Assignment 1

Report

Exercise 1:

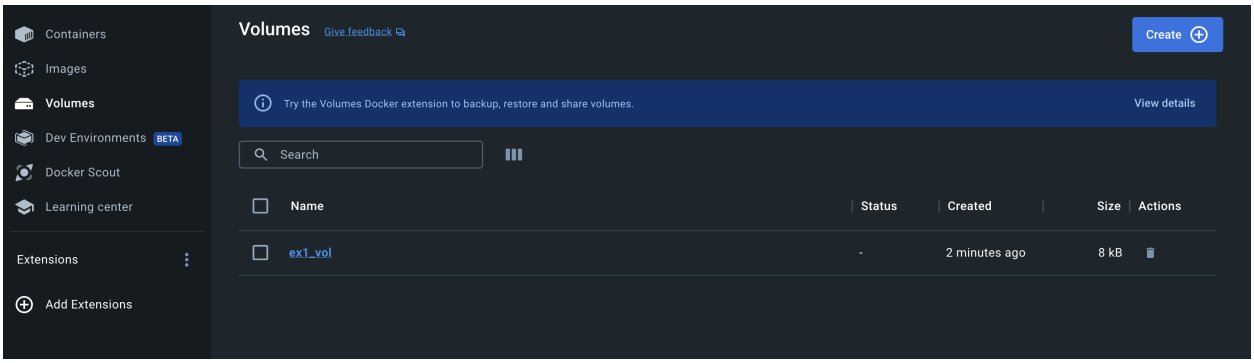
Task 1:

To create a volume in Docker named “ex1_vol”, run the following command

```
docker volume create ex1_vol
```

The output in the terminal will look like this, and we will also be able to see it in the Volume section on Docker Desktop

```
alexodonnell@Alexs-MacBook-Pro source code % docker volume create ex1_vol
ex1_vol
alexodonnell@Alexs-MacBook-Pro source code %
```



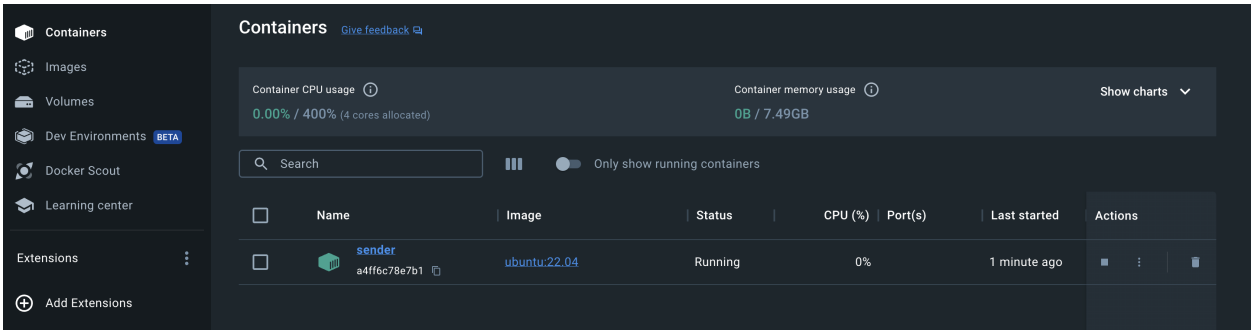
Task 2:

To start an ubuntu:22.04 container named sender that is mounted to the volume we just created, we run the following command:

```
docker run -it --name sender -v ex1_vol:/data ubuntu:22.04
```

The output will show us inside the container in the terminal, and the container will be shown running in Docker Desktop

```
[alexodonnell@Alexs-MacBook-Pro source code % docker volume create ex1_vol
ex1_vol
[alexodonnell@Alexs-MacBook-Pro source code % docker run -it --name sender -v ex1_vol:/data ubuntu:22
.04
Unable to find image 'ubuntu:22.04' locally
22.04: Pulling from library/ubuntu
bfbe77e41a78: Pull complete
Digest: sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
Status: Downloaded newer image for ubuntu:22.04
root@a4ff6c78e7b1:/#
```



Task 3:

Firstly, navigate to the data directory of the Volume:

```
[root@a4ff6c78e7b1:/# ls
bin   data  etc   lib    mnt  proc  run   srv  tmp  var
boot  dev   home  media  opt  root  sbin  sys  usr
[root@a4ff6c78e7b1:/# cd data
[root@a4ff6c78e7b1:/data# ls
root@a4ff6c78e7b1:/data#
```

To create a text file with my name as the contents, we run the following command:

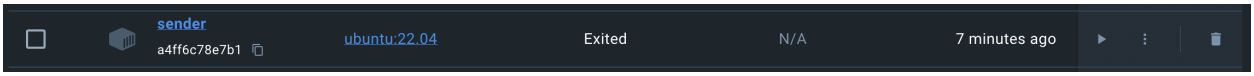
`echo "Alex O'Donnell" > alex_odonnell.txt`

```
[root@a4ff6c78e7b1:/data# echo "Alex O'Donnell" > alex_odonnell.txt
[root@a4ff6c78e7b1:/data# ls
alex_odonnell.txt
root@a4ff6c78e7b1:/data#
```

As we can see, alex_donnell.txt is now in the data directory of the container. We can now exit using the exit command.

`exit`

This also kills the container, which we can see in Docker Desktop that it is “Exited”



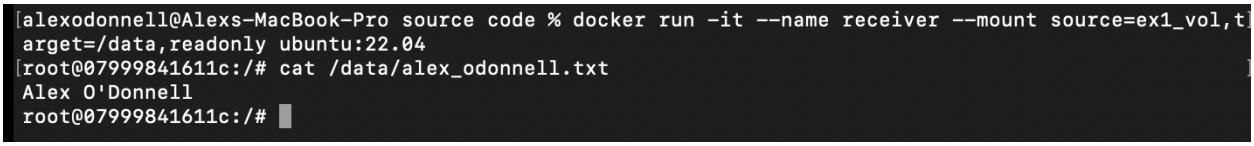
Now to start another container named “receiver” using the same image as before in interactive mode, mounted to the same volume with readonly permissions, we fun the following command:

```
docker run -it --name receiver --mount source=ex1_vol,target=/data,readonly ubuntu:22.04
```

Once inside, we can read the contents of the txt file we already created within the data directory of the container with the following command:

```
cat /data/alex_odonnell.txt
```

Running these two commands will produce the contents of the txt file within the data folder, which is just my name:



Exercise 2:

Task 1:

In order to 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

We need to create a Dockerfile with the following contents:

```
FROM ubuntu
COPY install.sh /install.sh
```

```
RUN chmod +x install.sh
RUN /install.sh
```

Task 2:

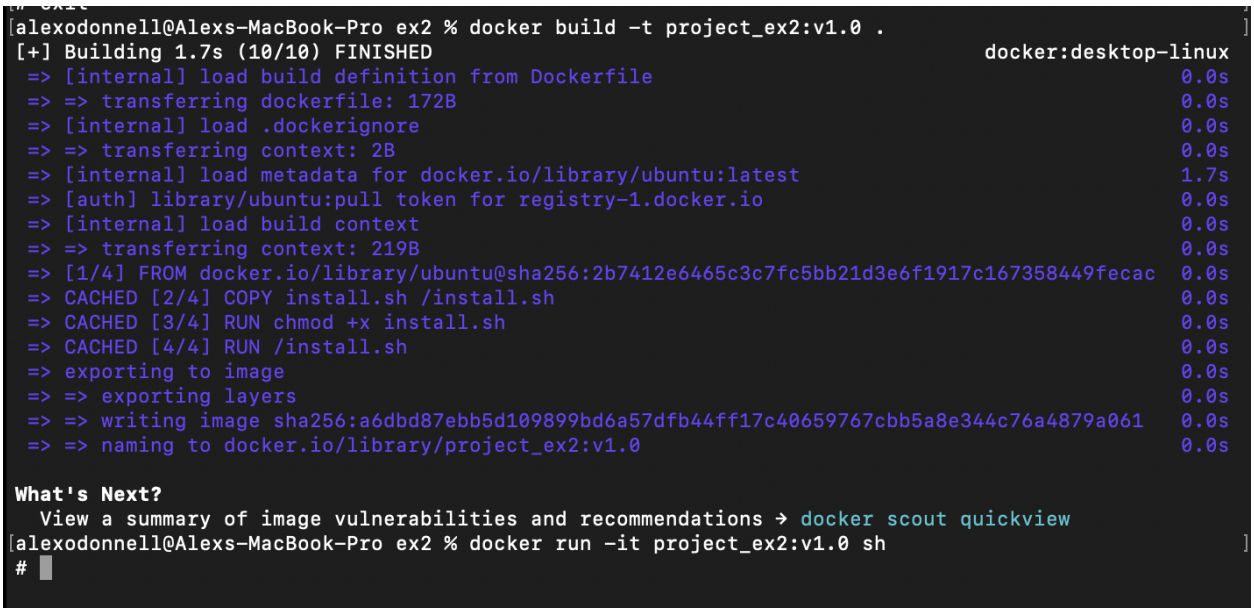
We want to run the shell application (sh), which means that when you will run your container it will immediately attach your terminal to the container's terminal. There are two methods to achieve this:

Method 1:

Specifying `sh` in the `docker run` command to attach the terminal:

```
docker build -t project_ex2:v1.0 .
docker run -it project_ex2:v1.0 sh
```

This method results in the following output:



As you can see, we have entered the shell inside the container. There are some pros to using this method, such as:

1) We can modify or inspect the image before running it because there is clear separation between building and running the container

2) We can choose different commands to run when starting the container which could be useful for testing and debugging

Method 2:

Altering our Dockerfile to contain the following code, so that it is specified to run /bin/sh when the container starts:

```
FROM ubuntu
COPY install.sh /install.sh
RUN chmod +x install.sh
RUN /install.sh
CMD ["/bin/sh"]
```

And run the following commands:

`docker build -t project_ex2:v1.0 .`

`docker run -it project_ex2:v1.0`

As you can see, it produces the same result as method 1:

```
alexodonnell@Alexs-MacBook-Pro ex2 % docker run -it project_ex2:v1.0
Unable to find image 'project_ex2:v1.0' locally
docker: Error response from daemon: pull access denied for project_ex2, repository does not exist or
 may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
alexodonnell@Alexs-MacBook-Pro ex2 % docker build -t project_ex2:v1.0 .
[+] Building 1.6s (10/10) FINISHED                                  docker:desktop-linux
 => [internal] load .dockerignore                                                  0.0s
 => => transferring context: 2B                                                    0.0s
 => [internal] load build definition from Dockerfile                               0.0s
 => => transferring dockerfile: 172B                                               0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                   1.5s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                      0.0s
 => [internal] load build context                                                  0.0s
 => => transferring context: 219B                                                  0.0s
 => [1/4] FROM docker.io/library/ubuntu@sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac  0.0s
 => CACHED [2/4] COPY install.sh /install.sh                                       0.0s
 => CACHED [3/4] RUN chmod +x install.sh                                           0.0s
 => CACHED [4/4] RUN /install.sh                                                   0.0s
 => exporting to image                                                             0.0s
 => => exporting layers                                                            0.0s
 => => writing image sha256:a6dbd87ebb5d109899bd6a57dfb44ff17c40659767cbb5a8e344c76a4879a061   0.0s
 => => naming to docker.io/library/project_ex2:v1.0                                0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
alexodonnell@Alexs-MacBook-Pro ex2 % docker run -it project_ex2:v1.0
# 
```

There are also some prod to using this method too, such as:

1) Specifying the default command within the Dockerfile simplifies the `docker run` command

2) It provides consistency because every time the container is run, it runs the set command, eliminating the risk of human error

Exercise 3:

Task 1&2:

In order to deploy an application that is composed of three components: a php apache server, a mysql database, and a php admin. We 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.

Firstly, we need to define the mysql server with 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

We also need to 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`

This is what our Docker-compose file should look like:

```
version: '3.9'
services:
    php-apache-environment:
        container_name: php-apache
        build:
            context: .
            dockerfile: Dockerfile
        depends_on:
            - db
        volumes:
            - ./php/src:/var/www/html/
        ports:
            - 8000:80
    db:
        image: mysql:latest
```

```
        container_name: db
        restart: always
        ports:
            - "9906:3306"
        environment:
            MYSQL_ROOT_PASSWORD: MYSQL_ROOT_PASSWORD
            MYSQL_DATABASE: MYSQL_DATABASE
            MYSQL_USER: MYSQL_USER
            MYSQL_PASSWORD: MYSQL_PASSWORD
    phpmyadmin:
        container_name: phpmyadmin
        image: phpmyadmin/phpmyadmin
        restart: always
        ports:
            - "8080:80"
        depends_on:
            - db
        environment:
            PMA_HOST: db
```

We now need to start the app:

Navigate to directory where exercise folder is located:

`docker build -t project_ex3:v1.0 .`

`docker-compose up`

This will produce the following output in the terminal:

```
alexodonnell@Alexs-MacBook-Pro ex3 % docker build -t project_ex3:v1.0 .
[+] Building 2.1s (8/8) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 307B                                              0.0s
 => [internal] load .dockerignore                                                 0.0s
 => => transferring context: 2B                                                   0.0s
 => [internal] load metadata for docker.io/library/php:8.0-apache                 2.1s
 => [auth] library/php:pull token for registry-1.docker.io                        0.0s
 => [1/3] FROM docker.io/library/php:8.0-apache@sha256:61b257bd20a9ab5a19f24cb2f8610595183b31  0.0s
 => => resolve docker.io/library/php:8.0-apache@sha256:61b257bd20a9ab5a19f24cb2f8610595183b31  0.0s
 => CACHED [2/3] RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli  0.0s
 => CACHED [3/3] RUN apt-get update && apt-get upgrade -y                         0.0s
 => exporting to image                                                            0.0s
 => => exporting layers                                                           0.0s
 => => writing image sha256:4fb612b2e1cbabff2ebf7cc7ff256d2719dc1f7bee4bda11af949065f0702060   0.0s
 => => naming to docker.io/library/project_ex3:v1.0                               0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
alexodonnell@Alexs-MacBook-Pro ex3 % docker-compose up
[+] Running 30/30
 ✔ phpmyadmin 18 layers [####################]      0B/0B      Pulled                11.5s
   ✔ faef57eae888 Pull complete                                                    4.4s
   ✔ 989a1d6c052e Pull complete                                                    0.5s
   ✔ 0705c9c2f22d Pull complete                                                    5.8s
   ✔ 621478e043ce Pull complete                                                    1.3s
   ✔ 98246dcca987 Pull complete                                                    5.0s
   ✔ bfed8c155cb6 Pull complete                                                    4.9s
   ✔ 7a7c2e908867 Pull complete                                                    5.4s
   ✔ d176994b625c Pull complete                                                    6.2s
   ✔ 2d8ace6a2716 Pull complete                                                    6.1s
   ✔ c70df516383c Pull complete                                                    6.8s
   ✔ 15e1b44fe4c7 Pull complete                                                    6.7s
   ✔ 65e50d44e95a Pull complete                                                    6.7s
   ✔ 77f68910bc0a Pull complete                                                    7.2s
   ✔ 605dd3a6e332 Pull complete                                                    7.4s
   ✔ 99ce27188f07 Pull complete                                                    7.3s
   ✔ 74d64e32c5d5 Pull complete                                                    8.2s
   ✔ ef5fc9928b9f Pull complete                                                    7.8s
   ✔ 163f3256e112 Pull complete                                                    8.0s
 ✔ db 10 layers [##########]      0B/0B      Pulled                               21.2s
   ✔ 89ec84aa94fe Pull complete                                                    9.6s
   ✔ 2047d0a85fcf Pull complete                                                    8.3s
   ✔ a981666a88bb Pull complete                                                    8.6s
   ✔ 7aa9dbd75443 Pull complete                                                    9.5s
   ✔ b8ed90386e32 Pull complete                                                    9.5s
   ✔ 0b36981941b7 Pull complete                                                   10.0s
   ✔ ca996c23fc53 Pull complete                                                   14.7s
   ✔ 15050e9043e6 Pull complete                                                   10.2s
   ✔ 3ecfc97ced27 Pull complete                                                   14.0s
   ✔ e435e3bf4d77 Pull complete                                                   11.1s
[+] Building 0.7s (7/7) FINISHED                                docker:desktop-linux
 => [php-apache-environment internal] load build definition from Dockerfile       0.0s
 => => transferring dockerfile: 307B                                              0.0s
 => [php-apache-environment internal] load .dockerignore                          0.0s
 => => transferring context: 2B                                                   0.0s
 => [php-apache-environment internal] load metadata for docker.io/library/php:8.0-apache  0.6s
 => [php-apache-environment 1/3] FROM docker.io/library/php:8.0-apache@sha256:61b257bd20a9ab5  0.0s
 => => resolve docker.io/library/php:8.0-apache@sha256:61b257bd20a9ab5a19f24cb2f8610595183b31  0.0s
 => CACHED [php-apache-environment 2/3] RUN docker-php-ext-install mysqli && docker-php-ext-e  0.0s
 => CACHED [php-apache-environment 3/3] RUN apt-get update && apt-get upgrade -y  0.0s
 => [php-apache-environment] exporting to image                                   0.0s
```

```
[+] Running 5/2
 ✔ Network ex3_default                                                    Created0.2s                                                    0.2s
 ✔ Container db                                                           Created0.2s
 ✔ Container phpmyadmin                                                   Created0.0s
 ✔ Container php-apache                                                   Created0.0s
 ! phpmyadmin The requested image's platform (linux/amd64) does not match the detected host platform
 (linux/arm64/v8) and no specific platform was requested 0.0s
Attaching to db, php-apache, phpmyadmin
db           | 2023-10-19 17:30:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.
0-1.el8 started.
db           | 2023-10-19 17:30:54+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
db           | 2023-10-19 17:30:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.
0-1.el8 started.
db           | 2023-10-19 17:30:54+00:00 [Note] [Entrypoint]: Initializing database files
db           | 2023-10-19T17:30:54.308458Z 0 [System] [MY-015017] [Server] MySQL Server Initializatio
n - start.
db           | 2023-10-19T17:30:54.309223Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-c
ache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0
 instead.
db           | 2023-10-19T17:30:54.309277Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8
.1.0) initializing of server in progress as process 79
db           | 2023-10-19T17:30:54.313504Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has
started.
php-apache   | AH00558: apache2: Could not reliably determine the server's fully qualified domain nam
e, using 172.18.0.4. Set the 'ServerName' directive globally to suppress this message
php-apache   | AH00558: apache2: Could not reliably determine the server's fully qualified domain nam
e, using 172.18.0.4. Set the 'ServerName' directive globally to suppress this message
php-apache   | [Thu Oct 19 17:30:54.393372 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.56
(Debian) PHP/8.0.30 configured -- resuming normal operations
php-apache   | [Thu Oct 19 17:30:54.393581 2023] [core:notice] [pid 1] AH00094: Command line: 'apache
2 -D FOREGROUND'
db           | 2023-10-19T17:30:54.496916Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has
ended.
phpmyadmin   | AH00558: apache2: Could not reliably determine the server's fully qualified domain nam
e, using 172.18.0.3. Set the 'ServerName' directive globally to suppress this message
phpmyadmin   | AH00558: apache2: Could not reliably determine the server's fully qualified domain nam
e, using 172.18.0.3. Set the 'ServerName' directive globally to suppress this message
phpmyadmin   | [Thu Oct 19 17:30:54.709905 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57
(Debian) PHP/8.2.8 configured -- resuming normal operations
phpmyadmin   | [Thu Oct 19 17:30:54.712085 2023] [core:notice] [pid 1] AH00094: Command line: 'apache
2 -D FOREGROUND'
db           | 2023-10-19T17:30:55.004502Z 6 [Warning] [MY-010453] [Server] root@localhost is created
 with an empty password ! Please consider switching off the --initialize-insecure option.
db           | 2023-10-19T17:30:56.561855Z 0 [System] [MY-015018] [Server] MySQL Server Initializatio
n - end.
db           | 2023-10-19 17:30:56+00:00 [Note] [Entrypoint]: Database files initialized
db           | 2023-10-19 17:30:56+00:00 [Note] [Entrypoint]: Starting temporary server
db           | 2023-10-19T17:30:56.601301Z 0 [System] [MY-015015] [Server] MySQL Server - start.
db           | 2023-10-19T17:30:56.806135Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-c
ache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0
 instead.
db           | 2023-10-19T17:30:56.807794Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8
.1.0) starting as process 123
db           | 2023-10-19T17:30:56.817807Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has
started.
db           | 2023-10-19T17:30:56.953412Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has
ended.
db           | 2023-10-19T17:30:57.100581Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is
```

We can then input the following credentials:

Connected to MySQL server successfully!

Table users created successfully

ID: 20456522

Firstname: root

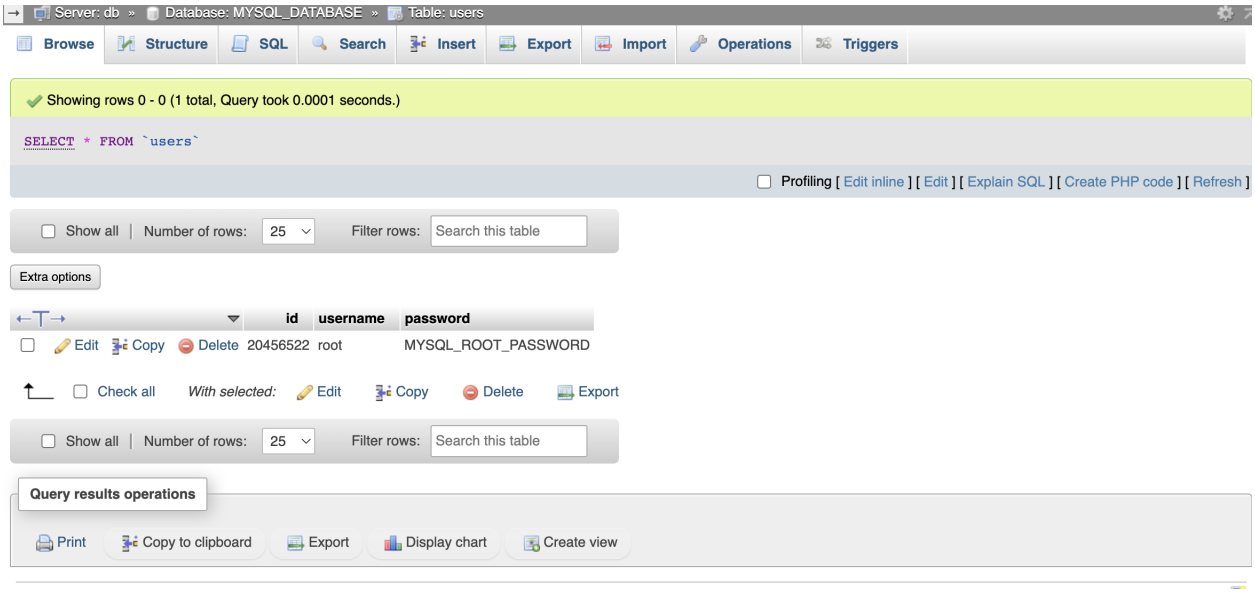
password: MYSQL_ROOT_PASSV

Submit

1 record added

TABLE DATA: id name pass
20456522 root MYSQL_ROOT_PASSWORD
Go back to insert more data

We can see that everything is operating as expected if we visit our database as a database admit through `http://localhost:8080:80` using the credentials we specified in our Docker-compose file



Exercise 4:

The command `minikube start --nodes 2` provided in the Exercise brief did not work for me, perhaps because I am using MACOS. The command I used to start two Minikube cluster with two nodes is:

```
minikube start --nodes 2 -p multinode-demo2
```

Which could be confirmed by the command `kubectl get nodes`

The output is as follows:

```
alexodonnell@Alexs-MacBook-Pro ex4 % minikube start --nodes 2 -p multinode-demo2
😄  [multinode-demo2] minikube v1.31.2 on Darwin 13.4.1 (arm64)
✨  Using the docker driver based on existing profile
👍  Starting control plane node multinode-demo2 in cluster multinode-demo2
🚜  Pulling base image ...
🤷  docker "multinode-demo2" container is missing, will recreate.
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring CNI (Container Networking Interface) ...
🌟  Enabled addons:
🔎  Verifying Kubernetes components...
❗  The cluster multinode-demo2 already exists which means the --nodes parameter will be ignored.
 Use "minikube node add" to add nodes to an existing cluster.
👍  Starting worker node multinode-demo2-m02 in cluster multinode-demo2
🚜  Pulling base image ...
🤷  docker "multinode-demo2-m02" container is missing, will recreate.
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🌐  Found network options:
    ▪ NO_PROXY=192.168.67.2
🐳  Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
    ▪ env NO_PROXY=192.168.67.2
E1019 19:01:35.775983   11336 node.go:121] unable to delete node "m02": nodes "multinode-demo2-m0
2" not found
E1019 19:01:35.776032   11336 start.go:316] error removing existing worker node before rejoining
cluster, will continue anyway: nodes "multinode-demo2-m02" not found
🔎  Verifying Kubernetes components...
🏄  Done! kubectl is now configured to use "multinode-demo2" cluster and "default" namespace by d
efault
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl get nodes
NAME                STATUS   ROLES           AGE   VERSION
multinode-demo2     Ready    control-plane   50s   v1.27.4
multinode-demo2-m02   Ready    <none>          18s   v1.27.4
alexodonnell@Alexs-MacBook-Pro ex4 %
```

Task 1:

`kind` specifies the type of resource, and in this case, it's set to "Deployment." A Deployment is used to manage and update a set of replicated pods in a declarative manner. It ensures that a specified number of pod replicas are running and facilitates updates or rollbacks

`spec.replicas` defines the desired number of replicas for the pods managed by this Deployment. In this case, it's set to 2, which means that the Deployment should maintain two running pods with the specified configuration

`spec.strategy` defines the update strategy for the Deployment. In this YAML, the strategy is set to "RollingUpdate." Rolling updates are performed in a way that allows a gradual replacement of old pods with new ones to minimise disruptions during updates

`spec.template.spec.affinity` scap.template is part of the pod template specification within the Deployment. Inside spec.template, .affinity is used to define rules for how pods should be scheduled and distributed across nodes in the cluster. Specifically, it configures "pod anti-affinity," which means that it enforces that pods with the label app.hello (defined as selector) should not be scheduled on the same node. This is specified by using the podAntiAffinity field with requiredDuringSchedulingIgnoredDuringExecution, which enforces this anti-affinity rule during pod scheduling

`spec.template.spec.containers` spec.template also contains the containers field, which defines the pod's containers. In this case, there is a single container named "hello-from." It specifies the Docker image (pbitty/hello-from:latest) to run, exposes port 80 for HTTP traffic, and sets a termination grace period of 1 second. This container is part of the pods created by this Deployment.

(No Task 2)

Task 3:

We now need to make a new deployment to Kubernetes using the aforementioned configuration file, which can be done by running the following command:

```
kubectl apply -f hello-deployment.yaml
```

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl get nodes
NAME                 STATUS   ROLES           AGE   VERSION
multinode-demo2      Ready    control-plane   50s   v1.27.4
multinode-demo2-m02  Ready    <none>          18s   v1.27.4
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl apply -f hello-deployment.yaml
deployment.apps/hello created
alexodonnell@Alexs-MacBook-Pro ex4 %
```

And then we can view the running pods by running:

`kubectl get pods -o wide`

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl apply -f hello-deployment.yaml
deployment.apps/hello created
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE                 NOMINATED NODE   READINESS GATES
hello-77c947d946-6ngxr   1/1     Running   0          72s   10.244.1.2   multinode-demo2-m02  <none>           <none>
hello-77c947d946-wfnt4   1/1     Running   0          72s   10.244.0.3   multinode-demo2      <none>           <none>
alexodonnell@Alexs-MacBook-Pro ex4 %
```

(a) Delete the previous deployment and make sure no other pod is running, by running the command:

`kubectl delete deployment hello`

This yields the following output:

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl delete deployment hello
deployment.apps "hello" deleted
alexodonnell@Alexs-MacBook-Pro ex4 %
```

We can confirm that the previous deployment has indeed been deleted by running the previous command:

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl delete deployment hello
deployment.apps "hello" deleted
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl get pods -o wide
No resources found in default namespace.
alexodonnell@Alexs-MacBook-Pro ex4 %
```

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

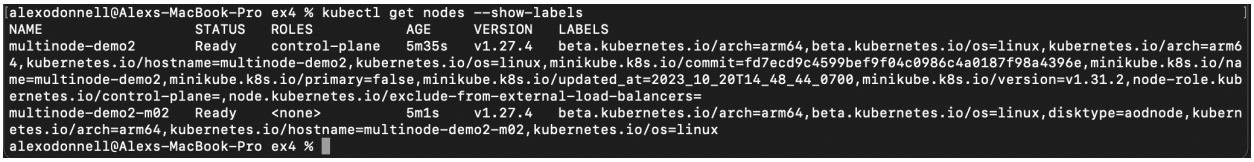
`kubectl label nodes multinode-demo2-m02 disktype=aodnode`

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl label nodes multinode-demo2-m02 disktype=aodnode
node/multinode-demo2-m02 labeled
alexodonnell@Alexs-MacBook-Pro ex4 %
```

(c) We can then confirm this by viewing the new label by running the command:

`kubectl get nodes --show-labels`

Which produces the output:



(d) To create a new configuration file with the name "hello–deployment_updated.yaml", we need to 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

The contents of hello–deployment_updated.yaml should look like this:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 100%
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: disktype
                  operator: In
                  values:
```

```
                - aodnode
      containers:
      - name: hello-from
        image: pbitty/hello-from:latest
        ports:
          - name: http
            containerPort: 80
      terminationGracePeriodSeconds: 1
```

Now with the altered code, we can run the previous commands again except with the new YAML file, so the chain of commands would be:

`kubectl apply -f hello-deployment_updated.yaml`

`kubectl get pods -o wide`

```
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl apply -f hello-deployment_updated.yaml
deployment.apps/hello created
alexodonnell@Alexs-MacBook-Pro ex4 % kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE                   NOMINATED NODE   READINESS GATES
hello-b899fd44b-frmzw    1/1     Running   0          13s   10.244.1.4   multinode-demo2-m02    <none>           <none>
hello-b899fd44b-rtbgm    1/1     Running   0          13s   10.244.1.3   multinode-demo2-m02    <none>           <none>
alexodonnell@Alexs-MacBook-Pro ex4 %
```

So now we can see, both pods are running on one node `multinode-demo2-mo2` as expected.