Docker

AI Education

Контрольные вопросы по прошедшему занятию

- Что такое **контейнер**?
- Что нужно, чтобы **запустить контейнер**?
- Что такое **образ**?
- Что такое **реестр**?

План занятия

- Образы (продолжение)
- Образы -> Контейнеры
- Собственный образ

Образы (откуда берутся)

- Репозиторий
  $ docker pull <название образа>
- Контейнер
  $ docker commit <название контейнера>
- Dockerfile

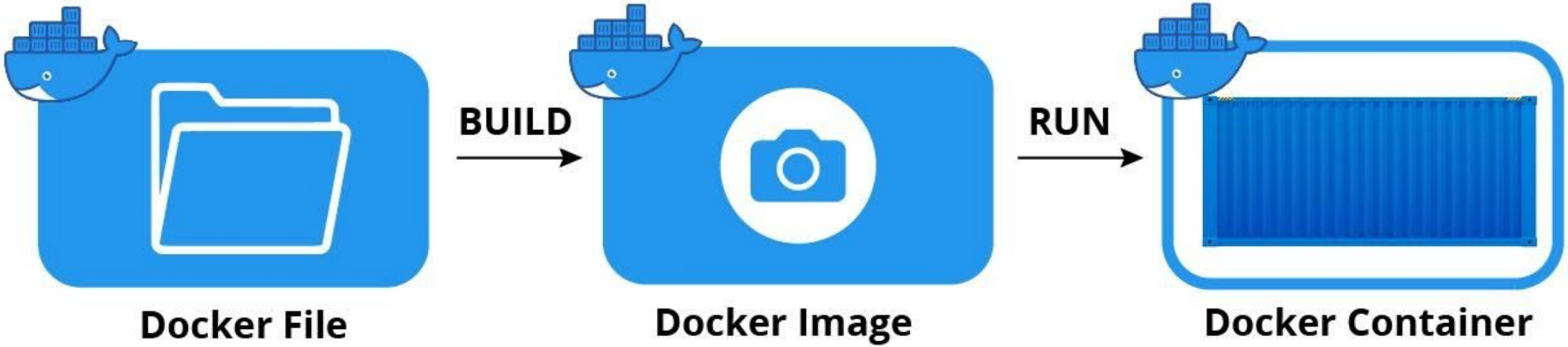
Образы (как опознать)

- REPOSITORY
- TAG
- IMAGE ID
- CREATED
- SIZE

Образы (слои)

- Образ как слоеный пирог
- Всегда есть базовый слой
- Каждый слой это diff!
- Слои в итоговом образе создают только инструкции:
  FROM
  RUN
  COPY
  ADD

Манифест



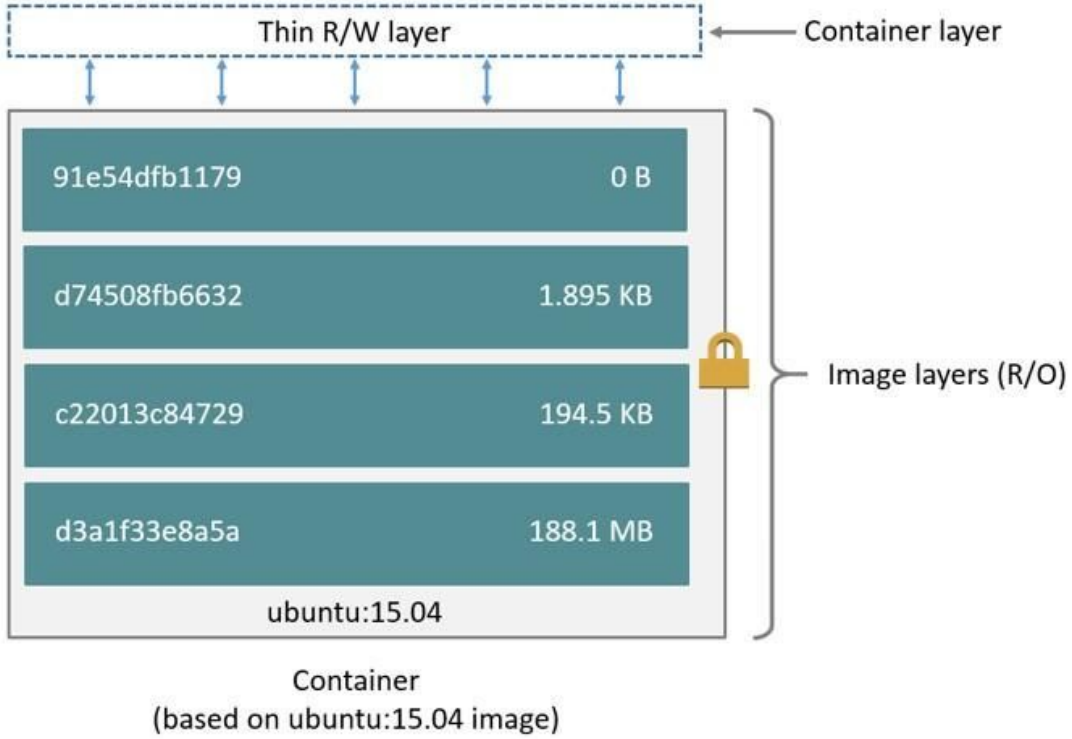
Dockerfile (что это)

- Dockerfile — это конфигурационный файл, в котором описаны инструкции, которые будут применены при сборке образа и запуске контейнера

- Создается в корневой директории проекта

- Выполняется сверху вниз

Структура контейнера



Dockerfile (основные инструкции)

- **FROM** — задает базовый (родительский) образ
- **LABEL** — описывает метаданные
- **ENV** — устанавливает постоянные переменные среды
- **RUN** — выполняет команду и создает слой образа
- **COPY** — копирует в контейнер файлы и папки
- **ADD** — копирует файлы и папки в контейнер
- **CMD** — описывает команду с аргументами, которую нужно выполнить когда контейнер будет запущен
- **WORKDIR** — задаёт рабочую директорию для следующей инструкции
- **ARG** — задает переменные для передачи Docker во время сборки образа
- **ENTRYPOINT** — предоставляет команду с аргументами для вызова во время выполнения контейнера
- **EXPOSE** — указывает на необходимость открыть порт
- **VOLUME** — создает точку монтирования для работы с постоянным хранилищем

FROM, LABEL и другие

- Dockerfile должен начинаться с инструкции FROM, или с инструкции ARG, за которой идет инструкция FROM

- Пример

```
FROM Ubuntu:22.04
LABEL org.opencontainers.image.authors="org@example.com"
ENV ADMIN="student"
COPY . /app
RUN make /app
RUN rm -r $HOME/.cache
CMD python /app/app.py
```

DEMO (в PyCharm)

- Предобработка данных
  eda.py

- Построение модели
  model.py

- Оборачивание модели в REST сервис
  app.py

- Пишем Dockerfile
  Dockerfile

Dockerfile

```
FROM python:3

RUN pip install --no-cache-dir flask catboost pandas scikit-learn


COPY app.py /app/app.py

COPY trained_model.cbm /app/trained_model.cbm


WORKDIR /app

EXPOSE 5000

ENTRYPOINT ["python"]

CMD ["app.py"]
```

Сборка и запуск

- Собираем образ

  $ docker build -t mlmodel:latest .

- Запускаем контейнер

  $ docker run -p 5000:5000 mlmodel:latest &

- Тестируем что получилось

  $ diff <(curl -d '{"PassengerId": 762, "Pclass":3, "Name": "Nirva, Mr. Lisakki Antino Aijo", "Sex": "female", "Age": 34, "SibSp": 4, "Parch": 3, "Ticket": "a", "Fare": 1.0, "Cabin": "A", "Embarked": "A"}' -H 'Content-Type: application/json' http://localhost:5000/predict | tr -d "\n") <(echo '{"Survived_Probability":0.6485056503550757}' | tr -d "\n")