Docker: Full Documentation

What is Docker?

Docker is an open-source platform that automates the deployment, scaling, and management of applications within lightweight, portable containers. Containers package an application and its dependencies, ensuring that the application runs consistently across different computing environments, from a developer's local machine to production.

Why Use Docker?

- **Consistency**: Docker ensures that your application runs the same way, regardless of where it's deployed, eliminating the "it works on my machine" problem.
- **Isolation**: Containers isolate applications from each other and the host system, reducing conflicts and enhancing security.
- **Efficiency**: Docker containers are lightweight and share the host system's OS kernel, making them faster and more resource-efficient compared to traditional virtual machines.
- **Portability**: Docker containers can be easily moved between environments, such as from a developer's laptop to a production server or cloud.
- **Scalability**: Docker makes it easy to scale applications by spinning up or down additional containers.

Components of Docker

1. **Docker Engine**: The core part of Docker that creates and manages containers.
2. **Docker Images**: Read-only templates that contain the application and its dependencies.
3. **Docker Containers**: Instances of Docker images running as isolated processes.
4. **Dockerfile**: A script that contains instructions to create a Docker image.
5. **Docker Hub**: A cloud-based repository for sharing Docker images.

Setting Up Docker

1. Installation

- **Linux**:

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

  Add your user to the `docker` group:

```
sudo usermod -aG docker $USER
```

Test Docker installation:

```
docker --version
```

- **macOS**:

  1. Download Docker Desktop from Docker's official website.
  2. Install and start Docker Desktop.
  3. Test installation with:

  ```
  docker --version
  ```

- **Windows**:

  1. Download Docker Desktop from Docker's official website.
  2. Install and start Docker Desktop.
  3. Test installation with:

  ```
  docker --version
  ```

2. Basic Docker Commands

- **Pull an Image**:

  ```
  docker pull <image_name>
  ```

  Example:

  ```
  docker pull nginx
  ```

- **Run a Container**:

  ```
  docker run <image_name>
  ```

  Example:

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

  Flags:

  - `-d`: Run container in detached mode (in the background).
  - `-p`: Map container port to host port.

- **List Running Containers**:

  ```
  docker ps
  ```

- **List All Containers**:

  ```
  docker ps -a
  ```

- **Stop a Container**:

  ```
  docker stop <container_id>
  ```

- **Remove a Container**:

  ```
  docker rm <container_id>
  ```

- **Remove an Image**:

  ```
  docker rmi <image_id>
  ```

- **Build an Image**:

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

- **Tag an Image**:

  ```
  docker tag <image_name> <repository>/<image_name>:<tag>
  ```

- **Push an Image to Docker Hub**:

  ```
  docker push <repository>/<image_name>:<tag>
  ```

- **Login to Docker Hub**:

  ```
  docker login
  ```

Dockerfile

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

Example Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80
```

```
# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you can use a YAML file to configure your application’s services, and then create and start all the services with a single command.

Installation

- **Linux**:

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

- **macOS** and **Windows**: Docker Compose is included with Docker Desktop.

Basic Docker Compose Commands

- **Start Services**:

  ```
  docker-compose up
  ```

- **Start Services in Detached Mode**:

  ```
  docker-compose up -d
  ```

- **Stop Services**:

  ```
  docker-compose down
  ```

- **View Logs**:

  ```
  docker-compose logs
  ```

- **Build Services**:

  ```
  docker-compose build
  ```

Example `docker-compose.yml` File

```yaml
version: '3'
services:
  web:
    image: nginx
    ports:
      - "80:80"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

Conclusion

Docker simplifies the process of managing, deploying, and scaling applications. It ensures consistency across various environments, making it an essential tool for developers and operations teams. Docker Compose further extends Docker's capabilities by making it easy to manage multi-container applications.