

Exercises to OD-01

Operating Systems and Distributed Systems

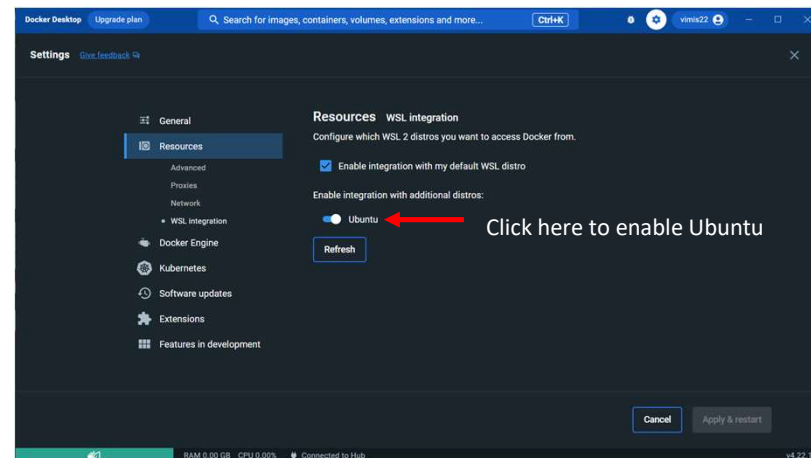
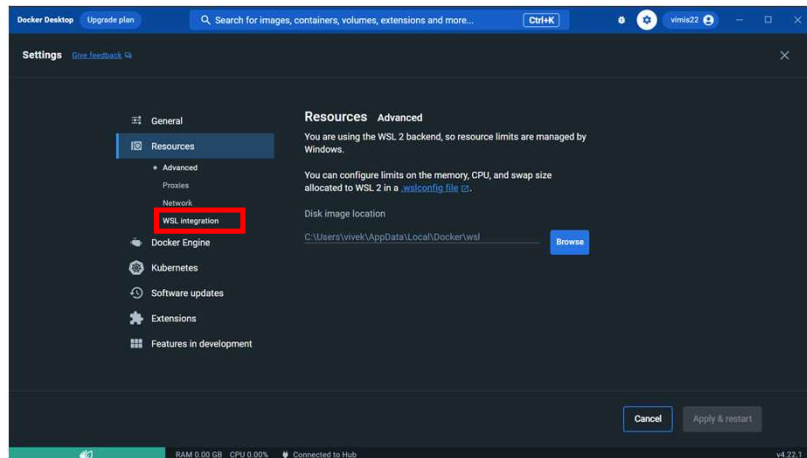
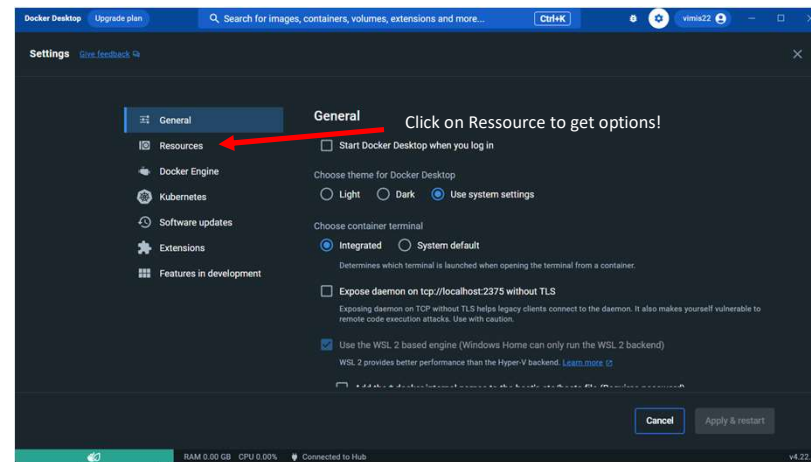
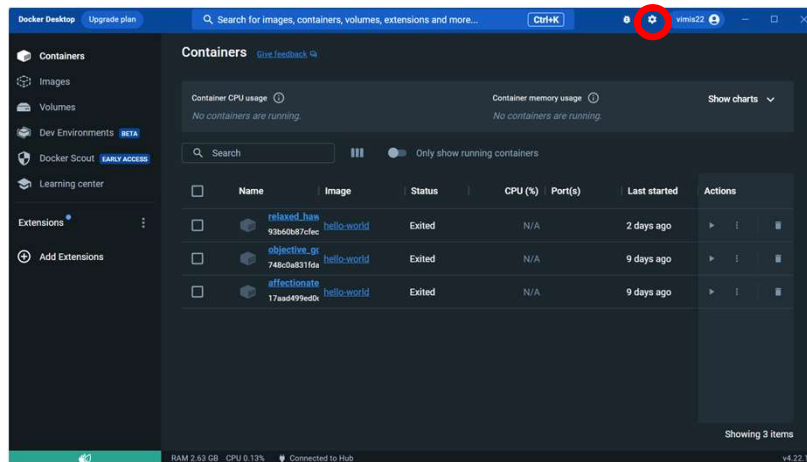
What is the purpose?

- The main purpose of this præsentation is to show solutions to all the tasks made in the Exercise-hours, so that the student is able to solve the questions during the final exams!
- The Lecture and Exercise-Hours take place on Tuesdays!

Task 1 and Task 2

- We need to install Linux on either a Virtual Box or a Virtual Machine.
 - That is actually done, and can be done on Oracle Virtual Box where Ubuntu can be included.
 - It is also possible to install linux on the PowerShell Terminal by simply writing: `wsl –install`.
- We need to install Docker Desktop and it is possible through the following link: [Docker Desktop: The #1 Containerization Tool for Developers | Docker](#)
 - Remind you that if we are using powershell with linux on windows, we need to make sure that our Ubuntu is connected with wsl. (Look at the next slide).

Task 1 and Task 2 – Docker Desktop



Task 3, Part 1

- In this task we need to print the big text from "hello-world".
 - We need to write the following code in a VS-Code Studio Docker File.

FROM Ubuntu

RUN echo "Some text" > helloWorld

CMD cat helloWorld

- Look at the next page, and see code written in the terminal!

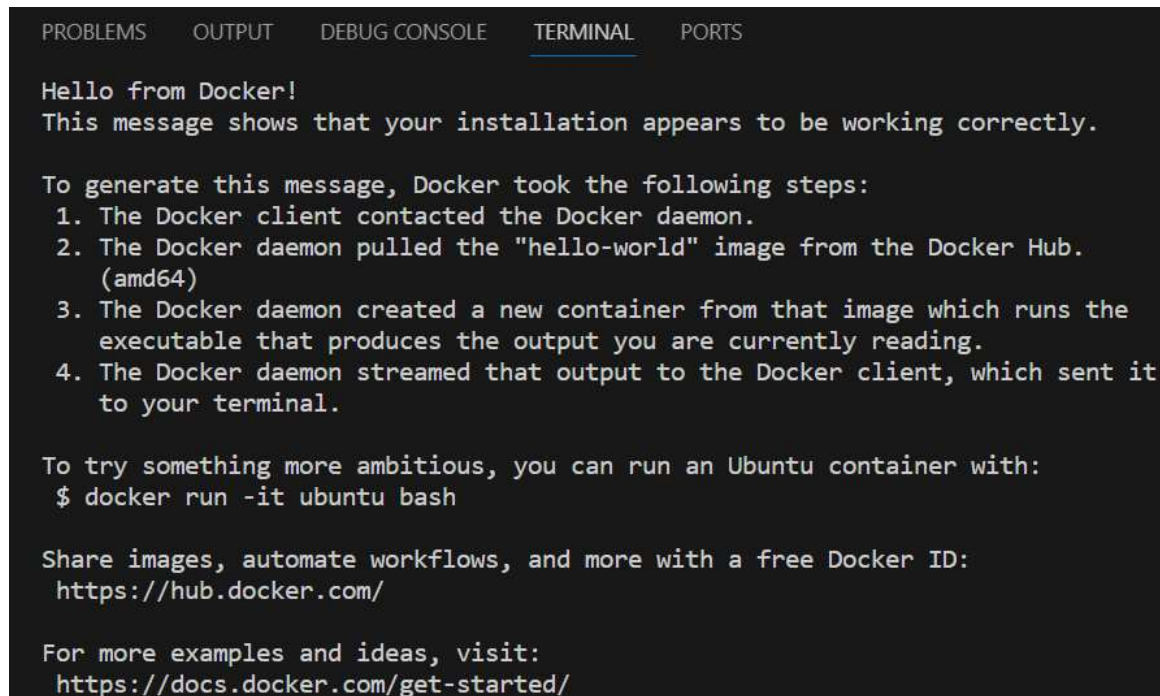
Task 3, Part 1 – Terminal Commands

- First, we need to find the file in which we are working with.
 - I have used the command `cd "directory_path"` to find the location of the VS-Code Dockerfile.

```
PS C:\Users\vivek> cd "OneDrive - Syddansk Universitet"
PS C:\Users\vivek\OneDrive - Syddansk Universitet> cd "docker_projects"
PS C:\Users\vivek\OneDrive - Syddansk Universitet\docker_projects> cd "lesson1_task3_dockerfile"
```

Lesson 1: Task 3, Part 1 – Terminal Commands

- Now we will write a command, which we print Hello-World.
 - It is important to understand that when we work with Dockerfiles, we are converting from files to image to execution. So when we run the command we are running the document.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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/
```

Task 4, Part 2

- In the previous part, we had runned a dockerfile from image to execution. Now it is important to emphasize, that we are taking one step back and building an docker image.
 - COMMAND: `docker build dockerfile .`

```
vimis22@VivekMisraPC:/mnt/c/Users/vivek/OneDrive - Syddansk Universitet/docker_projects/lesson1_task3_dockerfile$ docker build -t dockerfile .
[+] Building 0.2s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile            0.1s
=> => transferring dockerfile: 107B                            0.0s
=> [internal] load .dockerignore                              0.1s
=> => transferring context: 2B                                  0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
=> [1/2] FROM docker.io/library/ubuntu                        0.0s
=> CACHED [2/2] RUN echo "Some text" > helloWorld            0.0s
=> exporting to image                                         0.0s
=> => exporting layers                                         0.0s
=> => writing image sha256:0003f79d070166adbccd132fd624e14f7c79a7905a105888fc0ca4bac5164ff0 0.0s
=> => naming to docker.io/library/dockerfile                  0.0s
```

What's Next?

View summary of image vulnerabilities and recommendations → [docker scout quickview](#)

Task 4, Part 2

- Here we have written the command, so that we are able to change the name of the dockerfile.





```
vimis22@VivekMisraPC:/mnt/c/Users/vivek/OneDrive - Syddansk Universitet/docker_projects/lesson1_task3_dockerfile$ docker build -f dockerfile .
[+] Building 0.1s (2/2) FINISHED                                docker:default
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [internal] load build definition from dockerfile             0.0s
=> => transferring dockerfile: 2B                                0.0s
ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount2135120214/dockerfile: no such file or directory
```

Task 5

- First we need to write which file we are extracting data from.
- Then we need to copy the code where the data is taken from.
- Then we will connect to a port-number from the localhost.

```
1 FROM php:7.2-apache
2
3 COPY ./L1E5 /var/www/html
4
5 RUN echo 'ServerName localhost' >> /etc/apache2/apache2.conf
6
```

- The following image shows the arrangement of the files in accordance with the filesystem meant by the task.

 L1E5		16-09-2023 18:27	Filmappe	
 Dockerfile		18-09-2023 20:45	Fil	1 KB

Task 5 – Terminal Commands

- Here we have the commands for Ubuntu WSL Terminal.
 - We have runned the tests and have been successful.
 - Though our execution results have vanished, and therefore we cannot show the results!

```
8  #The following should be written in the Ubuntu(WSL) Terminal!
9  #1.) sudo docker build -t lesson1exercise5
10 #2.) sudo docker run lesson1exercise5
11 #3.) sudo docker run -p 8080:80 lesson1exercise5
12 #4.) http://localhost:8080
13
14 #Note that step 4.), needs to be written inside the link-bar in edge, chrome etc.
```

- Results in WebPage: Hello World

Task 6 – Terminal Commands

- First we will write a code in Python that can count up to 10, while printing the results in the terminal!

```
1 import time
2
3 # We have a number which is being counted up to.
4 num_counts = 10
5
6 # Here we are looping through the counts!
7 for count in range(1, num_counts + 1):
8     print(f"Count: {count}")
9
10     #How fast should the results be printed and be executed!
11     time.sleep(1)
12
13
```

Task 6 – Terminal Results

- Here is the following results which I have got from the Command in Thonny!

```
>>> %Run lesson1_task6_python.py  
Count: 1  
Count: 2  
Count: 3  
Count: 4  
Count: 5  
Count: 6  
Count: 7  
Count: 8  
Count: 9  
Count: 10
```

Task 7 - Defintions

1. Explain what a Dockerfile is
 - a) How does it work?
 - b) Which instructions can be used? What do they do?
2. Explain how a Docker Image is built
 - a) Which options do you know?
 - b) What are they used for?
3. Explain how a Docker container is run
 - a) Which options do you know?
 - b) What are they used for?

Task 7.1

- 7.1: According to docs.docker a Dockerfile is a text document which has all the informations and commands which a user can used to assemble an image in the Docker-World.
- 7.1.A: After the creating of a Dockerfile the user can continue on building the dockerfile through a docker-image which is a mockup for how the container should be able to look.
- 7.1.B: Instructions which can be used, can forexample be "sudo docker run <nameoffile> which runs the image and goes to the container. The "sudo docker build -t <nameoffile> makes sure that docker image is built after the files creation. And "sudo docker -rm <nameoffile> removes the containers!

Task 7.2

- 7.2: A Dockerimage is build by the command "sudo docker build -t <nameoffile> which actually builds the a dockerimage over the dockerfile.
- 7.2.A: As mentioned we know commands such as: "sudo docker build -t <nameoffile> to execute the build-process for the image.
- 7.2.B: The actual purpose of the dockerimage is to create a mockup so that we can define how the docker container is going to look!

Task 7.3

- 7.3: A Docker container is executed by commands such as COPY where we copy a directory path from the host and to another path in the container.
- 7.3.A: As mentioned we know options such as COPY, --RM, and etc
- 7.3.B: The first command is mentioned in 7.3. Where as --RM removes the container while existing.

Tak for i dag!

Operating Systems and Distributed Systems