Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL605	**Course Name:**	Skill Based Lab course : Cloud Computing

Name of Student:	Soham Ajit Dahanukar
Roll No. :	13
Experiment No.:	9
Title of the Experiment:	To study and Implement Containerization using Docker
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty :

Signature :

Date :

Experiment No. 9

Aim: To study and Implement Containerization using Docker

Objective: To know the basic differences between Virtual machine and Container. It involves demonstration of creating, finding, building, installing, and running Linux/Windows application containers inside a local machine or cloud platform.

Theory:

- open platform for developing, shipping and running applications
- enables you to separate your applications from your infrastructure so you can deliver software quickly
- you can manage your infrastructure in the same ways you manage your applications
- Docker provides the ability to package and run an application in a loosely isolated environment called a container.
- Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host.
- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

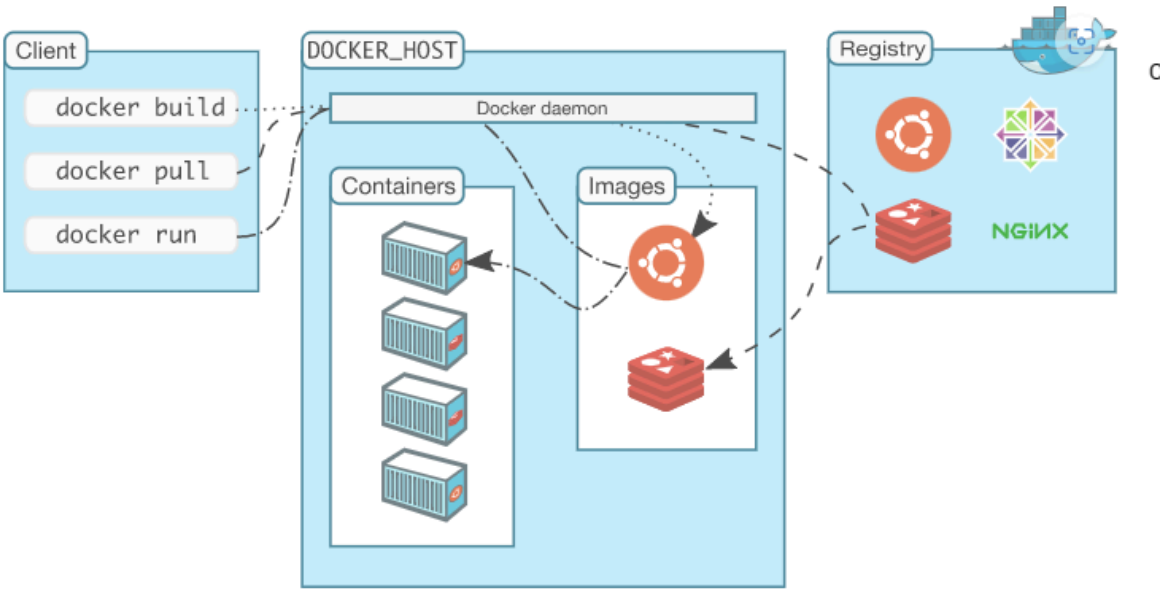
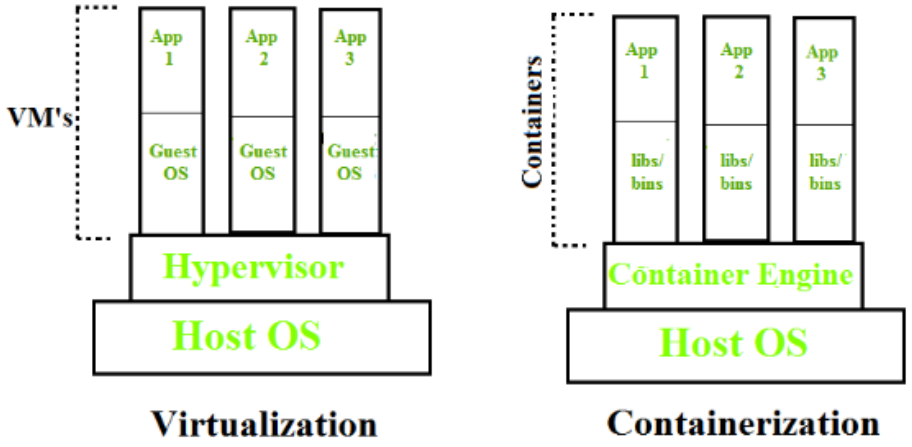


Figure 1: Docker Architecture

Containerization:

- Containerization is OS-based virtualization that creates multiple virtual units in the user space, known as Containers.
- Containers share the same host kernel but are isolated from each other through private namespaces and resource control mechanisms at the OS level.
- Container-based Virtualization provides a different level of abstraction in terms of virtualization and isolation when compared with hypervisors.
- Hypervisors use a lot of hardware which results in overhead in terms of virtualizing hardware and virtual device drivers.
- containers implement isolation of processes at the operating system level, thus avoiding such overhead.
- Containerization has better resource utilization compared to VMs and a short boot-up process. It is the next evolution in virtualization.
- Containers can run virtually anywhere, greatly easy development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or bare metal, on a developer's machine or in data centers on-premises; and of course, in the public cloud.
- Containers virtualize CPU, memory, storage, and network resources at the OS level, providing developers with a sandboxed view of the OS logically isolated from other applications.
- Docker is the most popular open-source container format available and is supported on Google Cloud Platform and by Google Kubernetes Engine.



Steps:

1. Open docker.com Scroll down, Click on 'Get started for free' tab.
2. Click on Docker Desktop, Download it

3. After downloading, Open 'Docker Desktop Installer' & start installation
4. After Installation, Restart your device.
5. Accept the terms and conditions, Click on Accept
6. Click on the link - https://aka.ms/wsl2kernel. (Do not close this window).
7. Download the WSL2 Linux kernel update package for x64 machines.
8. A ft e r D o w nlo a d is c o m ple t e, R u n t h e . m s i p a c k a g e. Click on next
9. After, the setup is complete, Click on finish.
10. Open Powershell as an Administrator
11. Run the following Command:
    wsl --set-default-version 2

0. Now, Click on Restart
0. Docker should now restart. Click on Start.
    0. Open Command Prompt, run the following commands:
    1. To check the version of Docker:
       docker --version

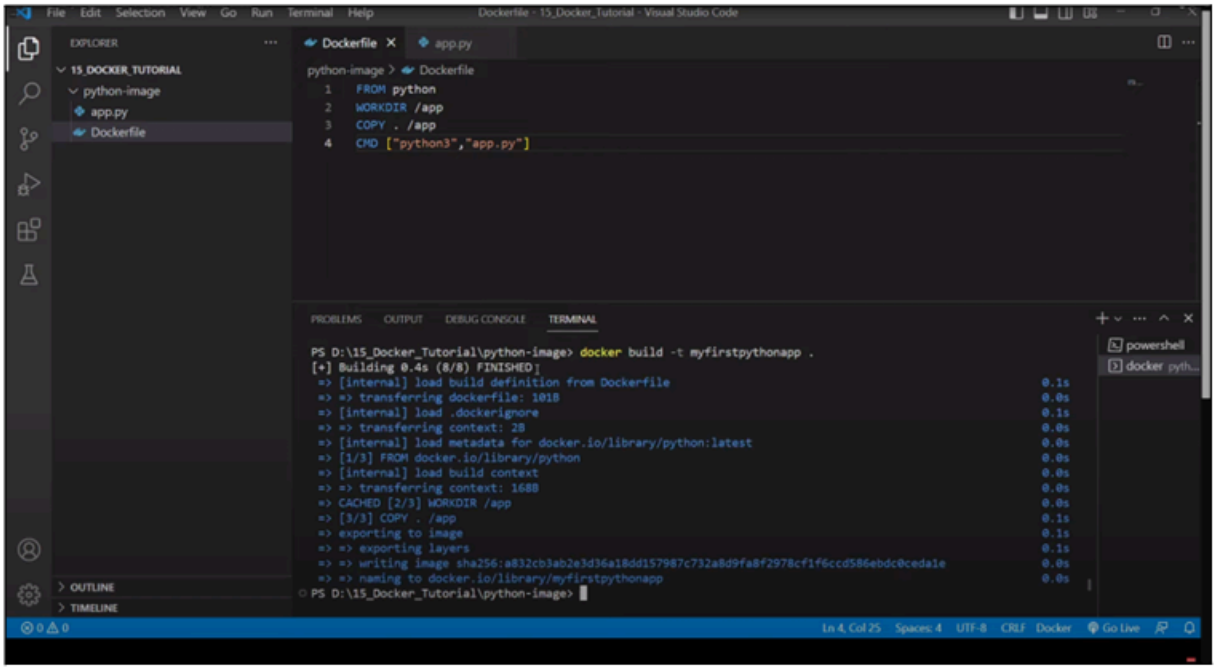
0. To install image of ubuntu
       docker pull ubuntu
0. Check downloaded images
       docker images
0. Run ubuntu OS
       docker run -it ubuntu /bin/bash
0. Open another Command Prompt and follow the steps shown below
       -docker ps

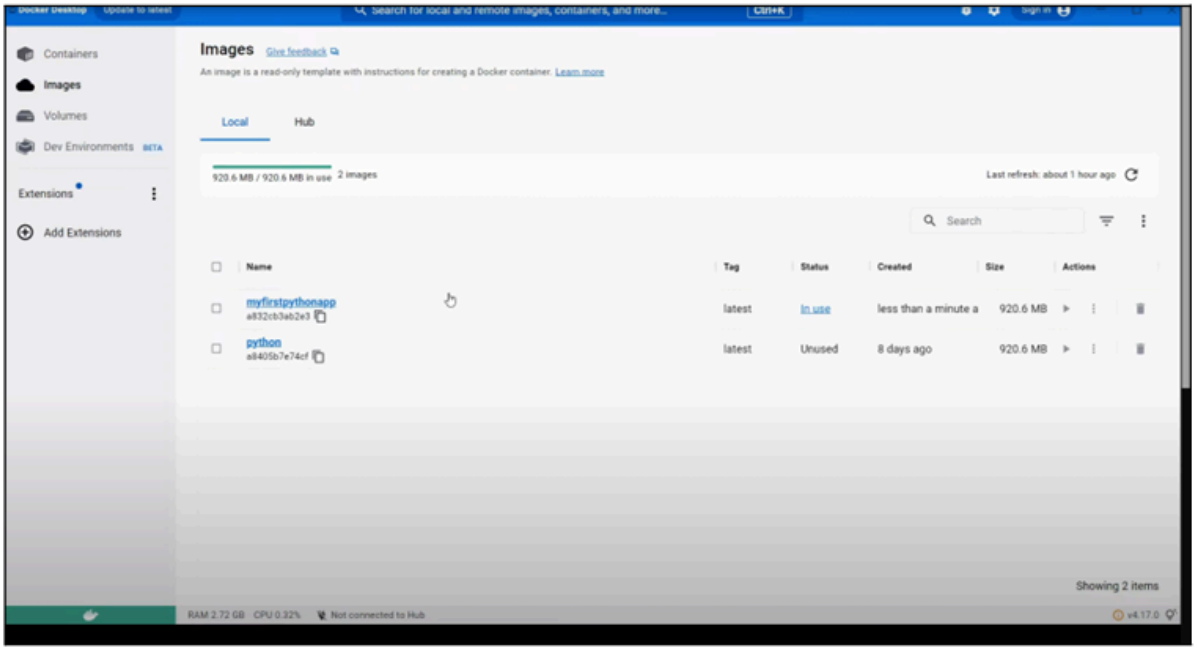
       docker container ls –a

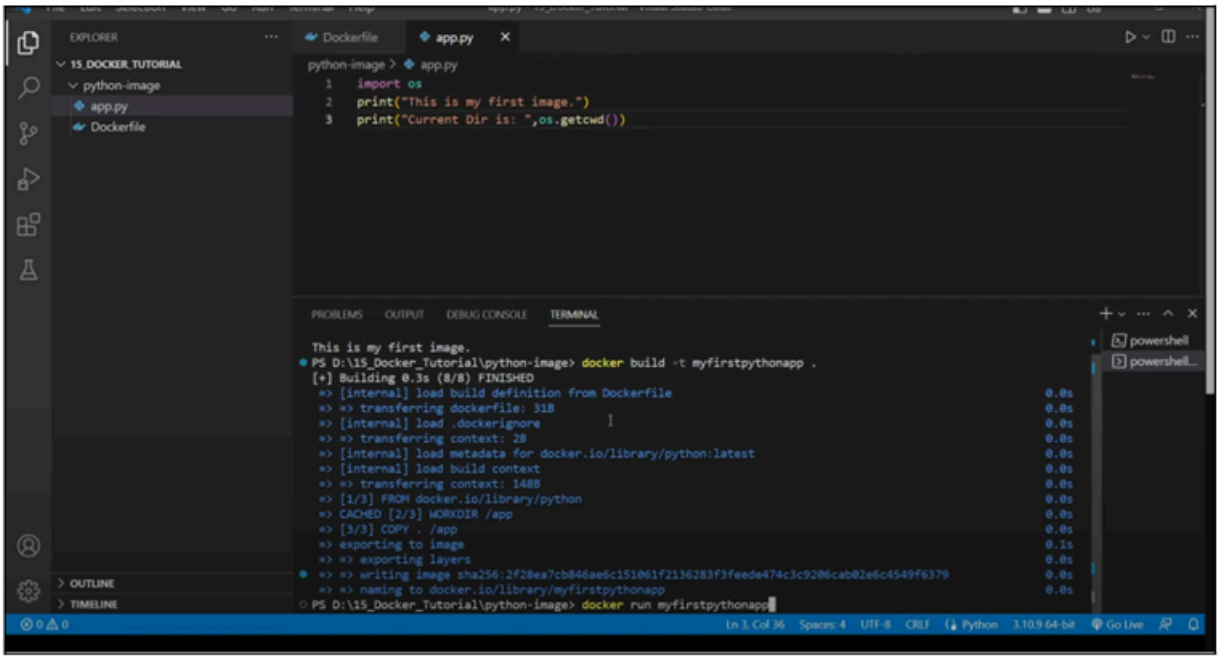
       docker container rm b71e3e6b1118 //copy docker id for remove but first
       (Use your container ID in the above command)

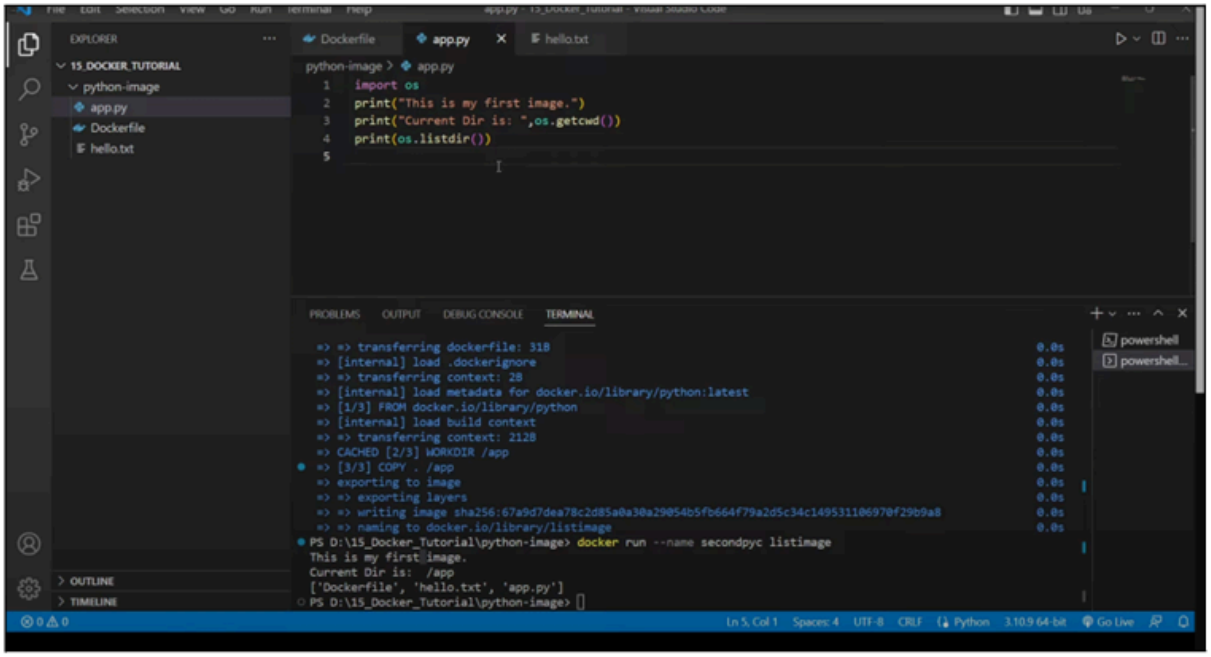
       stop your docker
       - docker container stop b71e3e6b1118
       - docker container rm b71e3e6b1118
       - docker ps
       - docker //list all docker commands
       - docker images
       - docker image rm ff0fea8310f3 // copy image id from previous output
       (Use your image ID in the above command)
       - docker run -it ubuntu /bin/bash //check output

Output/Observation:






EXPLORER
15_DOCKER_TUTORIAL
python-image
app.py
Dockerfile
Dockerfile
app.py
python-image > app.py
1 import os
2 print("This is my first image.")
3 print("Current Dir is: ",os.getcwd())
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
This is my first image.
PS D:\15_Docker_Tutorial\python-image> docker build -t myfirstpythonapp .
[+] Building 0.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 318
=> [internal] load .dockerignore
=> => transferring context: 28
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load build context
=> => transferring context: 1488
=> [1/3] FROM docker.io/library/python
=> CACHED [2/3] WORKDIR /app
=> [3/3] COPY . /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:2f28ea7cb846ae6c151061f2136283f3feede474c3c9206cab02e6c4549f6379
=> => naming to docker.io/library/myfirstpythonapp
PS D:\15_Docker_Tutorial\python-image> docker run myfirstpythonapp
powershell
powershell...
OUTLINE
TIMELINE


EXPLORER
15_DOCKER_TUTORIAL
python-image
app.py
Dockerfile
hello.txt
Dockerfile
app.py
hello.txt
python-image > app.py
1 import os
2 print("This is my first image.")
3 print("Current Dir is: ",os.getcwd())
4 print(os.listdir())
5
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
=> => transferring dockerfile: 318
=> [internal] load .dockerignore
=> => transferring context: 28
=> [internal] load metadata for docker.io/library/python:latest
=> [1/3] FROM docker.io/library/python
=> [internal] load build context
=> => transferring context: 2128
=> CACHED [2/3] WORKDIR /app
=> [3/3] COPY . /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:67a9d7dea78c2d85a0a30a29054b5fb664f79a2d5c34c14953110670f29b9a8
=> => naming to docker.io/library/listimage
PS D:\15_Docker_Tutorial\python-image> docker run --name secondpyc listimage
This is my first image.
Current Dir is: /app
['Dockerfile', 'hello.txt', 'app.py']
PS D:\15_Docker_Tutorial\python-image>
powershell
powershell...
OUTLINE
TIMELINE

Conclusion:

Implementing containerization using Docker involves creating lightweight, portable containers to package and deploy applications along with their dependencies. It streamlines the development, deployment, and scaling of applications by isolating them from the underlying infrastructure, ensuring consistency across different environments. Docker simplifies the process of building, sharing, and running containers, enhancing development agility and operational efficiency. Additionally, Docker's ecosystem includes tools for container orchestration, networking, and security, empowering organizations to build and manage containerized applications at scale with ease.