

Ulzhana Kylyshbek (ulzhik5566)

<https://hub.docker.com/r/ulzhik5566/hello-go>

Assignment 1, Web Application Development

Intro to Containerization: Docker

Exercise 1: Installing Docker

```
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker run hello-world
[sudo] password for ulzhana:

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

1. **Objective:** Install Docker on your local machine.
2. **Steps:**
 - Follow the installation guide for Docker from the official website, choosing the appropriate version for your operating system (Windows, macOS, or Linux).
 - After installation, verify that Docker is running by executing the command `docker --version` in your terminal or command prompt.
 - Run the command `docker run hello-world` to verify that Docker is set up correctly.
3. **Questions:**
 - What are the key components of Docker (e.g., Docker Engine, Docker CLI)?
 - Docker Engine, Docker CLI, Docker Compose and Docker Hub.
 - How does Docker compare to traditional virtual machines? Docker is faster and more resource-efficient than traditional virtual machines because virtual machines run full OS instances with their own kernel.
 - What was the output of the `docker run hello-world` command, and what does it signify? It gave: Hello world Docker! It says that everything is okay and ready with Docker.

Exercise 2: Basic Docker Commands

```

ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo 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: sha256:04ba374043ccd2fc5c593885c0eacddeabd5ca375f9323666f28dfd5a9710e3
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "h
mission denied
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
nginx                latest          39286ab8a5e1   5 weeks ago     188MB
hello-world          latest          d2c94e258dcb   17 months ago   13.3kB
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker run -d nginx
d7d92aba51843df74459b2f6f3b3dfe4314118b63cacb9c51723b99f0db62c23
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS          NAMES
d7d92aba5184   nginx     "/docker-entrypoint..." 56 seconds ago   Up 55 seconds   80/tcp         zealous_banza
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker stop nginx
Error response from daemon: No such container: nginx
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker stop ^C
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker stop op ^C
Error response from daemon: No such container: op
Error response from daemon: No such container: ^C
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ : command not found
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~$ sudo docker stop d7d92aba5184
d7d92aba5184

```

1. **Objective:** Familiarize yourself with basic Docker commands.
2. **Steps:**
 - Pull an official Docker image from Docker Hub (e.g., `nginx` or `ubuntu`) using the command `docker pull <image-name>`.
 - List all Docker images on your system using `docker images`.
 - Run a container from the pulled image using `docker run -d <image-name>`.
 - List all running containers using `docker ps` and stop a container using `docker stop <container-id>`.
3. **Questions:**
 - What is the difference between `docker pull` and `docker run`? Docker pull downloads a Docker image to the local machine. Docker run starts a new container from an existing image.
 - How do you find the details of a running container, such as its ID and status? Docker ps command lists all active containers with their info.
 - What happens to a container after it is stopped? Can it be restarted? It remains in a "stopped" state on the system and can be restarted using the docker start command.

Exercise 3: Working with Docker Containers

- Stop and remove the container using `docker stop <container-id>` and `docker rm <container-id>`.

3. Questions:

- How does port mapping work in Docker, and why is it important? Port mapping allows Docker containers to expose their internal ports to the host machine, making the container's services accessible. It's important for enabling communication between containers and external systems or users.
- What is the purpose of the `docker exec` command? This command allows to run additional commands inside an already running container.
- How do you ensure that a stopped container does not consume system resources? We can remove it using `docker rm` because stopped containers still use disk space until they are deleted.

Dockerfile

Exercise 1: Creating a Simple Dockerfile

```

ulzhana@ulzhana-Lenovo-YOGA-530-14IKB: ~/Downloads/docker-python-app
File Edit View Search Terminal Help
GNU nano 2.9.3 app.py
print("Hello, Docker!")

ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ ls
app.py
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ nano Dockerfile
Use "fg" to return to nano.

[7]+ Stopped nano Dockerfile
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ ls
app.py Dockerfile
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker build -t hello-docker .
[+] Building 11.5s (7/7) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 112B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim 3.2s
=> [internal] load build context 0.0s
=> => transferring context: 57B 0.0s
=> [1/2] FROM docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c45 7.8s
=> => resolve docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c45 0.0s
=> => sha256:2851c06da1fdc3c451784beef8aa31d1a313d8e3f 10.41kB / 10.41kB 0.0s
=> => sha256:d465e807ab2e72c74ec6fa81d1d2751108c7861a9c0 1.75kB / 1.75kB 0.0s
=> => sha256:397ed8d3163622f16a7ad7f8d235cb365b893a589ce 5.22kB / 5.22kB 0.0s
=> => sha256:0fa26e0a6c779a41b265beaf4e11ac2899b82fc487f 3.51MB / 3.51MB 1.8s
=> => sha256:a657783e238bb29a690f2f314f9eb40a7bf9ab06 14.74MB / 14.74MB 5.0s
=> => sha256:b665d04ddefb24a5af0c944a98df2ebfb1e3a26e0a54657 251B / 251B 0.3s
=> => extracting sha256:0fa26e0a6c779a41b265beaf4e11ac2899b82fc487fde96e 0.6s
=> => extracting sha256:a657783e238bb29a690f2f314f9eb40a7bf9ab06f795e95 2.5s
=> => extracting sha256:b665d04ddefb24a5af0c944a98df2ebfb1e3a26e0a546573 0.0s
=> [2/2] COPY app.py /app.py 0.3s
=> => exporting layers 0.0s
=> => writing image sha256:25edc7232474ed5f50f4e51b7a576955433c4e815f1ae 0.0s
=> => naming to docker.io/library/hello-docker 0.0s
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ docker run hello-docker
WARNING: Error loading config file: /home/ulzhana/.docker/config.json: open /home/ulzhana/.docker/config.json: permission
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http:
k: connect: permission denied.
See 'docker run --help'.
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker run hello-docker
Hello, Docker!

```

1. **Objective:** Write a Dockerfile to containerize a basic application.
2. **Steps:**
 - Create a new directory for your project and navigate into it.

- Create a simple Python script (e.g., `app.py`) that prints "Hello, Docker!" to the console.
- Write a Dockerfile that:
 - Uses the official Python image as the base image.
 - Copies `app.py` into the container.
 - Sets `app.py` as the entry point for the container.
- Build the Docker image using `docker build -t hello-docker ..`
- Run the container using `docker run hello-docker`.

3. Questions:

- What is the purpose of the `FROM` instruction in a Dockerfile? It specifies the base image from which to build the new image.
- How does the `COPY` instruction work in Dockerfile? It copies files from the host filesystem into the container's filesystem at the specified path during the image build process.
- What is the difference between `CMD` and `ENTRYPOINT` in Dockerfile? `CMD` provides default arguments for a container when it starts and can be overridden, while `ENTRYPOINT` specifies the command to run as the main process of the container, ensuring it cannot be easily overridden.

Exercise 2: Optimizing Dockerfile with Layers and Caching

```
File Edit View Search Terminal Help
GNU nano 2.9.3 Dockerfile
FROM python:3.9-slim
COPY app.py /app.py
ENTRYPOINT ["python", "/app.py"]
```

```

ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ nano Dockerfile
Use "fg" to return to nano.

[12]+  Stopped                  nano Dockerfile
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker build -t pythonapp
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker build -t pythonapp .
[+] Building 2.4s (7/7) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 112B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 59B                                                  0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim               2.3s
=> [1/2] FROM docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c45         0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 27B                                                  0.0s
=> CACHED [2/2] COPY app.py /app.py                                             0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:25edc7232474ed5f50f4e51b7a576955433c4e815f1ae       0.0s
=> => naming to docker.io/library/pythonapp                                    0.0s
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker run pythonapp
Hello, Docker!
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/docker-python-app$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-docker        latest             25edc7232474       17 minutes ago     125MB
pythonapp           latest             25edc7232474       17 minutes ago     125MB
nginx               latest             39286ab8a5e1       5 weeks ago        188MB
hello-world         latest             d2c94e258dcb       17 months ago      13.3kB

```

1. **Objective:** Learn how to optimize a Dockerfile for smaller image sizes and faster builds.
2. **Steps:**
 - Modify the Dockerfile created in the previous exercise to:
 - Separate the installation of Python dependencies (if any) from the copying of application code.
 - Use a **.dockerignore file** to exclude unnecessary files from the image.
 - Rebuild the Docker image and observe the build process to understand how caching works.
 - Compare the size of the optimized image with the original.
3. **Questions:**
 - What are Docker layers, and how do they affect image size and build times? Docker images are made up of layers, with each layer representing a set of changes. Layers are cached and reused, affecting image size and build times; fewer and more efficient layers can reduce size and speed up builds.
 - How does Docker's build cache work, and how can it speed up the build process? Docker's build cache stores the results of each step in the Dockerfile, allowing unchanged layers to be reused in subsequent builds. This speeds up the build process by skipping steps that haven't changed, significantly reducing build time.
 - What is the role of the **.dockerignore** file? It specifies files and directories that should be excluded from the build context when creating a Docker image. This reduces the size of the context.

Exercise 3: Multi-Stage Builds

```
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB: ~/Downloads/go_app
File Edit View Search Terminal Help
GNU nano 2.9.3 main.go

package main
import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

```
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB: ~/Downloads/go_app
File Edit View Search Terminal Help
GNU nano 2.9.3 Dockerfile

#1
FROM golang:1.16 as builder
WORKDIR /app
COPY go.mod .
RUN go mod download
COPY main.go .
RUN go build -o main .

#2
FROM alpine:latest
WORKDIR /root/
COPY --from=builder /app/main .
CMD ["/main"]
```

```
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ nano go.mod
Use "fg" to return to nano.

[23]+ Stopped nano go.mod
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker build -t hello-go .
[+] Building 4.8s (15/15) FINISHED
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 241B 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 0.9s
=> [internal] load metadata for docker.io/library/golang:1.16 0.9s
=> [builder 1/6] FROM docker.io/library/golang:1.16@sha256:5f6a4662de3ef 0.0s
=> [stage-1 1/3] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1d 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 82B 0.0s
=> CACHED [stage-1 2/3] WORKDIR /root/ 0.0s
=> CACHED [builder 2/6] WORKDIR /app 0.0s
=> [builder 3/6] COPY go.mod . 0.2s
=> [builder 4/6] RUN go mod download 0.9s
=> [builder 5/6] COPY main.go . 0.1s
=> [builder 6/6] RUN go build -o main . 2.1s
=> [stage-1 3/3] COPY --from=builder /app/main . 0.2s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:6c55bcd67a713503c7dc982ca748d2521401ad74b198 0.0s
=> => naming to docker.io/library/hello-go 0.0s
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ ls
Dockerfile go1.21.1.linux-amd64.tar.gz go.mod main.go
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker run hello-go
Hello, World!
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ ls
Dockerfile go1.21.1.linux-amd64.tar.gz go.mod main.go
```

1. **Objective:** Use multi-stage builds to create leaner Docker images.
2. **Steps:**

- Create a new project that involves compiling a simple Go application (e.g., a "Hello, World!" program).
- Write a Dockerfile that uses multi-stage builds:
 - The first stage should use a Golang image to compile the application.
 - The second stage should use a minimal base image (e.g., **alpine**) to run the compiled application.
- Build and run the Docker image, and compare the size of the final image with a single-stage build.

3. Questions:

- What are the benefits of using multi-stage builds in Docker? They allow to create smaller, more efficient images by separating the build environment from the runtime environment, reducing complexity and improving security.
- How can multi-stage builds help reduce the size of Docker images? By using multi-stage builds, you can copy only the necessary artifacts from the build stage to the final stage, excluding unnecessary files. This results in a smaller final image size.
- What are some scenarios where multi-stage builds are particularly useful? For example, in compiling code in Go, Java, where you want to compile in one stage and run in another.

Exercise 4: Pushing Docker Images to Docker Hub


```
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: ulzhik5566
Password:
WARNING! Your password will be stored unencrypted in /home/ulzhana/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store


Login Succeeded
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker tag hello-go
"docker tag" requires exactly 2 arguments.
See 'docker tag --help'.

Usage:  docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]


Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker tag hello-go ulzhik5566/hello-go
ulzhana@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$ sudo docker push ulzhik5566/hello-go
Using default tag: latest
The push refers to repository [docker.io/ulzhik5566/hello-go]
def3171452a9: Pushed
5f70bf18a086: Pushed
63ca1fbb43ae: Mounted from library/alpine
latest: digest: sha256:309ef2c965c3d602e8a4c386335a2fa757416b9c974914a4d6a9d957a1642275 size: 945
ulzhik5566@ulzhana-Lenovo-YOGA-530-14IKB:~/Downloads/go_app$
```

Explore / ulzhik5566/hello-go




ulzhik5566/hello-go



By **ulzhik5566** · Updated 2 minutes ago



[Manage Repository](#)

 Pulls 0

[Overview](#)
[Tags](#)



No overview available

This repository doesn't have an overview

Docker Pull Command

```
docker pull ulzhik5566/hello-go
```

[Copy](#)

1. **Objective:** Learn how to share Docker images by pushing them to Docker Hub.
2. **Steps:**
 - Create an account on Docker Hub.
 - Tag the Docker image you built earlier with your Docker Hub username (e.g., **docker tag hello-docker <your-username>/hello-docker**).
 - Log in to Docker Hub using **docker login**.

- Push the image to Docker Hub using `docker push <your-username>/hello-docker`.
- Verify that the image is available on Docker Hub and share it with others.

3. **Questions:**

- What is the purpose of Docker Hub in containerization? It is a cloud-based registry service that allows users to store, share, and manage Docker images.
- How do you tag a Docker image for pushing to a remote repository? `docker tag` command used to push to a remote repository. Example: `docker tag hello-go ulzhik5566/hello-go`.
- What steps are involved in pushing an image to Docker Hub? `docker login`, `docker tag hello-go ulzhik5566/hello-go`, `docker push ulzhik5566/hello-go`.