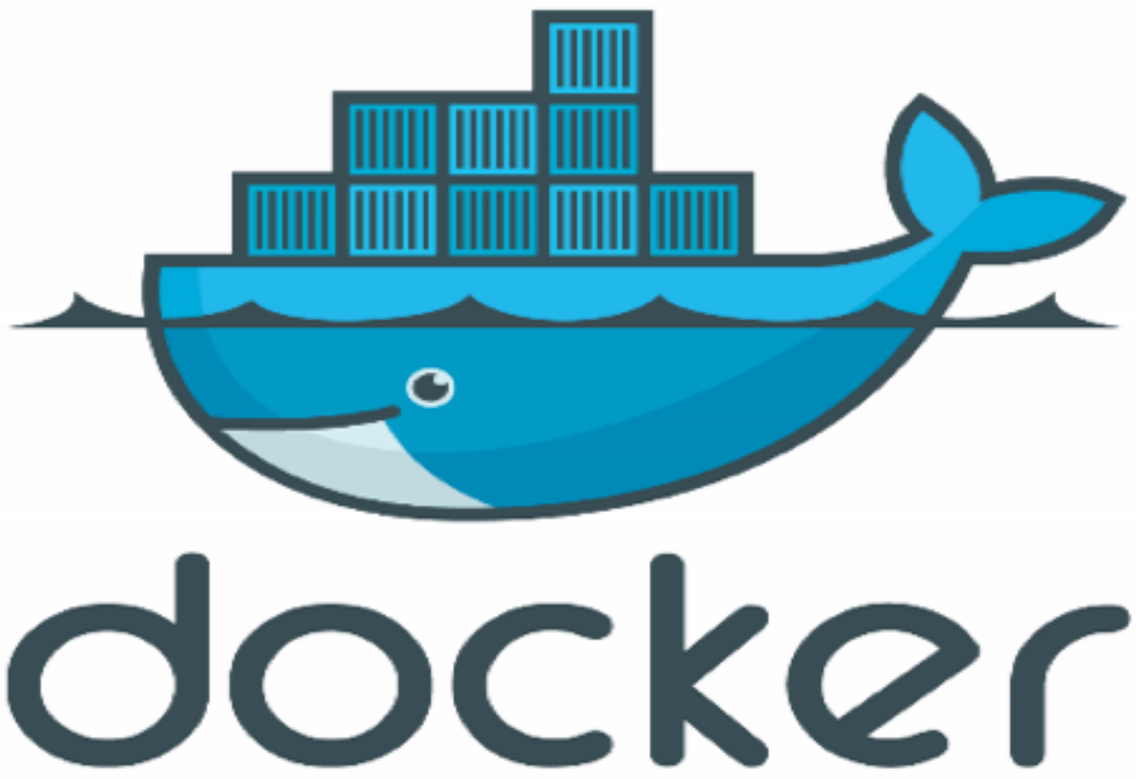
ПромПрог

Лекция 1. Базовые вещи


docker

Что такое Docker?

- Докер — это открытая платформа для разработки, доставки и эксплуатации приложений.

- отделить ваше приложение от вашей инфраструктуры

- позволяет запускать практически любое приложение, безопасно изолированное в контейнере

Когда Docker полезен?

- упаковывание приложения (и так же используемых компонент) в docker контейнеры;
- раздача и доставка этих контейнеров вашим командам для разработки и тестирования;
- размещение контейнеров на серверах, как в дата центры так и в облака.

Внутри Докера

- образы (images)

- реестр (registries)
- контейнеры

Как работает Docker?

- Каждый образ состоит из набора уровней.
- Docker использует union file system для сочетания этих уровней в один образ.
- В основе каждого образа находится базовый образ.
- использовать образы как базу для создания новых образов.

Docker

- run - запускает контейнер
  - -i - interactive
  - -t - tty
  - —rm - удаляем образ после запуска
  - -p, -P - проброс портов
  - —name - имя контейнера
  - -e - environment

Пример

- docker run -it --rm --name myubuntu -e TEST=test ubuntu /bin/bash

Docker

- docker ps - список контейнеров
- docker images - список images
- docker stop - остановка койнтера
- docker rm - удаление контейнера
- docker rmi - удаление image’a

Dockerfile

```
FROM eva-dock.sberned.ru/smartdata/conda-classifier:0.0.2

ADD requirements.txt requirements.txt
RUN pip install --upgrade pip
RUN pip install -r requirements.txt

ADD . /Service
WORKDIR /Service

RUN mkdir -p /root/.keras && touch ~/.keras/keras.json && \
    echo '{ "floatx": "float32", "epsilon": 1e-07, "backend": "tensorflow",
"image_dim_ordering": "tf" }' > ~/.keras/keras.json


EXPOSE 8888
ENTRYPOINT [ "/opt/conda/bin/python", "__main__.py"]
```

Сборка image’a

- docker build -t тег .

requirements.txt

- Файлик с библиотеками python

Docker Compose

- Поднятие нескольких контейнеров одновременно
- Прокидывание сетевого взаимодействия

ComposeFile

```
version: "3.3"

services:

  cppapp:
    build: .
    restart: always
    entrypoint: python -m service
    networks:
      - backend

  integration_tests:
    build: ./integration_tests
    links:
      - cppapp:cppapp
    environment:
      - SERVICE_HOST=cppapp
    networks:
      - backend

networks:
  backend:
```

Базы Данных

SQL

- PostgreSQL
- реляционная
- SQL
- легко и понятно

CAP

- Теорема CAP (известная также как теорема Брюера) — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

  - согласованность данных (англ. consistency) — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;

  - доступность (англ. availability) — любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;

  - устойчивость к разделению (англ. partition tolerance) — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

ACID

- Atomicity — Атомарность. Атомарность гарантирует, что никакая транзакция не будет зафиксирована в системе частично.

- Consistency — Согласованность. Не та же, что в CAP

- Isolation — Изолированность. Во время выполнения транзакции параллельные транзакции не должны оказывать влияние на её результат.

- Durability — Устойчивость. Независимо от проблем на нижних уровнях (к примеру, обесточивание системы или сбои в оборудовании) изменения, сделанные успешно завершённой транзакцией, должны остаться сохранёнными после возвращения системы в работу.

NoSQL

- MongoDB
- Легко кластеризуется
- Документоориентированная
- Без SQL

Прочие БД

- InfluxDB - TimeSeries база данных

Очереди

- RabbitMQ

Терминология

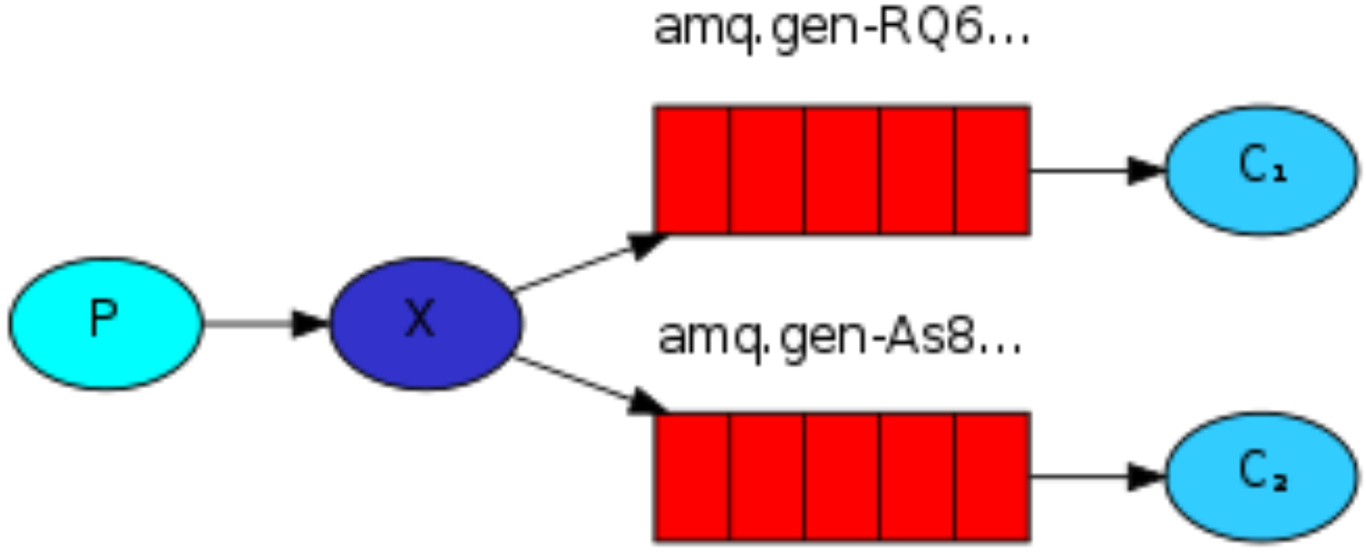
- Producer (поставщик) - программа, отправляющая сообщения
- Queue (очередь) – буффер, хранящий сообщение
- Consumer (подписчик) - программа, принимающая сообщения.

Точка Доступа (Exchange)

- Основная идея в модели отправки сообщений Rabbit – Поставщик(producer) никогда не отправляет сообщения напрямую в очередь. Фактически, довольно часто поставщик не знает, дошло ли его сообщение до конкретной очереди.

- Вместо этого поставщик отправляет сообщение в точку доступа. В точке доступа нет ничего сложного. Точка доступа выполняет две функции:

Exchange



pika

- pika - python обвязка для работы с кроликом

Producer

```
connection = pika.BlockingConnection(
    pika.URLParameters("amqp://guest:guest@localhost:
32769")
)
channel = connection.channel()
channel.queue_declare(queue='hello')
data += "a" * (random.randrange(1, 100 * 1024))
channel.basic_publish(exchange='',
    routing_key='hello',
    body=data.encode()
)
```

Consumer

```
connection = pika.BlockingConnection(
    pika.URLParameters("amqp://guest:guest@localhost:32769")
)
channel = connection.channel()
channel.queue_declare(queue='hello')

def callback(ch, method, properties, body):
    now_time = datetime.datetime.now().timestamp()
    recieved = (body.decode())
    recieved_time = float(recieved)

    print(now_time-recieved_time)

channel.basic_consume(callback,
            queue='hello',
            no_ack=True)

channel.start_consuming()
```

ДЗ

- Написать 2 программы
  - Одна принимает на вход строчку и кладет ее в очередь
  - Вторая слушает очередь и кладет строчки в базу