Dockerfile

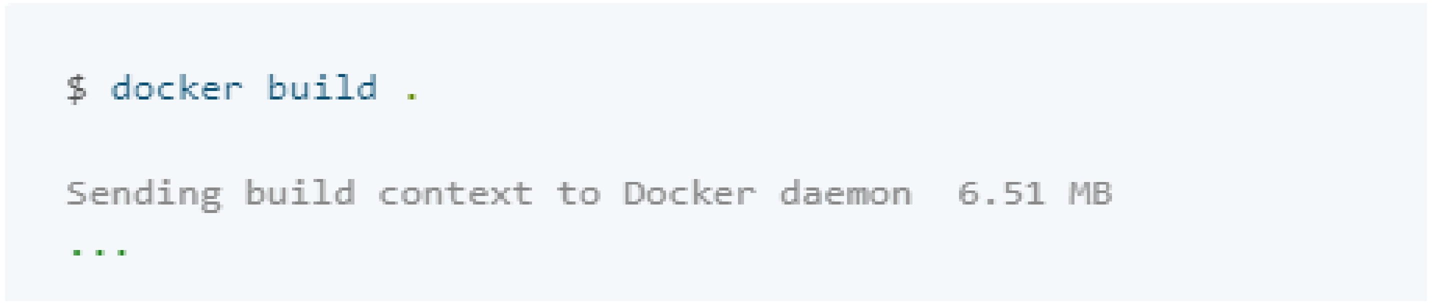
OW

Dockerfile

- Docker can build images automatically by reading the instructions from a Dockerfile.
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- Using docker build users can create an automated build that executes several command-line instructions in succession.

Docker build

- The docker build command builds an image from a Dockerfile and a context. The build's context is the set of files at a specified location PATH or URL. The PATH is a directory on your local filesystem. The URL is a Git repository location.

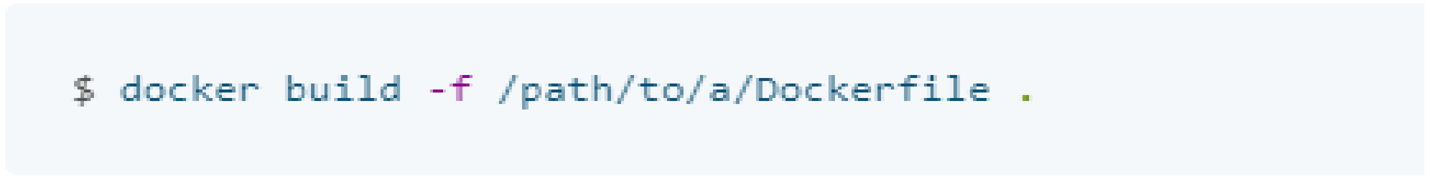


```
$ docker build .

Sending build context to Docker daemon  6.51 MB
...
```

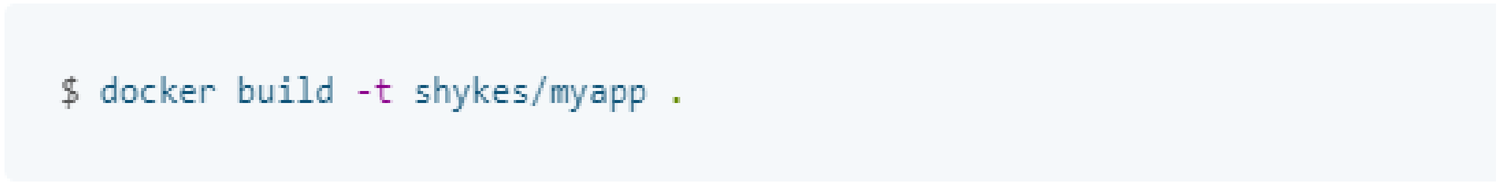
..

- use the -f flag with docker build to point to a Dockerfile anywhere in your file system.



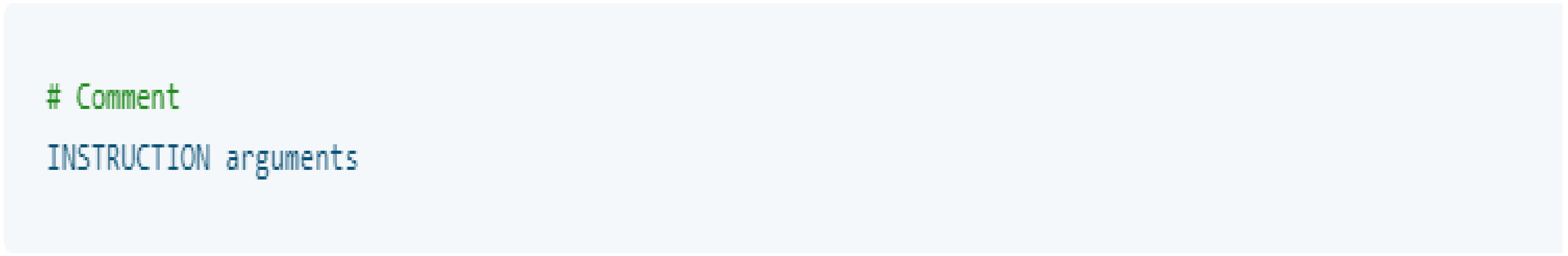
```
$ docker build -f /path/to/a/Dockerfile .
```

- specify a repository and tag at which to save the new image if the build succeeds:



```
$ docker build -t shykes/myapp .
```

Dockerfile format



```
# Comment
INSTRUCTION arguments
```

- The instruction is not case-sensitive. However, convention is for them to be UPPERCASE to distinguish them from arguments more easily.
- A dockerfile must begin with a FROM instruction. The FROM instructions specifies the Parent image from which we are building
- Docker treats lines begins with # as a comment

RUN

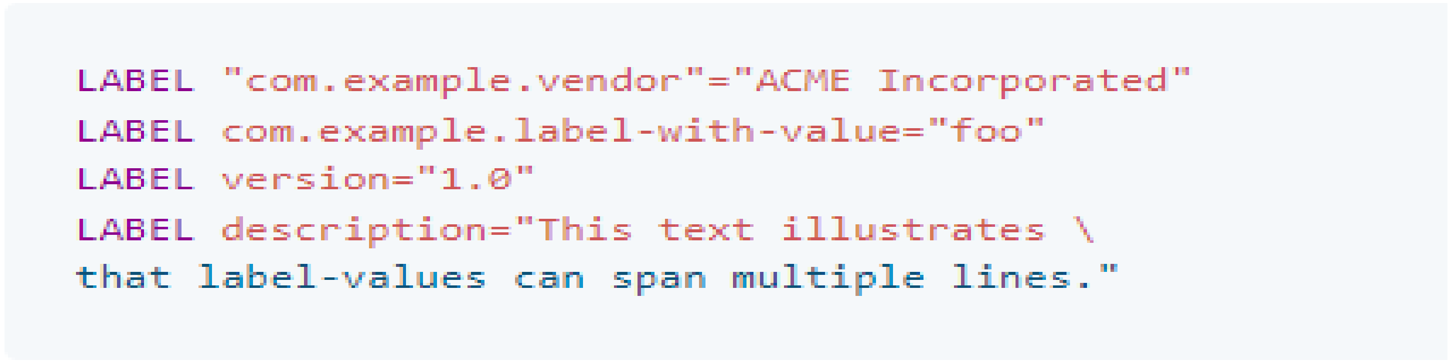
- RUN has 2 forms:
  - RUN <command> (shell form, the command is run in a shell, which by default is /bin/sh -c on Linux or cmd /S /C on Windows)
  - RUN ["executable", "param1", "param2"] (exec form)
- The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile.
- The exec form makes it possible to avoid shell string munging, and to RUN commands using a base image that does not contain the specified shell executable.

CMD

- The CMD instruction has three forms:
  - CMD ["executable","param1","param2"] (exec form, this is the preferred form)
  - CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
  - CMD command param1 param2 (shell form)
- There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.

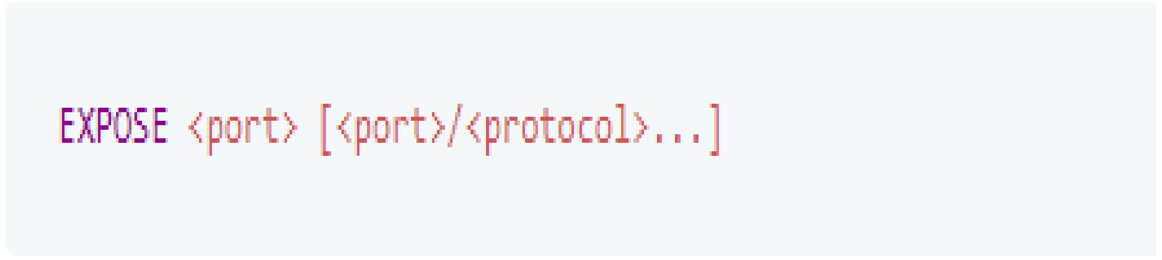
LABEL

- LABEL <key>=<value> <key>=<value> <key>=<value> ...
- The LABEL instruction adds metadata to an image. A LABEL is a key-value pair. To include spaces within a LABEL value, use quotes and backslashes as you would in command-line parsing.



```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

EXPOSE

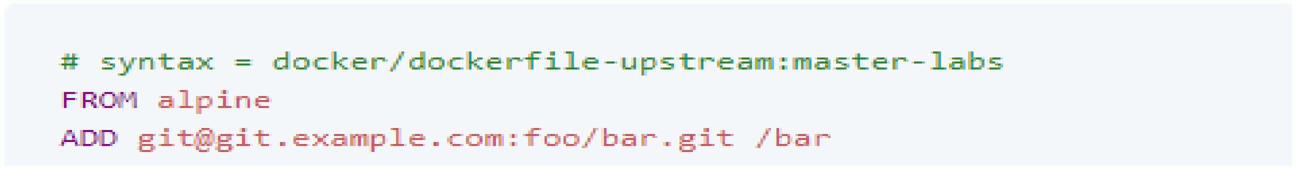


```
EXPOSE <port> [<port>/<protocol>...]
```

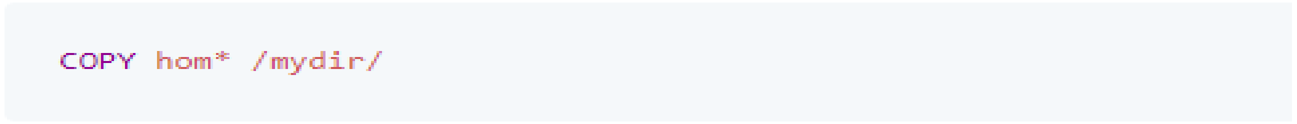
The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime. You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not specified.

ADD and COPY

- The ADD instruction copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.
- The COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.



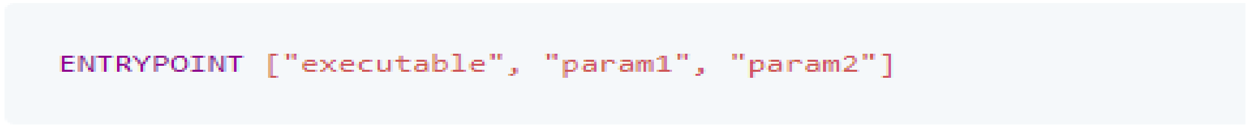
```
# syntax = docker/dockerfile-upstream:master-labs
FROM alpine
ADD git@git.example.com:foo/bar.git /bar
```



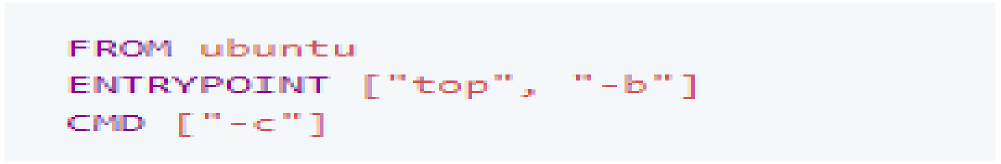
```
COPY hom* /mydir/
```

ENTRYPOINT

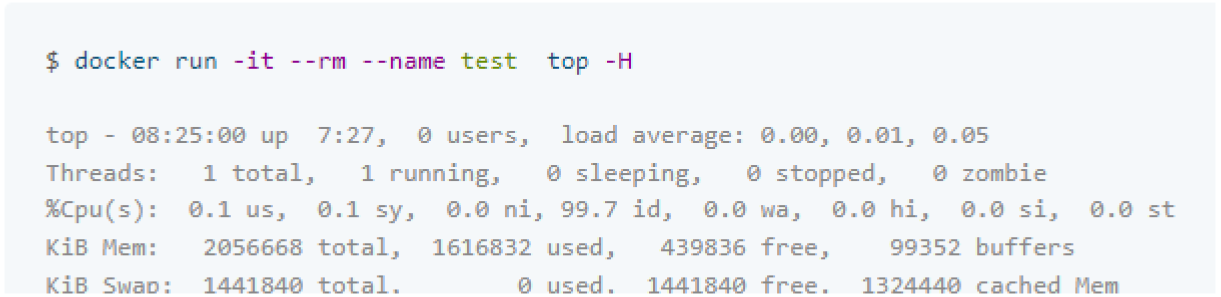
- An ENTRYPOINT allows you to configure a container that will run as an executable.



```
ENTRYPOINT ["executable", "param1", "param2"]
```



```
FROM ubuntu
ENTRYPOINT ["top", "-b"]
CMD ["-c"]
```



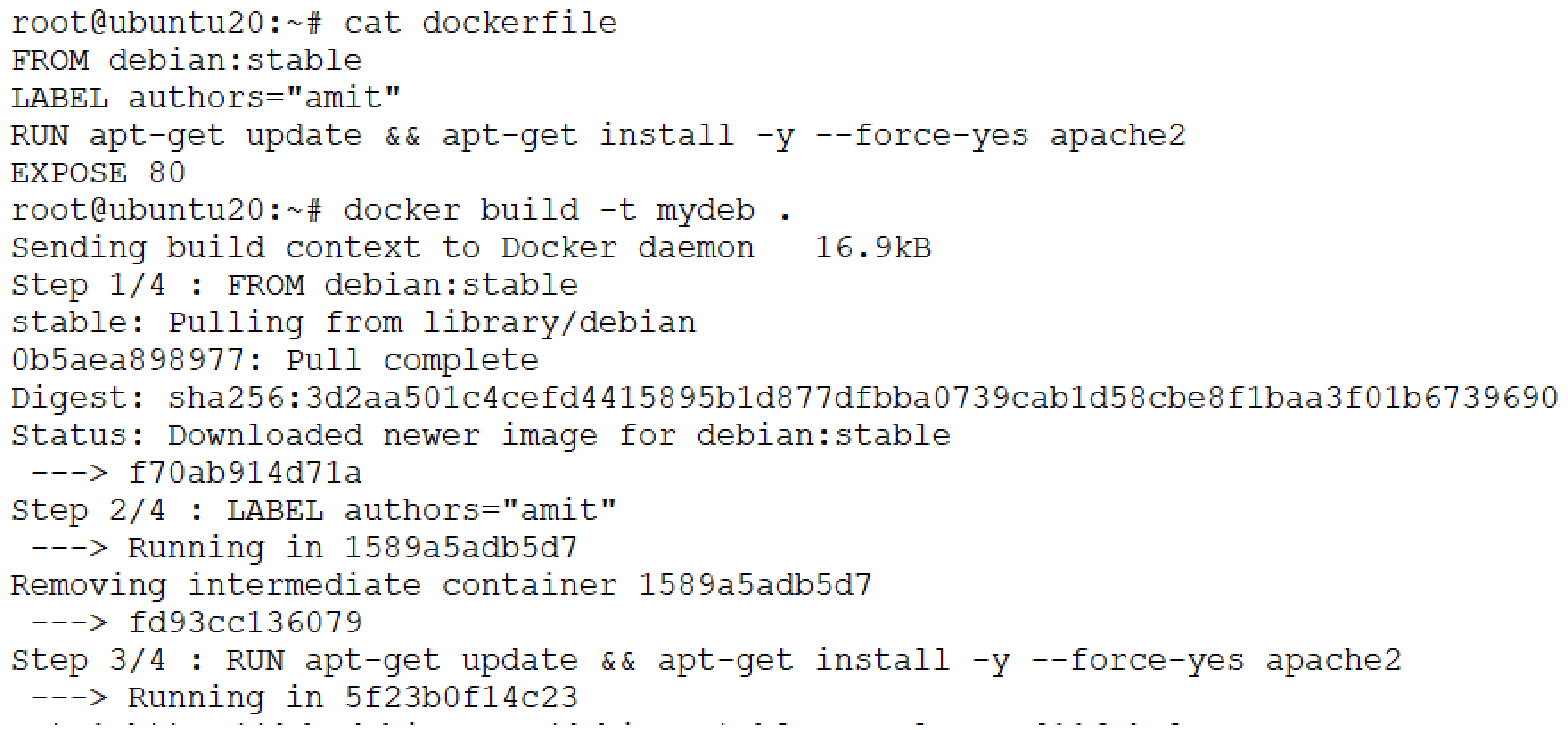
```
$ docker run -it --rm --name test  top -H

top - 08:25:00 up  7:27,  0 users,  load average: 0.00, 0.01, 0.05
Threads:   1 total,   1 running,   0 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.1 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2056668 total,  1616832 used,   439836 free,    99352 buffers
KiB Swap:  1441840 total,        0 used,  1441840 free.  1324440 cached Mem
```

Understand how CMD and ENTRYPOINT interact

- Both CMD and ENTRYPOINT instructions define what command gets executed when running a container. There are few rules that describe their co-operation.
  - Dockerfile should specify at least one of CMD or ENTRYPOINT commands.
  - ENTRYPOINT should be defined when using the container as an executable.
  - CMD should be used as a way of defining default arguments for an ENTRYPOINT command or for executing an ad-hoc command in a container.

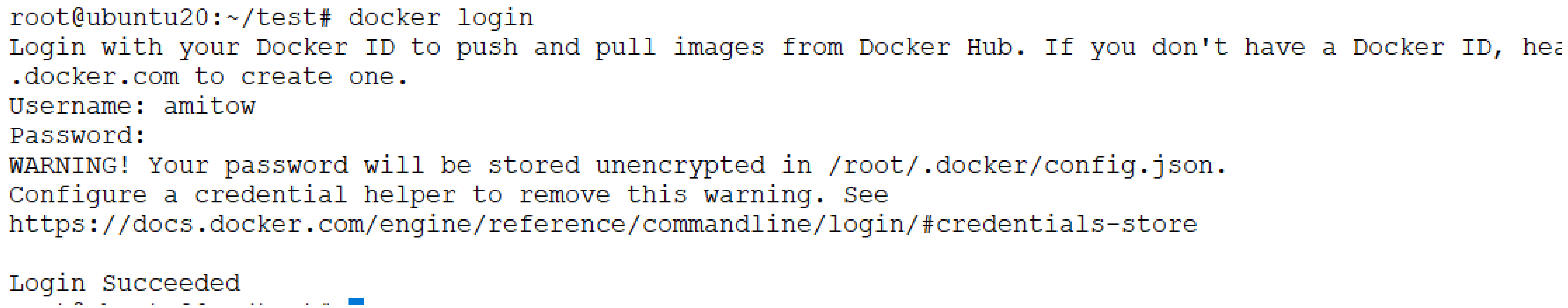
Examples



```
root@ubuntu20:~# cat dockerfile
FROM debian:stable
LABEL authors="amit"
RUN apt-get update && apt-get install -y --force-yes apache2
EXPOSE 80
root@ubuntu20:~# docker build -t mydeb .
Sending build context to Docker daemon   16.9kB
Step 1/4 : FROM debian:stable
stable: Pulling from library/debian
0b5aea898977: Pull complete
Digest: sha256:3d2aa501c4cefd4415895b1d877dfbba0739cab1d58cbe8f1baa3f01b6739690
Status: Downloaded newer image for debian:stable
 ---> f70ab914d71a
Step 2/4 : LABEL authors="amit"
 ---> Running in 1589a5adb5d7
Removing intermediate container 1589a5adb5d7
 ---> fd93cc136079
Step 3/4 : RUN apt-get update && apt-get install -y --force-yes apache2
 ---> Running in 5f23b0f14c23
```

Docker hubs

- Signup to docker hubs
- docker login



```
root@ubuntu20:~/test# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, hea
.docker.com to create one.
Username: amitow
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

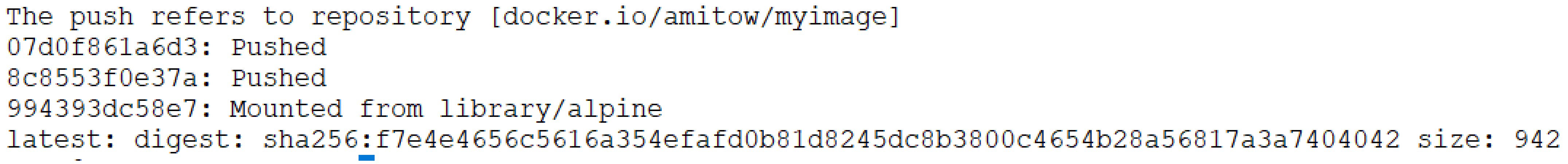
Push image to docker hub

- Tag the image with the host name or IP address, and the port of the registry

  docker image tag myimage amitow/myimage

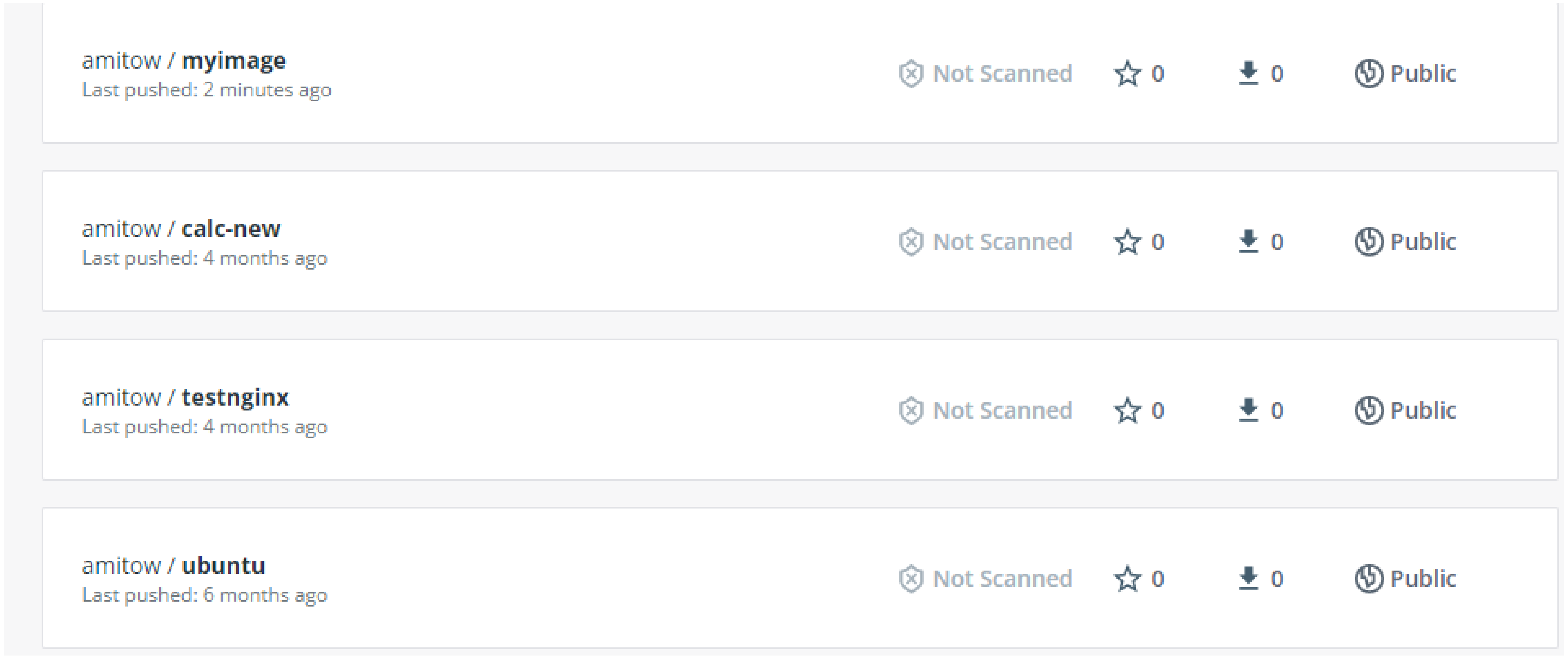
- Push the image

  docker image push amitow/myimage



```
The push refers to repository [docker.io/amitow/myimage]
07d0f861a6d3: Pushed
8c8553f0e37a: Pushed
994393dc58e7: Mounted from library/alpine
latest: digest: sha256:f7e4e4656c5616a354efafd0b81d8245dc8b3800c4654b28a56817a3a7404042 size: 942
```

• •


amitow / **myimage**
Last pushed: 2 minutes ago
Not Scanned ☆ 0 0 Public

amitow / **calc-new**
Last pushed: 4 months ago
Not Scanned ☆ 0 0 Public

amitow / **testnginx**
Last pushed: 4 months ago
Not Scanned ☆ 0 0 Public

amitow / **ubuntu**
Last pushed: 6 months ago
Not Scanned ☆ 0 0 Public