# Jenkins and Docker Project

-Pranjali Datkhile

Project 2

## Introduction

This project is a simple **To-Do List REST API** built using **Java Spring Boot**. It allows users to create, view, and retrieve tasks through HTTP endpoints. The project is integrated with **GitHub** for version control, **Jenkins** for automated CI/CD, and **Docker** for containerization. The final Docker image is pushed to **Docker Hub** for easy deployment. This demonstrates a complete development-to-deployment workflow using modern DevOps tools.

## Technologies Used

|  |  |
| --- | --- |
| **Category** | **Tool/Technology** |
| Backend | Java, Spring Boot |
| Build Tool | Maven |
| Version Control | Git, GitHub |
| CI/CD | Jenkins |
| Containerization | Docker |
| Deployment | Docker Hub |
| Java Version | Java 17 |

## Code

### ToDoListApplication.java

package com.example.to\_do.list;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ToDoListApplication {

public static void main(String[] args) {

SpringApplication.run(ToDoListApplication.class, args);

}

}

### ToDo.java

package com.example.to\_do.list;

public class Todo {

private Long id;

private String task;

private boolean completed;

public Todo() {}

public Todo(Long id, String task, boolean completed) {

this.id = id;

this.task = task;

this.completed = completed;

}

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTask() { return task; }

public void setTask(String task) { this.task = task; }

public boolean isCompleted() { return completed; }

public void setCompleted(boolean completed) { this.completed = completed; }

}

### TodoController.java

package com.example.to\_do.list;

import com.example.to\_do.list.Todo;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/todos")

public class TodoController {

private Map<Long, Todo> todoMap = new HashMap<>();

private Long idCounter = 1L;

@GetMapping

public Collection<Todo> getAllTodos() {

return todoMap.values();

}

@PostMapping

public Todo createTodo(@RequestBody Todo todo) {

todo.setId(idCounter++);

todoMap.put(todo.getId(), todo);

return todo;

}

@GetMapping("/{id}")

public Todo getTodoById(@PathVariable Long id) {

return todoMap.get(id);

}

}

### Jenkins file(CI/CD pipeline)

pipeline {

agent any

environment {

IMAGE\_NAME = 'yourdockerhubusername/todo-list-api'

}

stages {

stage('Checkout') {

steps {

checkout([

$class: 'GitSCM',

branches: [[name: '\*/main']],

userRemoteConfigs: [[

url: 'https://github.com/pranjalixxx/Docker\_project.git',

credentialsId: 'github-credentials'

]] ]) } }

stage('Build with Maven') {

steps {

sh './mvnw clean package -DskipTests'

} }

stage('Build Docker Image') {

steps {

script {

docker.build("${IMAGE\_NAME}")

} } }

stage('Push to Docker Hub') {

steps {

script {

docker.withRegistry('https://index.docker.io/v1/', 'dockerhub-credentials') {

docker.image("${IMAGE\_NAME}").push("latest")

} } } } } }

### Pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>to-do-list</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>to-do-list</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

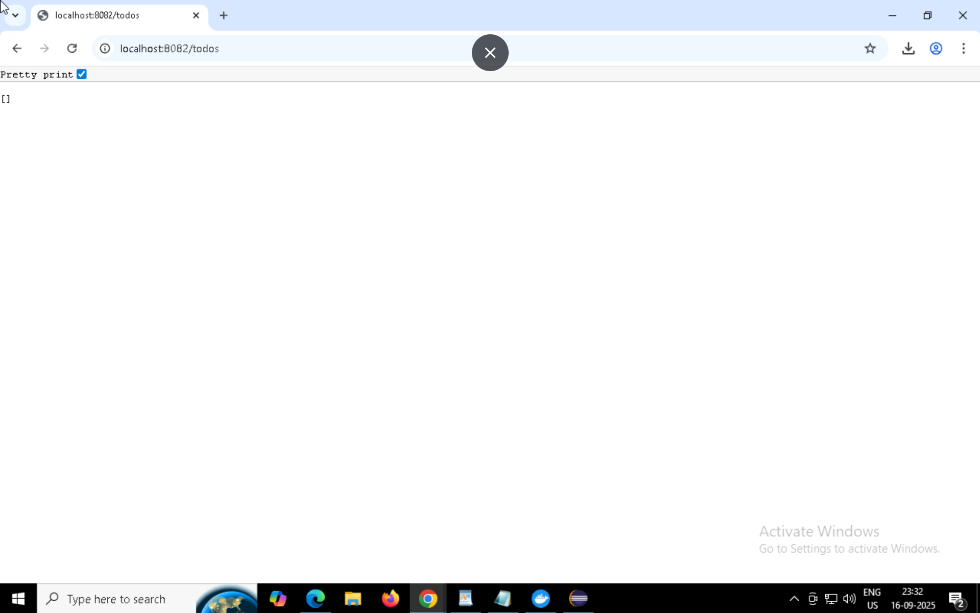
</plugin>

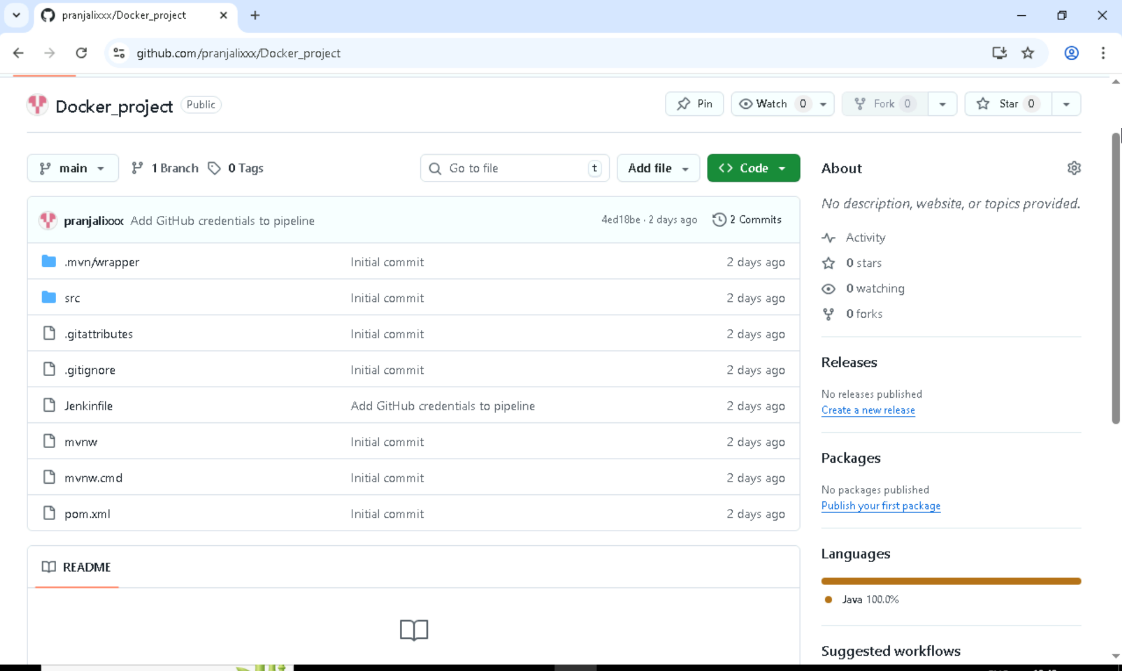
</plugins>

</build>

</project>

## Output





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Conclusion

The project features a **REST API built using the modern Java Spring Boot stack**, providing a simple and efficient way to manage to-do tasks. It includes **end-to-end CI/CD automation through Jenkins**, which handles building, testing, and deploying the application seamlessly. The application is **containerized using Docker**, ensuring consistent environments across development and production. Finally, the Docker image is **publicly hosted on Docker Hub**, making it easily accessible for deployment on any compatible platform.

## Future Scope

To further improve the project, several enhancements can be implemented. These include adding **Update** and **Delete** endpoints to complete the full set of CRUD operations, and integrating the application with a **real database** such as **MySQL** or **PostgreSQL** for persistent data storage. Incorporating **unit and integration tests** would help ensure code quality and reliability. Finally, the project can be scaled and made production-ready by deploying it to **Kubernetes** or popular **cloud platforms** like **AWS**, **Google Cloud (GCP)**, or **Microsoft Azure**.