Docker: Definition and Concepts

- **Docker** is a platform that allows you to package applications and their dependencies into containers, ensuring they run the same everywhere.
- **Image:** A read-only template (blueprint) with instructions for creating a container. Built from a Dockerfile.
- **Container:** A running instance of an image. It's isolated, lightweight, and contains everything needed to run your app.

Difference: Image vs Container

Image (Blueprint)	Container (Running Instance)
Built from a Dockerfile	Created from an image
Read-only, never changes	Read-write, can be started/stopped
Can be stored and shared	Runs your app, isolated environment
Example: php:8.2-apache	Example: laravel_app

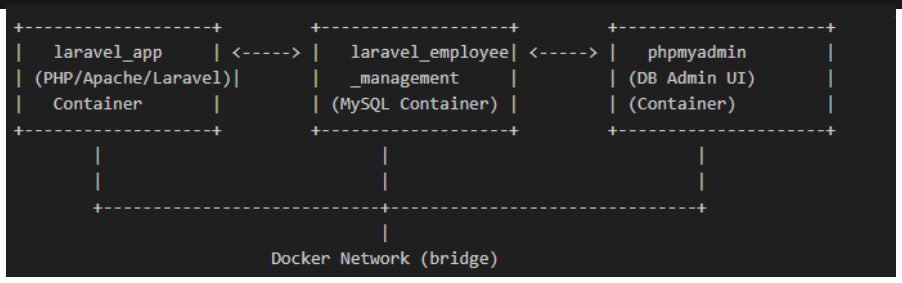
Docker Workflow Diagram

For exemple :

In your project:

- You write a **Dockerfile** for Laravel.
- Docker builds an **image** from it.
- Docker runs a **container** from that image.

Your Laravel Docker Setup: Architecture Diagram



Key Docker Commands (with explanations)

Command	Description
docker-compose build	Build images as defined in docker-compose.yml and Dockerfile.
docker-compose up -d	Start all services in detached mode (background).
docker-compose down	Stop and remove all containers and networks.
docker-compose ps	List running containers.
docker-compose logs app	View logs for the Laravel app container.
docker-compose exec app bash	Open a shell inside the app container.
docker-compose exec app php artisan migrate	Run Laravel migrations inside the container.
docker-compose exec app npm install	Install Node.js dependencies (if using Laravel Mix/Vite).
docker-compose exec app npm run dev	Build frontend assets (if using Laravel Mix/Vite).
docker volume ls	List all Docker volumes.
docker volume inspect <volume_name>	Show details about a Docker volume.

Summary Table for Revision

Step	What You Did	Why
Dockerfile edits	Set up PHP, Apache, Composer, permissions	To run Laravel in a container
docker-compose.yml	Defined services, ports, volumes	To manage all containers together
Key commands	Build, run, debug containers	For development workflow
Troubleshooting	Fixed permissions, port conflicts	To ensure app runs smoothly

Best Practices & Troubleshooting

- **Forbidden Error:** Usually caused by wrong Apache config or permissions. Fixed by setting DocumentRoot to /var/www/html/public and correcting permissions.
- **Port Conflicts:** If port 3306 is in use (e.g., by XAMPP), change to another port in docker-compose.yml and .env.
- **Database Data Location:** MySQL data is stored in Docker volumes, not your project folder. Example: C:\ProgramData\Docker\volumes\first_project_db_data_data
- **phpMyAdmin:** Access at http://localhost:8080 to manage your database visually.