**Assignment**

1. What is Docker, and why is it used?

Docker is a containerization platform that allows the developers to packaging applications and there dependences, lightweight, potable containers that can run consistently across different environment

* The application run the same way in development, testing and production environment
* Each container runs independently. Preventing the conflicts

1. How is Docker different from a virtual machine (VM)?

|  |  |
| --- | --- |
| **Docker** | **VMware** |
| Resource usage low | Resources usage high |
| Docker takes seconds to start | VMware takes 1-2 minutes to start |
| Runs anywhere | Hard to move |
| Easily scalable | Requires more hardware |
| Free and open source | Expensive |

1. What are the main components of Docker?
2. **Docker Engine**

The core component that runs and manages containers

1. **Docker Images**

A **blueprint** for containers that includes everything needed to run an application

1. **Docker Containers**

A running instance of a docker image

1. **Docker file**

A script that defines how to build a docker image

1. **Docker Compose**

A tool to define and run multi-container applications using a docker-compose.yml file

1. **Docker Hub**

A cloud-based registry where you can store and share Docker image

1. Explain the difference between Docker images and Docker containers.

|  |  |
| --- | --- |
| Docker images | Docker Container |
| A **read-only** blueprint/template used to create containers | A **running instance** of a Docker image. |
| Static (doesn’t change after creation). | Dynamic (can change during execution). |
| Cannot be modified; remains the same until rebuilt. | Can be started, stopped, modified, and deleted. |
| Stored on disk and used to launch containers | Runs in memory (unless explicitly stored or committed). |
| An Ubuntu image that contains all the OS files. | A running instance of that Ubuntu image where you can execute commands. |

1. What is a Dockerfile?

A Dockerfile is a script-like text file that contains a set of instructions to automate the creation of a docker images. It defines everything needed to set up an environment, including the base image, dependencies. Configurations, and commands

1. What command is used to build a Docker image?

Docker build -t image\_name

1. How do you run a container from an image?

Docker run -itd –name container\_name -p portid:container\_numder imageid

1. How do you list all running containers?

Docker ps

1. What command is used to stop a running container?

Docker stop container\_id

1. How do you remove a Docker container?

Docker rmi container\_id

1. What is the difference between CMD and ENTRYPOINT in a Dockerfile?

**CMD (**Default command)

* Used to provide default arguments for a container
* Can be overridded when running docker run
* If multiple CMD instructions are present, only the last one is used

**ENTRYPOINT (**Fixed Command)

* Used to define a fixed command that cannot be overridden easily
* Additional arguments are treated as parameters
* If you still need to overrided it, you must use the –entrypoint flag

1. What is a Docker volume, and why is it used?

A **Docker volume** is a storage mechanism used by Docker to **persist data** beyond the lifecycle of a container. Volumes allow containers to store and share data even when they are stopped or removed.

1. How do you persist data in Docker containers?

Docker **volumes** are the best way to persist data because they are managed by Docker and stored in /var/lib/docker/volumes/

* Using volume

Docker run -d –name container\_name -v volume:/data image\_id

1. What is a Docker Compose file? How is it used?

A **Docker Compose file** (docker-compose.yml) is a YAML configuration file used to **define and manage multi-container Docker applications**

* **Simplifies Multi-Container Management** – Define multiple services in one file.
* **Easier Deployment** – Start all services with a single command
* **Reusable & Portable** – Works across different environments.

1. How do you scale services using Docker Compose?

Docker compose up –scale wed=3 -d

1. How do you check the logs of a running container?

Docker logs container\_id

1. What is the purpose of the .dockerignore file?

The **.dockerignore** file is used to **exclude files and directories** from being copied into a Docker image when building it using a Dockerfile. This helps optimize the image by preventing unnecessary files from being included, which can reduce the size of the image and speed up the build process.

1. What are the different networking modes in Docker?

**Bridge Network (**default network mode)

* Used for standalone containers.
* Containers in the same bridge network can communicate using container names.
* The default bridge network is called bridge, but custom bridge networks can be created.

**Host Network**

* The container shares the host machine’s network stack.
* No network isolation between the container and the host.
* Used for performance-sensitive applications.

**None Network**

* The container has no network access.
* Used for security or when networking is unnecessary.

**Network plugins**

* Supports third-party network drivers (e.g., Calico, Weave, Cilium).
* Used for advanced networking features in Kubernetes and large-scale deployments.

1. How do you expose ports in a Docker container?

Docker run -itd --name container\_name -p port\_number:container\_number image\_id

1. What is the difference between docker stop and docker kill?

**Docker stop**

* Sends a **SIGTERM** signal to the container’s main process
* Allows the process to perform cleanup (e.g., saving data, closing connections).
* After a timeout (default: 10 seconds), if the container hasn’t stopped, Docker sends **SIGKILL** to force termination.

Docker stop <container\_id>

**Docker kill**

* Immediately stops the container by sending a **SIGKILL** signal.

Docker kill <container\_id>