# Basic Commands

Certainly! Here are some commonly used Docker commands for managing containers:

1. **Image Management:**

* docker pull <image>: Download an image from a registry.
* docker images: List the downloaded images on your system.
* docker rmi <image>: Remove an image from your system.

1. **Container Lifecycle:**

* docker run <image>: Create and start a new container from an image.
* docker ps: List running containers.
* docker stop <container>: Stop a running container.
* docker start <container>: Start a stopped container.
* docker restart <container>: Restart a running container.
* docker rm <container>: Remove a stopped container.

1. **Container Interaction and Information:**

* docker exec -it <container> <command>: Run a command inside a running container.
* docker logs <container>: View the logs of a container.
* docker inspect <container>: Display detailed information about a container.
* docker port <container>: List the ports exposed by a container.

1. **Container Networking:**

* docker network ls: List the available networks.
* docker network create <network>: Create a new network.
* docker network connect <network> <container>: Connect a container to a network.
* docker network disconnect <network> <container>: Disconnect a container from a network.

1. **Docker Compose (for multi-container applications):**

* docker-compose up: Create and start containers defined in a Compose file.
* docker-compose down: Stop and remove containers defined in a Compose file.
* docker-compose logs: View the logs of containers defined in a Compose file.

These are just a few examples of Docker commands. Docker provides a wide range of functionality for container management and orchestration. You can explore more commands and options in the Docker documentation and reference guides.

# Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define a set of services, their configurations, and their dependencies in a YAML file. With a single command, you can create and manage multiple containers that work together as a cohesive application.

Here’s an example of a Docker Compose file for a FastAPI application:

version: '3'  
services:  
 app:  
 build:  
 context: .  
 dockerfile: Dockerfile  
 ports:  
 - '8000:8000'  
 volumes:  
 - .:/app  
 depends\_on:  
 - db  
 db:  
 image: postgres:13  
 environment:  
 - POSTGRES\_USER=myuser  
 - POSTGRES\_PASSWORD=mypassword  
 - POSTGRES\_DB=mydb  
 volumes:  
 - db\_data:/var/lib/postgresql/data  
volumes:  
 db\_data:

Explanation of the Docker Compose file:

* version: '3': Specifies the version of the Docker Compose file syntax.
* services: Defines the individual services or containers in your application.
  + app: Defines the service for the FastAPI application.
    - build: Specifies the build configuration for the service.
      * context: .: Sets the build context to the current directory (where the Docker Compose file is located).
      * dockerfile: Dockerfile: Specifies the Dockerfile to use for building the image.
    - ports: Maps the host’s port to the container’s port. In this example, the application inside the container will be accessible on localhost:8000.
    - volumes: Mounts the current directory to the /app directory inside the container, allowing live code changes without rebuilding the image.
    - depends\_on: Defines the dependency of the app service on the db service. Docker Compose ensures that the db service starts before the app service.
  + db: Defines the service for a PostgreSQL database.
    - image: postgres:13: Specifies the image to use for the db service.
    - environment: Sets the environment variables for the PostgreSQL container.
    - volumes: Mounts a volume to persist the PostgreSQL data.
* volumes: Defines the named volume db\_data for persisting the PostgreSQL data.

To use this Docker Compose file, save it as docker-compose.yml in your project directory. Then, from the same directory, you can run the following command to start the application:

docker-compose up

Docker Compose will build the Docker image for the FastAPI application, start the PostgreSQL container, and create the necessary network and dependencies between the services.

This example demonstrates how Docker Compose simplifies the management of multi-container applications by defining their configurations, dependencies, and network in a single YAML file.