# Linux Advanced Writeup

## 0. Foreword

In order to pass my exams I kinda have to study but its just something I just don’t do.. I’m the type of person who doesn’t care about studying and just likes to do put everything into practical use and that’s what I’ll do for this writeup of Linux Advanced.

In here I’ll write down all commands with screenshots, code and explanations in order for other people and myself to learn from it.

## 1. Introduction

In this course we will go over:

* Docker
* Inodes and linking
* Network management
* Package management
* Memory management
* Disk management
* Scheduling
* Logging
* SSH, scp, VNC

But mostly Docker

## 2. Docker

### 2.1 Docker: What is docker and installation

#### 2.1.1 What is docker?

Docker is an open source framework which makes it possible for an application to be placed in a lightweight moveable container.   
You can easily more this container between platforms as long as the Docker engine is being run on it!

Docker prevents the overheat that results in using multiple virtual machines that need their own OS, which not only spares resources bit applications work much faster due to not requiring a hypervisor!

Docker engine is a tool that works with 3 technologies:

* Namespaces
* Cgroups
* Capabilities

We will see more about this later

A docker container makes use of a kernel so Linux containers can’t work in a docker environment from Windows.

Before 1 physical machine was used for each application:

* This was an Apache server + Nginx server
* This gave problems with libraries and dll-files
* Much of the server resources was never used
* Many apps in the same OS don’t work well
* You couldn’t run Linux and Windows on the same server

After that we used Hypervisor-virtualization, multiple virtual server per physical machine:

* This made better use of the servers resources
* A lot of overheat due to having to run a separate OS for each application
* You could now run multiple Windows and Linux distributions

Now we have Container-Virtualization, an application in a runtime environment:

* Its lightweight and it uses less resources than a VM
* It uses one physical machine with only 1 OS that can run tens or more containers with their own application.

When we used virtual machines we had an infrastructure which is the server and its hardware, on top of that we had our Hypervisor, mostly Windows and Linux and inside of our hypervisor we where then running multiple virtual machines each running their own Operating system with the application on top.

Graphical user interface

Description automatically generated

With docker we have our infrastructure and on top of that our operating system, on our operating system we have docker and docker will hold all the containers with the applications inside.

Chart

Description automatically generated with medium confidence

There are different docker components to take note of:

* Docker Engine
  + Also named: Docker Daemon and Docker Runtime
  + Responsible for working with root-filesystem
* Docker Image
  + Defines OS of the container
  + run a docker image with   
    | docker run -it ubuntu /bin/bash
  + See all your images with  
    | docker images
  + Download an image  
    | docker pull <ubuntu>
* Containers
  + Gets build with downloaded images
  + Running copy of images
  + start with  
    | docker run <hello world>
* Repository
  + Images are found and found from here
  + there are standard repo’s for Ubuntu, ngrix, …
  + Different repo versions available
* Registry
  + Find docker registry at hub.docker.com
  + Repo’s are held in the registry
  + In the Ubuntu repo there are different Ubuntu versions

#### 2.1.2 Docker installation

For commands we will go through a little faster because a lot has been touched on in my Linux Essentials writeup

Go in as root  
| sudo su

Download docker (If you get a notification that hashes mismatch you will just have to try again)  
| wget -qO- https://get.docker.com | sh

Give normal users permission to use docker so you don’t have to give root access with  
| usermod -aG docker daan

when we did this we can leave root with   
| exit  
And see if docker has installed correctly with   
| docker version

We can see the rights of the user at docker.sock  
| ls -l /var/run/docker.sock

We can already download an Ubuntu dock upfront so we can use it later  
| docker pull ubuntu

But we can also run docks before we pulled them, docker will then automatically pull and run the container  
| docker run hello-world

If we now want to see all active containers we can use  
| docker ps

If we want to see all containers currently on the system we can use  
| docker ps -a

We can see all of our downloaded images with docker images  
| docker images

if we put --help behind one of these commands we can get more info  
| docker images help

### 2.2 Docker Hub & Images

#### 2.2.1 Docker Hub

Docker hub is ofcorse where the images are being downloaded.  
We can check out https://hub.docker.com/ and check out Explore

Graphical user interface, text, application, email

Description automatically generated

We can also search explore through the terminal, for example we are looking for an Ubuntu image  
| docker search ubuntu

Graphical user interface, text, application, email

Description automatically generated

We can then search for ubuntu and choose the rolling version  
In this example we will go for Ubuntu 16.04

If we now want this Ubuntu version we can go in Linux and do  
| docker pull ubuntu:16.04

#### 2.2.2 Docker images

If we done that we can see our new downloaded ubuntu image with  
| docker images

When we don’t add a version behind it will pull the latest version  
| docker pull ubuntu

We can also remove our images with  
| docker image rm ubuntu:16.04

If we want to remove all images that aren’t currently in use we use  
| docker image prune -a

We can also save an image in a tar file  
| docker save -o ubuntu.tar ubuntu

If we then remove our ubuntu image  
| image rm ubuntu  
  
We can easily load our ubuntu image again with  
| docker load -i ubuntu.tar

### 2.3 Working with containers

We can simply run ubuntu with   
| docker run ubuntu

But lets say we want to send a command directly after run  
| docker run ubuntu /bin/echo “Hello World”  
| docker run ubuntu:14.04 cat /etc/os-release  
| docker run ubuntu /bin/sleep 20

We can also detach the console from the container after running a command so the console doesn’t have to wait  
| docker run -d ubuntu /bin/sleep 20

we can also enter the bash itself with  
| docker run -it ubuntu /bin/bash  
You can use exit or CTRL + P + Q  
| exit

We can get access again to the shell with docker attach <CONTAINER ID>  
| docker attach 16e1985d16c7

we can find all active containers with  
| docker ps  
| docker container ls

Or we can ofcorse check for all containers with  
| docker ps -a  
| docker container ls -a

We can also remove a container with rm   
| docker container rm 16e1985d16c7

We can give another name so we don’t have to use the id  
| docker run --name 16e1985d16c7 PostgreSQLContainer

And ofcorse rename the container  
| docker container rm PostgreSQLContainer PostgreSQLC

We can pause a container with  
| docker container pause PostgreSQLC

And resume the container with  
| docker container unpause PostgreSQLC

Stop a container with  
| docker container stop PostgreSQLC

Start a container with  
| docker container start PostgreSQLC

Get the stats with  
| docker container stats PostgreSQLC

And logs with  
| docker container logs PostgreSQLC

We can throw al containers that aren’t running away with  
| docker container prune

To remove a specific container we can do   
| docker rm PostgreSQLC

If the container is still running you can’t instantly throw it away we can force it with  
| docker rm -f PostgreSQLC

### 2.4 Containerdata, Layers & OverlayFS

#### 2.4.1 Data stays in your container

Ofcorse when we add data to a container the data stays when we exit the console.

We can try this with he following commands  
| docker run -it ubuntu /bin/bash  
| echo Hello > /tmp/testfile  
| exit

We have now created a file inside of our Ubuntu   
If we now use   
|docker diff ubuntuContainer  
We can see the changes done to the filesystem  
Now we can start the container again with the prompt  
| docker start -i UbuntuContainer  
We can now still see that the file still exists after we restarted it  
| ls /tmp

#### 2.4.2 Images layers

A docker image is build up on multiple image layers via the  
Overlay2-filesystem, this works with union layers

Image layers are build up out of:

* Bottom layer: The rootfs from the OS
* Middle layer(s): the application
* top layer(s): updates and bugfixes

These layers are stacked on top of each other aka. Union mounts, they are transparent for the end user

During runtime when a container starts on top of the layers one more gets placed that monitors all additions and changes to the filesystem (This is the only accessible layer)

## 3. Ubuntu