SOURCE CODE FOR CI/CD DEPOLYEMENT WITH SPRINGBOOOT APPLICATION

1. Controller.java

|  |
| --- |
| package com.tomgregory.controller; |
|  |  |
|  | import com.tomgregory.entity.ThemeParkRide; |
|  | import com.tomgregory.repository.ThemeParkRideRepository; |
|  | import org.springframework.http.HttpStatus; |
|  | import org.springframework.http.MediaType; |
|  | import org.springframework.web.bind.annotation.\*; |
|  | import org.springframework.web.server.ResponseStatusException; |
|  |  |
|  | import javax.validation.Valid; |
|  |  |
|  | @RestController |
|  | public class ThemeParkRideController { |
|  | private final ThemeParkRideRepository themeParkRideRepository; |
|  |  |
|  | public ThemeParkRideController(ThemeParkRideRepository themeParkRideRepository) { |
|  | this.themeParkRideRepository = themeParkRideRepository; |
|  | } |
|  |  |
|  | @GetMapping(value = "/ride", produces = MediaType.APPLICATION\_JSON\_VALUE) |
|  | public Iterable<ThemeParkRide> getRides() { |
|  | return themeParkRideRepository.findAll(); |
|  | } |
|  |  |
|  | @GetMapping(value = "/ride/{id}", produces = MediaType.APPLICATION\_JSON\_VALUE) |
|  | public ThemeParkRide getRide(@PathVariable long id){ |
|  | return themeParkRideRepository.findById(id).orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND, String.format("Invalid ride id %s", id))); |
|  | } |
|  |  |
|  | @PostMapping(value = "/ride", consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_VALUE) |
|  | public ThemeParkRide createRide(@Valid @RequestBody ThemeParkRide themeParkRide) { |
|  | return themeParkRideRepository.save(themeParkRide); |
|  | } |
|  | } |

1. Entity

|  |
| --- |
| package com.tomgregory.entity; |
|  |  |
|  | import lombok.Getter; |
|  | import lombok.NoArgsConstructor; |
|  | import lombok.ToString; |
|  |  |
|  | import javax.persistence.Entity; |
|  | import javax.persistence.GeneratedValue; |
|  | import javax.persistence.GenerationType; |
|  | import javax.persistence.Id; |
|  | import javax.validation.constraints.NotEmpty; |
|  |  |
|  | @Entity |
|  | @Getter |
|  | @ToString |
|  | @NoArgsConstructor |
|  | public class ThemeParkRide { |
|  | @Id |
|  | @GeneratedValue(strategy=GenerationType.AUTO) |
|  | private Long id; |
|  | @NotEmpty |
|  | private String name; |
|  | @NotEmpty |
|  | private String description; |
|  | private int thrillFactor; |
|  | private int vomitFactor; |
|  |  |
|  | public ThemeParkRide(String name, String description, int thrillFactor, int vomitFactor) { |
|  | this.name = name; |
|  | this.description = description; |
|  | this.thrillFactor = thrillFactor; |
|  | this.vomitFactor = vomitFactor; |
|  | } |
|  |  |
|  | } |

1. Repository

|  |
| --- |
| package com.tomgregory.repository; |
|  |  |
|  | import com.tomgregory.entity.ThemeParkRide; |
|  | import org.springframework.data.repository.CrudRepository; |
|  | import org.springframework.stereotype.Repository; |
|  |  |
|  | import java.util.List; |
|  |  |
|  | @Repository |
|  | public interface ThemeParkRideRepository extends CrudRepository<ThemeParkRide, Long> { |
|  | List<ThemeParkRide> findByName(String name); |
|  | } |

1. application.java

|  |
| --- |
| package com.tomgregory; |
|  |  |
|  | import com.tomgregory.entity.ThemeParkRide; |
|  | import com.tomgregory.repository.ThemeParkRideRepository; |
|  | import org.springframework.boot.CommandLineRunner; |
|  | import org.springframework.boot.SpringApplication; |
|  | import org.springframework.boot.autoconfigure.SpringBootApplication; |
|  | import org.springframework.context.annotation.Bean; |
|  |  |
|  | @SpringBootApplication |
|  | public class ThemeParkApplication { |
|  | public static void main(String[] args) { |
|  | SpringApplication.run(ThemeParkApplication.class); |
|  | } |
|  |  |
|  | @Bean |
|  | public CommandLineRunner sampleData(ThemeParkRideRepository repository) { |
|  | return (args) -> { |
|  | repository.save(new ThemeParkRide("Rollercoaster", "Train ride that speeds you along.", 5, 3)); |
|  | repository.save(new ThemeParkRide("Log flume", "Boat ride with plenty of splashes.", 3, 2)); |
|  | repository.save(new ThemeParkRide("Teacups", "Spinning ride in a giant tea-cup.", 2, 4)); |
|  | }; |
|  | } |
|  | } |

1. Dockerfile

|  |
| --- |
| FROM openjdk:17 |
|  | ARG JAR\_FILE |
|  | COPY ${JAR\_FILE} app.jar |
|  | ENTRYPOINT ["java", "-jar", "/app.jar"] |

1. Jenkinsfile

|  |
| --- |
| pipeline { |
|  | agent any |
|  | environment { |
|  | AWS\_ACCOUNT\_ID="044003570833" |
|  | AWS\_DEFAULT\_REGION="us-east-1" |
|  | IMAGE\_REPO\_NAME="jenkins-pipeline-build-demo" |
|  | IMAGE\_TAG="latest" |
|  | REPOSITORY\_URI = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${IMAGE\_REPO\_NAME}" |
|  | } |
|  |  |
|  | stages { |
|  |  |
|  | stage('Logging into AWS ECR') { |
|  | steps { |
|  | script { |
|  | sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com" |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | stage('Cloning Git') { |
|  | steps { |
|  | checkout([$class: 'GitSCM', branches: [[name: '\*/main']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: '', url: 'https://github.com/amitk95/jenkinswithdocker.git']]]) |
|  | } |
|  | } |
|  |  |
|  | // Building Docker images |
|  | stage('Building image') { |
|  | steps{ |
|  | script { |
|  | dockerImage = docker.build "${IMAGE\_REPO\_NAME}:${IMAGE\_TAG}" |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // Uploading Docker images into AWS ECR |
|  | stage('Pushing to ECR') { |
|  | steps{ |
|  | script { |
|  | sh "docker tag ${IMAGE\_REPO\_NAME}:${IMAGE\_TAG} ${REPOSITORY\_URI}:$IMAGE\_TAG" |
|  | sh "docker push ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${IMAGE\_REPO\_NAME}:${IMAGE\_TAG}" |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |