8.4
dvc==1.9.1
pyyaml==5.3.1
kedro==0.16.6
fastapi==0.61.2
pydantic==1.7.2
starlette==0.13.6
uvicorn==0.12.3
```

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный
- Не забываем о .dockerignore
- Установка ! только ! нужных библиотек
- Один контейнер – одна задача

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный
- Не забываем о .dockerignore
- Установка ! только ! нужных библиотек
- Один контейнер – одна задача
- Минимальное кол-во слоёв / шагов при сборки

DOCKERFILE BEST PRACTICES

- Явный запуск процессов (без nohup)
- Контейнер – эфемерный
- Не забываем о .dockerignore
- Установка ! только ! нужных библиотек
- Один контейнер – одна задача
- Минимальное кол-во слоёв / шагов при сборке
- CMD – для запуска процесса внутри контейнера
- ENTRYPOINT для DevOps, не для DS !

DOCKERFILE

```
FROM python:3.7

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update \
    && apt-get -y install gcc make \
    && rm -rf /var/lib/apt/lists/*
RUN pip install --no-cache-dir --upgrade pip

WORKDIR /model

# copy & install requirements.txt
COPY deployml_course/requirements.txt /model/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# copy project
COPY deployml_course/ /model

EXPOSE 6789

# run model with API
ENTRYPOINT [ "python" ]
CMD [ "runner.py"]
```

DOCKERFILE

```
FROM python:3.7

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update \
    && apt-get -y install gcc make \
    && rm -rf /var/lib/apt/lists/*
RUN pip install --no-cache-dir --upgrade pip

WORKDIR /model

# copy & install requirements.txt
COPY deployml_course/requirements.txt /model/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# copy project
COPY deployml_course/ /model

EXPOSE 6789

# run model with API
ENTRYPOINT [ "python" ]
CMD [ "runner.py"]
```

?

DOCKERFILE

```
FROM python:3.7

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update \
    && apt-get -y install gcc make \
    && rm -rf /var/lib/apt/lists/*
RUN pip install --no-cache-dir --upgrade pip

WORKDIR /model

# copy & install requirements.txt
COPY deployml_course/requirements.txt /model/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# copy project
COPY deployml_course/ /model

EXPOSE 6789

# run model with API
ENTRYPOINT [ "python" ]
CMD [ "runner.py"]
```

DOCKERFILE

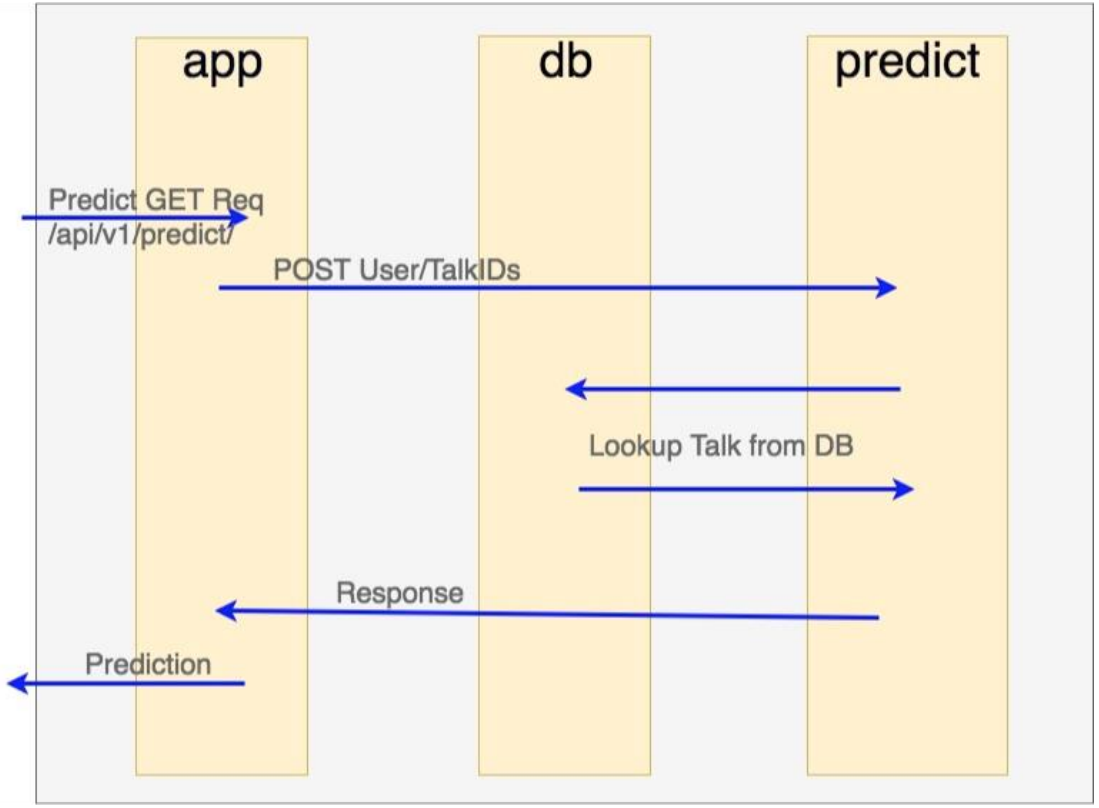
```
FROM python:3.7

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update \
    && apt-get -y install gcc make \
    && rm -rf /var/lib/apt/lists/*
RUN pip install --no-cache-dir --upgrade pip

WORKDIR /model

# copy & install requirements.txt
COPY deployml_course/requirements.txt /model/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# copy project
COPY deployml_course/ /model

EXPOSE 6789

# run model with API
ENTRYPOINT [ "python" ]
CMD [ "runner.py"]
```

DOCKERFILE

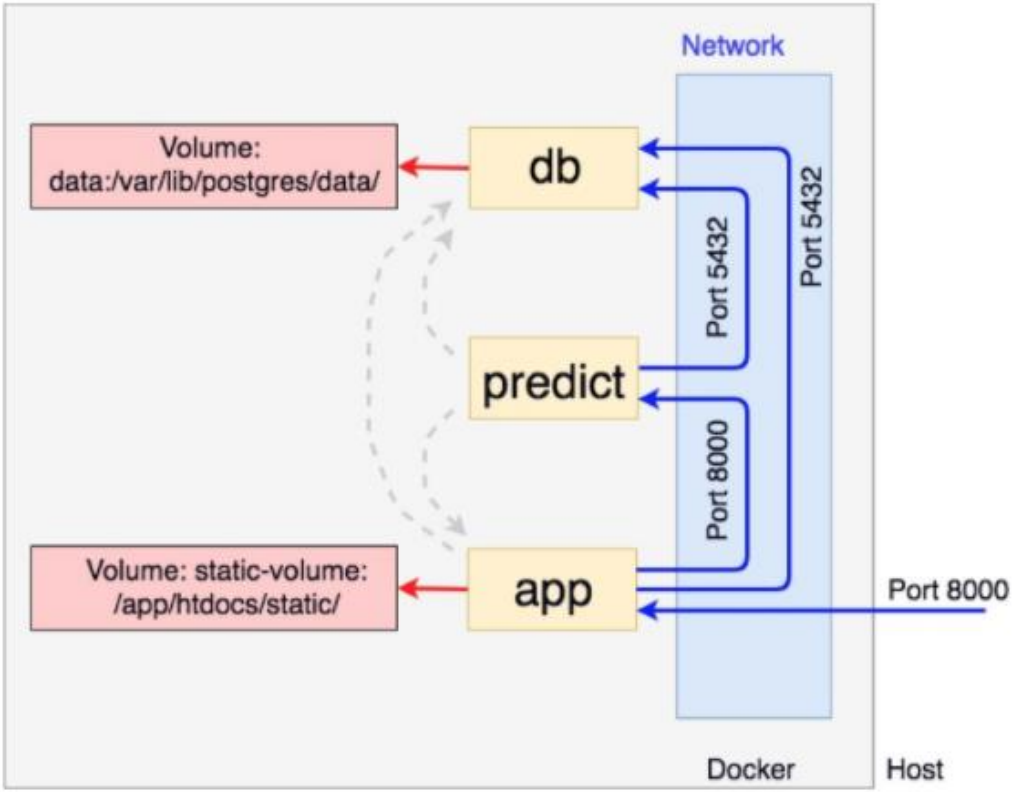
```
FROM python:3.7

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update \
    && apt-get -y install gcc make \
    && rm -rf /var/lib/apt/lists/*
RUN pip install --no-cache-dir --upgrade pip

WORKDIR /model

# copy & install requirements.txt
COPY deployml_course/requirements.txt /model/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# copy project
COPY deployml_course/ /model

EXPOSE 6789

# run model with API
ENTRYPOINT [ "python" ]
CMD [ "runner.py"]
```

DOCKER & ДОКУМЕНТАЦИЯ

Вся модель

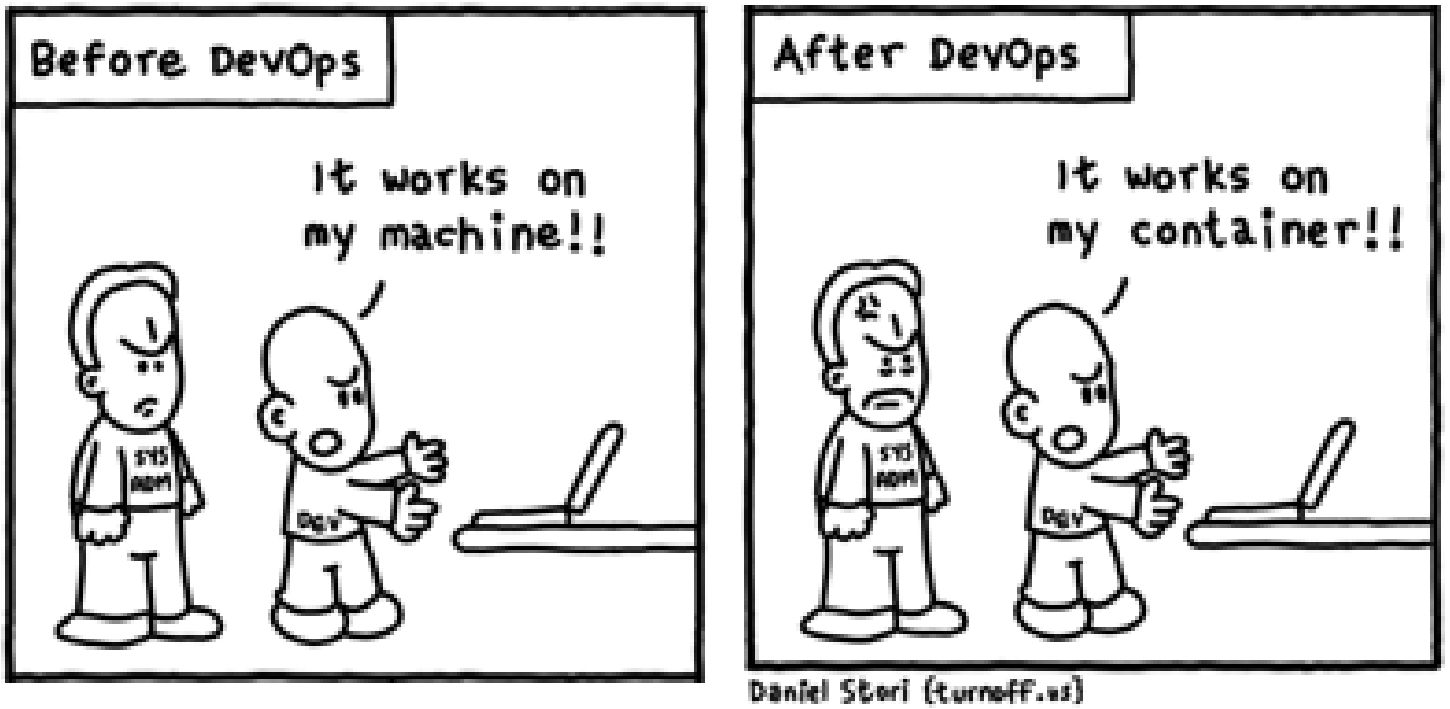


Контейнер + сеть

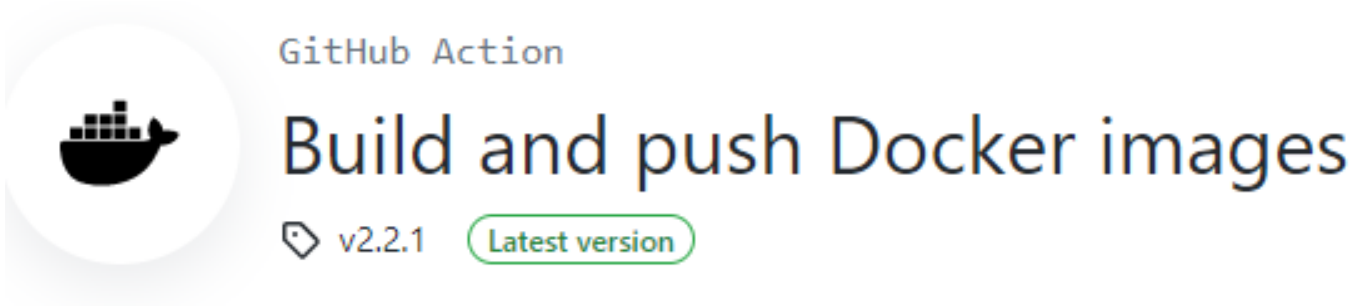


CI/CD & DOCKER

CI/CD & DOCKER


Before DevOps
It works on my machine!!
SYS ADM
DEV
After DevOps
It works on my container!!
SYS ADM
DEV
Daniel Stori (turnoff.us)

CI/CD & DOCKER



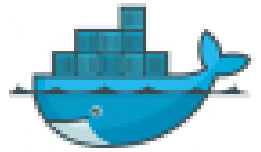
https://github.com/marketplace/actions/build-and-push-docker-images

CI/CD & DOCKER

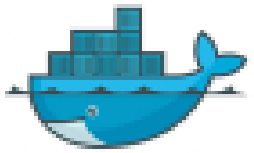
.dockerignore

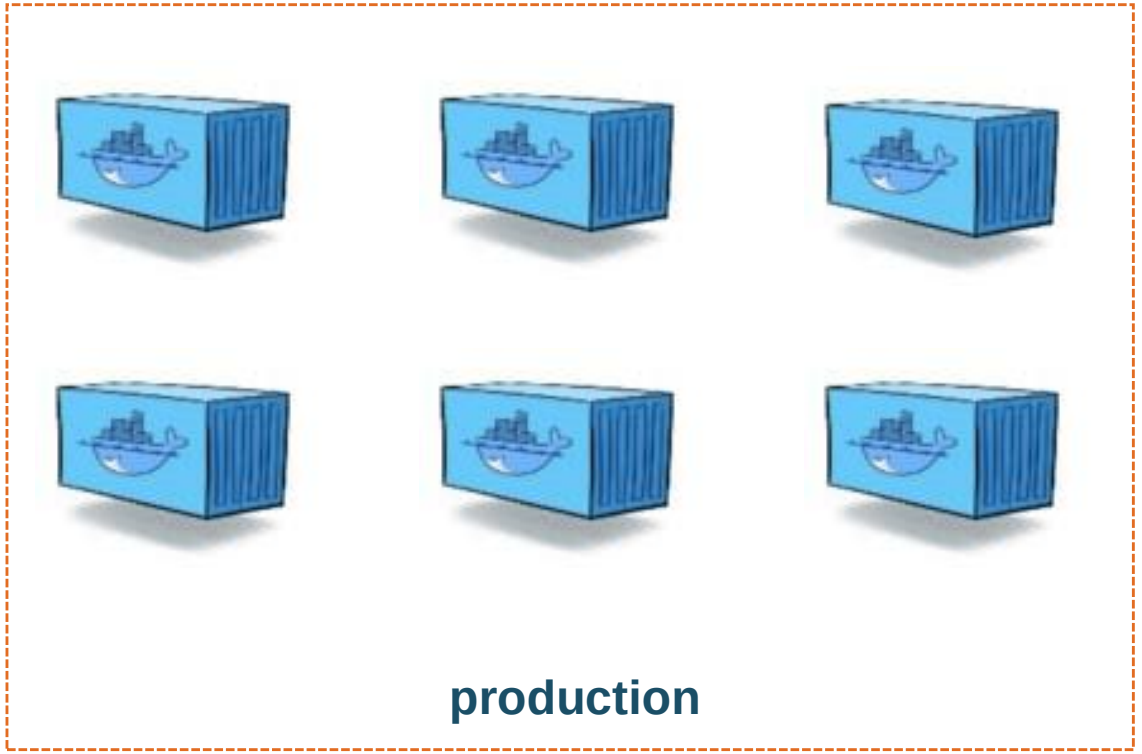
- checkpoints
- conf
- data
- docs
- expectations
- logs
- notebooks
- plugins
- src
- uncommitted
- .coverage
- .coveragerc
- .gitignore
- .kedro.yml
- great_expectations.yml
- info.log
- kedro_cli.py
- pyproject.toml
- README.md
- requirements.txt
- runner.py
- setup.cfg

```
name: Build and push
id: docker_build
uses: docker/build-push-action@v2
with:
  push: true
  tags: user/app:latest
  build-args: |
    arg1=value1
    arg2=value2

name: Image digest
run: echo ${{ steps.docker_build.outputs.digest }}
```

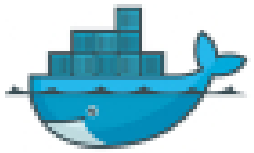


CI/CD & DOCKER

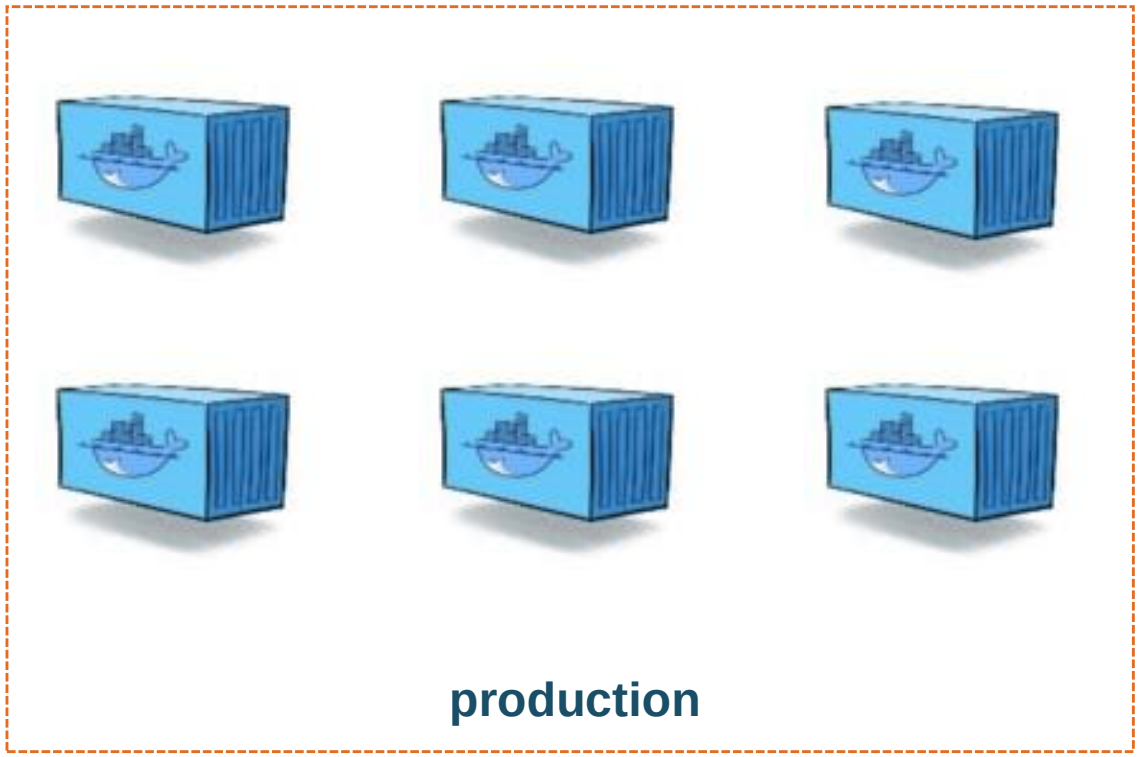




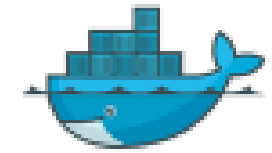
CI/CD & DOCKER & KUBER



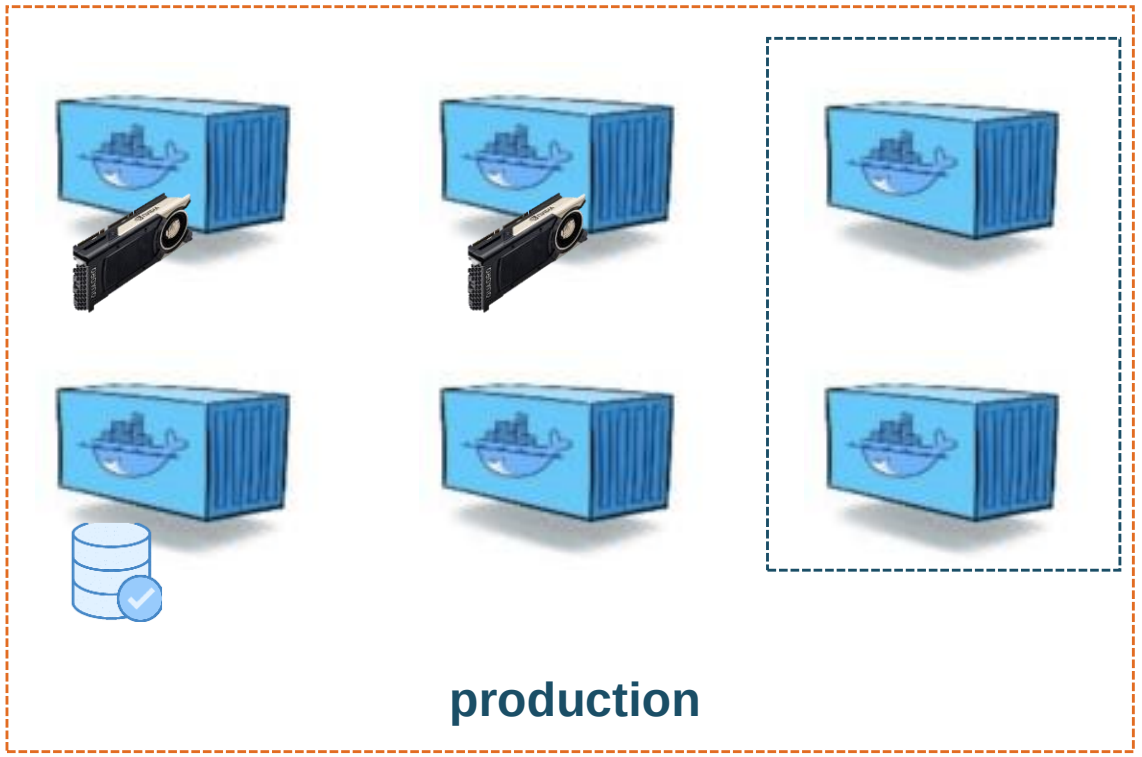
kubectl



CI/CD & DOCKER & KUBER



kubectl



CI/CD & DOCKER & KUBER

Docker is deprecated? - https://habr.com/ru/company/southbridge/blog/531820/

ДОПОЛНЕНИЕ

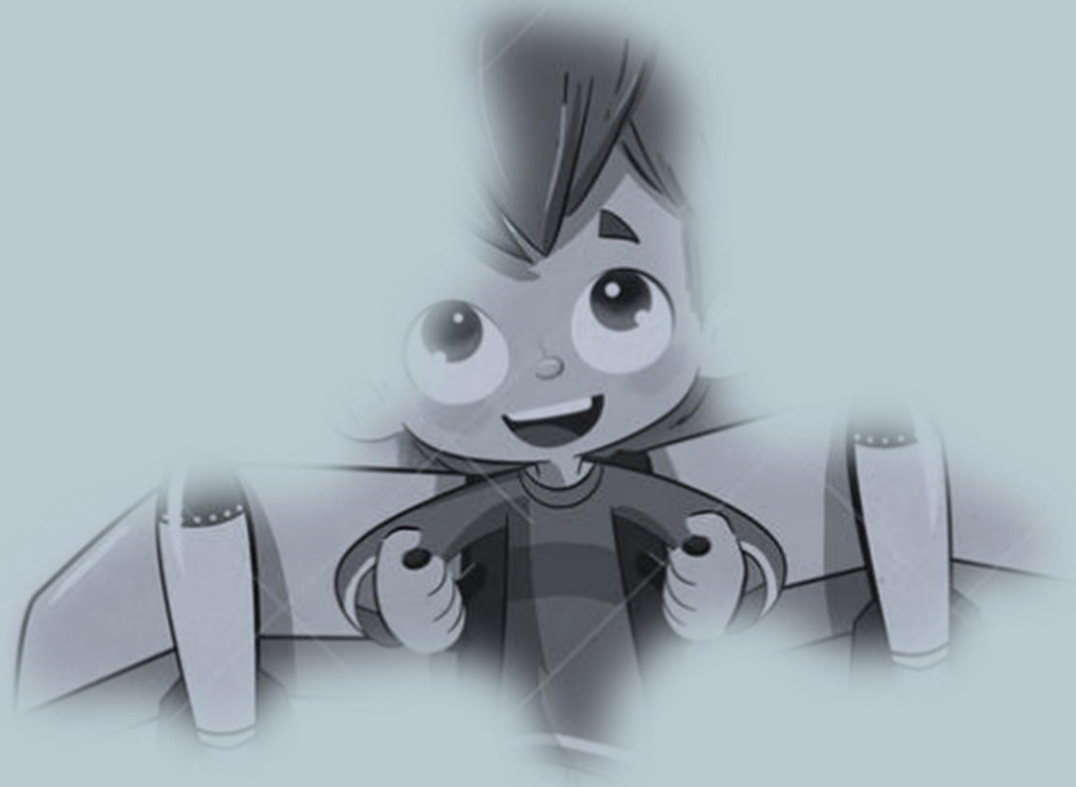
Docker. Полезные ссылки. #5

Open **NameArtem** opened this issue 2 hours ago · 0 comments

NameArtem commented 2 hours ago • edited

- Docker Cheet Sheet
- Best practices for writing Dockerfiles
- GitHub Action - Docker
- Docker Labs - сборник примеров от сообщества Docker

ЗАДАНИЕ



МОДЕЛЬ В КОНТЕЙНЕР

- **Создать Dockerfile. В качестве базового образа использовать: python3.№ или alpine**
- **Используя .dockerignore скопировать в image, только необходимые для работы файлы (для API предикта + передачи данных)**
- **Создать Docker Image с моделью**
- **Запустить Docker Container на основе созданного Docker Image. Модель должна возвращать результат по API**