Project one

Short description

Lab focus: This project covers Docker, docker-compose, and Kubernetes.
Submission deadline: 23:59 20 October,
Credits: 12%. For every day you delay the submission you will lose 0.5%
Submission guidelines:

- Create a folder with name `PROJECT1_FirstName_LastName_StudentID` and inside this folder create a folder named *exY* for each exercise `Y`. Also, for each exercise submit the source code you created. Note that, we need to be able to run your code and reproduce your results.
- You need to submit a report in **PDF** format (Note: other types of format will not be accepted). Create only one report for all exercises.
- The report should include a description for each exercise you have attempted to solve. You need to include a description of how your results will be re-produced and show how you executed each step as well as a description of the results.
- For each exercise record a video that demonstrates your solution. *You will be evaluated only for the tasks that are presented in the video and report.* You can use Zoom to record the videos but you are free to use any other tool you are familiar with. The video format should be mp4.
- After finishing your work, zip the main folder and submit your work via Brightspace.

Exercise One - Images & Containers (2%)

Task 1 (0.5 %): Using docker commands create a volume named `ex1_vol`.

Task 2 (0.5 %): With docker run start an `ubuntu:22.04` container named `sender`. Additionally, mount to the container the `\data` to the volume we created and make sure to start your container in interactive mode.

Task 3 (1 %): Inside the container go to `\data` directory and create a file with content of your selection (e.g., a txt file with your name). Since this directory is mounted to a volume, its content will remain even after when your container is killed. Exit from the `sender` container with the `exit` command (Note this way you also kill the container). Then, start another container named `receiver` using the same image in interactive mode, as well. However, for this container when mounting the volume give only read permissions. Inside the container read the content of the message (i.e the file you created) in the `sender` container.

NOTE! In the report describe step-by-step the commands you run and explain the purpose of each step. Demonstrate all tasks in a single video.

Exercise Two - `Dockerfile` (3%)

Task 1 (2%): Create a Dockerfile with the following attributes:

1. Use as base image `ubuntu`
2. Copy the install.sh file in the '/' directory of the container.
3. Add executable permissions in the install.sh file.
4. use RUN to execute the install.sh script file.

Task 2 (1%): Finally, for this image/container we want to run the shell application (sh), which means that when you will run your container it will immediately attach your terminal to the container's terminal. Demonstrate the two ways we used in the practicals to do that. Show both solutions, describe them in your report and discuss in which cases each solution is more suitable.

HINT ;) install.sh can be found in the given ex2 folder.

Exercise Three - `docker-compose` - Dockerize a Web application (4%)

In this exercise, you will deploy an application that is composed of three components: a php apache server, a `mysql` database, and a `php admin`. Thus, you need to define three containerised services (i.e, php apache server container, mysql database, and php admin container) and connect them accordingly using the code inside the `docker-compose` file. Inside file *ex3*, you will find a code "skeleton". This will give you a guideline on how you should complete this exercise. Note that the given code is not completed. Update the docker-compose file following the below steps. Note that the the service regarding the php apache server (i.e, named php-apache-environment) is already defined. You have to complete the code accordingly for the remaining services.

Task 1 (2%): Define the `mysql` server. Complete the code that corresponds to the `db` service in the skeleton code. This service should have the following characteristics:

- The container is created using the mysql image (from public repo)
- Define the restart policy as always
- Map the ports "9906:3306"
- Set the following variables:

```
MYSQL_ROOT_PASSWORD: MYSQL_ROOT_PASSWORD
MYSQL_DATABASE: MYSQL_DATABASE
MYSQL_USER: MYSQL_USER
MYSQL_PASSWORD: MYSQL_PASSWORD
```

Task 2 (2%): Update the code for the service regarding the php admin (service name: `phpmyadmin`) with the following characteristics:

- Use the phpmyadmin/phpmyadmin image

- Define the restart policy as always
- Define an export port for the host machine (note it has to be different form the one of the php server)
- Add a dependency on the database image
- Set the following environment variables:
  `PMA_HOST: db`

HINT ;) When you have completed the compose file and started the app, you will see that a folder `php/src` is created. Manually copy to this folder the php source codes that are included in the ex3 folder. Visit `http://localhost:8000` to view the website. Enter as a database admit through `http://localhost:<xx>`, where `<xx>` is the port you have selected to expose the php admin container. Then, write as username `root` and the password is `MYSQL_ROOT_PASSWORD`.

Exercise Four - Kubernetes - scale-up your app (3%)

First step: Start a new Minikube cluster with two nodes, using the following command:

```
minikube start --nodes 2
```

Make sure that you have deleted any previous Minikube images/containers. If the Minikube does not start with two nodes, try to restart you machine. You can view the available nodes using the following command:

```
kubectl get nodes
```

Wait until all nodes are ready. It may take some time...

Task 1 (1.5%): Inside folder `ex4`, there is a Kubernetes configuration file named "hello–deployment.yaml". Describe what the deployment does and specifically what is the purpose of the following fields:

1. `kind`
2. `spec.replicas`
3. `spec.strategy`
4. `spec.template.spec.affinity`
5. `spec.template.spec.containers`

Task 3 (1.5%): Make a new deployment to Kubernetes using the aforementioned configuration file. Then, use the following command to view the running pods:

```
kubectl get pods -o wide
```

You will see that the two pods are running into two different nodes to balance the load. We want to change that and assign the two pods to one specific node.
To do so follow the following steps:

(a) Delete the previous deployment and make sure no other pod is running.

(b) You should associate the preferred node (i.e, the one that we will assign both pods) with a label. You can do so using the command:

```
kubectl label nodes <your-node-name> disktype=<new-label>
```

(c) View the new label using the command:

```
kubectl get nodes --show-labels
```

(d) Create a new configuration file with the name "hello–deployment_updated.yaml" Copy inside it the code from "hello–deployment.yaml", and make adjustments to associate this two pods with the node that has been received the <new-label>.

HINT ;) There is a configuration inside the hello–deployment.yaml that ensures the pods will land on separate hosts. Find this configuration and delete it.
