# 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

## 3. Ubuntu