Project Name: CI/CD Deployment for Springboot Application.

GitHub Link:

<https://github.com/tamasjit/SpringJenkinDocker>

## Source Code:

### HelloWorldController.java

package com.javatpoint.controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloWorldController

{

@RequestMapping("/")

public String hello()

{

return "Welcome to Tamasjit's AWS";

}

}

### SpringBootHelloWorldExampleApplication.java

package com.javatpoint;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldExampleApplication

{

public static void main(String[] args)

{

SpringApplication.run(SpringBootHelloWorldExampleApplication.class, args);

}

}

### application.properties:

server.port=8090

server.error.whitelabel.enabled=false

### 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>2.2.2.BUILD-SNAPSHOT</version>

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

    </parent>

    <groupId>com.javatpoint</groupId>

    <artifactId>spring-boot-hello-world-example</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>spring-boot-hello-world-example</name>

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

    <properties>

        <java.version>1.8</java.version>

    </properties>

    <dependencies>

        <dependency>

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

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

    </dependency>

          <dependency>

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

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

<version>2.2.1.RELEASE</version>

<type>pom</type>

</dependency>

<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>

            <exclusions>

                <exclusion>

                    <groupId>org.junit.vintage</groupId>

                    <artifactId>junit-vintage-engine</artifactId>

                </exclusion>

            </exclusions>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

    <repositories>

        <repository>

            <id>spring-milestones</id>

            <name>Spring Milestones</name>

            <url>https://repo.spring.io/milestone</url>

        </repository>

        <repository>

            <id>spring-snapshots</id>

            <name>Spring Snapshots</name>

            <url>https://repo.spring.io/snapshot</url>

            <snapshots>

                <enabled>true</enabled>

            </snapshots>

        </repository>

    </repositories>

    <pluginRepositories>

        <pluginRepository>

            <id>spring-milestones</id>

            <name>Spring Milestones</name>

            <url>https://repo.spring.io/milestone</url>

        </pluginRepository>

        <pluginRepository>

            <id>spring-snapshots</id>

            <name>Spring Snapshots</name>

            <url>https://repo.spring.io/snapshot</url>

            <snapshots>

                <enabled>true</enabled>

            </snapshots>

        </pluginRepository>

    </pluginRepositories>

</project>

### Jenkinsfile:

node {

    def WORKSPACE = "/var/lib/jenkins/workspace/springboot-deploy"

    def dockerImageTag = "springboot-deploy${env.BUILD\_NUMBER}"

    //def DOCKERHUB\_CREDENTIALS=credentials('docker-hub-credentials')

    try{

//          notifyBuild('STARTED')

         stage('Clone Repo') {

            // for display purposes

            // Get some code from a GitHub repository

            git url: 'https://github.com/tamasjit/SpringJenkinDocker.git',

                credentialsId: 'springdeploy-user',

                branch: 'main'

         }

          stage('Build docker') {

                 dockerImage = docker.build("springboot-deploy:${env.BUILD\_NUMBER}")

          }

          stage('Deploy docker'){

                  echo "Docker Image Tag Name: ${dockerImageTag}"

                  sh "docker stop springboot-deploy || true && docker rm springboot-deploy || true"

                  sh "docker run --name springboot-deploy -d -p 8081:8081 springboot-deploy:${env.BUILD\_NUMBER}"

          }

          stage('Push image') {

        /\* Finally, we'll push the image with two tags:

         \* First, the incremental build number from Jenkins

         \* Second, the 'latest' tag.

         \* Pushing multiple tags is cheap, as all the layers are reused. \*/

         environment {

                DOCKER\_HUB\_LOGIN = credentials('docker-hub-credentials')

            }

            sh "docker tag springboot-deploy:${env.BUILD\_NUMBER} tamasjit/springboot-deploy"

            sh "docker login --username=tamasjit --password=123456789"

            sh "docker push tamasjit/springboot-deploy"

        }

    }

    catch(e){

//         currentBuild.result = "FAILED"

        throw es

    }finally{

//         notifyBuild(currentBuild.result)

    }

}

def notifyBuild(String buildStatus = 'STARTED'){

// build status of null means successful

  buildStatus =  buildStatus ?: 'SUCCESSFUL'

  // Default values

  def colorName = 'RED'

  def colorCode = '#FF0000'

  def now = new Date()

  // message

  def subject = "${buildStatus}, Job: ${env.JOB\_NAME} FRONTEND - Deployment Sequence: [${env.BUILD\_NUMBER}] "

  def summary = "${subject} - Check On: (${env.BUILD\_URL}) - Time: ${now}"

  def subject\_email = "Spring boot Deployment"

  def details = """<p>${buildStatus} JOB </p>

    <p>Job: ${env.JOB\_NAME} - Deployment Sequence: [${env.BUILD\_NUMBER}] - Time: ${now}</p>

    <p>Check console output at "<a href="${env.BUILD\_URL}">${env.JOB\_NAME}</a>"</p>"""

  // Email notification

    emailext (

         to: "admin@gmail.com",

         subject: subject\_email,

         body: details,

         recipientProviders: [[$class: 'DevelopersRecipientProvider']]

       )

}

### Dockerfile:

# Docker Build Stage

FROM maven:3-jdk-8-alpine AS build

# Build Stage

WORKDIR /opt/app

COPY ./ /opt/app

RUN mvn clean install -DskipTests

# Docker Build Stage

FROM openjdk:8-jdk-alpine

COPY --from=build /opt/app/target/\*.jar app.jar

ENV PORT 8081

EXPOSE $PORT

ENTRYPOINT ["java","-jar","-Xmx1024M","-Dserver.port=${PORT}","app.jar"]

### Pull from docker hub and run inside any aws instance:

