Experiment – 9

Name: Ronak Surve
Roll No: 64
Batch: C
Subject: Cloud Computing

Title: Implement Containerization using Docker	
Learning Objective:	To study and Implement Containerization using Docker
Learning Outcome:	Students will be able to understand the Security practices available in public cloud platforms and to demonstrate various Threat detection.
Course Outcome:	**CSL605 .6**
Program Outcome:	1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. 2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. 3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
Bloom's Taxonomy Level:	Apply
Theory:	Docker is a software platform that allows developers to create, deploy, and run applications in containers. It is an open-source platform that provides an isolated environment for running applications, making it easier to manage and deploy software across different environments. Here are some key concepts and features of Docker: 1. Containerization: Docker uses containerization technology to create an isolated environment for applications to run in. Containers are lightweight and self-contained, which means they can run anywhere, regardless of the underlying operating system. 2. Docker Image: A Docker image is a lightweight, standalone, and executable package

	that contains all the necessary components to run an application. It includes the application code, libraries, dependencies, and configuration files. Docker images can be built and shared easily, making it easier to deploy applications across different environments. 3. Docker Hub: Docker Hub is a cloud-based registry that allows developers to store, share, and manage Docker images. It provides a centralized location for managing Docker images, making it easier to collaborate with other developers and share applications. 4. Dockerfile: A Dockerfile is a text file that contains instructions for building a Docker image. It includes a set of commands that define how the Docker image should be built, what software should be installed, and how the application should be configured. 5. Docker Compose: Docker Compose is a tool that allows developers to define and run multi-container Docker applications. It simplifies the process of running complex applications by defining all the necessary components in a single file. 6. Portability: Docker provides a high level of portability, allowing developers to run applications in different environments without modification. Docker containers can be easily moved between different machines or cloud platforms, making it easier to deploy applications in different environments. 7. Scalability: Docker provides a scalable infrastructure for running applications. Developers can easily spin up multiple containers to handle increased traffic or load, and Docker's orchestration tools make it easier to manage and scale container-based applications. Overall, Docker provides a powerful platform for managing and deploying applications, with features like containerization, Docker images, Docker Hub, Dockerfile, Docker Compose, portability, and scalability. These features make it easier for developers to build, deploy, and manage applications across different environments, leading to faster development cycles and more efficient operations.
Procedure	**Students will be able 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 local machine or cloud platform**
Steps	Here are the basic steps for using Docker: 1. Install Docker: To use Docker, you first need to install it on your computer or server. Docker provides installation packages for Windows, Mac, and Linux operating systems. You can download the installation package from the Docker website and follow the installation instructions. 2. Create a Dockerfile: A Dockerfile is a text file that contains instructions for building a Docker image. You can create a Dockerfile using a text editor or an integrated development environment (IDE). The Dockerfile contains a set of commands that define how the Docker image should be built, what software should be installed, and

how the application should be configured.

3. Build a Docker image: Once you have created a Dockerfile, you can use the docker build command to build a Docker image. The docker build command reads the Dockerfile and builds an image based on the instructions in the file. The Docker image is a lightweight, standalone, and executable package that contains all the necessary components to run an application.

4. Run a Docker container: To run a Docker container, you can use the docker run command. The docker run command creates a new container based on the Docker image and starts the application inside the container. You can specify various options with the docker run command, such as the port mappings, environment variables, and volume mounts.

5. Manage Docker containers: You can use various Docker commands to manage Docker containers. For example, you can use the docker ps command to list all running containers, the docker stop command to stop a container, and the docker rm command to remove a container.

6. Use Docker Compose: Docker Compose is a tool that allows you to define and run multi-container Docker applications. You can create a docker-compose.yml file that defines all the necessary components of your application and use the docker-compose up command to start the application.

These are the basic steps for using Docker. Once you have mastered these steps, you can explore more advanced Docker features, such as Docker networking, Docker volumes, and Docker swarm mode.

Outcome:

```
dbit@complab1-13:~$ sudo apt install docker
[sudo] password for dbit:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin docker-scan-plugin gir1.2-goa-1.0 libfwupdplugin1
  libxmlb1 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  wmdocker
The following NEW packages will be installed:
  docker wmdocker
0 upgraded, 2 newly installed, 0 to remove and 6 not upgraded.
Need to get 14.3 kB of archives.
After this operation, 58.4 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 wmdocker amd64 1.5-2 [13.0 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 docker all 1.5-2 [1,316 B]
Fetched 14.3 kB in 0s (34.1 kB/s)
Selecting previously unselected package wmdocker.
(Reading database ... 223452 files and directories currently installed.)
Preparing to unpack .../wmdocker_1.5-2_amd64.deb ...
Unpacking wmdocker (1.5-2) ...
Selecting previously unselected package docker.
Preparing to unpack .../archives/docker_1.5-2_all.deb ...
Unpacking docker (1.5-2) ...
Setting up wmdocker (1.5-2) ...
Setting up docker (1.5-2) ...
Processing triggers for man-db (2.9.1-1) ...
dbit@complab1-13:~$ mkdir hello-docker
dbit@complab1-13:~$ cd hello-docker
```

```
dbit@complab1-13:~/hello-docker$ nano app.py
dbit@complab1-13:~/hello-docker$ python app.py

Command 'python' not found, did you mean:

  command 'python3' from deb python3
  command 'python' from deb python-is-python3

dbit@complab1-13:~/hello-docker$ python3 app.py
25
```

```
  GNU nano 4.8                                                         app.py
a=15
b=10
c = a+b
print(c)
```

```
dbit@complab1-13:~/hello-docker$ docker build -t hello-docker
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
dbit@complab1-13:~/hello-docker$ sudo chmod 666 /var/run/docker.sock
```

```
dbit@complab1-13:~/hello-docker$ nano dockerfile
dbit@complab1-13:~/hello-docker$ docker build -t hello-docker .
[+] Building 12.7s (8/8) FINISHED
 => [internal] load build definition from dockerfile                                                                0.3s
 => => transferring dockerfile: 99B                                                                                 0.0s
 => [internal] load .dockerignore                                                                                   0.2s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/python:alpine                                                    2.8s
 => [1/3] FROM docker.io/library/python:alpine@sha256:cf1bda3cbfe9a65c5c1541233e75e4ad7384b60e592d3afc16b50b0197fe98b8  7.1s
 => => resolve docker.io/library/python:alpine@sha256:cf1bda3cbfe9a65c5c1541233e75e4ad7384b60e592d3afc16b50b0197fe98b8  0.1s
 => => sha256:faa62a21887c6ce0730e40a70103b17132a4cebc1656e1efbf7199613925721c 6.32kB / 6.32kB                      0.0s
 => => sha256:cf1bda3cbfe9a65c5c1541233e75e4ad7384b60e592d3afc16b50b0197fe98b8 1.65kB / 1.65kB                      0.0s
 => => sha256:d811538656b665bba649b3ac2c3fc8a5187e24cd510a34d7e905d7ec4534fb89 1.37kB / 1.37kB                      0.0s
 => => sha256:e9f9cf5029d271a7f108bc0ad050e3772c5047c61a6e1f7413bc751dc4f01653 622.93kB / 622.93kB                  1.3s
 => => sha256:d47ea573a7911457dc1b9f91c991dd22366e36b455251f21e6d0b3a056b3a4c4 14.63MB / 14.63MB                    3.2s
 => => sha256:37906e6a7a2e6f773bfef2bcb87c5fa77cb2150296f96a8d7ecb9cc5d4eea5a9 241B / 241B                          1.6s
 => => extracting sha256:e9f9cf5029d271a7f108bc0ad050e3772c5047c61a6e1f7413bc751dc4f01653                           0.2s
 => => sha256:ab9f45de3756d3acbcbac0257fc4686a28fc5ada48917ee9eb4a77afa9ce011d 3.08MB / 3.08MB                      2.6s
 => => extracting sha256:d47ea573a7911457dc1b9f91c991dd22366e36b455251f21e6d0b3a056b3a4c4                           0.2s
 => => extracting sha256:37906e6a7a2e6f773bfef2bcb87c5fa77cb2150296f96a8d7ecb9cc5d4eea5a9                           0.0s
 => => extracting sha256:ab9f45de3756d3acbcbac0257fc4686a28fc5ada48917ee9eb4a77afa9ce011d                           0.2s
 => [internal] load build context                                                                                   0.3s
 => => transferring context: 164B                                                                                   0.0s
 => [2/3] COPY . /app                                                                                               0.9s
 => [3/3] WORKDIR /app                                                                                              0.4s
 => exporting to image                                                                                              0.8s
 => => exporting layers                                                                                             0.6s
 => => writing image sha256:5b7c5ed6c54e134c0cdc4287541c167534a014c7c174bcb1883c3489e5b1a510                        0.0s
 => => naming to docker.io/library/hello-docker                                                                     0.0s
dbit@complab1-13:~/hello-docker$ docker run hello-docker
25
```

```
  GNU nano 4.8                                                    dockerfile
FROM python:alpine
COPY . /app
WORKDIR /app
CMD python app.py
```

Conclusion:	Docker provides a consistent and isolated environment for running tasks, reducing variability and improving reproducibility. By leveraging containerization technology, Docker can also enable faster deployment and scaling of experiments, allowing for more efficient use of resources and better experimental outcomes.
References:	https://youtu.be/pTFZFxd4hOI

Rubrics for Assessment

Timely Submission	Submitted after 2 weeks 0	Submitted after deadline 1	On time Submission 2
Understanding	Student is confused about the concept 0	Students has justifiably understood the concept 2	Students is very clear about the concepts 3
Performance	Students has not performed the Experiment 0	Student has performed with help 2	Student has independently performed the experiment 3
Development	Students struggle to run virtual machines. 0	Student can write steps the requirement stated 1	Student can write exceptional steps with his own ideas 2