My-first-docker-image

$HOME/Library/Containers/com.docker.docker/Data/com.docker.driver.amd64-linux/Docker.qcow2

Commander:

Docker run -it //交互
example：docker run -it alpine /bin/sh
Docker ps -a. // 查看所有的历史
Docker stop image_id(container id)
Docker rm image_id(container id)

View the docker port:

Docker port container_id.

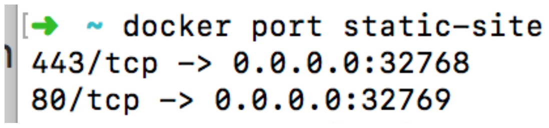
Ex: docker run —name static-site -e AUTHOR="elliot" -d - P sequence/static-site

In the above command:

- -d will create a container with the process detached from our terminal
- -P will publish all the exposed container ports to random ports on the Docker host
- -e is how you pass environment variables to the container
- --name allows you to specify a container name
- AUTHOR is the environment variable name and Your Name is the value that you can pass

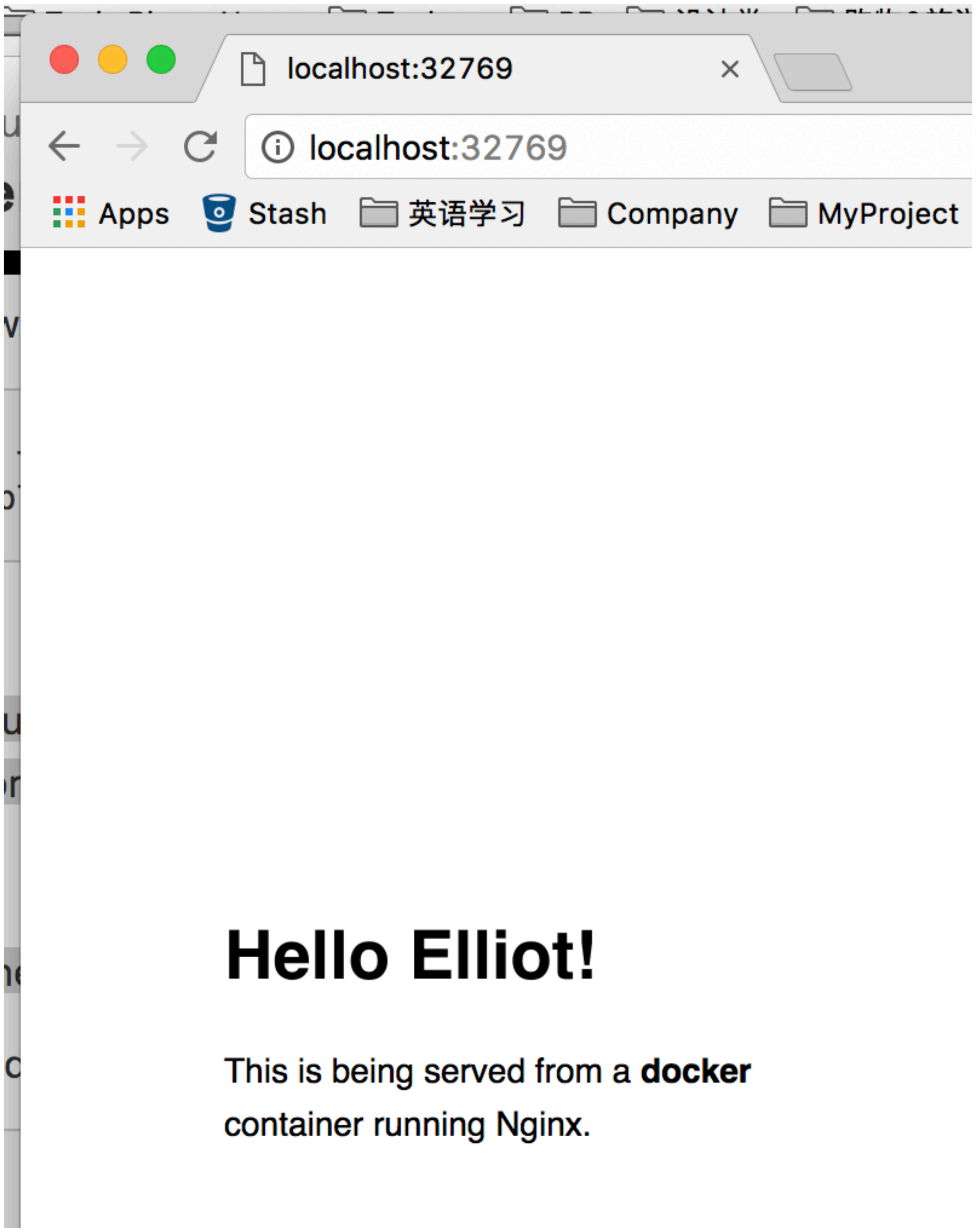
Now you run:
Docker port static-site

docker 内部的80/tcp expose 到外面的端口 32769



```
➜  ~ docker port static-site
443/tcp -> 0.0.0.0:32768
80/tcp -> 0.0.0.0:32769
```

so you can open below on chrome browser to do the testing.



Hello Elliot!

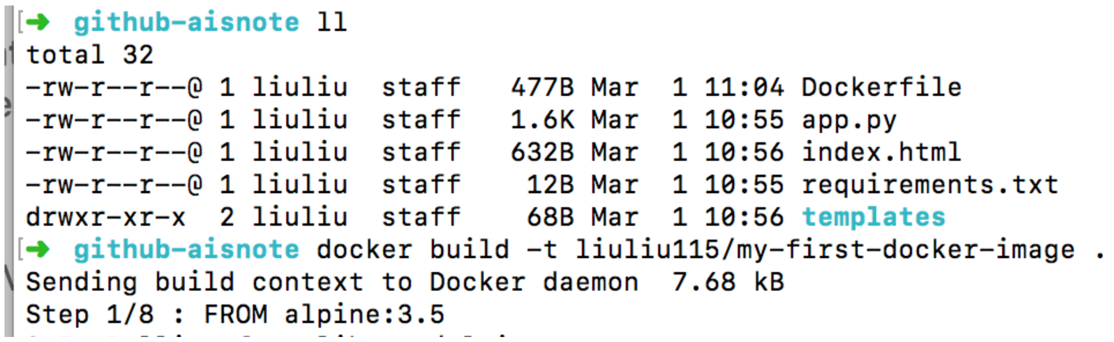
This is being served from a **docker** container running Nginx.

Docker search **.   Can search the images from hub.docker.io**

Create your own docker image:

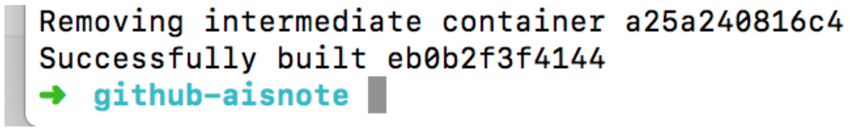
A Dockerfile is a text file that contains a list of commands that the Docker daemon calls while creating an image. The Dockerfile contains all the information that Docker needs to know to run the app — a base Docker image to run from, location of your project code, any dependencies it has, and what commands to run at start-up. It is simple way to automate the image creation process. The best part is that the commands you write in a Dockerfile are *almost* identical to their equivalent Linux commands. This means you don't really have to learn new syntax to create your own Dockerfiles.

Need the docker hub account, and do as below:



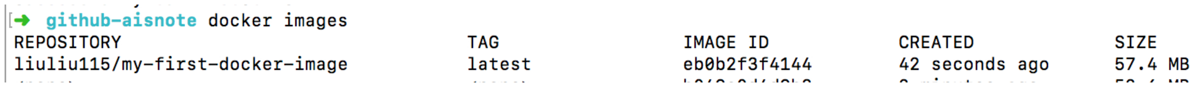
```
➜  github-aisnote ll
total 32
-rw-r--r--@ 1 liuliu  staff   477B Mar  1 11:04 Dockerfile
-rw-r--r--@ 1 liuliu  staff   1.6K Mar  1 10:55 app.py
-rw-r--r--@ 1 liuliu  staff   632B Mar  1 10:56 index.html
-rw-r--r--@ 1 liuliu  staff    12B Mar  1 10:55 requirements.txt
drwxr-xr-x  2 liuliu  staff    68B Mar  1 10:56 templates
➜  github-aisnote docker build -t liuliu115/my-first-docker-image .
Sending build context to Docker daemon  7.68 kB
Step 1/8 : FROM alpine:3.5
```

If success, it will show as below:



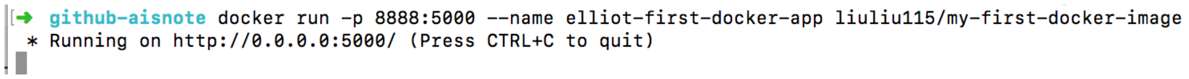
```
Removing intermediate container a25a240816c4
Successfully built eb0b2f3f4144
➜  github-aisnote
```

And you can run docker images to see:



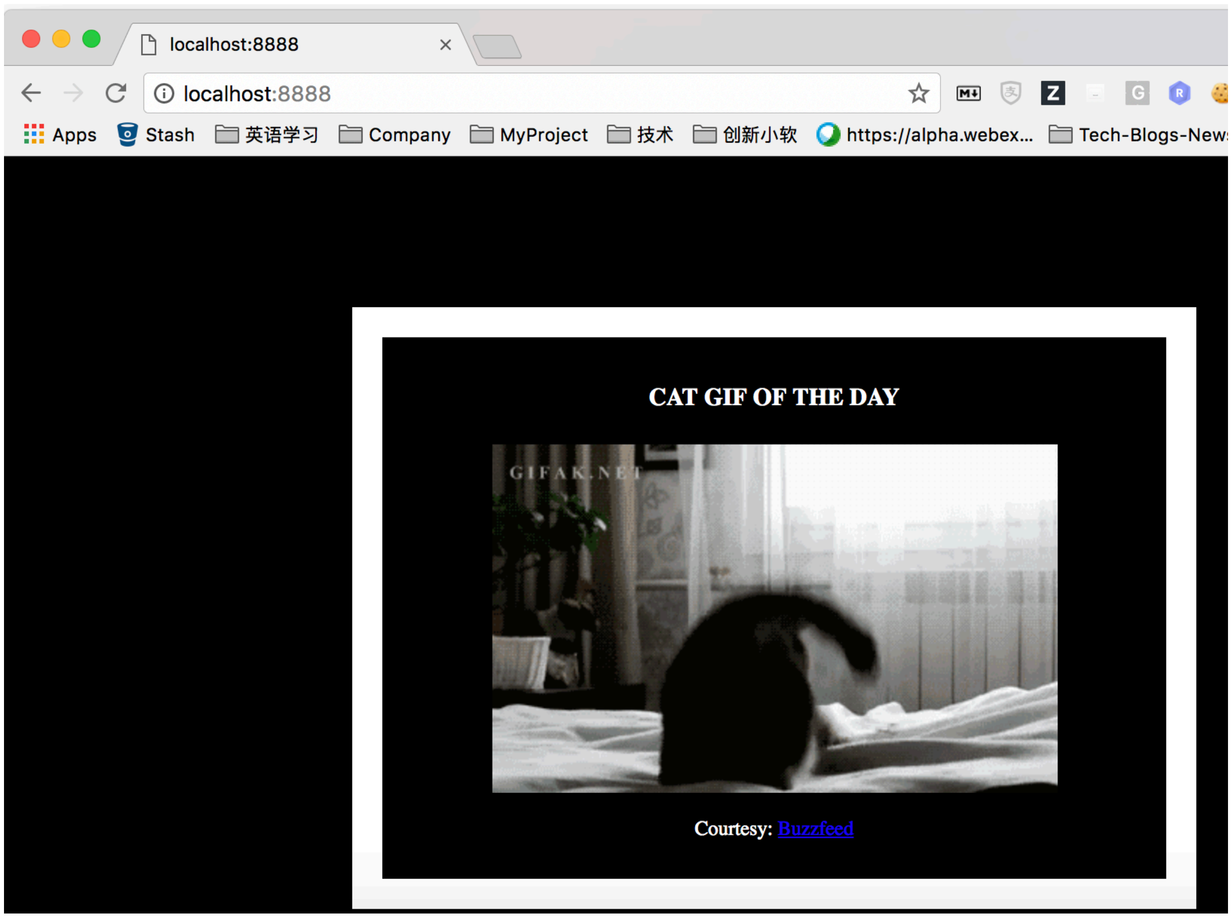
```
➜  github-aisnote docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
liuliu115/my-first-docker-image     latest              eb0b2f3f4144        42 seconds ago      57.4 MB
```

Now it's time to run your-own image:



```
➜  github-aisnote docker run -p 8888:5000 --name elliot-first-docker-app liuliu115/my-first-docker-image
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

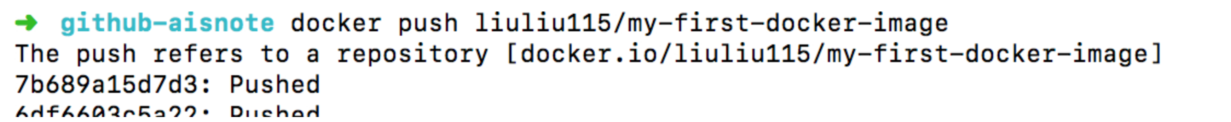
You will see below on browser:



CAT GIF OF THE DAY

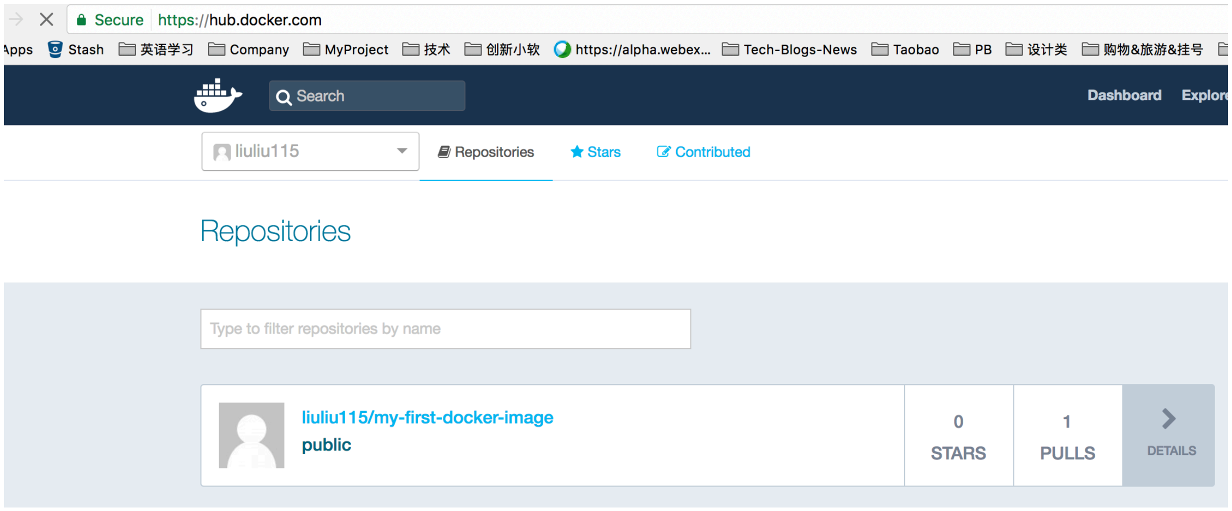
Courtesy: Buzzfeed

Push your docker to hub.docker



```
➜  github-aisnote docker push liuliu115/my-first-docker-image
The push refers to a repository [docker.io/liuliu115/my-first-docker-image]
7b689a15d7d3: Pushed
6df6603c5a22: Pushed
```

Go to hub.docker.com to check your image:



liuliu115/my-first-docker-image
public

0 STARS
1 PULLS

Now you can stop and remove your images on local:
Docker rm -f $(docker ps -a -q)

Summary for Dockerfile:

Here's a quick summary of the few basic commands we used in our Dockerfile.

- FROM starts the Dockerfile. It is a requirement that the Dockerfile must start with the FROM command. Images are created in layers, which means you can use another image as the base image for your own. The FROM command defines your base layer. As arguments, it takes the name of the image. Optionally, you can add the Docker Hub username of the maintainer and image version, in the format username/imagename:version.

- RUN is used to build up the Image you're creating. For each RUN command, Docker will run the command then create a new layer of the image. This way you can roll back your image to previous states easily. The syntax for a RUN instruction is to place the full text of the shell command after the RUN (e.g., RUN mkdir /user/local/foo). This will automatically run in a /bin/sh shell. You can define a different shell like this: RUN /bin/bash -c 'mkdir /user/local/foo'

- COPY copies local files into the container.

- CMD defines the commands that will run on the Image at start-up. Unlike a RUN, this does not create a new layer for the Image, but simply runs the command. There can only be one CMD per a Dockerfile/Image. If you need to run multiple commands, the best way to do that is to have the CMD run a script. CMD requires that you tell it where to run the command, unlike RUN. So example CMD commands would be:

```
CMD ["python", "./app.py"]

CMD ["/bin/bash", "echo", "Hello World"]
```

- EXPOSE opens ports in your image to allow communication to the outside world when it runs in a container.

- PUSH pushes your image to Docker Hub, or alternately to a private registry

More link: https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/