**Docker Basics Tutorial Documentation**

**Introduction**

This document provides a comprehensive guide to Docker basics based on the official Docker documentation. It covers core concepts, practical examples, and step-by-step instructions for working with Docker containers, images, and Docker Compose.

**1. Docker Core Concepts**

**1.1 What is a Container?**

A container is a standardized unit of software that packages an application and all its dependencies, ensuring it works seamlessly across different environments.

**Key Characteristics:**

* Lightweight and portable
* Isolated from other containers and the host system
* Shares the host OS kernel
* Contains everything needed to run an application

**1.2 What is a Docker Image?**

A Docker image is a read-only template containing instructions for creating a Docker container.

**Key Points:**

* Immutable (cannot be changed once created)
* Built from a series of layers
* Can be versioned using tags
* Served from container registries (like Docker Hub)

**1.3 What is Docker Compose?**

Docker Compose is a tool for defining and running multi-container Docker applications.

**Benefits:**

* Configure all services in a single YAML file
* Create and start all services with a single command
* Scale specific services independently
* Maintain consistency across environments

**2. Installation Guide**

**2.1 Installing Docker Desktop on Windows**

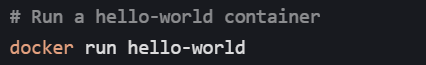
1. Download Docker Desktop from the [official website](https://docs.docker.com/desktop/install/windows-install/)
2. Run the installer and follow the installation wizard
3. Ensure "Use WSL 2 instead of Hyper-V" is selected (recommended)
4. Complete the installation and restart your computer

**2.2 Configuring WSL 2 Integration**

1. Open Docker Desktop
2. Go to Settings > Resources > WSL Integration
3. Enable integration with your preferred WSL 2 distributions
4. Click "Apply & Restart"

**3. Working with Containers**

**3.1 Running Your First Container**



This command:

1. Checks if the hello-world image exists locally
2. If not, pulls it from Docker Hub
3. Creates a new container from the image
4. Runs the container, which prints a message and exits

**3.2 Publishing Ports**

To make a container's service accessible from the host machine:



* The -p flag maps port 8080 on the host to port 80 in the container
* Access the service at [http://localhost:8080](http://localhost:8080/)

**3.3 Overriding Container Defaults**

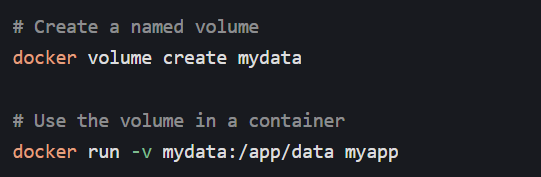
Override a container's default command:



* -it flags provide interactive terminal access
* /bin/bash overrides the default command, starting a bash shell

**3.4 Persisting Container Data**

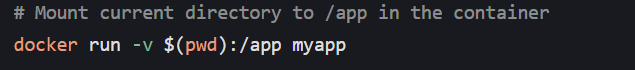
Use volumes to persist data beyond a container's lifecycle:



Volumes are managed by Docker and stored in a part of the host filesystem that's managed by Docker.

**3.5 Sharing Local Files**

Mount a local directory into a container:

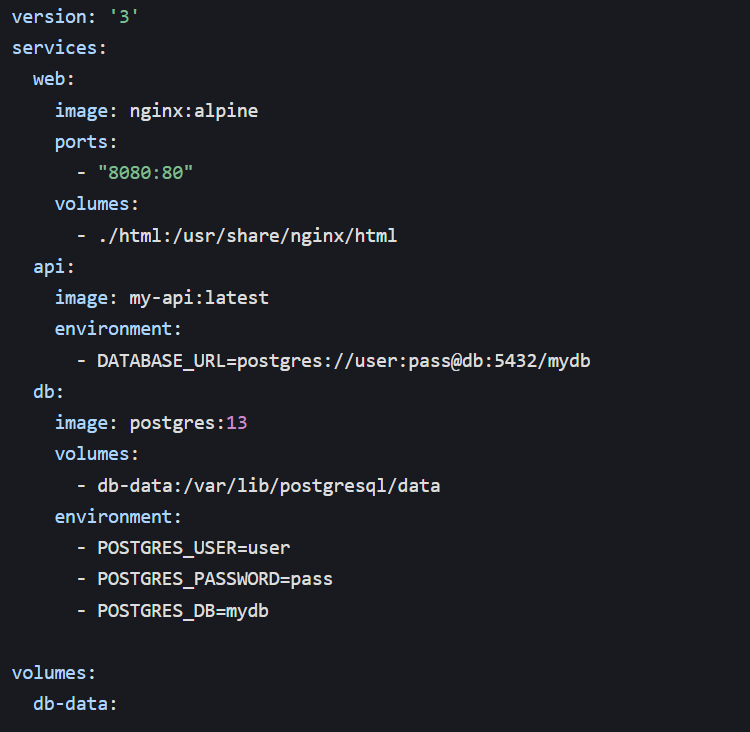


* Changes made in the container's /app directory affect the local directory
* Changes in the local directory are immediately visible in the container

**4. Multi-Container Applications**

**4.1 Using Docker Compose**

Create a docker-compose.yml file:



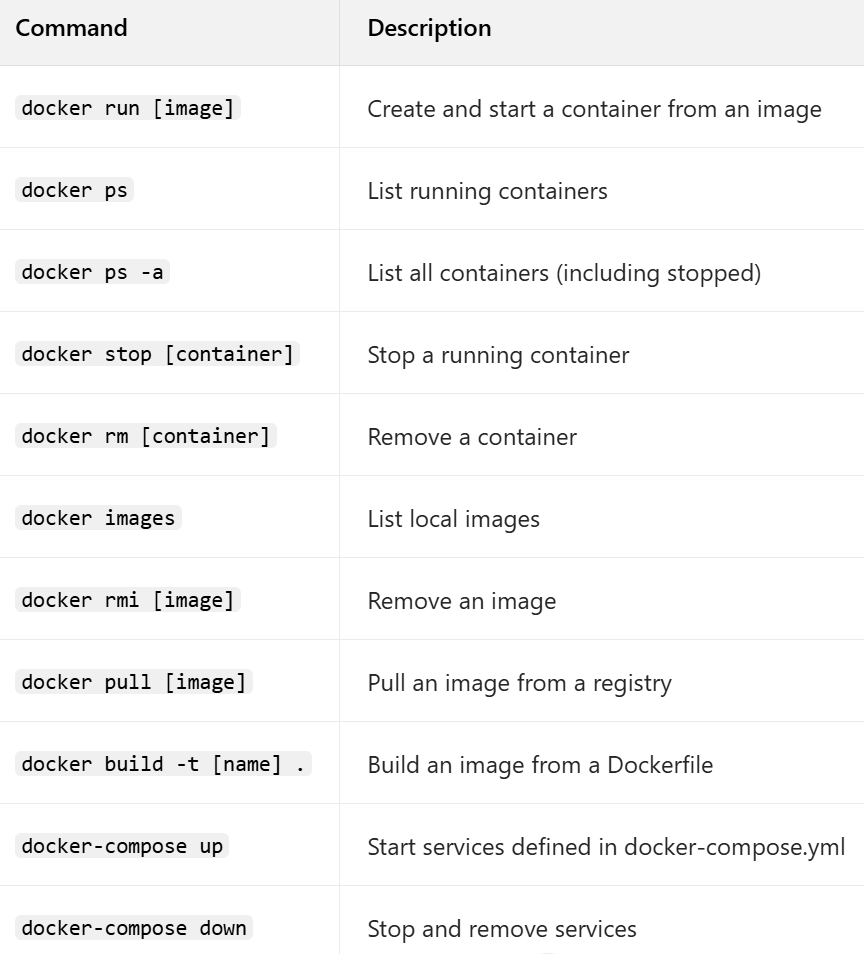
Start the application:



Stop the application:



**5. Key Commands Summary**



**Conclusion**

This guide covers the fundamental concepts and operations needed to work effectively with Docker. By leveraging containers, images, and Docker Compose, you can create consistent, portable, and scalable applications that work seamlessly across different environments.