# Write a Dockerfile for a simple web application and build a Docker image

## Table of Contents



## Description

This section provides instructions to create a Dockerfile for a simple **Java-based TodoApp**. It also covers the steps to build and run the Docker image.

## Problem Statement

You want to containerize a simple **Java-based TodoApp**. Using Docker will help ensure that the app runs consistently in any environment, eliminating the "works on my machine" problem. This document explains how to create the required Dockerfile, build the image, and run the container.

## Prerequisites

Completion of all previous lab guides (up to Lab Guide-01) is required before proceeding with Lab Guide-02.

### Software Requirement

Before beginning the installation, ensure the following software is available and compatible:

1. **Docker Desktop**
2. **Windows 10 (64-bit) or higher**: Required for WSL 2 integration.
3. **Windows Subsystem for Linux 2 (WSL 2)**: This enables Linux container support on Windows systems.
4. **Python (Optional)**: If working with Python-based applications within Docker, install Python.

Docker Desktop can run either Windows containers or Linux containers; Linux containers are the default and require WSL 2 for optimal performance on Windows systems.

### Hardware Requirement

Ensure your system meets the following hardware specifications to run Docker Desktop efficiently:

- **CPU**: 64-bit processor with virtualization support (Intel VT-x/AMD-V).
- **RAM**: 4 GB RAM (8 GB or higher recommended).
- **Storage**: 10 GB free disk space for Docker images and containers.
- **Virtualization**: Ensure that hardware virtualization is enabled in BIOS. This is critical for Docker to function.

## Implementation Steps

---

### Step 1: Downloading the TodoApp with H2 Database

To download the **TodoApp** with an H2 database, please follow the steps below:

1. **Click the Download Link**: Download the project from the following link: Download TodoApp with H2 Database
2. **Extract the Zip File**:
    - After downloading, go to your downloads folder.
    - Find the file named `TodoApp_H2-main.zip`.
    - Right-click on the zip file and choose **Extract All** or use any extraction tool to unzip the folder.
3. **Navigate to the Extracted Folder**:
    - Open the folder `TodoApp_H2-main` which contains the project files.
4. **Open the Project in Your IDE**:
    - Open your preferred IDE (e.g., IntelliJ IDEA, Eclipse, or VSCode).
    - Choose **File > Open** and select the extracted `TodoApp_H2-main` folder.

Now you have the **TodoApp with H2 database** downloaded and ready for setup!

### Step 2: Writing the Docker File

- We are working with a simple Todo application, which is a Java-based project built using Maven. The application uses H2 as the database, along with Hibernate for ORM (Object-Relational Mapping).We will write a Dockerfile for the project and proceed to build a Docker image, which will package the entire application and its environment for deployment in any containerized setup.

```
FROM openjdk:11.0.15-jre
ADD target/*.jar app.jar
ENTRYPOINT ["java","-jar","app.jar"]
```

- **FROM openjdk:11.0.15-jre:** This sets the base image for the container, using the Java Runtime Environment (JRE) version 11.0.15. It provides the necessary environment to run Java applications.

- **ADD target/*.jar app.jar:** This copies the JAR file generated by Maven from the **target/** directory on your local machine into the Docker image, renaming it to **app.jar**.
- **ENTRYPOINT ["java", "-jar", "app.jar"]:** This defines the command that runs when the container starts. It tells the container to execute the **app.jar** file using the Java runtime.

### Step 3: Building The Docker Image

- Docker builds images by reading the instructions from a Dockerfile. A Dockerfile is a text file containing instructions for building your source code.
- Open cmd prompt or cd to the source folder path.
- Docker desktop should be running in the background.

```
docker build -t todoapp .
```

The command **docker build -t todoapp .** is used to build a Docker image from a Dockerfile.

- **docker build:** This is the Docker command to build an image.
- **-t todoapp:** The **-t** option tags the image with a name (**todoapp** in this case). It helps identify the image when running or managing containers later.
- **. (dot):** This specifies the build context, which means the current directory. Docker will look for the **Dockerfile** and other necessary files in this directory to create the image.


## References

For more detailed information on Dockerfile best practices and Docker command usage, refer to:

Docker Official Documentation: **click here**