Если вы видите что-то необычное, просто сообщите мне.

???????? ???????????? ??? ?????? ??????????.

????????? ??????:

Нам понадобится приложение,которое мы собираемся развернуть на проекте.

```
git clone https://github.com/gasick/docker-compose-exempleapp --branch onlyapp
```

Если вы не знаете с чего начать, тогда смотрите в README проекта. Обычно в нем разрабочики указывают подготовительные шаги для того, чтобы развернуть приложени локально. Это позволит набросать какой-то простой пример Dockerfile от которого уже можно будет двигаться дальше. Ошибки при создании докер файла будет говорить, о том, что нам необхдимо добавить в докер, чтобы приложение могло запуститься.

> “ Наша задача автоматизировать шаги развертывания приложения.

Результатом нашей работы должна получиться подобнаяструктура папок:

```
.
├── docker-compose.yml
├── Dockerfile
├── example.env
└── todoapp
    ├── go.mod
    ├── go.sum
    └── todo.go
```

- `todoapp` - папка в которой хранится проект который необходимо развернуть

- `Dockerfile` - файл создания контейнера с приложением внутри
- `docker-compose.yml` - файл конфигурацией инфраструктуры нашего приложения, то есть наше приложение обернутое в контейнера, а так же все сервисы необходимые для его работы.
- `example.env` - пример настроек для простоты перемещения проекта.

?????????? ?????????? ? docker ?????????

В нашем примере, мы воспользуемся двумя контейнерами. Один будет использоваться для построения приложения, другой уже для его запуска.

Dockerfile

```
# Указываем какой образ мы будем использовать в качестве основы проекта, и присвоем ему имя build
FROM golang:1.16 as build
# Устанавливаем необходимое по, в данном случае это только git но этот список может быть сильно больше.
RUN apt update && apt install -y git
# Указываем рабочую папку
WORKDIR /app
# Копируем в рабочую папку необходимые файлы
COPY todoapp/go.mod .
COPY todoapp/go.sum .
# Подготавливаем окружение, скачиваем необхдимые для построения зависимости.
RUN go mod download
# Копируем проект
COPY todoapp .
# Запускаем build проекта
RUN go build -o /out/app /app

# Теперь на основе нового образа будем создавать сам рабочий контейнер.
FROM fedora
# Копируем из базового образа наше приложение для указания образа используется ключ --from=build
```

```
COPY --from=build /out/app /app
# Указываем директиву с которой будет запускаться проект
CMD ["/app"]
# Так как проект слушает порт для подключения клиентов, выставляем его.
EXPOSE 8000
```

Тепреь давайте прверим, что Dockerfile коректен и мы можем упаковать наше приложение:

```
docker build -t test .
```

Если билд прошел, и все шаги отработали идем дальше. Запускать проект пока не нужно, он всё равно выдаст ошибку, так как для работы необходим Postgres.

??????? docker-compose, ????????? ??????????? ? ???? ?????? ? ???? ??????????????.

docker-compose.yml

```
version: "3"

services:
  todoapp:
    build: .
    ports:
      - 8000:8000
    env_file:
      - example.env
    depends_on:
      - postgres

  postgres:
    image: postgres:13-alpine
```

```
    restart: always
    env_file:
      - example.env
    volumes:
    - ./postgres/data:/var/lib/postgresql/data
    - ./postgres/dumps:/dumps
```

example.env

Для полноценной работы в проекте нам нужен example.env.

Вынесение переменных окружения позволяет легко переносить проект с сервера на сервер, так же упрощает обслуживание. И настройку приложения в процессе работы, не прибегая к помощи разработчиков.

Для данного проекта файл будет являтся ключничей, из него postgres будет знать с какими данными создавать пользователь/пароль/бд, а приложение будет знать с какими параметрами подключаться к ней:

```
POSTGRES_DB=db
POSTGRES_USER=user
POSTGRES_PASSWORD=password
POSTGRES_HOST=postgres
POSTGRES_PORT=5432
```

Во время разрабтки, и работы сервера в целом, мы привыкли к тому, что все общение различных сервисов происходит через `localhost`. В случае с docker-compose это не работает. Для понимания того, что произойдет после поднятия docker-compose, представьте себе локальную сеть, в которй большое количество серверов. Все сервисы запущенные на одной машиние, теперь разнесены таким образом, чтобы на каждой машине в сети работал только один сервис, то есть общение может быть только исключительно через локальную сеть, в реальной сети такое общение будет происходить с помощью ip, в docker-compose мы указываем названия сервисов.

> То есть во время работы приложение и бд не будут доступны через localhost, друг для друга они будут доступны только через свои имена:

> todoapp, postgres.

????????? ??????

??? ??????? ??????? ?????????:

```
docker-compose up
```

??? ???????? ??????? ? ???????? ?????????:

```
curl http://localhost:8000/ -v
```

В ответе должна быть строка: `Всё работает!`

??? ????, ????? ????????? ????? ?????????? ? ??, ????????????? ?????? ????????:

```
curl -H "Content-Type: application/json" http://localhost:8000/todos/ -d '{"name":"Wash the garbage","description":"Be especially thorough"}' -v
```

В ответ прилетит json с ID нашего todo

Чтобы проверить, что у нас всё работает, из текущей папки:

1. входим в docker контейнер

```
docker-compose exec postgres psql -U user db
```

2. Просим показать содержание таблицы todos

```
select *from todos;
```

В ответе видим таблицу с нашими вводными данными:

```
 id |       name       |      description
----+------------------+-----------------------
  1 | Wash the garbage | Be especially thorough
(1 row)
```

Revision #4

Created 22 September 2021 11:55:26 by gasick

Updated 8 November 2023 16:10:47 by gasick