|  |
| --- |
| **SimpliLearn-DevOps-Project**  **CI/CD Deployment for Springboot Application**  **Submitted by: KAVIN K R** |
| Date of submission : 11-10-2023  GitHub Project Repository URL : [GitHub Link](https://github.com/kavink-r/SimpliLearn-DevOps-Project.git) |
| **Backend Rest API:**  **Project structure:** |
| **productController.java**  package com.example.demo.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.ResponseEntity;  import org.springframework.web.bind.annotation.CrossOrigin;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import com.example.demo.entity.product;  import com.example.demo.service.productService;  @RestController  @RequestMapping("/api/product")  @CrossOrigin(allowedHeaders = "\*")  public class productController {  @Autowired  productService service;  @GetMapping("/")  public String handshake() {  return "Hello";  }    @GetMapping("/all")  public ResponseEntity<List<product>> getAllProducts(){  return ResponseEntity.ok(service.getAllProduct());  }    @GetMapping("/{id}")  public ResponseEntity<product> getById(@PathVariable(name="id")int id){  return ResponseEntity.ok(service.findById(id));  }    @PostMapping("/add")  public ResponseEntity<product> addProduct(@RequestBody product p){  return ResponseEntity.ok(service.addProduct(p));  }  @DeleteMapping("/delete/{id}")  public ResponseEntity<String> deleteProduct (@PathVariable(name="id")int id){  product pr = service.findById(id);  service.deleteProduct(pr);  return ResponseEntity.ok("Deleted id="+pr.getId());  }  } |
| **Product.java**  package com.example.demo.entity;  import javax.persistence.Entity;  import javax.persistence.Id;  @Entity  public class product {  @Id  int id;  String productName;  public product() {  // TODO Auto-generated constructor stub  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getProductName() {  return productName;  }  public void setProductName(String productName) {  this.productName = productName;  }    } |
| **Installing Jenkins on Devops-Demo-Server (AWS EC2 instance):**  curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \  /usr/share/keyrings/jenkins-keyring.asc > /dev/null  echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  /etc/apt/sources.list.d/jenkins.list > /dev/null  sudo apt-get update  sudo apt-get install jenkins |
| **Installing Docker on Devops-Demo-Server and api-server (AWS EC2 instances):**  # Add Docker's official GPG key:  sudo apt-get update  sudo apt-get install ca-certificates curl gnupg  sudo install -m 0755 -d /etc/apt/keyrings  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  sudo chmod a+r /etc/apt/keyrings/docker.gpg  # Add the repository to Apt sources:  echo \  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  "$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  sudo apt-get update  sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin |
| **Dockerfile for building sample Springboot app:**  FROM eclipse-temurin:17-jdk-alpine  WORKDIR /app  VOLUME api  COPY target/\*.jar app.jar  ENTRYPOINT ["java","-jar","/app/app.jar"] |
| **Jenkinsfile for building pipeline:**  pipeline {  agent any  tools {  maven 'maven'  }  stages {  stage("Building the project"){  steps{  bat 'mvn clean package'  }  }  stage("Creating docker image"){  steps{  bat 'docker build -t kavinkr/devops-project:api .'  }  }  stage("Pushing docker image to DockerHub"){  steps{  bat 'docker push kavinkr/devops-project:api'  }  }  stage("Pulling image from docker hub and executing container on api server over SSH connection"){  steps{  sshPublisher(publishers: [sshPublisherDesc(configName: 'api-host', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: 'sudo docker pull kavinkr/devops-project:api', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false, removePrefix: '', sourceFiles: ''), sshTransfer(cleanRemote: false, excludes: '', execCommand: 'sudo docker run -d -p 8090:8080 --env-file appenv.txt --name api kavinkr/devops-project:api', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false, removePrefix: '', sourceFiles: '')], usePromotionTimestamp: false, useWorkspaceInPromotion: false, verbose: false)])  }  }    }  } |