Group 5: Feng Li, Deshui Yu, Gavin Li, Yaohuang Chen, Katsia Ilunga and Constance Benson

Docker

1. What is Docker?

Docker is a platform to **build, ship, and run applications in containers**. Containers are lightweight, isolated environments with everything your app needs (code, libraries, dependencies). Unlike virtual machines, containers share the host OS kernel but remain isolated.

Advantages:

- Portability (works anywhere, no need to install dependencies on every machine)
- Fast startup (containers start in seconds)
- Lightweight (less resource usage than virtual machines)
- Easy dependency management (libraries and runtime are packaged inside the container)

---

2. Installing Docker

On Windows

1. Download **Docker Desktop**: https://www.docker.com/products/docker-desktop
2. Install and enable **WSL2 integration** (Windows Subsystem for Linux). This allows Linux containers to run efficiently on Windows.
3. Open PowerShell and check version:

```
docker --version   # Check Docker Engine version
docker compose version   # Check Docker Compose version
```

On macOS

1. Download **Docker Desktop for Mac**: same link above.
2. Install and run. Docker will create a virtual environment for containers.
3. Check version:

```
docker --version   # Verify installation
docker compose version   # Verify Compose installation
```

On Linux (Ubuntu example)

```
# Update system packages
sudo apt update

# Install dependencies needed for Docker
sudo apt install apt-transport-https ca-certificates curl software-properties-common

# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-

# Add Docker repository to APT sources
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/u

# Update again and install Docker
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Check Docker version
docker --version

# Check Docker Compose version
docker compose version
```

```
# Optional: run Docker commands without sudo
sudo usermod -aG docker $USER
```

---

3. Docker Basic Concepts

Concept	Description
Image	A read-only template used to create containers (like a blueprint)
Container	A running instance of an image
Dockerfile	File to define image building steps
Volume	Persistent storage for containers, keeps data after container stops
Network	Enables communication between containers

---

4. Common Docker Commands

Managing Images

```
docker images   # List all downloaded images
docker pull ubuntu:22.04   # Download an image from Docker Hub
docker build -t my-app:1.0 .   # Build image from Dockerfile in current directory
docker rmi my-app:1.0   # Remove an unused image
```

Managing Containers

```
docker run -it --name mycontainer ubuntu:22.04 /bin/bash   # Run interactive container
docker run -d --name webserver -p 8080:80 nginx   # Run container in background, map port 8080 to 80
docker ps   # List running containers
docker ps -a   # List all containers, including stopped

docker stop mycontainer   # Stop a running container
docker start mycontainer   # Start a stopped container
docker rm mycontainer     # Remove a container
```

Logs and Monitoring

```
docker logs mycontainer   # Show container logs
docker logs -f mycontainer   # Follow logs in real-time
docker stats   # Monitor container CPU, memory, network usage
```

Docker Compose

docker-compose.yml

```
version: "3"
services:
  web:
    image: nginx
    ports:
      - "8080:80"
  db:
    image: mysql
    environment:
      MYSQL_ROOT_PASSWORD: example
```

```
docker compose up -d   # Start services in background
docker compose down     # Stop and remove containers
docker compose logs -f   # Follow logs
```

Volumes (Persistent Data)

```
docker volume create mydata   # Create a volume
docker run -d -v mydata:/data ubuntu   # Attach volume to container
docker volume ls              # List volumes
docker volume rm mydata       # Remove unused volume
```

Networks

```
docker network create mynetwork         # Create a custom network
docker run -d --network mynetwork --name web nginx   # Attach container to network
```

---

5. Simple Docker Example: Python App

Step 1: Create Project Folder

```
mkdir my-python-app
cd my-python-app
```

Step 2: Create Python Script

app.py

```
print("Hello from Docker!")
```

Step 3: Create Dockerfile

```
FROM python:3.11-slim    # Use small Python image
WORKDIR /app              # Set working directory in container
COPY app.py .             # Copy local file to container
CMD ["python", "app.py"]  # Command to run when container starts
```

> **Comment:** Always use a slim base image to reduce size and security risks.

Step 4: Build Docker Image

```
docker build -t my-python-app:1.0 .
```

> **Comment:** The dot `.` means Dockerfile is in the current directory.

Step 5: Run Container

```
docker run my-python-app:1.0
```

Output: `Hello from Docker!`

Step 6: Optional Interactive Mode

```
docker run -it my-python-app:1.0 /bin/bash
```

Step 7: Optional Docker Compose

docker-compose.yml

```yaml
version: "3"
services:
  app:
    build: .
```

```
# Start services in background
docker compose up -d

# Stop services
docker compose down

# View logs
docker compose logs -f
```