Intro to Containerization: Docker

Exercise 1: Installing Docker

```
Администратор: Командная строка
Microsoft Windows [Version 10.0.22631.4169]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\System32>docker --version
Docker version 27.2.0, build 3ab4256
```

Fig. 1: Docker version

```
C:\Windows\System32>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.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


C:\Windows\System32>
```

Fig. 2: Docker run hello-world

Q1: Docker engine, docker CLI, images, containers, dockerhub

Q2: Docker containers are lighter and faster because they consume fewer resources and require less memory than virtual machines

Q3: Output will be like in figure 3. This message signifies that you successfully connected to Docker.

Exercise 2: Basic Docker commands

```
C:\Windows\System32>docker pull nginx
Using default tag: latest
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
docker.io/library/nginx:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview nginx

C:\Windows\System32>docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    39286ab8a5e1   5 weeks ago     188MB
hello-world   latest    d2c94e258dcb   16 months ago   13.3kB

C:\Windows\System32>docker run -d nginx
f27ceddf037c7cf058ebf6908ae941e711bebfaeb4b643dcf4e56776b7551ed2

C:\Windows\System32>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
f27ceddf037c   nginx     "/docker-entrypoint.…"   8 seconds ago   Up 8 seconds   80/tcp    intelligent_euler

C:\Windows\System32>docker stop f27ceddf037c
f27ceddf037c

C:\Windows\System32>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Windows\System32>
```

Fig.3: Basic Docker commands

Q1: Docker pull just download the image. Docker run can download if it's necessary and then run container from image.

Q2: By using command docker ps.

Q3: Container change it's state from running to stopped. Yes, it can be restarted.

Exercise 3: Working with Docker Containers

```
C:\Windows\System32>docker run -d -p 8080:80 nginx
ae376b057aaf4a14afba118acc70fea0572667a0d712152ba05f38cc9cae4ba5

C:\Windows\System32>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                   NAMES
ae376b057aaf   nginx     "/docker-entrypoint.…"   30 minutes ago   Up 30 minutes   0.0.0.0:8080->80/tcp   infallible_robinson

C:\Windows\System32>docker exec -it ae376b057aaf

What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug ae376b057aaf
    Learn more at https://docs.docker.com/go/debug-cli/
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Execute a command in a running container

C:\Windows\System32>docker exec -it ae376b057aaf /nib/bash
OCI runtime exec failed: exec failed: unable to start container process: exec: "/nib/bash": stat /nib/bash: no such file or directory: unknown

What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug ae376b057aaf
    Learn more at https://docs.docker.com/go/debug-cli/

C:\Windows\System32>docker exec -it ae376b057aaf /bin/bash
root@ae376b057aaf:/#
root@ae376b057aaf:/#
root@ae376b057aaf:/# aaaa
bash: aaaa: command not found
root@ae376b057aaf:/# docker stop ae376b057aaf
bash: docker: command not found
root@ae376b057aaf:/# /#
bash: /#: No such file or directory
root@ae376b057aaf:/# exit
exit

What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug ae376b057aaf
    Learn more at https://docs.docker.com/go/debug-cli/

C:\Windows\System32> docker stop ae376b057aaf
ae376b057aaf

C:\Windows\System32>docker rm ae476b057aaf
Error response from daemon: No such container: ae476b057aaf

C:\Windows\System32>docker rm ae376b057aaf
ae376b057aaf

C:\Windows\System32>
```

Fig. 4: Working with Docker Containers

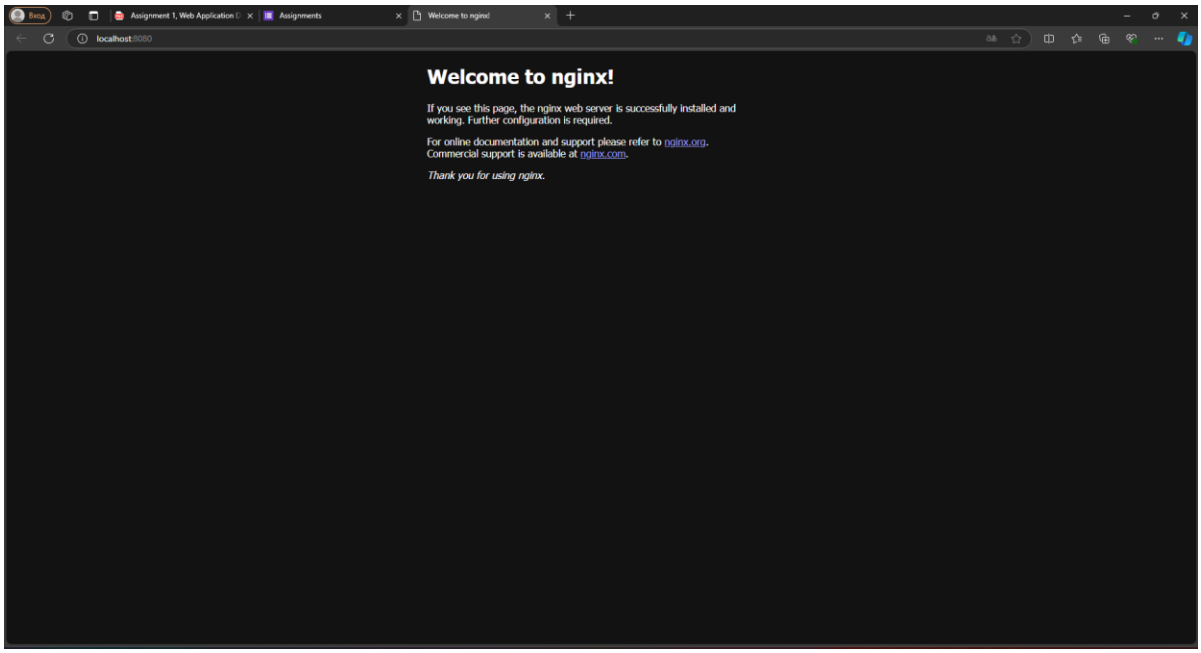

Fig.5: http://localhost:8080 in my browser

Q1: Port mapping using after flag –p and you type your port. It's important for communication between Docker and outside world.

Q2: To run a command in already running container

Q3: By removing this container with command rm <container-id>

Dockerfile

Exercise 1: Creating a simple Docekrfile

```
PS C:\Users\ravil\Downloads\hw_wb_1> docker build -t hello-docker .
[+] Building 1.0s (8/8) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 112B                                                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                  0.8s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [1/3] FROM docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c451784beef8aa31d1a313d8e3fc122e4a1891085a1  0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 27B                                                                                    0.0s
 => CACHED [2/3] WORKDIR /app                                                                                       0.0s
 => CACHED [3/3] COPY app.py .                                                                                      0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:3c33aa663000875afc25dadea1f37fd384b005b95b99e8ddfef7af51eab2a25a                        0.0s
 => => naming to docker.io/library/hello-docker                                                                     0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ravil\Downloads\hw_wb_1> docker run hello-docker
Hello, Docker!
PS C:\Users\ravil\Downloads\hw_wb_1>
```

Fig. 6: Creating Dockerfile

Q1: For defining the foundation of your Docker image

Q2: COPY <source> <destination> where <source> is the path from which files will be copied, and <destination> is a path inside Docker image where files will be copied.

Q3: CMD defines default commands that can be overridden. ENTRYPOINT sets a command that is always executed when the container starts.

Exercise 2: Optimizing Dockerfile with Layers and Caching

```
PS C:\Users\ravil\Downloads\hw_wb_1> docker build -t hello-docker-optimized .
[+] Building 32.7s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 191B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
 => => transferring context: 71B
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c451784beef8aa31d1a313d8e3fc122e4a1891085a104b7cfb
 => [internal] load build context
 => => transferring context: 119B
 => CACHED [2/5] WORKDIR /app
 => [3/5] COPY requirements.txt .
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => writing image sha256:026b6a71a56a4a5306339ca38ff5c97efc7821f8f0f8cc1f9126b23104cd62b3
 => => naming to docker.io/library/hello-docker-optimized

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ravil\Downloads\hw_wb_1>
```

Fig. 7: Creating optimized Dockerfile

```
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ravil\Downloads\hw_wb_1> docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
hello-docker-not-optimized  latest    946a7a14283f   23 seconds ago   218MB
hello-docker-optimized      latest    3e6402368bd0   2 minutes ago    201MB
<none>                      <none>    b8702bcedfed   58 minutes ago   125MB
hello-docker                latest    3c33aa663000   58 minutes ago   125MB
nginx                       latest    39286ab8a5e1   5 weeks ago      188MB
hello-world                 latest    d2c94e258dcb   16 months ago    13.3kB
PS C:\Users\ravil\Downloads\hw_wb_1>
```

Fig.8: Comparison of optimized and non-optimized Dockerfile

Q1: Docker layers are read-only components of an image that affect size and build times by enabling caching of unchanged layers.

Q2: The Docker build cache interrupts the build process by reusing unchanged pieces from previous builds.

Q3: .dockerignore file specifies which files and directories should be excluded from the build

Exercise 3: Multi-Stage builds

```
PS C:\Users\ravil\Downloads\hw_wb_1\go> docker build -t hello-docker-go .
[+] Building 6.7s (15/15) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 229B                                                                                0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                                    1.5s
 => [internal] load metadata for docker.io/library/golang:1.20-alpine                                               1.5s
 => [auth] library/alpine:pull token for registry-1.docker.io                                                       0.0s
 => [auth] library/golang:pull token for registry-1.docker.io                                                       0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [builder 1/5] FROM docker.io/library/golang:1.20-alpine@sha256:e47f121850f4e276b2b210c56df3fda9191278dd84a3a4  0.0s
 => CACHED [stage-1 1/2] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc5  0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 255B                                                                                   0.0s
 => CACHED [builder 2/5] WORKDIR /app                                                                               0.0s
 => [builder 3/5] COPY . .                                                                                          0.1s
 => [builder 4/5] RUN go mod init my-go-app                                                                         0.4s
 => [builder 5/5] RUN go build -o main .                                                                            4.3s
 => [stage-1 2/2] COPY --from=builder /app/main /main                                                               0.1s
 => exporting to image                                                                                              0.1s
 => => exporting layers                                                                                             0.1s
 => => writing image sha256:fd441e989ff37d9204f326020ef82387ad694878cd08cafe9a60c161bc83521a                        0.0s
 => => naming to docker.io/library/hello-docker-go                                                                  0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ravil\Downloads\hw_wb_1\go> docker run hello-docker-go
Hello, World!
```

Fig. 9: Build and run hello-docker-go image

```
PS C:\Users\ravil\Downloads\hw_wb_1\go> docker build -t hello-docker-go .
[+] Building 6.7s (15/15) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 229B                                                                                0.0s
 => [internal] load metadata for docker.io/library/alpine:latest                                                    1.5s
 => [internal] load metadata for docker.io/library/golang:1.20-alpine                                               1.5s
 => [auth] library/alpine:pull token for registry-1.docker.io                                                       0.0s
 => [auth] library/golang:pull token for registry-1.docker.io                                                       0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [builder 1/5] FROM docker.io/library/golang:1.20-alpine@sha256:e47f121850f4e276b2b210c56df3fda9191278dd84a3a4  0.0s
 => CACHED [stage-1 1/2] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc5  0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 255B                                                                                   0.0s
 => CACHED [builder 2/5] WORKDIR /app                                                                               0.0s
 => [builder 3/5] COPY . .                                                                                          0.1s
 => [builder 4/5] RUN go mod init my-go-app                                                                         0.4s
 => [builder 5/5] RUN go build -o main .                                                                            4.3s
 => [stage-1 2/2] COPY --from=builder /app/main /main                                                               0.1s
 => exporting to image                                                                                              0.1s
 => => exporting layers                                                                                             0.1s
 => => writing image sha256:fd441e989ff37d9204f326020ef82387ad694878cd08cafe9a60c161bc83521a                        0.0s
 => => naming to docker.io/library/hello-docker-go                                                                  0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ravil\Downloads\hw_wb_1\go> docker run hello-docker-go
Hello, World!
PS C:\Users\ravil\Downloads\hw_wb_1\go> docker images
REPOSITORY                  TAG       IMAGE ID       CREATED             SIZE
hello-docker-go             latest    fd441e989ff3   2 minutes ago       9.65MB
hello-docker-not-optimized  latest    946a7a14283f   14 minutes ago      218MB
hello-docker-optimized      latest    3e6402368bd0   16 minutes ago      201MB
hello-docker                latest    3c33aa663000   About an hour ago   125MB
nginx                       latest    39286ab8a5e1   5 weeks ago         188MB
PS C:\Users\ravil\Downloads\hw_wb_1\go>
```

Fig.10: Size comparison

Q1: Multi-stage builds enable smaller images and improved efficiency by separating build and runtime environments.

Q2: It reduce image size by including only necessary artifacts in the final image, excluding build tools and dependencies.

Q3: Reducing image size, compiling applications.

Exercise 4: Pushing Docker images to Docker Hub

```
PS C:\Users\ravil\Downloads\hw_wb_1> docker tag hello-docker lotur1/hello-docker
PS C:\Users\ravil\Downloads\hw_wb_1> docker login
Authenticating with existing credentials...
Login Succeeded
PS C:\Users\ravil\Downloads\hw_wb_1> docker push lotur1/hello-docker
Using default tag: latest
The push refers to repository [docker.io/lotur1/hello-docker]
f43d1d07ad67: Pushed
88576f3feb0e: Pushed
c33557cc9b97: Mounted from library/python
774888eabf21: Mounted from library/python
c559d9567165: Mounted from library/python
8e2ab394fabf: Mounted from library/python
latest: digest: sha256:194edc811a9071d5481201696db117c70c539333228ad948b9ee97f1089f7f4f size: 1572
PS C:\Users\ravil\Downloads\hw_wb_1>
```

Fig.11: Pushing images to dockerhub

Q1: For storing, sharing, collaboration, versioning Docker images.

Q2:docker tag < image> <username>/<image>

Q3: login, tag, push