Microservices Architecture and Programming

**Lab Practical and date** – Practical 1, Friday 24th July 2020

**Name and Roll Number**- Het Shah, 17BIT103

**Practical Objective**- Experimenting with Containers and understanding its fundamentals

**Steps Involved-**

We perform the experiment on the Play-with-docker (PWD) website to get a hands on experience with the docker platform and concepts.

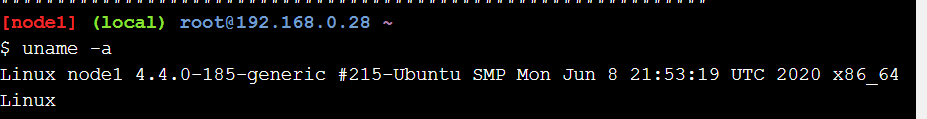
**Background**

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

**Steps/Commands-**

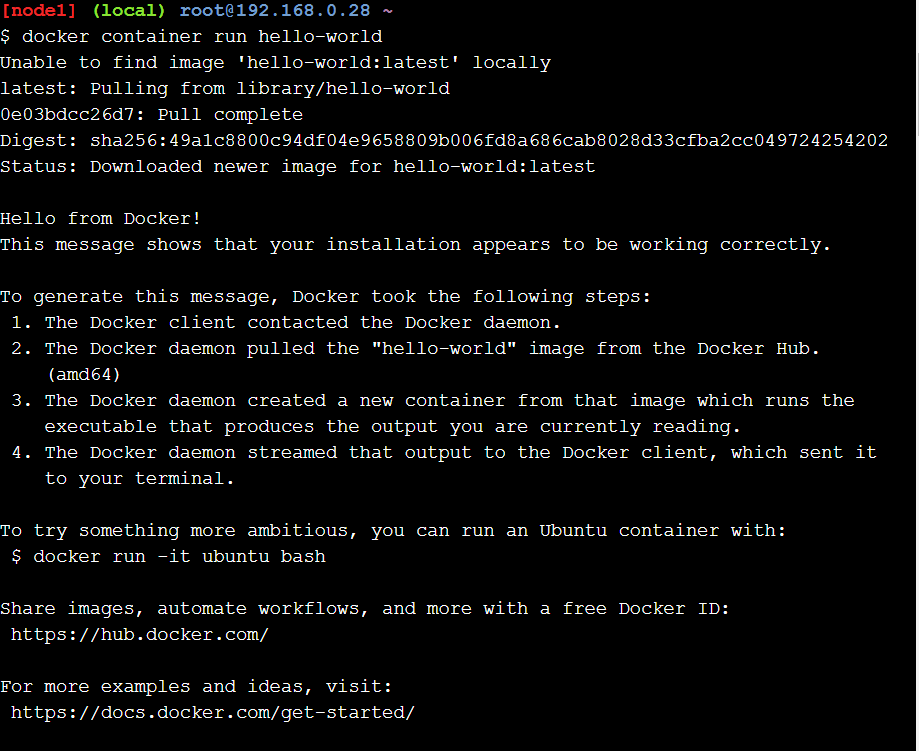
1. **Uname -a**

Basic linux commands that gives information about the system



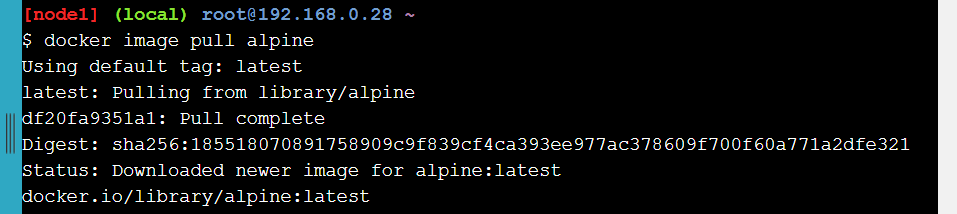
1. **Docker container run hello-world**

Create an image of the container hello-world. It is a default container that can be found on the docker platform



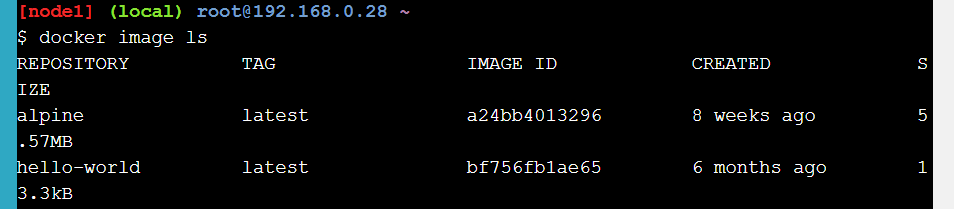
1. **Docker Image Pull Alpine**

Creates an container of the form Alpine linux, which is a linux distribution used for quick pull down and run.



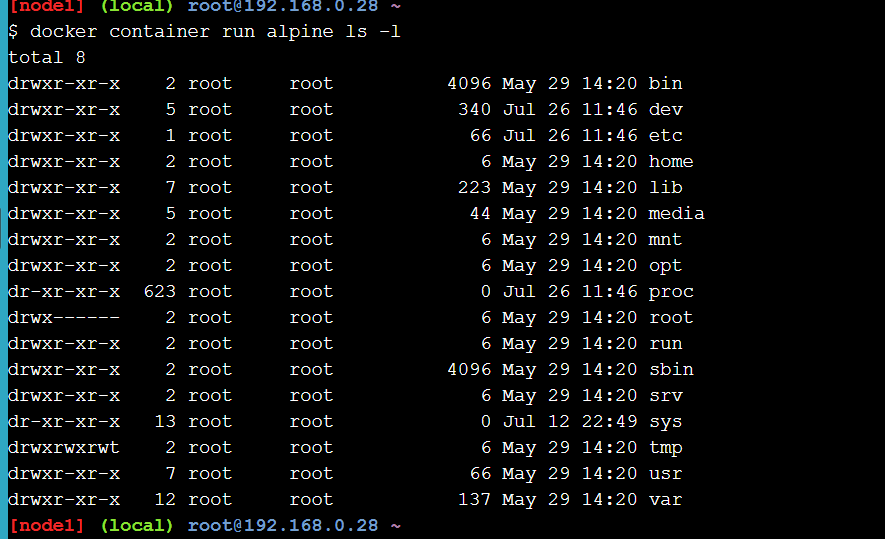
1. **Docker image ls**

Shows all the images of containers present in the system currently. We created two images of the form hello-world and alpine which is shown here.



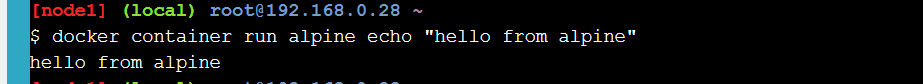
1. **Docker Container run alpine ls -l**

Runs the container alpine and then executes the command ls, which shows all the files present in the present working directory. After the command is executed the container shuts down.



1. **Docker container run alpine echo “hello from alpine”**

Just like the previous steps, runs the command echo



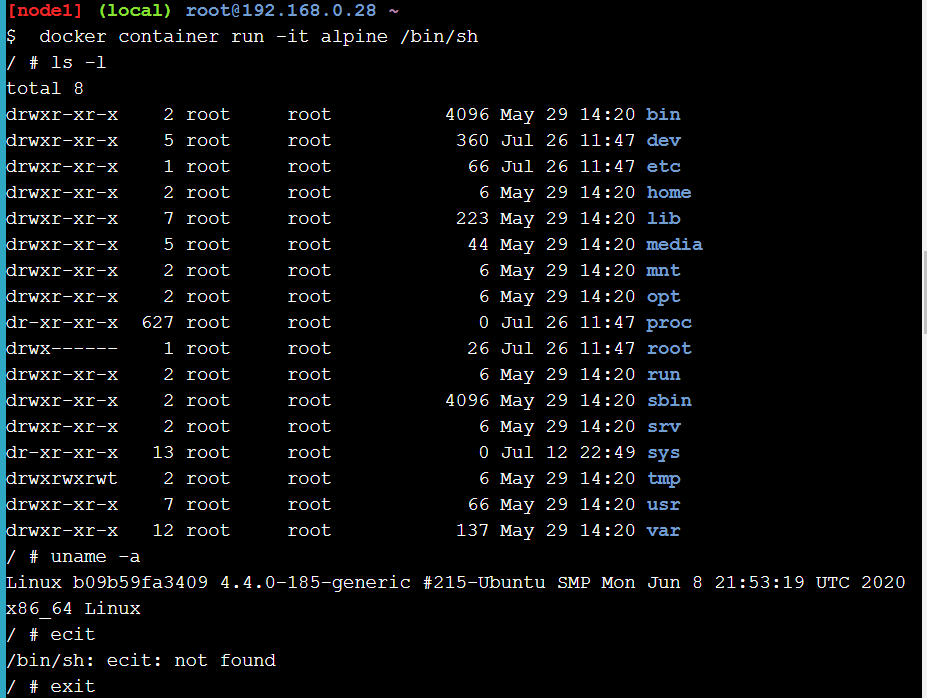
1. **Docker container run alpine /bin/sh**

/bin/sh is a shell in the alpine linux and we launched the shell but didn’t provide any commands, so it opened and then it terminated



1. **Docker container run -it alpine /bin/sh**

This starts the interactive version of the shell where we can put in multiple commands. I put ls-l and uname -a as example. To exit the shell, we type exit.



1. **Docker container ls -a**

Displays a list of all the containers that we have ran recently along with the commands that we used

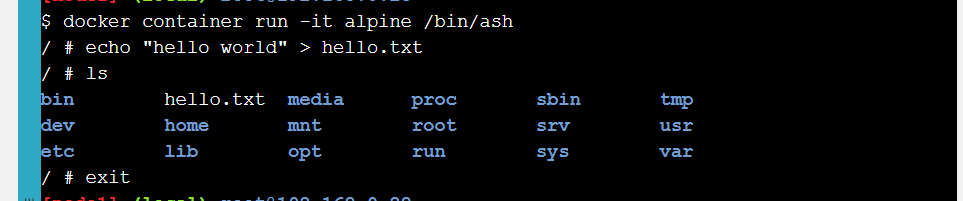


1. **Docker container run -it alpine /bin/ash**

**Echo “hello world” > hello.txt**

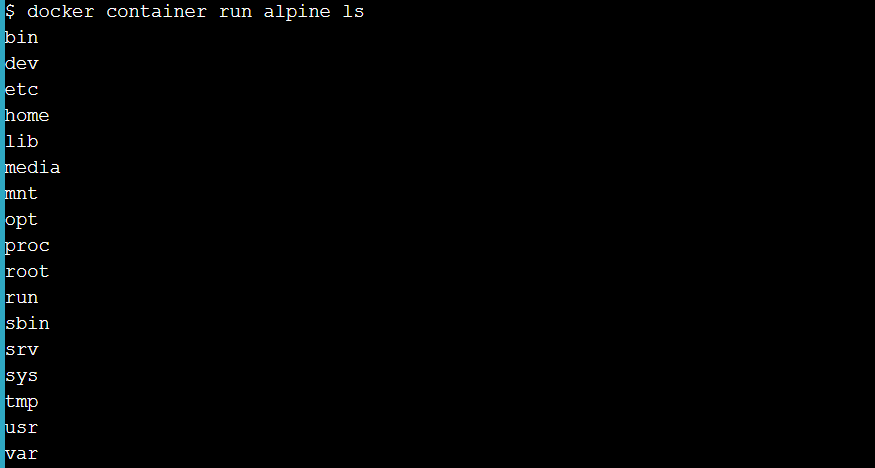
**ls**

Started the interactive shell and echoed the text “hello world” in the file hello.txt. Since it did not initially exist, it was created. Ls commands shows that the hello file was created.



1. **Docker Container run alpine ls**

As we can see, the hello.txt file is missing even though it’s the same container alpine. Docker creates a new instance everytime



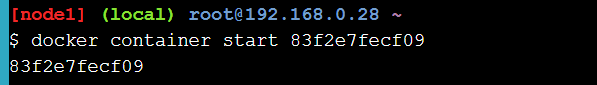
1. **Docker container ls -a**

To get back the container that stored the file hello.txt, we list all the containers and we note the container ID when we ran the command /bin/ash



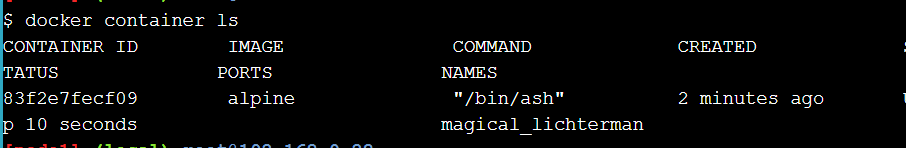
1. **Docker container start 83f2e7fecf09**

Running the container with the ID found in the previous step.



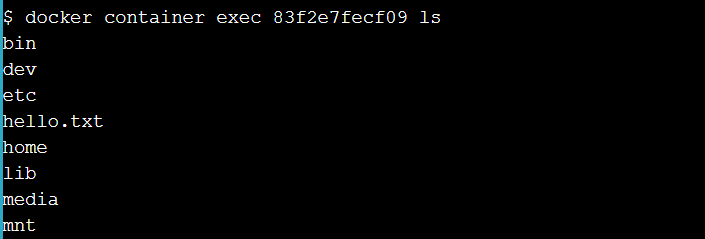
1. **Docker container ls**

Lists all the running containers.



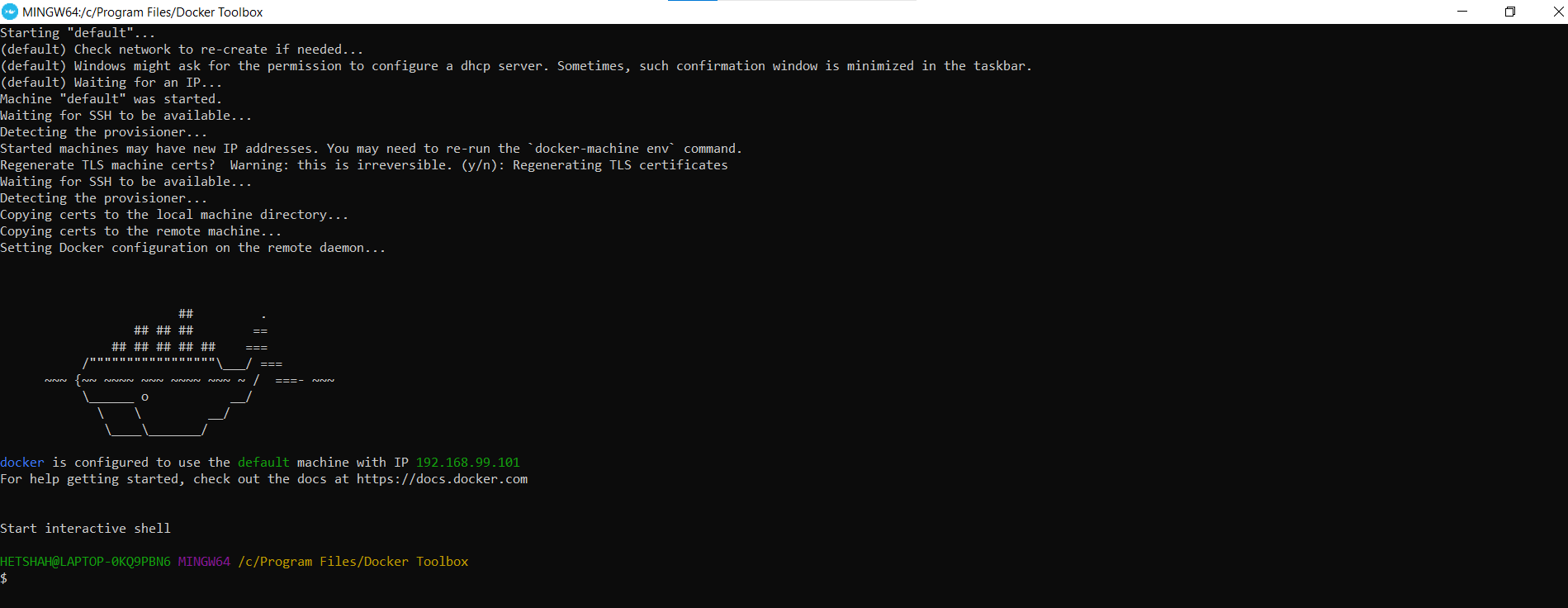
1. **Docker container exec 83f2e7fecf09 ls**

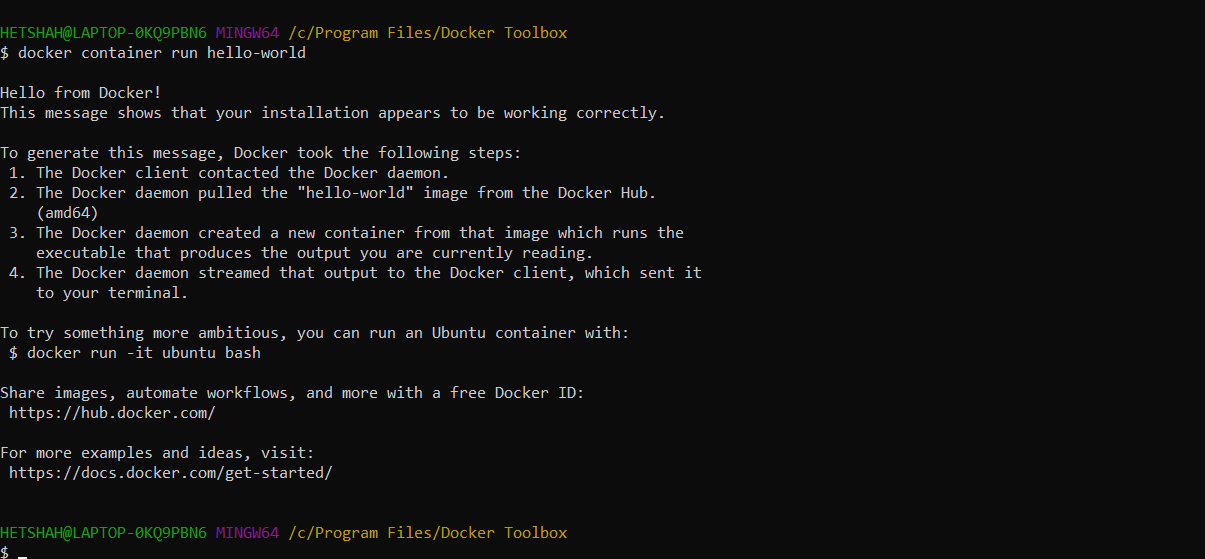
Execute the command ls on the docker container with the file hello.txt. As we can see the file is listed this time.



**Installing docker in the system**

As a part of the practical as well, we installed the docker toolbox on our windows system





Running docker container run hello-world on the local machine

**Conclusion**

In this practical we learned about the docker platform and how to create and manage containers. We also learned about how isolation works in the docker platform. In the end, we installed the docker platform on our local machine as well.