Docker Basics: Getting Started

Introduction to Docker

Docker is a containerization platform that packages your application and all its dependencies into a lightweight, portable container. This ensures your application runs consistently across different environments.

Key Concepts

- **Container**: A lightweight, standalone, executable package containing everything needed to run an application
- **Image**: A blueprint/template for creating containers
- **Registry**: A repository for storing and sharing Docker images
- **Layer**: Docker images are built in layers, each representing a set of changes

Installation

Windows Installation

```
# Download Docker Desktop from https://www.docker.com/products/docker-desktop
# Or use Windows Package Manager
winget install Docker.DockerDesktop
```

Linux Installation (Ubuntu)

```
# Update package manager
sudo apt-get update

# Install dependencies
sudo apt-get install -y apt-transport-https ca-certificates curl software-pro

# Add Docker repository
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/u

# Install Docker
```

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# Start Docker daemon
sudo systemctl start docker

# Add current user to docker group (optional, requires logout/login)
sudo usermod -aG docker $USER
```

macOS Installation

```
# Using Homebrew
brew install docker
brew install --cask docker

# Or download Docker Desktop directly
# https://www.docker.com/products/docker-desktop
```

First Steps: Verify Installation

```
# Check Docker version
docker --version

# Example output:
# Docker version 26.0.0, build 1a79695

# Check Docker info
docker info

# Example output:
# Client:
#  Version:    26.0.0
#  Context:    default
#  Debug Mode: false
#  Plugins:
#   buildx: Docker Buildx (Docker Inc.)
#   compose: Docker Compose (Docker Inc.)
# Server:
#  Containers: 2
#  Running: 0
#  Paused: 0
#  Stopped: 2
#  Images: 5
```

Your First Container

Your First Container

Pull an Image

```
# Download the official Hello World image
docker pull hello-world

# Output:
# Using default tag: latest
# latest: Pulling from library/hello-world
# c1f3f02b5547: Pull complete
# Digest: sha256:d000bc569937abbe195e20322a526...
# Status: Downloaded newer image for hello-world:latest
```

Run Your First Container

```
# Run the hello-world container
docker run hello-world

# Output:
# Hello from Docker!
# This message shows that your installation appears to be working correctly.
#
# To generate this message, Docker took the following steps:
#  1. The Docker client contacted the Docker daemon
#  2. The Docker daemon pulled the "hello-world" image from the Docker Hub
#  3. The daemon created a new container from that image
#  4. The daemon streamed that output to the Docker client
```

Common Basic Commands

Image Management

```
# List all images
docker images

# Output:
# REPOSITORY          TAG       IMAGE ID       CREATED      SIZE
# ubuntu              latest    ba6acccedd29   2 weeks ago  77.8MB
# hello-world         latest    d2c94e258dcb   4 months ago 13.3kB
# nginx               latest    605c77e624dd   3 days ago   187MB

# Search for images on Docker Hub
docker search nginx
```

```
# Example output (first few results):
# NAME                              DESCRIPTION                     STARS   AUTO
# nginx                             Official build of Nginx.        19k     [OK]
# bitnami/nginx                     Bitnami nginx Docker Image      539
# ubuntu/nginx                      Nginx, a high-performance...    73

# Remove an image
docker rmi hello-world

# Tag an image
docker tag ubuntu:latest myubuntu:v1.0
```

Container Management

```
# List running containers
docker ps

# Output:
# CONTAINER ID   IMAGE     COMMAND            CREATED      STATUS      PORTS
# a1b2c3d4e5f6   nginx:latest   "nginx -g..."   2 hours ago  Up 2 hours  0.0.

# List all containers (including stopped)
docker ps -a

# Output:
# CONTAINER ID   IMAGE         COMMAND                  CREATED       STATUS
# f1e2d3c4b5a6   ubuntu:latest "/bin/bash"              3 days ago    Exited
# a1b2c3d4e5f6   nginx:latest  "nginx -g 'daemon..."   2 hours ago   Up 2 hou

# Run a container with a name
docker run --name my-ubuntu -it ubuntu:latest /bin/bash

# Start a stopped container
docker start f1e2d3c4b5a6

# Stop a running container
docker stop a1b2c3d4e5f6

# Remove a container
docker rm f1e2d3c4b5a6

# View container logs
docker logs a1b2c3d4e5f6

# Example output:
# 2024-02-17 10:15:23.123456789 +0000 UTC: INFO Starting service...
```

```
# 2024-02-17 10:15:24.123456789 +0000 UTC: INFO Service ready on port 8080

# Inspect a container
docker inspect a1b2c3d4e5f6 | head -30

# Example output (abbreviated):
# [
#     {
#         "Id": "a1b2c3d4e5f6...",
#         "Created": "2024-02-17T10:00:00...",
#         "State": {
#             "Status": "running",
#             "Running": true,
#             "Paused": false
#         },
#         "Image": "sha256:1234567890...",
#         "Name": "/my-nginx",
#         "Config": {
#             "Image": "nginx:latest",
#             "Hostname": "a1b2c3d4e5f6",
#             "Env": [...]
#         }
#     }
# ]
```

Interactive Mode

```
# Run a container in interactive mode
docker run -it ubuntu:latest /bin/bash

# Output: You're now inside the container
# root@a1b2c3d4e5f6:/#

# Inside the container, you can run commands
# root@a1b2c3d4e5f6:/# ls
# bin   dev  home  lib  media  proc  run  srv  sys  tmp  usr  var
# root@a1b2c3d4e5f6:/# uname -a
# Linux a1b2c3d4e5f6 5.15.0-1234-generic #1234-Ubuntu SMP x86_64 GNU/Linux

# Exit the container
# root@a1b2c3d4e5f6:/# exit
```

Background Mode (Detached)

```
# Run a container in the background
```

```
docker run -d --name web-server nginx:latest

# Output: The container ID (first 12 characters shown)
# a1b2c3d4e5f6a0b1c2d3

# Check its status
docker ps

# Output:
# CONTAINER ID   IMAGE          COMMAND                  CREATED         STAT
# a1b2c3d4e5f6   nginx:latest   "nginx -g 'daemon..."   5 seconds ago   Up 4
```

Understanding Container Lifecycle

```
# 1. Create a container
docker create --name my-app nginx:latest
# Output: a1b2c3d4e5f6a0b1c2d3

# 2. Start the container
docker start my-app

# 3. Check it's running
docker ps

# 4. Pause the container
docker pause my-app

# 5. Unpause the container
docker unpause my-app

# 6. Restart the container
docker restart my-app

# 7. Stop the container gracefully (15 second timeout)
docker stop my-app

# 8. Kill the container (immediately)
docker kill my-app

# 9. Remove the container
docker rm my-app
```

Next Steps

- Learn how to create custom images with Docker Images & Dockerfile

- Understand container management in Docker Containers