Docker Guide for Beginners

Docker Guide for Beginners

Anonymous · 09/09/2026

Docker Guide for Beginners

This guide provides an introduction to Docker for developers who are new to containerization. It covers basic concepts, essential commands, and best practices for working with Docker in the context of our PHP-Caddy server environment.

Introduction to Docker

Docker is a platform that enables developers to build, package, and run applications in containers. Containers are lightweight, portable, and self-sufficient environments that include everything an application needs to run.

Benefits of using Docker:

- **Consistency**: Applications run the same way across different environments.
- **Isolation**: Applications run in isolated environments without interfering with each other.
- **Portability**: Containers can run on any system that has Docker installed.
- **Efficiency**: Containers share the host system's kernel and use fewer resources than virtual machines.

Key Concepts

- **Container**: A runnable instance of an image that is isolated from the host and other containers. Think of it as a lightweight virtual machine.
- **Image**: A read-only template used to create containers. Images include the application code, runtime, libraries, and dependencies.
- **Dockerfile**: A text file that contains instructions to build a Docker image.
- **Docker Compose**: A tool for defining and running multi-container Docker applications using a YAML file.
- **Volume**: A persistent data storage mechanism that exists outside of containers.
- **Registry**: A repository for storing and distributing Docker images (e.g., Docker Hub).

Installation

macOS

1. Download Docker Desktop from https://www.docker.com/products/docker-desktop
2. Install the application
3. Start Docker Desktop

Windows

1. Download Docker Desktop from https://www.docker.com/products/docker-desktop
2. Install the application
3. Enable WSL 2 (Windows Subsystem for Linux) if prompted
4. Start Docker Desktop

Linux (Ubuntu)

```
# Update package index.
sudo apt-get update

# Install dependencies.
sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

# Add Docker's official GPG key.
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg

# Set up the stable repository.
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine.
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io

# Install Docker Compose.
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-$(uname -
s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Basic Docker Commands

Check Docker installation

```
docker --version
docker compose version
```

Working with Images

List all images

```
docker images
```

Pull an image from a registry

```
docker pull php:8.5-fpm
```

Build an image from a Dockerfile

```
docker build -t my-app:latest .
```

Remove an image

```
docker rmi image_name
```

Working with Containers

List running containers

```
docker ps
```

List all containers (including stopped)

```
docker ps -a
```

Create and start a container

```
docker run -d --name my-container image_name
```

Stop a container

```
docker stop container_name
```

Start a stopped container

```
docker start container_name
```

Remove a container

```
docker rm container_name
```

Execute a command in a running container

```
docker exec -it container_name bash
```

View container logs

```
docker logs container_name
```

Follow container logs in real-time

```
docker logs -f container_name
```

Working with Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file to configure all the application's services, networks, and volumes.

Basic Docker Compose Commands

Start all services defined in docker-compose.yml

```
docker compose up -d
```

The `-d` flag runs containers in the background (detached mode).

Stop all services

```
docker compose down
```

View logs from all services

```
docker compose logs
```

View logs from a specific service

```
docker compose logs service_name
```

Follow logs in real-time

```
docker compose logs -f
```

Rebuild services

```
docker compose build --no-cache
```

Restart a specific service

```
docker compose restart service_name
```

Execute a command in a service container

```
docker compose exec service_name command
```

Example: Access bash in the webserver service

```
docker compose exec webserver bash
```

Docker Compose Override

This project uses Docker Compose's override functionality to separate different configurations:

Use both docker-compose.yml and docker-compose.override.yml

```
docker compose up -d
```

Use only docker-compose.yml

```
docker compose -f docker-compose.yml up -d
```

Docker Compose Environment Variables

Docker Compose can use environment variables from:

1. An `.env` file in the same directory.

2. Environment variables set in the shell.
3. Default values specified in the docker-compose.yml file.

Example .env File

```
SERVER_NAME=derafu-sites-server
HTTP_PORT=8080
HTTPS_PORT=8443
SSH_PORT=2222
```

Variable Substitution in docker-compose.yml

```
services:
  webserver:
    ports:
      - "${HTTP_PORT:-8080}:80"
```

The syntax `${VARIABLE:-default}` means “use the value of VARIABLE if set, otherwise use ‘default’”.

Common Workflows

Initial Setup

1. Clone the repository:

   ```
   git clone https://github.com/derafu/docker-php8.5-caddy-server.git
   cd docker-php8.5-caddy-server
   ```

2. Copy the example .env file:

   ```
   cp .env-dist .env
   ```

3. Add your SSH public key for deployment access:

   ```
   cat ~/.ssh/id_rsa.pub > config/ssh/authorized_keys
   ```

4. Build and start the containers:

   ```
   docker compose up -d
   ```

Adding a New Site

1. Create the site directory structure:

   ```
   mkdir -p sites/www.newsite.com/public
   echo "<?php phpinfo();" > sites/www.newsite.com/public/index.php
   ```

2. For local development, add to your hosts file:

   ```
   127.0.0.1 www.newsite.com.local
   ```

3. Access the site at https://www.newsite.com.local:8443

Updating After Configuration Changes

```
docker compose build --no-cache
docker compose up -d
```

Accessing the Container

```
docker compose exec webserver bash
```

Checking Logs

```
# View Caddy logs.
docker compose exec webserver cat /var/log/caddy/access.log

# Follow PHP-FPM logs.
docker compose exec webserver tail -f /var/log/php-fpm.log
```

Troubleshooting

Container Won’t Start

1. Check for port conflicts:

   ```
   netstat -tuln | grep 8080
   ```

2. Check container logs:

```
docker compose logs webserver
```

Permission Issues

If you encounter permission issues with mounted volumes:

```
docker compose exec webserver chown -R admin:admin /var/www/sites
```

Network Issues

If containers can't communicate:

1. Check if containers are on the same network:

   ```
   docker network ls
   docker network inspect <network_name>
   ```

2. Check if services are running:

   ```
   docker compose ps
   ```

Rebuilding from Scratch

If you need to start over completely:

```
# Stop and remove containers, networks, volumes, and images.
docker compose down -v --rmi all

# Rebuild and start
docker compose up -d --build
```

Best Practices

1. **Use .dockerignore files**: Exclude unnecessary files from the build context.
2. **Minimize image layers**: Combine RUN commands where possible.
3. **Use specific tags for base images**: Avoid using 'latest' which can change unexpectedly.
4. **Keep containers stateless**: Store persistent data in volumes.
5. **Use environment variables** for configuration that changes between environments.
6. **Regularly update base images** to get security patches.
7. **Use health checks** to ensure services are running correctly.

8. **Limit container privileges** for better security.

Further Resources

- Official Docker Documentation
- Docker Compose Documentation
- Docker Hub - Repository of Docker images.
- Docker Curriculum - Comprehensive Docker tutorial.
- Play with Docker - Browser-based Docker playground.

Last updated on 09/09/2026