Docker Get Started

Contents

Time required: 60 minutes

What is Docker?

Docker is a platform that allows you to package an application and its dependencies into a "container." A container is a lightweight, standalone, and executable package that includes everything needed to run the application: code, runtime, libraries, and system tools.

Why Use Docker?

1. **Consistency**: Containers ensure that your application runs the same way, regardless of where it's deployed. This eliminates the "it works on my machine" problem.

2. **Isolation**: Each container runs in its own isolated environment, which means you can run multiple containers on the same host without conflicts.
3. **Portability**: Containers can run on any system that supports Docker, making it easy to move applications between different environments (development, testing, production).
4. **Efficiency**: Containers are lightweight and use fewer resources compared to traditional virtual machines.

How Does Docker Work?

1. **Dockerfile**: A text file that contains instructions for building a Docker image. It specifies the base image, application code, dependencies, and commands to run.

- **Docker Image**: A read-only template created from a Dockerfile. It contains the application and its environment.
- **Docker Container**: A running instance of a Docker image. It includes everything needed to execute the application.

Example Use Case

Imagine you're developing a web application. With Docker, you can:

1. Create a Dockerfile that specifies the environment (e.g., Python, Node.js).
2. Build a Docker image from the Dockerfile.
3. Run the image as a container on your local machine.
4. Share the image with your team, who can run it on their machines without worrying about setup or dependencies.

Summary

Docker simplifies the process of developing, testing, and deploying applications by providing a consistent and isolated environment. It's a powerful tool for modern software development, especially when working with microservices and cloud-native applications.

Setup Docker on Kali Linux

Login to your Kali Linux VM.

Step 1: Update Your System

Update your package list and upgrade your existing packages.

```
sudo apt update
sudo apt upgrade -y
```

Step 2: Install Docker

Install Docker using the following commands:

```
sudo apt install -y docker.io
```

Step 3: Start and Enable Docker

Start the Docker service and enable it to start on boot:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Step 4: Verify Docker Installation

Check if Docker is installed correctly by running:

```
docker --version
```

Step 5: Run a Test Container

Run the Hello World container to ensure Docker is working:

```
sudo docker run hello-world
```

If you see a welcome message, Docker is successfully installed and running on your Kali Linux virtual machine.

Additional Tips

- **Running Docker without sudo**: To run Docker commands without sudo, add your user to the docker group:

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

Log out and back in for the changes to take effect.

Hello Docker!

Let's create a simple Python program and then containerize it using Docker.

Step 1: Create a Python Program

Let's create a basic Python program. We'll make a directory for our project and create a simple script.

1. **Create a Project Directory**:

```
mkdir hello_docker
cd hello_docker
```

2. **Create a Python Script**: Create a file named app.py with the following content. You can also use an existing Program.

```
# app.py
print("Hello, Docker!")
```

Step 2: Create a Dockerfile

A Dockerfile is a text document that contains all the commands to assemble a Docker image.

1. **Create a Dockerfile**: In the same directory, create a file named Dockerfile with the following content:

```
# Use the official Python image from the Docker Hub
FROM python:3.13.2-slim

# Set the working directory in the container
WORKDIR /app

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

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

Step 3: Build the Docker Image

Build the Docker image using the Dockerfile.

1. **Build the Image**: Run the following command in your project directory. Keep the . (period) in the command.

```
sudo docker build -t hello_docker .
```

Step 4: Run the Docker Container

Run the Docker container using the image we just built.

1. **Run the Container**:

```
sudo docker run hello_docker
```

You should see the output:

Hello, Docker!

Assignment Submission

1. Attach screenshots showing the successful operation of the program.
2. Submit in Blackboard.