Training Project Lab, 2^{nd} & 3^{rd} tasks

The purpose of this file is to unofficially-document the undergoing process in the Training Project Lab, here I write what I find important to remember for further work. I would like to highlight that various sources were used to write this document, mainly Wikipedia and Docker official documentation.

Serverless computing: describing the native architecture of the cloud. What we mean by 'serverless' is that vendors take the responsibility of managing all the behind the scene work starting from operating system management to server maintenance. This enables the users to build their application focusing on its core, rather than worrying about the process of deployment. It is important not to confuse this concept with other networking models which use no actual server.

Docker: "Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code."

https://opensource.com/resources/what-docker

- Starting with Docker: Installing, building an image, managing it.
- More Docker: edit Dockerfile (write a simple dockerfile that uses a container from dockerfile), setting up a container, managing ports, setting environment variables, mounting volumes.
    - Dockerfile: "a text-based script of instructions that is used to create a container image."
      Some commands:

```
# Comment
FROM {base-image (e.g.: scratch/ubuntu)}
MAINAINER {personal information}
RUN {with respect to the base (e.g.: yarn to install dependencies)}
CMD ["command", "command"]
```

- Database persisting: With each container instance created, a new filesystem is created, which means that we need to find a way to persist our data.
- "Volumes provide the ability to connect specific filesystem paths of the container back to the host machine. If a directory in the container is mounted, changed on that directory are also seen on the host machine. If we mount that same directory across container restarts, we'd see the same files"
- Creating a volume?
    - `docker volume create todo-db`
- Then?
    - `'-v' flag when 'run' command to specify a volume mount`
- This was the usage of 'named volume', think of it as a data bucket

- Else? There is 'bind mounts', a very common technique for development setups. Instead of having to installing all building tools, we use a single docker command, where the environment is prepared and ready to use.
- Next? Multi-container applications, for that we use Docker Compose.
- Writing a Docker Compose file means that we are writing a definition for the multiple services we are going to use.
- Docker Compose file is written using YAML (YAML Ain't Markup Language). and has '.yml'/'.yaml' extension.
- YAML, originally was the abbreviation for 'Yet Another Markup Language'. However, that was changed later to distinguish its purpose as data-oriented, rather than document markup.
- Quick guide to start with YAML: https://rollout.io/blog/yaml-tutorial-everything-you-need-get-started/
- Docker Compose file example:

```yaml
version: "3.8"
services:
  web:
    build: .
    ports:
      - "5000:5000"
  redis:
    image: "redis:alpine"
```

- This example is from the official docker documentation, let's go through it.
  - The version depends on the current docker version being used.
  - Two services defined in this file,
    - the `web` uses the already built image (`build: .`) and binds the container and the host to some port.
    - `redis`: uses an image that is going to be pulled from the Docker Hub registry.