# Lab-12-1: Docker files

# Objectives

* Create Docker containers using Docker files
* Deploy Docker images
* Automate docker build
* Deploy bind volmes

--------------------------------------------------------------------------------------------------

**Today:**

Login to your ubuntu installation (either locally, through Azure/AWS instance)

**Task1: Create and Run Container**

1. Setup github repo
2. Create dockerfile for build context
3. Build image from dockerfile

Setup github and build context

---------------------------------------

* create a github repo named docker-lab, tick box for README
  + generate and register an ssh key and register with your account

[hints: https://help.github.com/articles/generating-ssh-keys]

* clone your repo to your Linux VM
  + sudo apt-get install git
  + git clone [git@github.come:<your-username./docker-lab.git](mailto:git@github.come:%3cyour-username./docker-lab.git)
* the above steps will create a directory which will be referred as your build context and you will put resources to build container here

Prepare Build Context

* Inside the build context create a file name ‘Dockerfile’ and a sample index.html file
* Add the following on the Docker file

FROM ubuntu:16.04

LABEL maintainer=”<yourname/email address>”

RUN apt-get -q update && apt-get -yq dist-upgrade

RUN apt-get -yq install apache2

ENV APACHE\_RUN\_USER www-data

ENV APACHE\_RUN\_GROUP www-data

ENV APACHE\_LOG\_DIR /var/log/apache2

ENV APACHE\_LOCK\_DIR /var/run/apache

ENV APACHE\_PID\_FILE /var/run/apache/httpd.pid

RUN mkdir /var/run/apache

ADD index.html /var/www/html/index.html

EXPOSE 80

ENTRYPOINT ["/usr/sbin/apache2"]

CMD ["-DFOREGROUND"]

Q1. What does the RUN command do?

ANS: 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

Q2. How can you add a directory instead of a file?

Ans: RUN mkdir /root/file/path/filename

Q3. What does the ENTRYPOINT and CMD commands do?

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

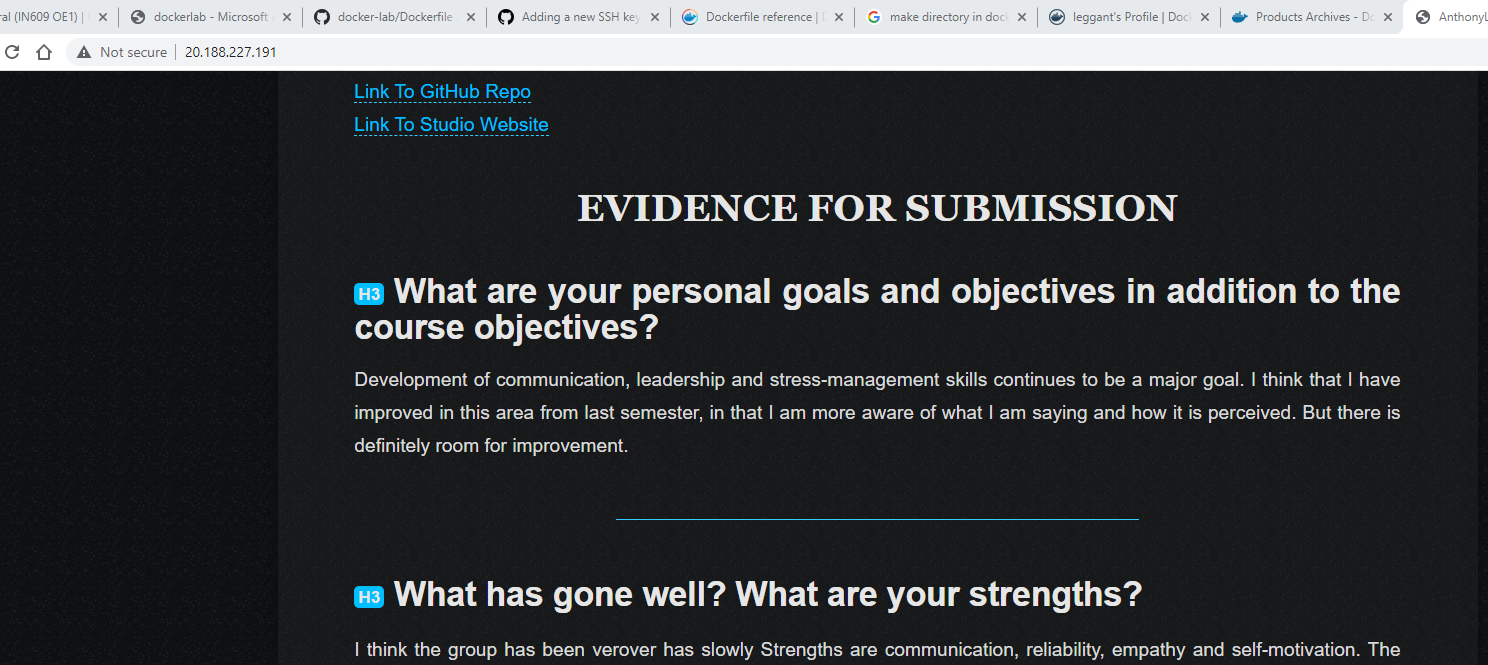
[hint: <https://docs.docker.com/engine/reference/builder/>.]

Build your image

* From your build context directory type the following
  + Docker build -t your-username/dockerlab .
* Run the container
* Use your browser to see the index.html file from the container.

Q4. Paste a screenshot of your browser for the above step

Ans:



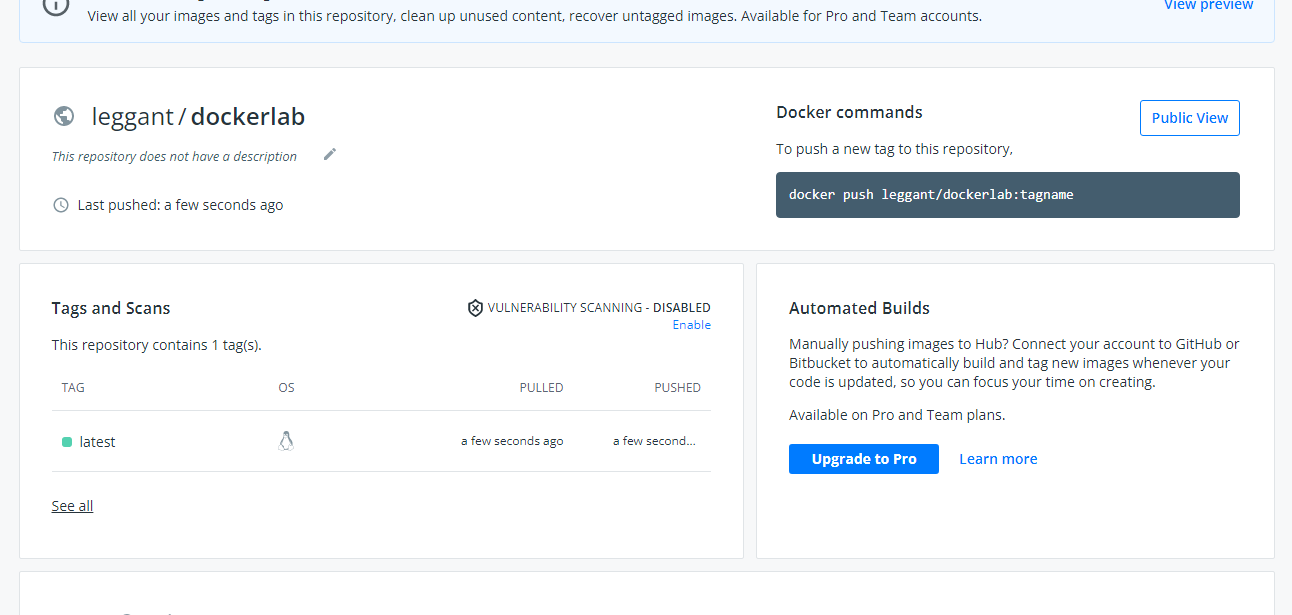
Automating Build:

We would like to push the images to dockehub, and automate the build everytime we make changes to our Dockerfile on our github repo

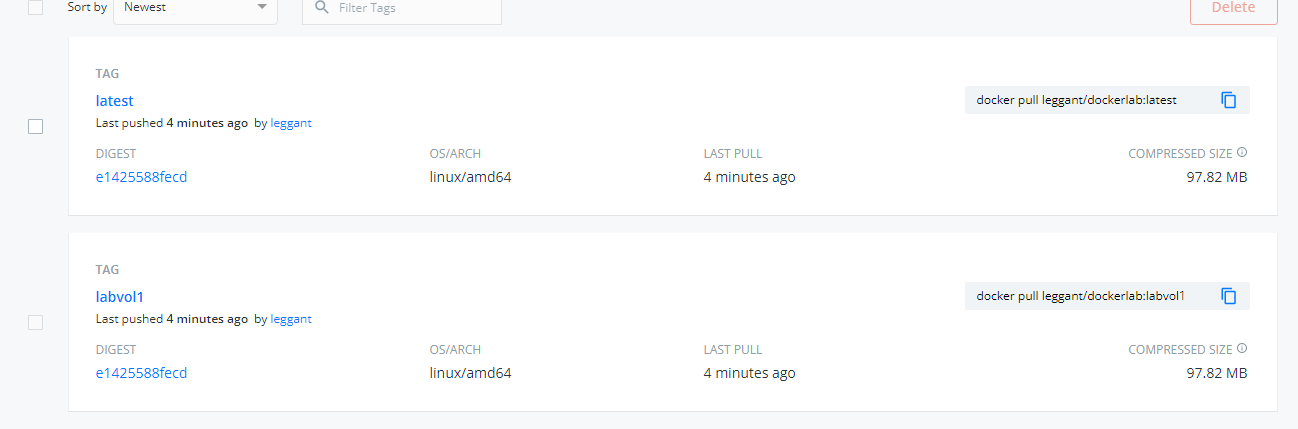
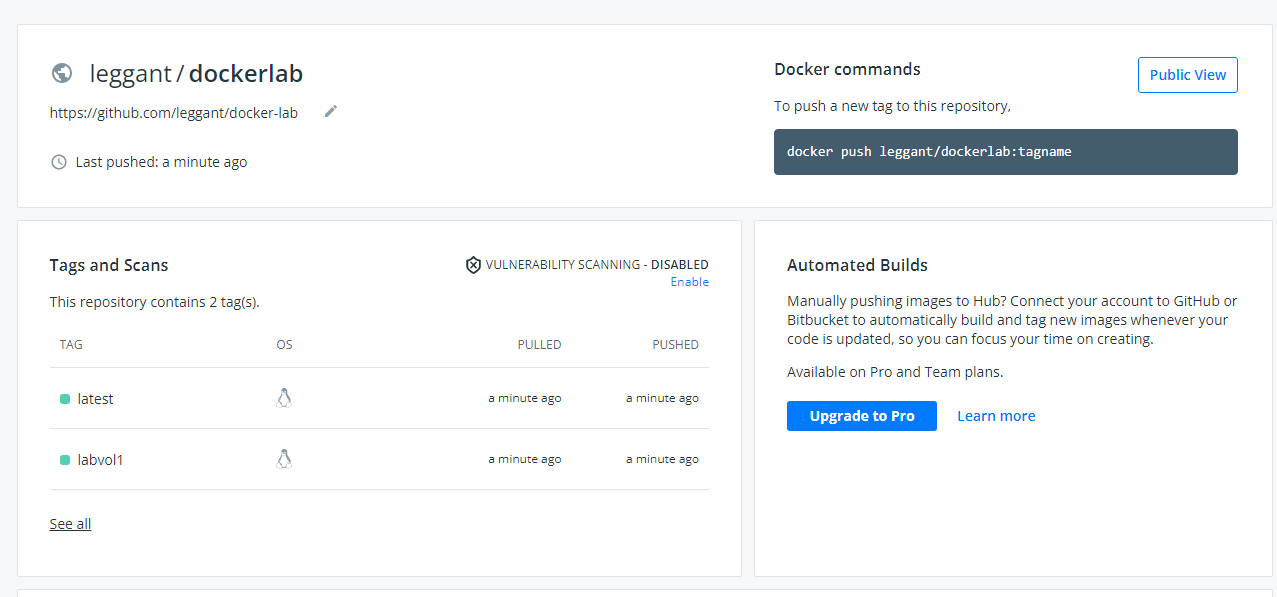
* Create an account on dockerhub
* Push your image to dockerhub account
  + Docker push yourusername/dockerlab

Q5. Print a screenshot of your docker repo verifying that your docker image has been pushed

Ans:



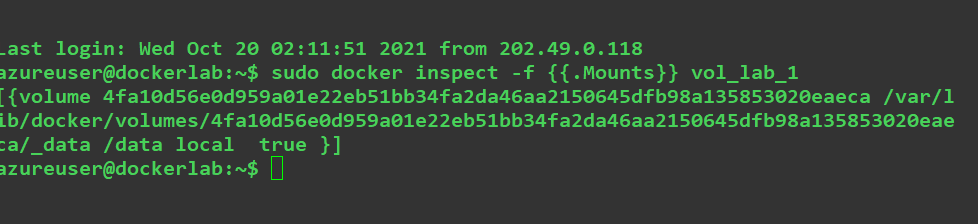
* On your DockerHub repo, select the “build” item and link it to your Github Repo.
* Make some changes on your Docker file and push it to gitrepo, verify that
  + Create a directory, /data inside your container.
  + In /data, create files named foo, bar, and baz.
  + Make /data a volume. [Hint: VOLUME /data]
* Push your Dockerfile and build an image and launch a container called vol\_lab\_1 from it.



Using the volume

Open another terminal session with this VM.

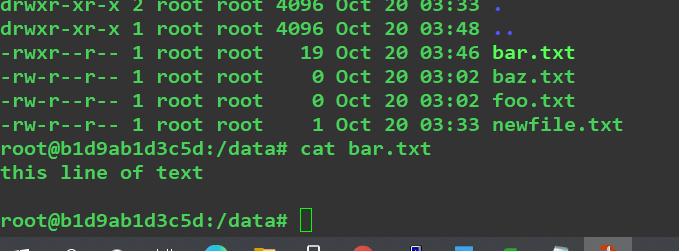
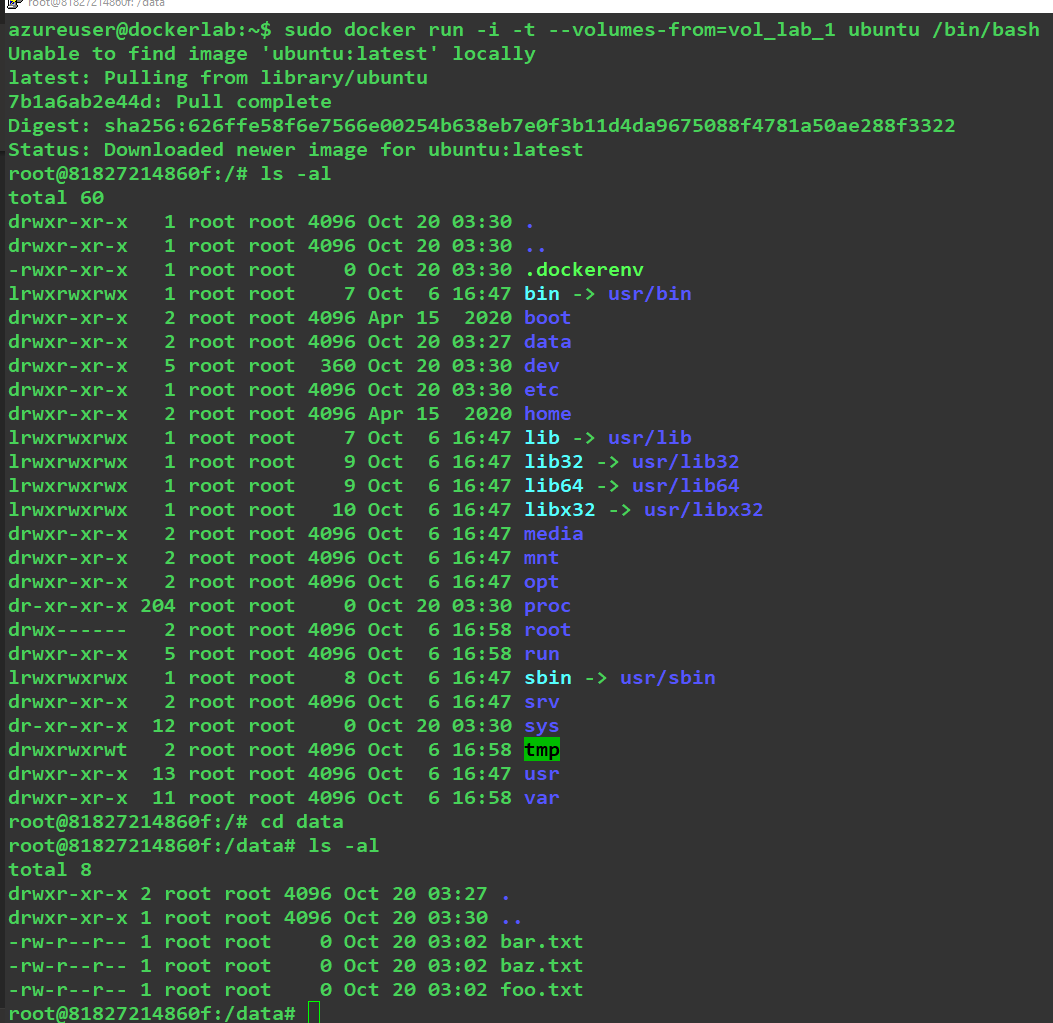
* Inspect your new container to find the volume
  + sudo docker inspect -f {{.Mounts}} vol\_lab\_1



* Shut down vol\_lab\_1. Start a new container with the command

sudo docker run -it --volumes-from=vol\_lab\_1 ubuntu /bin/bash

When your new container starts, inspect the /data volume inside it. Make some changes inside the volume and then shut down this container. Then, check the volume directory on the host system and see if your changes are visible there



Q6. What does the volumes-from option do?

This will run a ubuntu container, with the volume used in the vol\_lab\_1 container.

Bind Volumes

In this case, we will use an existing folder on the host to be available on containers so that files on the host’s directory becomes available for the containers.

- Create a directory, /home/user/mydata on your host system. Place some files in it.

- In your Dockerfile, remove the earlier commands that created the files in /data, but do create the/data directory.

- Rebuild your image and start a new container from it. Create a bind mount of /home/user/mydata onto /data [hint: sudo docker run -it --name vol-test3 n

--mount type=bind,source=/home/student/data,target=/data ubuntu /bin/bash

- On your host system, modify the contents of /home/user/mydata and then observe how those changes are visible inside your running container.

