**Домашна Работа бр. 1   
Анета Станковска  
Индекс: 249012**

**Запознавање со Docker, инсталација на Docker Desktop на локална машина, основни команди, креирање на Docker Container и Docker Image etc.**

**Инсталација и стартување на апликацијата**

* Download на docker desktop installer од официјалната web страна: <https://www.docker.com/>
* Со оглед на тоа што користам Windows 64-битен оперативен систем, одбирам опција Download for Windows - AMD-64.
* Го симнувам инсталерот и го пуштам .exe фајлот, а потоа ја селектирам опцијата Use WSL 2 instead of Hyper V со оглед на тоа што користам Windows 10 и мојата машина поддржува WSL2, а од друга страна е многу полесен за виртуелизација и овозможува да ранувам Linux kernel на мојата Windows машина. WSL2 е исто така и користи помалку ресурси, е побрз и по перформантен во извршување на Docker операции.
* Верзијата на Docker Desktop којашто ја инсталирам е всушност последната верзија од Docker Desktop, а тоа во моментот кога го пишувам ова е **4.36.0**
* Откако успешно ќе помине инсталацијата, инсталерот побара од мене да се одлогирам од Windows-от и повторно да се логирам.
* Откако успешно се логирав назад на Windows, сакав да се осигурам дека успешно сум инсталирала Docker на мојата машина. Го направив тоа со извршување на следната команда во терминал: **docker –version** по што во терминал го добив следниот аутпут
  + A black background with white text

    Description automatically generated
* Откако го направив тоа ја стартнав апликацијата по што добив Welcome екран кадешто Docker апликацијата побара да се регистрирам со службен или приватен емаил кадешто јас одбрав да се регистрирам со приватен Google account.
* Потоа одбрав корисничко име – anetastankovska по што ме навигираше назад на Docker Desktop апликацијата каде пополнив анкета за тоа која е мојата професија и за кои цели ќе го користам.
* По овој чекор го добивам почетниот екран на Docker Desktop апликацијата

**Креирање на image и на container, запирање на container, бришење на Image и на container, прикачување на DockerHub**

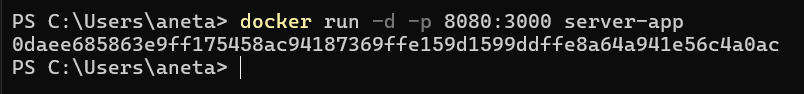
Креирање

* Креирав две мали апликации – server и client
* Server апликацијата всушност е express апликација којашто има неколку рути кои враќаат json како и готови html документи спремни за сервирање
* Client апликацијата всушност само има едно копче кое на клик прави fetch на api рута од серверската апликација
* За потребите на оваа домашна, креирав container и image само за сервер апликацијата бидејќи client апликацијата сеуште нема структура.
* Инавигирав до server фолдерот и од таму преку терминал ја извршив следната команда **docker build -t server-app:v1 .** со помош на која креирам image со таг server-app и додадов верзија v1
* За да се осигурам дека успешно сум креирала image, ја извршив следната команда: docker images по што го добив следното:

A black background with white text

Description automatically generated

* Откако успешно креирав image, следниот чекор беше да го ранувам тој image со помош на командата **docker run -d -p 8080:3000 server-app:v1** по што добив ваква порака:



Промени

* За да направам промени во container најпрво треба да го стартувам во interactive mode: **docker run -it server-app:v1 powershell**,да ги направам промените и да ги зачувам како нов image
* Оваа команда отвара shell внатре во самиот container и ни овозможува да инсталираме софтвер, да правиме промени итн.
* Откако сме ги направиле промените испишуваме **exit** во shell-от за да излеземе
* Ги листаме сите containers, го наоѓаме container-от кадешто сме ги направиле промените и ги комитираме со командата **docker commit 884a4d3b473b server-app:v2**
* Ги тестираме промените со **docker run -d -p 8080:80 server-app:v2**
* Исто така можеме и да го автоматизираме овој процес со апдејтирање на Dockerfile:
  + # Extend the existing image
  + FROM server-app:v1
  + # Add new content or software
  + RUN echo "Hello, Docker!" > /usr/share/nginx/html/index.html
* И на крај можеме и да го експортираме image-от: **docker save -o server-app-modified.tar server-app:v2** и по потреба да го импортираме подоцна: **docker load -i server-app-modified.tar**

Бришење

* За да избришам image требаше прво да проверам дали некој container го користи тој image и ако го користи треба да го запрам тој container. За таа цел прво да ги излистав сите активни containers со командата **docker ps** и кога го најдов container-от ја извршив командата **docker stop <imageId>,** после тоа ги излистав сите containers со помош на командата **docker ps -a** со цел да го најдам стопираниот container и да го избришамсо помош на командата **docker rm <containerId>.** После тоа го избришав и image-ot со помош на **docker rm <imageId>**

A screen shot of a computer

Description automatically generated

* За да го запрам container-от повторно ја повторив командата docker stop <containerId> за запирање. За бришење на image-ot само ги повторив командите за бришење од погоре.
* За да ги запреме сите containers можеме да ги извршиме следните команди:
* Запирање: **docker stop $(docker ps -q)** и бришење: **docker rm $(docker ps -aq)**

Docker Repository

* Во оваа фаза од проектот јас не користев ниту public ниту private Docker Repository, со оглед на тоа што images што ги креирав беа локално на мојата машина, и истите се достапни само таму, т.е не се достапни од екстерни места се додека јас не одлучам да ги експортирам и да ги споделам мануелно. Доказ за тоа е што не искористив ниту docker push, ниту docker pull команда
* Потоа одлучив да го прикачам image-от на Docker Hub. Постапката за тоа беше прво логирање на Docker Hub со командата docker login а после тоа и push на images на Docker Hub со командите **docker tag server-app:v1** **anetastankovska/server-app:v1** и **docker push anetastankovska/server-app:v1**

A black background with white text

Description automatically generated

* Откако направив push на image-от, истиот се појави во мојот профил на Docker Hub

A screenshot of a computer

Description automatically generated

Screenshots од дел од командите се ставени во images фолдерот.

Source кодот е поставен на моето GitHub репо, во рамки на FINKI-DevOps репозиториум.

[ЛИНК до репо](https://github.com/anetastankovska/FINKI-DevOps/tree/main/Homework_1)

**Преглед на најважните команди**

* Build an image: docker build -t <image\_name>:<tag> .
* List images: docker images
* Run a container: docker run -d -p <host\_port>:<container\_port> <image\_name>:<tag>
* Run a container interactively: docker run -it <image\_name>:<tag> /bin/bash
* List running containers: docker ps
* List all containers (including stopped): docker ps -a
* Stop a container: docker stop <container\_id>
* Remove a container: docker rm <container\_id>
* Remove all stopped containers: docker container prune
* Remove an image: docker rmi <image\_name>:<tag>
* Login to Docker Hub: docker login
* Add tag to image: docker tag <existing\_image\_name>:<tag> <new\_image\_name>:<new\_tag>
* Commit changes in a container to a new image: docker commit <container\_id> <new\_image\_name>:<tag>
* Push an image to Docker Hub: docker push <image\_name>:<tag>
* Save an image as a tar file: docker save -o <file\_name>.tar <image\_name>:<tag>
* Load an image from a tar file: docker load -i <file\_name>.tar
* Remove unused images, containers, and networks: docker system prune
* Check container logs: docker logs <container\_id>
* Execute a command inside a running container: docker exec -it <container\_id> <command>