GO-11 01: Заворачиваем наш сервис в docker-образ

Описание:

В текущем модуле мы будем «заворачивать» наш сервер в docker-образ и запускать из этого образа контейнер. Мы также научимся передавать нашему серверу в контейнере параметры конфигурации.
В первом задании создадим dockerfile - своего рода скрипт для описания параметров окружения, сборки нашего сервера, и собственно, его запуска. Затем соберем из этого файла полноценный docker-образ.
Образы Docker
Образы Docker сейчас являются самым удобным и распространенным способом развертывания серверного программного обеспечения.
Для образов есть своего рода репозитории (registry) для их хранения, например, самый известный - dockerhub, который и используется по умолчанию для поиска при сборке образов.
Он хранит в себе уже готовые образы самых разных продуктов и их сочетаний, так что, по большей части, вам фактически нужно собирать свои образы исключительно для своих же продуктов.
Свои образы изначально хранятся локально, но ничто не мешает выгрузить их, для примера, на dockerhub, чтобы этот образ был доступен и другим.
Dockerfile
Давайте рассмотрим простой докер-файл и разберем его команды:

```
FROM golang:alpine
WORKDIR /files
COPY hello.go /files
RUN go build -o /files/hello hello.go
ENTRYPOINT ["/files/hello"]
```

Первая строчка:

```
FROM golang:alpine
```

загружает готовый образ популярной минималистической сборки Linux ОС alpine из репозитория dockerhub (только если он не был загружен ранее), в этом образе также уже установлена последняя стабильная версия Go.
Далее:

```
WORKDIR /files
```

«говорит» сборщику, что надо создать корневую директорию /files и перейти в нее (сделать текущей). На самом деле, это удобная замена двум отдельным командам:

```
RUN mkdir /files
RUN cd /files
```

То есть, любая строка после RUN выполняется как в консоли. Следующая команда:

```
COPY hello.go /files
```

копирует нашу программу в только что созданную рабочую директорию.
hello.go - это простая программа на go, которая печатает "Hello, Docker!":

```
package main

import (
      "fmt"
)

func main() {
      fmt.Println("Hello, Docker!")
}
```

и располагается в той же директории, где и Dockerfile.
У нас остается:

```
RUN go build -o /files/hello hello.go
ENTRYPOINT ["/files/hello"]
```

Первая команда вам уже известна - собираем исполняемый файл hello и кладем его в рабочую директорию.
Последняя команда задает исполняемый файл, который будет запущен сразу после создания контейнера из образа: логично, что это будет наш hello.
Собираем образ
Итак, сейчас у нас есть два файла: hello.go и Dockerfile. Теперь пора собирать образ, и для этого нам нужен docker. Как его установить, рассказано вот здесь.
Собираем командой:

```
docker build -t hello_go:v1 .
```

которая говорит, что надо собрать образ из файла скрипта по умолчанию (Dockerfile) и присвоить ему тег hello_go:v1. Собственно, тег - это имя нашего образа и его версия после двоеточия указывается произвольно.
Если все сделано правильно, то далее после выполнения команды:

```
docker images
```

мы увидим что-то подобное:

```
$ docker images
REPOSITORY      TAG             IMAGE ID        CREATED
     SIZE
hello_go        v1              9bec016712c4    About a minute ago
312MB
golang                alpine          f56365ec0638    About a minute
ago   310MB
```

Теперь наш образ готов к запуску контейнера.
Запуск контейнера из образа
Запускать контейнер не сложнее его сборки:

```
docker run --rm hello_go:v1
```

где hello_go:v1 - это наш ранее собранный образ, а --rm - опция для удаления контейнера (не образа!) сразу после завершения его работы. В этой команде есть еще много опций для запуска, но с ними вы разберетесь уже самостоятельно (см. docker run --help).
В итоге, после запуска вы должны увидеть в консоли строку Hello, Docker!.

Полезные ссылки:

- Официальная документация по docker-у
- Изучаем Docker, часть 3: файлы Dockerfile

Задание:

1. Сделайте форк проекта module11 в группу golang_users_repos/<your_gitlab_id>.
2. Создайте в своем проекте module11 ветку module11_01.
3. Напишите и закоммитьте Dockerfile .
4. Запустите веб-сервис в докер-контейнере, чтобы он был доступен на порту 8080 (менять номер порта в коде не надо, все делается через docker).
5. Откройте браузер и удостоверьтесь, что веб-сервис доступен по адресу http://localhost:8080/hello.
6. В качестве ответа пришлите:

- ссылку на merge request в ветку master вашего проекта ветки module11_01;
- команду из консоли, с помощью которой вы собирали образ;
- команду из консоли, с помощью которой вы запускали контейнер из образа.