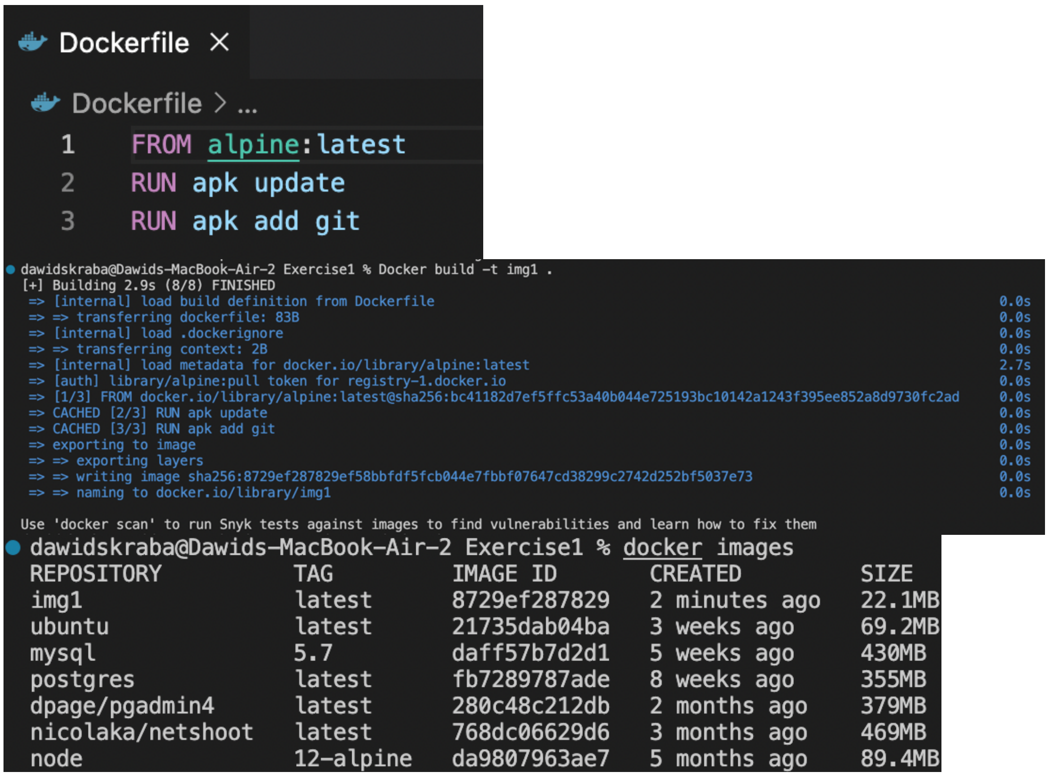
Cloud Computing Practical 1 Exercise 1

Dawid Skraba 19433692

Exercise 1.

Task 1:



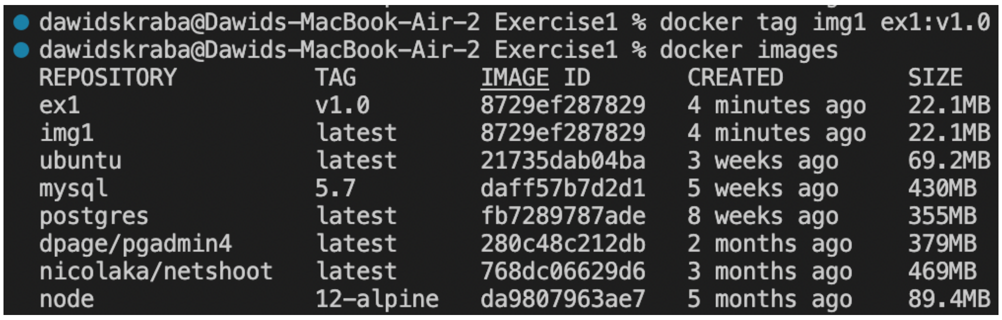
```
FROM alpine:latest
RUN apk update
RUN apk add git
```

```
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % Docker build -t img1 .
[+] Building 2.9s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 83B                                                                                           0.0s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                                              2.7s
 => [auth] library/alpine:pull token for registry-1.docker.io                                                                 0.0s
 => [1/3] FROM docker.io/library/alpine:latest@sha256:bc41182d7ef5ffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad        0.0s
 => CACHED [2/3] RUN apk update                                                                                               0.0s
 => CACHED [3/3] RUN apk add git                                                                                              0.0s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:8729ef287829ef58bbfdf5fcb044e7fbbf07647cd38299c2742d252bf5037e73                                  0.0s
 => => naming to docker.io/library/img1                                                                                       0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker images
REPOSITORY          TAG         IMAGE ID       CREATED          SIZE
img1                latest      8729ef287829   2 minutes ago    22.1MB
ubuntu              latest      21735dab04ba   3 weeks ago      69.2MB
mysql               5.7         daff57b7d2d1   5 weeks ago      430MB
postgres            latest      fb7289787ade   8 weeks ago      355MB
dpage/pgadmin4      latest      280c48c212db   2 months ago     379MB
nicolaka/netshoot   latest      768dc06629d6   3 months ago     469MB
node                12-alpine   da9807963ae7   5 months ago     89.4MB
```

Above I show that I created an image (using alpine as its base) with git installed. I Then built this image using the docker build with the tag "img1". Using the docker images command its confirmed the image has been created.

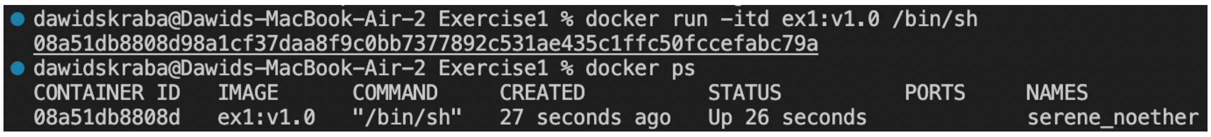
Task2:



```
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker tag img1 ex1:v1.0
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker images
REPOSITORY          TAG         IMAGE ID       CREATED          SIZE
ex1                 v1.0        8729ef287829   4 minutes ago    22.1MB
img1                latest      8729ef287829   4 minutes ago    22.1MB
ubuntu              latest      21735dab04ba   3 weeks ago      69.2MB
mysql               5.7         daff57b7d2d1   5 weeks ago      430MB
postgres            latest      fb7289787ade   8 weeks ago      355MB
dpage/pgadmin4      latest      280c48c212db   2 months ago     379MB
nicolaka/netshoot   latest      768dc06629d6   3 months ago     469MB
node                12-alpine   da9807963ae7   5 months ago     89.4MB
```

Here I had to tag the image using the tag "ex1:v1.0", and I did this using the docker tag command. This has created another image that points to the previous image we created, but now it has a different name/tag. This is helpful when we have multiple versions and we want to know exactly which version we are pulling. It helps maintain a build version much like GIT.

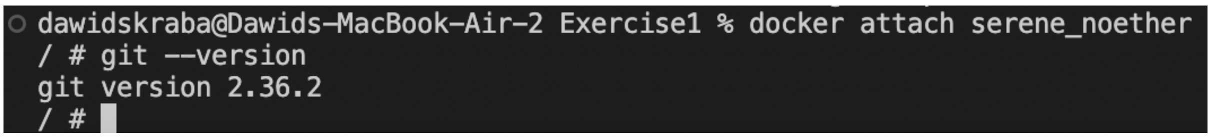
Task3:



```
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker run -itd ex1:v1.0 /bin/sh
08a51db8808d98a1cf37daa8f9c0bb7377892c531ae435c1ffc50fccefabc79a
dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker ps
CONTAINER ID   IMAGE      COMMAND     CREATED          STATUS          PORTS     NAMES
08a51db8808d   ex1:v1.0   "/bin/sh"   27 seconds ago   Up 26 seconds             serene_noether
```

Here I created a container based on the image “ex1:v1.0” and used the flag -itd to make it run in the background I also added that it will use the bash shell(for the next exercise).

Task4:



```
○ dawidskraba@Dawids-MacBook-Air-2 Exercise1 % docker attach serene_noether
  / # git --version
  git version 2.36.2
  / # 
```

Here I entered the container and it’s using bash. I entered the command to get the git version to confirm git is installed in the container.
The attach command attaches my terminal’s stdin, stdout and error to the specified container. This can be used to control the container or to see what’s going on in the container. For example if we attach to a container and we use curl to send some request to the container we will see it. This can be also used to test and debug.