# Docker Fundamentals Assignment

## 1. Introduction to Containerization & Docker Fundamentals

Containerization packages applications and dependencies into isolated units called containers. Docker is the most widely used container platform.

Key Concepts:

- **Image**: Blueprint for containers.
- **Container**: Lightweight, executable package.
- **Docker Engine**: Core service to run/manage containers.
- **Dockerfile**: Script to automate image creation.

Basic Commands:

- `docker --version`: Check Docker version.
- `docker pull <image>`: Download image.
- `docker run -it <image>`: Start interactive container.
- `docker ps -a`: List all containers.
- `docker stop <id>` / `docker rm <id>`: Stop/remove container.

## 2. Docker Installation & Container Operations

Installation:

- Windows/Mac: Docker Desktop.
- Linux: `sudo apt install docker.io`

Common Operations:

- `docker start <container>`: Start container.
- `docker exec -it <container> bash`: Access shell.
- `docker rm <container>`: Delete container.

Build from Dockerfile:

```dockerfile
FROM ubuntu
RUN apt update && apt install -y nginx
```

```
CMD ["nginx", "-g", "daemon off;"]
```

Run: `docker build -t my-nginx .`

## 3. DockerHub, Registry & Multi-Stage Build

DockerHub:

- `docker login`
- `docker push user/image`
- `docker pull user/image`

Multi-Stage Build Example:

```dockerfile
FROM golang:1.17 AS build
WORKDIR /src
COPY . .
RUN go build -o app

FROM alpine
COPY --from=build /src/app .
CMD ["./app"]
```

## 4. Creating Docker Images

Methods:

1. **From Dockerfile**: `docker build -t custom-img .`
2. **From Running Container**:
   - Make changes in container.
   - Save image: `docker commit <container> my-img`

## 5. DockerHub & Azure Container Registry (ACR)

DockerHub:

- Push: `docker push user/image`
- Pull: `docker pull user/image`

ACR:

- Login: `az acr login --name <acr>`
- Tag: `docker tag img <acr>.azurecr.io/img`
- Push: `docker push <acr>.azurecr.io/img`

### 6. Custom Docker Bridge Network

Create and Use Network:

- `docker network create --driver bridge my-net`
- `docker run --network my-net --name container1 ubuntu`

Containers on the same custom network can communicate by name.

### 7. Docker Volume & Mounting

Persistent Storage:

- Create: `docker volume create my-vol`
- Use: `docker run -v my-vol:/data ubuntu`
- Inspect: `docker volume inspect my-vol`

### 8. Docker Compose & Security Best Practices

Compose Setup (`docker-compose.yml`):

```yaml
version: '3'
services:
  web:
    image: nginx
    ports:
      - "80:80"
  db:
    image: mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
```

Run: `docker-compose up`

Best Practices:

- Use official base images.
- Minimize image size (multi-stage).
- Avoid running as root.
- Regular vulnerability scans.