# Spring Boot Application Deployment Using Jenkins CI & CD | Live Demo | Ashok IT

<https://youtu.be/6fcoYGqvg6o>

Project is created with SpringBoot, import as Existing Maven project and

Check into GitHub repository with git and then build a Jenkins job to clean build a war with Maven and deploy it to Tomcat 9 server. Then it can be used or consumed by Angular client.

1. Pick any good springboot project.

########## Git Command ##################

1. Check into the github with the following command:
2. Git init -b main; Git add . ; git commit -m “comments”;
3. Git push origin main/master depend on the remote branch.
4. Git status and git remote -v to check.
5. Create a new repository on GitHub, copy remote repository URL
6. Git remote add origin <Remote\_URL>; git remote -v to verify
7. Git push origin main ( push the code to git hub )

############ Setting up Jenkins ##################

1. Setting up Jenkins: for this project, I installed it on windows with msi installer under C:\Apps2022
2. <?xml version='1.1' encoding='UTF-8'?>

<jenkins.model.JenkinsLocationConfiguration>

<jenkinsUrl>http://localhost:8008/</jenkinsUrl>

</jenkins.model.JenkinsLocationConfiguration>

This is set in Jenkins.xml file; same directory as exe file.

1. Start Jenkins manually with port 8008; go to “localhost:8008”
2. From Jenkins dashboard, “Manage Jenkins” and “deploy” search to install deploy container plugin
3. From Jenkins dashboard, create a new job with “freestyle”
4. Source Code Management: Git and branch \*/main or feature.
5. Build Triggers: Poll SCM with Schedule
6. Build Envionment: delete workspace before build starts
7. If use Maven, Invoke top-level Maven targets: clean compile package
8. Post-Build Actions: WAR/EAR files, use ? suggestion and set context path like: SpringBootApp
9. Add Containers, in this case Tomcat 9. Apply and Save.
10. Build now and watch the console output. (Done!)

Jenkins stores the configuration for each job within an eponymous directory in jobs/. The job configuration file is config.xml

Building a Spring Boot application in Jenkins (part 1 of microservice devops series)

<https://tomgregory.com/building-a-spring-boot-application-in-jenkins/>

1. Created ThemePark gradle project in Eclipse.
2. Webservice works fine and tested. ./gradlew bootRun or run class.
3. Run Gradle test from eclipse not working but run command line:

./gradlew test worked

1. Get the report from build/reports/tests/test/index.html

Graphical user interface

Description automatically generated

## **3. Running Jenkins**

The next stage is to automatically build our Spring Boot application on commit, or in other words to use continuous integration. This will ensure that our master branch is always in a good state, and anything that gets deployed into production is a working version of our application.

A good devops philosophy is to define all infrastructure and configuration as code. So, we’ll bootstrap Jenkins to start up and configure whatever jobs we need automatically, with minimal manual setup required.

############# I decided to skip the last part of the project one and follow the project 2 with Docker and Jenkins:

<https://tomgregory.com/building-a-spring-boot-application-in-docker-and-jenkins/>