Making Your Golang API with Docker

This will be a quick tutorial to host your Golang API using Docker and DigitalOcean (or any Ubuntu server that you can SSH into.)

Be mindful, you will need to have your code hosted on a private Git repo to have better access for this project, as well as Docker downloaded to your computer (*I recommend to download Docker Desktop to better visualize running containers as well as images. You will also learn how to view these options from the command line locally*).

Dockerfile

Once your code is ready to be put into a `Container`, you must create a Dockerfile to hold the build process for the container.

This is how your directory should look after creating the Dockerfile

```
- your-api-dir
  - Dockerfile
  - api-files
```

```
FROM golang:1.12-alpine3.9        # Choose appropriate Golang version.
RUN mkdir /APP_NAME
ADD . /APP_NAME
WORKDIR /APP_NAME
RUN apk add git
RUN go get YOUR_GITHUB_LINK
ENV PORT ####                     # Replace the hashs with desired port
RUN go build -o main .            # Build your Go however you like.
CMD ["/APP_NAME/main"]
```

This sample Dockerfile first creates a directory app, then it does `go get` to your private repo.
Here you may be prompted to allow login with Git if this repo is private.

Setting Up Digital Ocean

To setup Digital Ocean, starting by making an account and entering a PayPal deposit of $5 minimum, or adding a credit card. Digital Ocean does this to prevent fraud. (*Not my policy so I dont care but I also think it is dumb. They probably dont want to pay for the extra insurance.*)

Once you have got your account setup, you can start by heading over to the main control panel and clicking `Droplets` on the left. Next, on the top right, you will see `Create Droplet`. Select the Marketplace image `Docker xx.xx.xx on Ubuntu xx.xx` and setup your droplet. A password should be simple enough for a low usage and exposure API, but you might want to upgrade to SSH keys for larger and more secure projects (*Also should consider a load balancer in these situations.*).

Now, you should have your Droplet's IPV4 address.

You will need to SSH into the server; however you choose to do it is up to you, but I prefer to use MobaXTerm for the universal master key, saved sessions, and overall pleasing UI, but again, this is just a preference.

Once you have SSH'ed into the server, you will want to clone your repository **(with your Dockerfile)** into a directory on the server, and you can use this command to do so.

```
git clone https://github.com/YOUR_USER/YOUR_REPO.git DIRECTORY_NAME
```

Then we can `cd` into our directory.

```
cd ./DIRECTORY_NAME
```

Once in, we can use `docker build -t CUSTOM_NAME .` at the command line to build our image.

Then we can use `docker run -d -p PORT_NUMBER:PORT_NUMBER CUSTOM_NAME` to begin running our container.

Note: If you are testing a docker build on local before you turn to production, use `docker run -it -p PORT_NUMBER:PORT_NUMBER CUSTOM_NAME`.

Finale

Well, now your Golang API should be live on your droplet via Docker and containerization of your API.
You can access your API through the `http://DROPLET_IP:ENV_PORT` link on the internet.

You can also keep track of your dockers with `docker ps` and you can keep track of the images made with `docker images`.