



# Assignment 1, Web Application Development

⚙️ Status	Planning
👤 Owner	🕒 Serzhan Yerasil

Serzhan Yerasil

## Exercise 1: Installing Docker

```
C:\Users\yerasil>docker --version
Docker version 24.0.5, build ced0996

C:\Users\yerasil>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:91fb4b041da273d5a3273b6d587d62d518300a6ad268b28628f74997b93171b2
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

1. Key components of Docker is Docker Engine, Docker CLI, Docker Images, Docker Containers, **Docker Hub**
2. Docker runs directly on the OS kernel, while the virtual machine runs a full-fledged OS based on the Hypervisor. Therefore, Docker containers are much lighter than a Virtual Machine, hence launching a Docker container is very fast

3. The output was "Hello from Docker!" . It signify that Docker Client contacted to Docker Daemon, it checks if we have the image locally, if not, it goes to Docker hub and pull it. After that it creates container from this image and start it. This image configured to print this phrase to Stdout

## **Exercise 2: Basic Docker Commands**

```

C:\Users\yerasil>docker pull alpine:latest
latest: Pulling from library/alpine
43c4264eed91: Pull complete
Digest: sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview alpine:latest

C:\Users\yerasil>docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	91ef0af61f39	2 weeks ago	7.8MB
mongo	latest	a31b196b207d	2 months ago	796MB
domake-myapp	latest	dd2ede772389	4 months ago	16.8MB
<none>	<none>	744c7862cbe6	4 months ago	16.6MB
<none>	<none>	afba279a768a	4 months ago	16.6MB
<none>	<none>	a8742f7d969e	4 months ago	16.6MB
<none>	<none>	786fed929112	4 months ago	16.6MB
<none>	<none>	d1c272ae9992	4 months ago	16.6MB
<none>	<none>	a5e7b10349e7	4 months ago	16.6MB
<none>	<none>	2e8afb9499a6	4 months ago	16.6MB
<none>	<none>	1159145f7364	4 months ago	16.6MB
<none>	<none>	5a807474f689	4 months ago	16.6MB
<none>	<none>	b12053e827af	4 months ago	16.6MB
postgres	16	0e536f047d0f	4 months ago	447MB
golang-project-app	latest	7fae9208e86e	4 months ago	16.6MB
<none>	<none>	200344db2803	4 months ago	16.6MB
<none>	<none>	0aadbf8b8efb	4 months ago	16.6MB
postgres	latest	b9390dd1ea18	7 months ago	431MB
migrate/migrate	latest	34463ca35334	9 months ago	59.8MB
jaegertracing/all-in-one	1.52.0	f54c2e9a1e62	9 months ago	74MB
otel/opentelemetry-collector	0.90.1	530437bc777a	9 months ago	102MB
grafana/grafana	10.2.2	06e5d59b720d	10 months ago	399MB
prom/prometheus	v2.48.0	620d5e2a39df	10 months ago	247MB
obsidiandynamics/kafdrop	4.0.1	a4ab0ddeba31	10 months ago	611MB
confluentinc/cp-kafka	7.4.3	788258c436f7	11 months ago	851MB
confluentinc/cp-zookeeper	7.4.3	3df461e5e1f2	11 months ago	851MB
postgres	16.0	fbdl1be2cbb1f	12 months ago	417MB
danielqsj/kafka-exporter	v1.7.0	4289127a8da8	16 months ago	22MB
hello-world	latest	d2c94e258dcb	16 months ago	13.3kB
confluentinc/cp-kafka	7.3.3	9d780d4dee74	18 months ago	839MB
docker.redpanda.com/vectorized/console	v2.1.1	8d34e3f990cb	22 months ago	146MB
tchiotludo/akhq	0.22.0	fc77462bc05c	2 years ago	523MB
vectorized/redpanda	v21.11.4	6bb03e476029	2 years ago	302MB
postgres	14.1	da2cb49d7a8d	2 years ago	374MB
docker.vectorized.io/vectorized/redpanda	v21.9.3	3272ec2e7432	2 years ago	289MB
redis	6.0.9-alpine	c678242f9116	3 years ago	31.6MB
wurstmeister/zookeeper	latest	3f43f72cb283	5 years ago	510MB
elevy/zookeeper	latest	6b8df71ed13e	6 years ago	91.7MB

```

C:\Users\yerasil>docker run -d 91ef0af61f39
bbb7a7792aba2a215815044bfc16aa13647b6c623d54e3a8bf4200ea5346e42b

C:\Users\yerasil>docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4bea3927fbc7c452322a4b1f3c86f6d8ce4a41c4ae7448e3945b1f7f0f549d9f	alpine:latest	sleep infinity	2 weeks ago	Up 2 weeks		alpine

```

C:\Users\yerasil>docker run -d a31b196b207d

```

1. Docker pull pulls up needed image from Docker Hub  
Docker run checks if we have the image locally, if not, it goes to Docker hub

- and pull it. After that it creates container from this image and start it
2. By command `docker ps` we can see running containers, also it has info about container id, used image, status, port etc.
  3. After the container stops, Docker saves information about the container so it can be restarted it in the future by command `docker start <cont_id>`

## Exercise 3: Working with Docker Containers

```
C:\Users\yerasil>docker run -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
a2318d6c47ec: Pull complete
095d327c79ae: Pull complete
bbfaa25db775: Pull complete
7bb6fb0cfb2b: Pull complete
0723edc10c17: Pull complete
24b3fdc4d1e3: Pull complete
3122471704d5: Pull complete
Digest: sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
Status: Downloaded newer image for nginx:latest
d35eb9ab243cf5da4b71b2a84446816304acd3f59310019839521fa9ac36a7f6

C:\Users\yerasil>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
d35eb9ab243c   nginx    "/docker-entrypoint..." 53 seconds ago Up 52 seconds 0.0.0.0:8080->80/tcp   fervent_tesla

C:\Users\yerasil>docker exec -it d35eb9ab243c /bin/bash
root@d35eb9ab243c:/# ls
bin      dev                docker-entrypoint.sh    home  lib64  mnt  proc  run  srv  tmp  var
boot    docker-entrypoint.d  etc                    lib   media  opt  root  sbin sys  usr

root@d35eb9ab243c:/# q
bash: q: command not found
root@d35eb9ab243c:/# exit
exit

C:\Users\yerasil>docker stop d35eb9ab243c
d35eb9ab243c

C:\Users\yerasil>docker rm d35eb9ab243c
d35eb9ab243c
```

1. Port mapping is the connection between a docker container and the outside world. Thus, we can work with the container application through the port to which we mapped. For example, if an application is running on port 80 in a container, and we want to map it to server port 8000, it will be 8000:80
2. The `docker exec` command allows to execute commands in running container. For example, we have container with Postgresql. We can connect by `docker exec -it postgres_container psql -U postgres` and work with database
3. A stopped container does not consume system resources because its processes are terminated.

But still continues to take up space. To completely clean up the resource container, you need to delete the container and volumes

## Exercise 1: Creating a Simple Dockerfile

```
app.py Dockerfile X
D: > WebDev2024 > Dockerfile > ...
1 FROM python:alpine3.10
2
3 WORKDIR /app
4
5 COPY app.py .
6
7 ENTRYPOINT ["python", "app.py"]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\yerasil> cd ../../
● PS C:\> cd D:
● PS D:\> cd .\WebDev2024\
● PS D:\WebDev2024> docker build -t hello-docker .
[+] Building 12.8s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 127B                                0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for docker.io/python:alpine3.10      3.3s
=> [1/3] FROM docker.io/python:alpine3.10@sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6f9c63 9.2s
=> => resolve docker.io/python:alpine3.10@sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6f9c63 0.0s
=> => sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 300.94kB / 300.94kB 0.8s
=> => sha256:0d32a27dde5abbed916f46fcb2c81121cc1d09851ba5c744322ab51207f7e295 22.57MB / 22.57MB 8.3s
=> => sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6f9c63decbbfbbf407b96a 1.65kB / 1.65kB 0.0s
=> => sha256:655edcda221823fcd79b61095dd77e6c767bf1543505dcf078f6945497c7fcf 1.37kB / 1.37kB 0.0s
=> => sha256:cf02325838736bb42f50cff8931c1b0e5363d6106283d98ba769ac81376e9125 7.14kB / 7.14kB 0.0s
=> => sha256:21c83c5242199776c232920ddb58cfa2a46b17e42ed831ca9001c8dbc532d22d 2.80MB / 2.80MB 1.5s
=> => sha256:2cb80a514e077437cb1c8bc06caa99b515ba357f530d64833423d3e3b5b60c1a 230B / 230B 1.3s
=> => sha256:d5d3b19aaadda1328d19cdfa8b1933e08c113e613efa7d0083e6b317cab8031a 1.93MB / 1.93MB 2.4s
=> => extracting sha256:21c83c5242199776c232920ddb58cfa2a46b17e42ed831ca9001c8dbc532d22d 0.1s
=> => extracting sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 0.1s
=> => extracting sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 0.1s
=> => extracting sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 0.1s
=> => extracting sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 0.1s
=> => extracting sha256:9a80d14c35bdcf3182348df62dae02f393b19e47d873d74d008296c54b784602 0.1s
=> => extracting sha256:0d32a27dde5abbed916f46fcb2c81121cc1d09851ba5c744322ab51207f7e295 0.5s
=> => extracting sha256:2cb80a514e077437cb1c8bc06caa99b515ba357f530d64833423d3e3b5b60c1a 0.0s
=> => extracting sha256:d5d3b19aaadda1328d19cdfa8b1933e08c113e613efa7d0083e6b317cab8031a 0.1s
=> [internal] load build context                                  0.0s
=> => transferring context: 56B                                       0.0s
=> [2/3] WORKDIR /app                                           0.1s
=> [3/3] COPY app.py .                                          0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:bd2590b5502d976ed4afb6f638e8874738b422dcb54d38991543fa85d30c509c5 0.0s
● => => naming to docker.io/library/hello-docker                  0.0s

○ What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
  PS D:\WebDev2024> docker run hello-docker
  Hello, Docker!
  PS D:\WebDev2024>
```

1. From means on the basis of which image the container will work. For example, if you select the image FROM python:alpine3.10, a container will be created based on the Alpine distribution, this OS will contain only the necessary components to run a Python application
2. COPY copies files or folders from the local environment to the container's working directory.
3. CMD can be overridden, while ENTRYPOINT can not. For example, if you use cmd and run through docker, run <im\_name> python main.go and the command that was passed through the console will take precedence. And at the entry point it will be launched with what is written in the dockerfile

## **Exercise 2: Optimizing Dockerfile with Layers and Caching**

```
app.py Dockerfile X
D: > WebDev2024 > Dockerfile > ...
1 FROM python:alpine3.10
2
3
4 WORKDIR /app
5
6 COPY requirements.txt .
7
8 RUN pip install --no-cache-dir -r requirements.txt
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

elevy/zookeeper latest 6b8df71ed13e 6 years ago 91.7MB
● PS D:\WebDev2024> docker run 5d132fdd7336
Hello, Docker!
● PS D:\WebDev2024> docker build .
[+] Building 0.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 207B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:alpine3.10
=> [1/5] FROM docker.io/library/python:alpine3.10@sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6fbe9c63decbbffbf407b96a
=> [internal] load build context
=> => transferring context: 92B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt .
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:5d132fdd733636cd4988f32406eaca737e5c7c265cd01bd2141ce69d0e313934

What's Next?
View summary of image vulnerabilities and recommendations → docker scout quickview
● PS D:\WebDev2024> docker build .
[+] Building 5.3s (10/10) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 207B
=> [internal] load metadata for docker.io/library/python:alpine3.10
=> [1/5] FROM docker.io/library/python:alpine3.10@sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6fbe9c63decbbffbf407b96a
=> [internal] load build context
=> => transferring context: 124B
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:ee553ce0cd98fc8d66e790f7bb0fd6ec6769d90cdbe697a98fa8709a0a08bca2
```

1. Docker layers are instructions that are executed sequentially. It has a useful property like caching.
2. If the file or folder has not changed since the previous time, docker will take it from the cache
3. Dockerignore is a file that will exclude files or folders for building the image

## Exercise 3: Multi-Stage Builds

The screenshot shows the Visual Studio Code interface with a Dockerfile being edited. The Explorer on the left shows the file structure: `GO` (selected), `Dockerfile`, `go.mod`, and `main.go`. The Dockerfile editor shows the following content:

```
1 FROM golang:1.23 as builder
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN go build -o hello .
8
9 FROM alpine:latest
10
11 WORKDIR /app
12
13 COPY --from=builder /app/hello .
14
15 CMD ["/hello"]
16
```

The bottom panel shows the **TERMINAL** tab with the following output:

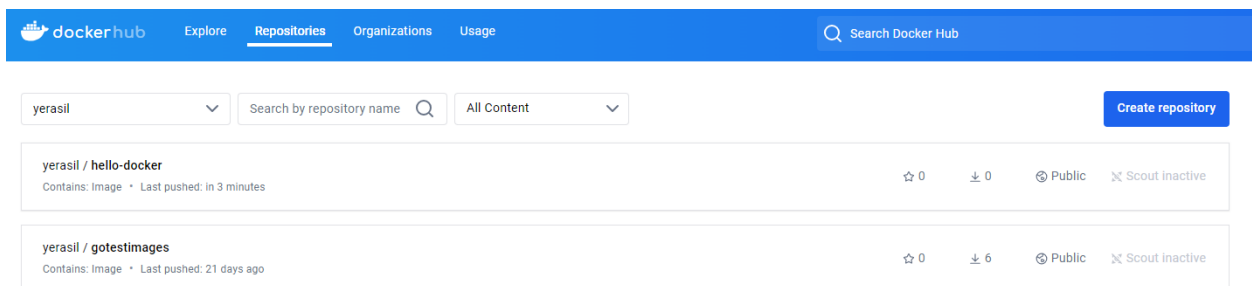
PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
			vectorized/redpanda	v21.11.4
			postgres	14.1
			docker.vectorized.io/vectorized/redpanda	v21.9.3
			redis	6.0.9-alpine
			wurstmeister/zookeeper	latest
			elevy/zookeeper	latest
			PS D:\WebDev2024\Go> docker run dbedb984983c	
			Hello, World!	
			PS D:\WebDev2024\Go>	



1. The benefit of multi-assembly is that we will not store the Go compiler in the final image, but simply a binary. Thus, we reduce the weight of the final image
2. The final image uses intermediate instructions to use useful values

## Exercise 4: Pushing Docker Images to Docker Hub

```
PS D:\WebDev2024\Go> docker tag dbedb984983c yerasil/hello-docker
PS D:\WebDev2024\Go> docker push yerasil/hello-docker
Using default tag: latest
The push refers to repository [docker.io/yerasil/hello-docker]
049f775d4c90: Pushed
36e27e339f40: Pushed
63ca1fbb43ae: Pushed
latest: digest: sha256:ebda30cc3e5ff04d688f3d022ac3cb13255038193fa722d5b41dab0a8ab77b79 size: 945
```



1. Docker-hub is needed to store images; for example, you can develop a useful image and upload it to the docker hub. Other people may use your image
2. I tagged with command `docker tag hello-docker <your-username>/hello-docker`
3. Create account in Docker hub, tag the image and push it by `docker push <your-username>/hello-docker`