**Installing Docker  
Learning Objectives:**

* Set up Docker on student machines.
* Understand the difference between Docker Desktop (Mac/Windows) and Docker CLI/Engine (Linux).
* Run the first container to verify installation.

**Topics to Cover**

**1. Platform Differences**

|  |  |  |
| --- | --- | --- |
| **Platform** | **Docker Tool** | **Notes** |
| Windows | Docker Desktop | Requires WSL2 for Linux containers |
| MacOS | Docker Desktop | Comes with a GUI + CLI |
| Linux | Docker Engine | No GUI; uses terminal-based CLI. Most lightweight and production-like |

## Installation Instructions (Step-by-Step)

**A. For Linux (Ubuntu/Debian)  
  
1. Update package index**

sudo apt update

**2. Install Docker**

sudo apt install docker.io -y

**3. Start the Docker service**

sudo systemctl start docker

**4. Enable Docker to start on boot**

sudo systemctl enable docker

**5. Add your user to the docker group (so you can run docker without sudo)**

sudo usermod -aG docker $USER

**6. Log out and back in (or run `newgrp docker`) to apply group changes**

### ****B. For Windows****

1. Download Docker Desktop from:  
    **https://www.docker.com/products/docker-desktop/**
2. Install Docker Desktop like any normal app.
3. During installation:
   * **Enable WSL 2** when prompted.
   * Choose the default container backend as “WSL 2.”
4. After install, **restart your system**.
5. Open **Docker Desktop** and ensure it's running.
6. Open **PowerShell or Terminal** and verify with the command below.

### ****C. For MacOS****

1. Download Docker Desktop from:  
    **https://www.docker.com/products/docker-desktop/**
2. Install it like a standard app (drag to Applications folder).
3. Open **Docker Desktop** and wait for it to initialize.
4. Verify installation with the command below.

## Hands-On Verification Tasks

### 1. ****Check Docker Version****

**Use this:**

docker –version

**Expected output:**

Docker version 24.x.x, build <sampledata>

### ****2**. **Run a Test Container****

**Use this:**

docker run hello-world

### Expected output includes:

Hello from Docker!

### ****This message shows that your installation appears to be working correctly.****

### 3. Check Docker Info

### See detailed system-wide Docker settings:

docker info

Docker Architecture & Key Commands

## 🎯 Learning Objectives:

* Explain how Docker works using the client-server model.
* Understand key Docker components: client, daemon, images, containers, registries.
* Grasp the concept of Docker’s layered image structure and caching.
* Use essential Docker CLI commands to interact with the Docker engine.

### ****Docker’s Client-Server Architecture****

Docker uses a **client-server** architecture:

Containers, Images, Networks, Volumes

docker CLI

dockerd (Docker Daemon)

Docker Hub / Private Registry

#### Docker Client (docker)

* The **CLI tool** that users interact with.
* Sends commands to the Docker Daemon via REST API.
* Examples: docker run, docker pull, docker build, etc.

#### Docker Daemon (dockerd)

* Runs in the background.
* Manages Docker **containers**, **images**, **networks**, and **volumes**.
* Listens on a Unix socket or TCP port for commands from the client.

#### 

#### Docker Registry

* Stores and shares container images.
* **Docker Hub** is the default public registry.
* Organizations can use **private registries** (e.g., Harbor, AWS ECR, GitHub Container Registry).

#### Docker Images

* Templates used to create containers.
* Composed of layers (like layers of a cake).
* Can be downloaded from registries or built locally using Dockerfile.

#### Docker Containers

* A **runtime instance** of an image.
* Isolated processes with their own filesystem, network stack, and resources.

**Key Docker Commands:**

|  |  |
| --- | --- |
| **Command** | **Purpose** |
| docker --version | Check installed version |
| docker info | System-wide Docker information |
| docker pull <image> | Download image from registry |
| docker images | List locally stored images |
| docker run <image> | Create and start a container |
| docker ps | List running containers |
| docker ps -a | List all containers (running & stopped) |
| docker stop <container\_id> | Stop a running container |
| docker rm <container\_id> | Remove a container |
| docker rmi <image\_id> | Remove an image |
| docker logs <container\_id> | View container logs |
| docker exec -it <container\_id> bash | Enter an interactive shell in a running container |

**Hands-On Tasks:**

**1. Pull an Image**

docker pull nginx

**2. Run a Container**

docker run -d -p 8080:80 nginx

**3. List Containers & Images**

docker ps

docker images

**4. Enter a Running Container**

docker exec -it <container\_id> bash

**5. Stop and Remove Container**

docker stop <container\_id>

docker rm <container\_id>

**Running Containers from Images**Learning Objectives:

* Run containers using public images from Docker Hub.
* Understand how to use image names, tags, and modes of running containers.
* Map container ports to the host.
* Pass environment variables into containers.

### ****Docker Hub Search****

* Visit: <https://hub.docker.com>
* Search for trusted official images like nginx, redis, mysql, node, httpd, etc.
* Explore:
  + Pull command
  + Default ports exposed
  + Environment variables supported
  + Documentation for usage examples