Docker Project

Since you all now have a little bit of experience starting docker containers (docker homework), I want to put you all to the test. I want you all to take a given repository, and dockerize it!

Docker Compose files (and Dockerfiles, at that) have a lot of different moving parts. I've tried to help you all out by leaving #TODOs scattered throughout, to help familiarize yourself with what is needed, and what it is expecting.

I want you to fork this repository, and we will be using your copy of the repo to check your files.

Steps To Do

1. Complete Dockerfile
2. Complete Compose.yaml
3. Add Your Name!
4. Deploy and Test!

Dockerfile

Part of the Dockerfile is provided for you - it's random things that I would guaranteed get an email about, such as "how do I compile Go code" or "am I using the right image??"

Within this Dockerfile, you'll be writing the second step - running the image.

The steps are provided in the file, but if you want a checklist to work off of:

- Set WORKDIR to the /app directory inside our image.
- Copy our current directory to /app/server (hint - use --from=builder to utilize step 1).
- Expose Port 8080
- Run the server (hint: CMD tag and ./server)

compose.yaml

Once again, part of the compose.yaml file has been provided - things that we did not cover in class, but are nice to have in a compose file.

DB

You'll need to define:

- image
- the ports that the DB is to run on
- environment variables:
    - POSTGRES_USER
    - POSTGRES_PASSWORD
    - POSTGRES_DB

- a volume attached to /var/lib/postgresql/data

APP

You'll need to define:

- environment variables:
  - DB_HOST
  - DB_PORT
  - DB_USER
  - DB_PASSWORD
  - DB_NAME
  - this container's port
- port (again 😦)

Volume

You'll need to define:

- the volume that you attached to DB

Adding Your Name

Within `main.go`, I want you to add a single logging statement:

At line 46, please add a log printing your name. You can find the formatting for logging in Go on line 45.

Deploying and Testing

The final stretch!

Since Go is a compiled language, you'd normally need to use `docker build .` to load it into an image, and compile it. By all means, go for it!

However, within our **compose.yaml** file, under **app**, we have a line that says `build: .`. This tells Docker Compose to find the Dockerfile in the current directory, and use it to create **app**.

All you should have to do is run `docker compose up`, and watch it boot up!

If you feel like testing my "definitely human created code hahaha", you can do it with the steps below!

- Adding users can be accomplished through a web browser or curl.
  - either `curl localhost:8080/add?name={enterNameHere}`, or navigate to `localhost:8080/add?name={enterNameHere}` in your web browser.
- Checking the current users can be accomplished by hitting:
  - `curl localhost:8080/users`
  - navigating to `localhost:8080/users` in your web browser.

Once you're finished testing, I recommend doing `docker compose down` to remove the application containers.

What to Turn In?

1. A link to your repository. a. Note: THIS REPOSITORY SHOULD BE PUBLIC. IF I HAVE TO EMAIL YOU TO MAKE IT PUBLIC I WILL HUNT YOU DOWN AND COUNT POINTS OFF.
2. A screenshot of your console, showing the log message containing your name.