1. В репозитории есть main.go файл для нашего приложения
2. За базовый образ можете взять golang:1.12.0-alpine3.9, но это не обязательно
3. Порт уже прописан в приложении, обратите на это внимание, когда будете делать проксирование в контейнер
4. Далее давайте сведём всё построение нашего Dockerfile примерно к одному стандарту:

- в контейнере нужно создать папку /app
- скопировать в /app наше приложение (можно всю текущую директорию)
- назначить эту директорию рабочей (текущей)
- скачать модуль download (go mod download)
- собрать наше приложение (go build -o main)
- точкой запуска, соответственно будет /app/main

1. После успешного билда образа, ставим на него тэг ваш_юзернейм/mygoapp
2. Загружаем наш образ в докерхаб
3. Запустить контейнер из образа, чтобы он отвечал на локальном порту 8888, работал в фоновом режиме
4. Докерфайл и все команды которые выполняли (можно скрины) загружайте в домашнее задание

---

1) Ставим docker с помощью скрипта.
2) Скачиваем main.go из дз
3) Создаем Dockerfile

```
FROM golang:1.12.0-alpine3.9 as builder

RUN mkdir /app

COPY . /app

WORKDIR /app
RUN go mod download
RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o main .

FROM scratch

COPY --from=builder /app/main /

EXPOSE 8081
CMD ["/main"]
```

4) Собираем image командой docker build . -t mygoapp

```
[root@centos7-8 dock]# docker build . -t mygoapp
Sending build context to Docker daemon  3.072kB
Step 1/10 : FROM golang:1.12.0-alpine3.9 as builder
 ---> 2205a315f9c7
Step 2/10 : RUN mkdir /app
 ---> Using cache
 ---> 1d558b322da4
Step 3/10 : COPY . /app
 ---> d11ddb3a864b
Step 4/10 : WORKDIR /app
 ---> Running in e5e92191e632
Removing intermediate container e5e92191e632
 ---> d41bbc51b58d
Step 5/10 : RUN go mod download
 ---> Running in 750343dfbe34
warning: pattern "all" matched no module dependencies
Removing intermediate container 750343dfbe34
 ---> dd082d1f9894
Step 6/10 : RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o main .
 ---> Running in bff3c84ba603
Removing intermediate container bff3c84ba603
 ---> 6ed6c3172b64
Step 7/10 : FROM scratch
 --->
Step 8/10 : COPY --from=builder /app/main /
 ---> Using cache
 ---> f2779913671a
Step 9/10 : EXPOSE 8081
 ---> Running in 2b1eea2112c5
Removing intermediate container 2b1eea2112c5
 ---> 460399fb8eba
Step 10/10 : CMD ["/main"]
 ---> Running in 44344b9e83c3
Removing intermediate container 44344b9e83c3
 ---> 0012eea764fb
Successfully built 0012eea764fb
Successfully tagged mygoapp:latest
```

5) Смотрим на получившийся образ командой docker images

```
[root@centos7-8 dock]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mygoapp             latest              0012eea764fb        9 minutes ago       7.31MB
<none>              <none>              6ed6c3172b64        9 minutes ago       363MB
<none>              <none>              5a8a0eddffdd        16 minutes ago      7.31MB
<none>              <none>              91b9f86fa8ee        16 minutes ago      363MB
<none>              <none>              bcc0dd79e1a3        32 minutes ago      7.37MB
<none>              <none>              1e461f3282d9        32 minutes ago      355MB
nginx               1.15                53f3fd8007f7        17 months ago       109MB
golang              1.12.0-alpine3.9    2205a315f9c7        19 months ago       347MB
```

6) Запускаем контейнер docker run -d -p 8888:8081 mygoapp

```
[root@centos7-8 dock]# docker run -d -p 8888:8081 mygoapp
0fee94a7c2e2e767d7f19c7b602eda5cd3acbcdb4bc4379cf16ab059c779fcc2
[root@centos7-8 dock]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
0fee94a7c2e2        mygoapp             "/main"             11 seconds ago      Up 9 seconds        0.0.0.0:8888->8081/tcp
  epic_elion
[root@centos7-8 dock]#
```

7) Проверяем работу контейнера

```
[root@centos7-8 dock]# curl localhost:8888
Hello, "/"[root@centos7-8 dock]#
```

8) После успешного запуска ставим новый таг командой
docker tag mygoapp aglayaq/mygoapp