

Cloud Computing

Practical 2

Using docker

Student No: 19396951

Exercise 1.

Task 1: To create an image with GIT installed I created a file called Dockerfile in the file is written some code as you can see in the picture below.

```
C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951> ex1 > Dockerfile
1 FROM alpine:3.5
2 RUN apk update
3 RUN apk add git
```

Task 2: I then put the dockerfile into folder "ex1" and opened up CMD where I then ran the file command - docker image build -t ex1:v1.0 ., as you can see below in the image.

```
C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951\ex1>docker image build -t ex1:v1.0 .
[+] Building 6.3s (8/8) FINISHED
=> [Internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 85B 0.0s
=> [Internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [Internal] load metadata for docker.io/library/alpine:3.5 2.3s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/3] FROM docker.io/library/alpine:3.5@sha256:66952b313e51c3bd1987d7c4ddf5dba9bc0fb6e524eed2448fa660246b3e76 0.5s
=> => resolve docker.io/library/alpine:3.5@sha256:66952b313e51c3bd1987d7c4ddf5dba9bc0fb6e524eed2448fa660246b3e76 0.0s
=> => sha256:66952b313e51c3bd1987d7c4ddf5dba9bc0fb6e524eed2448fa660246b3e76ec 433B / 433B 0.0s
=> => sha256:f7d2b5725685826823bc6b154c0de02832e5e6daf7dc25a00ab00f1158fabfc8 528B / 528B 0.0s
=> => sha256:f80194ae2e0ccf0f998baa6b981396dfbf6b16e6476164af72158577a7de2dd9 1.51kB / 1.51kB 0.0s
=> => sha256:8cae0e1ac61cead281f41115cc0ebd39117f7e54dffc8fd5e05a7590dca3cd4e 1.97MB / 1.97MB 0.3s
=> => extracting sha256:8cae0e1ac61cead281f41115cc0ebd39117f7e54dffc8fd5e05a7590dca3cd4e 0.1s
=> [2/3] RUN apk update 1.0s
=> [3/3] RUN apk add git 2.2s
=> exporting to image 0.2s
=> => exporting layers 0.1s
=> => writing image sha256:2c7c149453c4dca6de7b264249446cabea730bfa655cd81e439a28a46a0c889a 0.0s
=> => naming to docker.io/library/ex1:v1.0 0.0s
```

Task 3: To create a container based on the image, I ran the command - docker container run -itd ex1:v1.0, this created the container called upbeat_wright based off the image. This also allows the container to run in the background, as we can see in the image below.

```
C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951\ex1>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
3ae6764f1a8e   ex1:v1.0      "/bin/sh"               10 seconds ago Up 9 seconds   upbeat_wright

C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951\ex1>docker container run -itd ex1:v1.0
3ae6764f1a8e59e6e160c6da04f8eab6b3aaf3f104a41ae424fa7000782fdbdb0

C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951\ex1>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
3ae6764f1a8e   ex1:v1.0      "/bin/sh"               10 seconds ago Up 9 seconds   upbeat_wright
```

As we can see after running the command - docker container run -itd ex1:v1.0, I ran the command - docker ps, to check if there where any containers running in the background. The -itd is used to make to make the container interactive in order to be able to access bash and in order to make the container run in the background.

Task 4: To enter the container I used the docker attach command in the image below, and when I entered the container, I ran the command - git --version, which allowed us to verify if git was installed. The usability of docker attach command is to attach our terminals standard input and output and error to a running container using the containers ID, This allows us to view its ongoing output or to control it interactively, as though the commands were running directly in your terminal.

```
C:\Users\waleed\Desktop\PRACTICAL2_waleed_wazir_19396951\ex1>docker attach 3ae6764f1a8e59e6e160c6da04f8eab6b3aaf3f104a41ae424fa7000782fddb0
/ # git --version
git version 2.11.3
/ # ps
PID   USER     TIME   COMMAND
   1  root         0:00  /bin/sh
   8  root         0:00  ps
/ #
```