Assignment – 4

Submitted by ~ Ishika Dutta (CT_CSI_DV_5572)

Task 4.1

Introduction to containerization and Docker fundamentals, Basic Commands

Command	Purpose
docker --version	Check Docker version installed
docker pull <image>	Download an image (e.g. docker pull nginx)
docker images	List downloaded images
docker run <image>	Run a container (creates and starts a container)
docker run -d <image>	Run in detached mode (in background)
docker run -p 8080:80 <image>	Map port 8080 on host to 80 in container
docker ps	List running containers
docker ps -a	List all containers (including stopped ones)
docker stop <container_id>	Stop a running container
docker rm <container_id>	Remove a stopped container
docker rmi <image_id>	Remove an image
docker exec -it <container_id> /bin/bash	Open shell inside container
docker build -t <name> .	Build image from Dockerfile in current directory

Task 4.2

Docker installation and basic container operations, Build an image from Dockerfile

Command	Example
Pull an image	docker pull nginx
Run a container	docker run -d -p 8080:80 nginx
List running containers	docker ps
List all containers	docker ps -a
Stop a container	docker stop <container_id>
Remove a container	docker rm <container_id>

Remove an image	docker rmi <image_id>
Exec inside a container	docker exec -it <container_id> /bin/bash

Verify installation

```
docker --version
docker run hello-world
```

For Linux

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release


sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg


echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin


docker --version
sudo docker run hello-world

```

Container operations

```
docker pull nginx
docker run -d -p 8080:80 nginx
docker ps
docker stop <container_id>
docker rm <container_id>
```

Build an Image from a Dockerfile

Example Dockerfile

```
FROM python:3.9-slim

WORKDIR /app

COPY app.py /app/app.py

RUN pip install flask

EXPOSE 5000

CMD ["python", "app.py"]
```

Build the image

```
docker build -t my-flask-app .
```

Run the container

```
docker run -d -p 5000:5000 my-flask-app
```

Task 4.3

Docker Registry, DockerHub, Create a Multi-Stage Build

Basic workflow with Docker Hub

Login

```
docker login
```

Tag your image

```
docker tag my-flask-app username/my-flask-app:latest
```

Push to Docker Hub

```
docker push username/my-flask-app:latest
```

Pull from Docker Hub

```
docker pull username/my-flask-app:latest
```

Multi-Stage Build: Build multi-stage image

```
docker build -t my-multi-stage-app .
```

```
docker run -d -p 80:80 my-multi-stage-app
```

Task 4.4

Create a docker image from multiple methods likes Dockerfile, running containers.

Create an image using a Dockerfile

Create a file

```
FROM python:3.9-slim

WORKDIR /app
COPY app.py /app/

RUN pip install flask

EXPOSE 5000

CMD ["python", "app.py"]
```

Build the image

```
docker build -t my-flask-image .
```

Run the container

```
docker run -d -p 5000:5000 my-flask-image
```

Create an image from a running container (commit method)

Run a base container

```
docker run -it ubuntu /bin/bash
```

Inside the container, install something

```
apt-get update
apt-get install -y curl
```

Exit the container

```
exit
```

Find the container ID

```
docker ps -a
```

Commit the container as a new image

```
docker commit <container_id> my-ubuntu-curl
```

Run your new image

```
docker run -it my-ubuntu-curl /bin/bash
```

3t

Task 4.5

Push and pull image to Docker hub and ACR

Push and Pull to Docker Hub

Login

```
docker login
```

Tag the image

```
docker tag my-flask-app yourusername/my-flask-app:latest
```

Push to Docker Hub

```
docker push yourusername/my-flask-app:latest
```

Pull from Docker Hub

```
docker pull yourusername/my-flask-app:latest
```

Push and Pull to Azure Container Registry

Login to Azure

```
az login
```

Login to ACR

```
az acr login --name myregistry
```

Tag your image

```
docker tag my-flask-app myregistry.azurecr.io/my-flask-app:latest
```

Push to ACR

```
docker push myregistry.azurecr.io/my-flask-app:latest
```

Pull from ACR

```
docker pull myregistry.azurecr.io/my-flask-app:latest
```

```
docker run myregistry.azurecr.io/my-flask-app:latest
```

Task 4.6

Create a Custom Docker Bridge Network

Create

```
docker network create \
  --driver bridge \
  my_custom_bridge
```

Inspect the network

```
docker network inspect my_custom_bridge
```

Run containers on your custom bridge network

```
docker run -dit --name app1 --network my_custom_bridge alpine sh
docker run -dit --name app2 --network my_custom_bridge alpine sh
```

List networks

```
docker network ls
```

Remove networks

```
docker network rm my_custom_bridge
```

Task 4.7

Create a Docker volume and mount it to a container.

Create

```
docker volume create my_volume
```

List all volumes

```
docker volume ls
```

Inspect volume

```
docker volume inspect my_volume
```

Mount the volume

```
docker run -dit --name my_container \
  -v my_volume:/data \
  alpine sh
```

Reuse the volume with another container

```
docker run --rm -v my_volume:/data alpine cat /data/hello.txt
```

Task 4.8

Docker Compose for multi-container applications, Docker security best practices

Docker Compose

Flask Redis App

```yaml
version: '3'

services:
  web:
    image: my-flask-app
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5000:5000"
    depends_on:
      - redis

  redis:
    image: redis:alpine
```

Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY app.py /app/
RUN pip install flask redis
CMD ["python", "app.py"]
```

App.py

```
from flask import Flask
import redis

app = Flask(__name__)
 = redis.Redis(host='redis', port=6379)

@app.route('/')
def hello():
    r.incr('hits')
    return f'Hello! This page has been visited {r.get("hits").decode()} times.'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Build

```
docker-compose up --build
```

Docker Security Best Practices

Image security

- *Use official or trusted base images → Don't use random images from Docker Hub.*
- *Scan images for vulnerabilities → Use docker scan (powered by Snyk) or tools like Trivy.*
- *Keep images updated → Rebuild regularly to include latest security patches.*
- *Minimize image size → Use minimal base images (alpine, slim).*

Container runtime security

- *Run as non-root inside the container*

```
RUN adduser --disabled-password appuser
USER appuser
```

'

- *Limit container capabilities*

```
docker run --cap-drop ALL --cap-add NET_BIND_SERVICE ...
```

- *Use read-only filesystem where possible*

```
docker run --read-only ...
```

- *Limit resource usage*

```
 docker run --memory 256m --cpus 0.5 ...
```

Network security

- *Use custom networks for container-to-container communication.*
- *Close unnecessary ports; only expose what you need (-p).*
- *Use firewalls (host or cloud-level) to restrict external access.*

Secrets management

- *Don't hardcode secrets in images / env files.*
- *Use tools like:*

- *Docker secrets (in Swarm mode)*
- *HashiCorp Vault, AWS Secrets Manager, Azure Key Vault.*

Host security

- *Keep Docker, OS, and kernel up to date.*
- *Restrict who can access the Docker daemon (/var/run/docker.sock).*
- *Use AppArmor, SELinux, or seccomp profiles for added isolation.*