Docker Documentation Guide

This guide provides an overview of Docker and step-by-step instructions for using it effectively.

1. Introduction to Docker

Docker is a platform for developing, shipping, and running applications in lightweight, portable containers. It helps developers ensure consistency across environments and simplifies deployment.

Key Benefits:

- Lightweight and efficient
- Simplifies CI/CD workflows
- Cross-platform support

2. Installation

Prerequisites

- A 64-bit OS: Windows, macOS, or Linux.
- Virtualization enabled on the system.

Installation Steps

1. **Windows/Mac:**
    - Download Docker Desktop from Docker's official website.
    - Follow the installer and complete the setup.
2. **Linux:**
    - Use your distribution’s package manager:

```
sudo apt-get update

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

3. Key Docker Concepts

Containers

Lightweight, standalone executable packages that include everything needed to run an application.

Images

Templates used to create containers. Think of them as snapshots of a virtual machine.

Dockerfile

A text file containing instructions to build a Docker image.

Volumes

Used to persist data generated and used by Docker containers.

Docker Hub

A public repository to store and share container images.

4. Basic Commands

Command	Description
docker --version	Check the Docker version.
docker run <image>	Run a container from an image.
docker ps	List running containers.
docker ps -a	List all containers (running + stopped).
docker stop <container_id>	Stop a running container.
docker rm <container_id>	Remove a stopped container.
docker images	List all Docker images.
docker rmi <image_id>	Remove an image.
docker logs <container_id>	View logs of a container.
docker exec -it <container_id> bash	Access a running container's terminal.

5. Working with Docker Images

Pulling an Image

```
docker pull <image_name>:<tag>
```

Example:

```
docker pull nginx:latest
```

Building an Image

Create a Dockerfile:

```
FROM node:14
WORKDIR /app
COPY . .
RUN npm install
CMD ["npm", "start"]
```

1. Build the image:

   ```
   docker build -t <image_name>:<tag> .
   ```

   Example:

   ```
   docker build -t my-app:1.0 .
   ```

6. Managing Containers

Starting a Container

```
docker run -d -p <host_port>:<container_port> <image_name>
```

Example:

```
docker run -d -p 8080:80 nginx
```

Stopping a Container

```
docker stop <container_id>
```

Removing a Container

```
docker rm <container_id>
```

Viewing Logs

```
docker logs <container_id>
```

7. Using Docker Compose

Docker Compose simplifies managing multi-container applications.

Compose File (docker-compose.yml)

Example:

```
version: "3"
services:
      web:
        image: nginx
        ports:
         - "8080:80”
      app:
        build:
         context: .
        volumes:
         - "./app:/app"
        depends_on:
         - db
       db:
         image: postgres
         environment:
           POSTGRES_USER: user
           POSTGRES_PASSWORD: pass
```

Commands

Command	Description
`docker-compose up`	Start all services.
`docker-compose down`	Stop all services.
`docker-compose build`	Build or rebuild services.
`docker-compose logs`	View logs of services.

8. Best Practices

- Use lightweight base images (e.g., `alpine`).
- Keep Dockerfile commands simple and layered.
- Use `.dockerignore` to exclude unnecessary files.
- Always tag images with meaningful tags.
- Use volumes for persistent data storage.

---

9. Troubleshooting

Check Docker Logs

```
docker logs <container_id>
```

Inspect a Container

```
docker inspect <container_id>
```

Prune Unused Resources

```
docker system prune -f
```

Networking Issues

Check container connectivity:

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

10. Additional Resources

- Official Docker Documentation
- Docker Hub
- Docker Cheatsheet

This guide covers the essentials for getting started with Docker. For more complex scenarios, refer to Docker's official documentation or community forums.