**CI/CD** is an approach that increases the frequency of application distribution by introducing automation at the application development stage.

The main concepts related to the**CI/CD**approach are continuous integration, continuous distribution, and continuous deployment. The **CI/CD** represents a solution to the problems posed by the integration of new code segments for development and operations teams (the so-called “integration hell”).

Specifically, the **CI/CD** ensures continuous automation and monitoring throughout the application lifecycle, from the integration and testing phases to distribution and deployment. Together, these practices are often referred to as the CI/CD pipeline, and they rely on agile collaboration between development and operations teams.

Our application will have the following scenarios :

* Get random nations
* Get random currencies
* Get application version
* health check

Dependencies used :

<dependency>

<groupId>com.github.javafaker</groupId>

<artifactId>javafaker</artifactId>

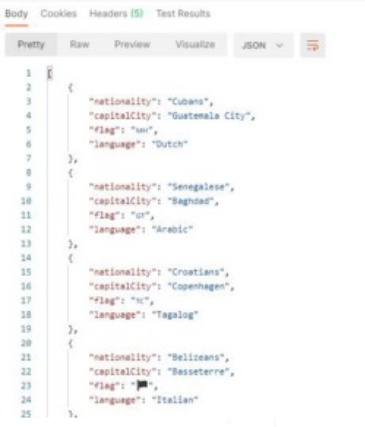
<version>1.0.2</version>

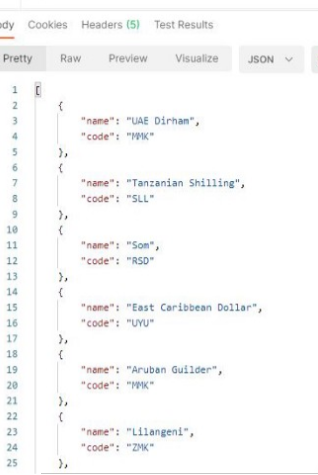
</dependency>

**DataController**class :

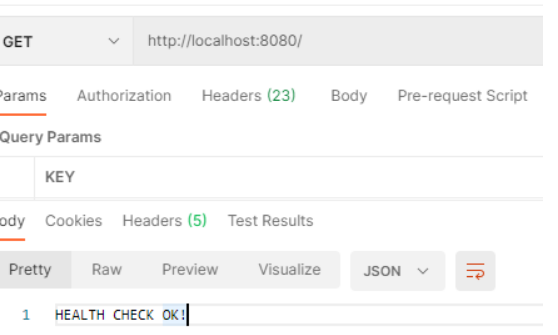
|  |  |
| --- | --- |
|  | @RestController |
|  | public class DataController { |
|  |  |
|  | @GetMapping("/") |
|  | public String healthCheck() { |
|  | return "HEALTH CHECK OK!"; |
|  | } |
|  |  |
|  | @GetMapping("/version") |
|  | public String version() { |
|  | return "The actual version is 1.0.0"; |
|  | } |
|  |  |
|  | @GetMapping("/nations") |
|  | public JsonNode getRandomNations() { |
|  | var objectMapper = new ObjectMapper(); |
|  | var faker = new Faker(new Locale("en-US")); |
|  | var nations = objectMapper.createArrayNode(); |
|  | for (var i = 0; i < 10; i++) { |
|  | var nation = faker.nation(); |
|  | nations.add(objectMapper.createObjectNode() |
|  | .put("nationality", nation.nationality()) |
|  | .put("capitalCity", nation.capitalCity()) |
|  | .put("flag", nation.flag()) |
|  | .put("language", nation.language())); |
|  | } |
|  | return nations; |
|  | } |
|  |  |
|  | @GetMapping("/currencies") |
|  | public JsonNode getRandomCurrencies() { |
|  | var objectMapper = new ObjectMapper(); |
|  | var faker = new Faker(new Locale("en-US")); |
|  | var currencies = objectMapper.createArrayNode(); |
|  | for (var i = 0; i < 20; i++) { |
|  | var currency = faker.currency(); |
|  | currencies.add(objectMapper.createObjectNode() |
|  | .put("name", currency.name()) |
|  | .put("code", currency.code())); |
|  | } |
|  | return currencies; |
|  |  |
|  | } |
|  |  |
|  | } |

----------------------------------------------------





**health check :**



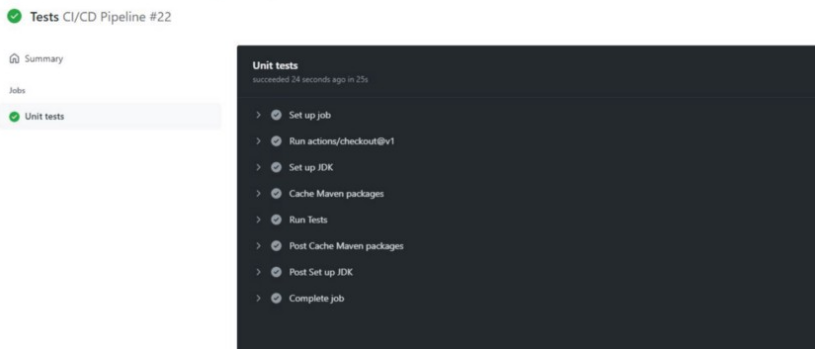
I have added unit tests to our RestController :

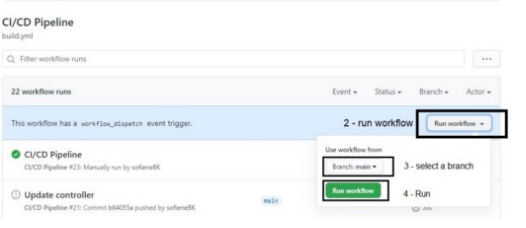
|  |
| --- |
| @SpringBootTest |
|  | class DataControllerTest { |
|  |  |
|  | @Autowired |
|  | DataController dataController; |
|  |  |
|  | @Test |
|  | void health() { |
|  | assertEquals( "HEALTH CHECK OK!", dataController.healthCheck()); |
|  | } |
|  |  |
|  | @Test |
|  | void version() { |
|  | assertEquals( "The actual version is 1.0.0", dataController.version()); |
|  | } |
|  |  |
|  | @Test |
|  | void nationsLength() { |
|  | Integer nationsLength = dataController.getRandomNations().size(); |
|  | assertEquals(10, nationsLength); |
|  | } |
|  |  |
|  | @Test |
|  | void currenciesLength() { |
|  | Integer currenciesLength = dataController.getRandomCurrencies().size(); |
|  | assertEquals(20, currenciesLength); |
|  | } |
|  |  |
|  | } |

--------------------------------------------------------------------------------------

|  |
| --- |
| #Workflow name |
|  | name: CI/CD Pipeline |
|  | on: |
|  | #Manually trigger workflow runs |
|  | workflow\_dispatch: |
|  | #Trigger the workflow on push from the main branch |
|  | push: |
|  | branches: |
|  | - main |
|  | jobs: |
|  | #Test's job |
|  | tests: |
|  | name: Unit tests |
|  | #Run on Ubuntu using the latest version |
|  | runs-on: ubuntu-latest |
|  | #Job's steps |
|  | steps: |
|  | #Check-out your repository under $GITHUB\_WORKSPACE, so your workflow can access it |
|  | - uses: actions/checkout@v1 |
|  | #Set up JDK 11 |
|  | - name: Set up JDK |
|  | uses: actions/setup-java@v1 |
|  | with: |
|  | java-version: '11' |
|  | #Set up Maven cache |
|  | - name: Cache Maven packages |
|  | #This action allows caching dependencies and build outputs to improve workflow execution time. |
|  | uses: actions/cache@v1 |
|  | with: |
|  | path: ~/.m2 |
|  | key: ${{ runner.os }}-m2-${{ hashFiles('\*\*/pom.xml') }} |
|  | restore-keys: ${{ runner.os }}-m2 |
|  | #Run Tests |
|  | - name: Run Tests |
|  | run: mvn -B test |

--------------------------------------------------------------------------------------------



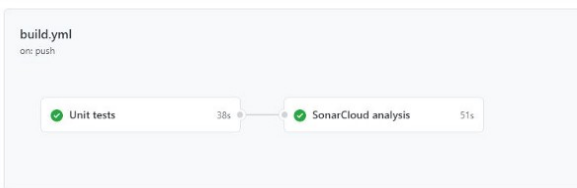


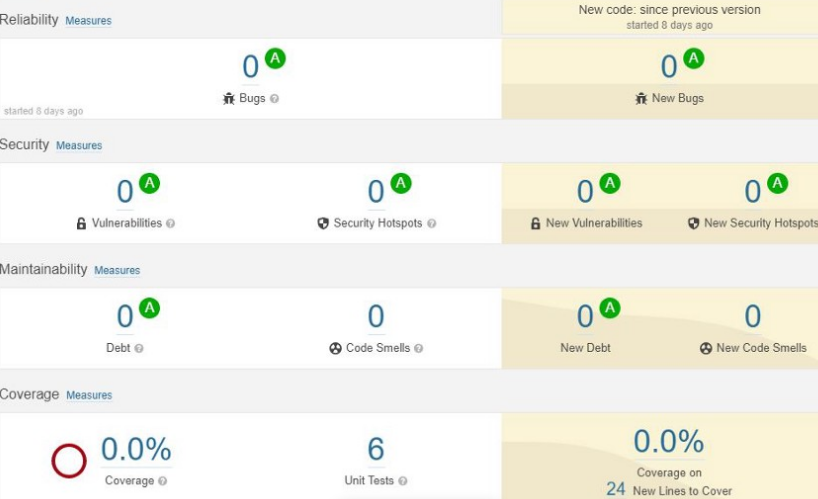
-----------------------------------------------------------------------------------------

in the build.yml file I have added a new job called sonar .

|  |
| --- |
| #Sonar's Job |
|  | sonar: |
|  | #Depends on test's job |
|  | needs: tests |
|  | name: SonarCloud analysis |
|  | #Run on Ubuntu using the latest version |
|  | runs-on: ubuntu-latest |
|  | #Job's steps |
|  | steps: |
|  | #Check-out your repository under $GITHUB\_WORKSPACE, so your workflow can access it |
|  | - uses: actions/checkout@v1 |
|  | #Set up JDK 11 |
|  | - name: Set up JDK |
|  | uses: actions/setup-java@v1 |
|  | with: |
|  | java-version: '11' |
|  | #Set up SonarCloud cache |
|  | - name: Cache SonarCloud packages |
|  | #This action allows caching dependencies and build outputs to improve workflow execution time. |
|  | uses: actions/cache@v1 |
|  | with: |
|  | path: ~/.sonar/cache |
|  | key: ${{ runner.os }}-sonar |
|  | restore-keys: ${{ runner.os }}-sonar |
|  | #Set up Maven cache |
|  | - name: Cache Maven packages |
|  | #This action allows caching dependencies and build outputs to improve workflow execution time. |
|  | uses: actions/cache@v1 |
|  | with: |
|  | path: ~/.m2 |
|  | key: ${{ runner.os }}-m2-${{ hashFiles('\*\*/pom.xml') }} |
|  | restore-keys: ${{ runner.os }}-m2 |
|  | #Analyze project with SonarCloud |
|  | - name: Analyze with SonarCloud |
|  | run: mvn -B verify sonar:sonar -Dsonar.projectKey=sofieneBK\_Hands-on-ci-cd-for-springboot-apps -Dsonar.organization=sofienebk -Dsonar.host.url=https://sonarcloud.io -Dsonar.login=$SONAR\_TOKEN |
|  | env: |
|  | GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN }} |
|  | SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }} |

----------------------------------------------------------------------------------------------



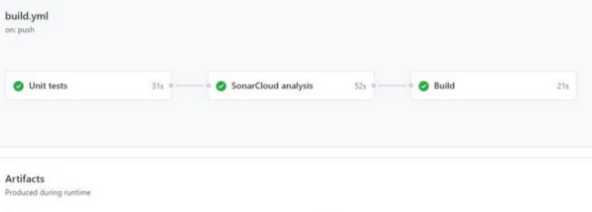


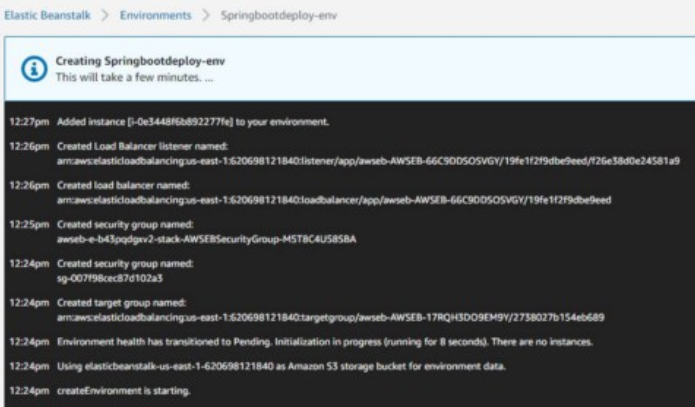
|  |
| --- |
| <plugin> |
|  | <groupId>org.jacoco</groupId> |
|  | <artifactId>jacoco-maven-plugin</artifactId> |
|  | <version>0.8.7</version> |
|  | <executions> |
|  | <execution> |
|  | <goals> |
|  | <goal>prepare-agent</goal> |
|  | </goals> |
|  |  |
|  | </execution> |
|  | <!-- attached to Maven test phase --> |
|  | <execution> |
|  | <id>report</id> |
|  | <phase>test</phase> |
|  | <goals> |
|  | <goal>report</goal> |
|  | </goals> |
|  | </execution> |
|  | </executions> |
|  | </plugin> |

--------------------------------------------------------------------------

|  |
| --- |
| #Build's job |
|  | build: |
|  | #Depends on sonar's job |
|  | needs: sonar |
|  | name: Build |
|  | #Run on Ubuntu using the latest version |
|  | runs-on: ubuntu-latest |
|  | steps: |
|  | #Check-out your repository under $GITHUB\_WORKSPACE, so your workflow can access it |
|  | - uses: actions/checkout@v1 |
|  | #Set up JDK 11 |
|  | - name: Set up JDK |
|  | uses: actions/setup-java@v1 |
|  | with: |
|  | java-version: '11' |
|  | #Set up Maven cache |
|  | - name: Cache Maven packages |
|  | #This action allows caching dependencies and build outputs to improve workflow execution time. |
|  | uses: actions/cache@v1 |
|  | with: |
|  | path: ~/.m2 |
|  | key: ${{ runner.os }}-m2-${{ hashFiles('\*\*/pom.xml') }} |
|  | restore-keys: ${{ runner.os }}-m2 |
|  | #Build the application using Maven |
|  | - name: Build with Maven |
|  | run: mvn -B package -DskipTests --file pom.xml |
|  | #Build the application using Maven |
|  | - name: Upload JAR |
|  | #This uploads artifacts from your workflow allowing you to share data between jobs and store data once a workflow is complete. |
|  | uses: actions/upload-artifact@v2 |
|  | with: |
|  | #Set artifact name |
|  | name: artifact |
|  | #From this path |
|  | path: target/data-0.0.1-SNAPSHOT.jar |

------------------------------------------------------------------------------

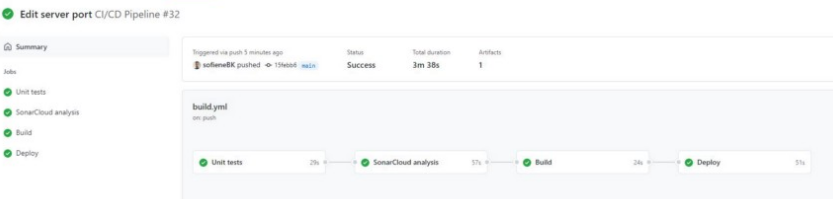




deployjob :

|  |
| --- |
| #Deploy's job |
|  | deploy: |
|  | #Depends on build's job |
|  | needs: build |
|  | name: Deploy |
|  | #Run on Ubuntu using the latest version |
|  | runs-on: ubuntu-latest |
|  | steps: |
|  | - name: Download JAR |
|  | #Download the artifact which was uploaded in the build's job |
|  | uses: actions/download-artifact@v2 |
|  | with: |
|  | name: artifact |
|  | #Deploy the artifact (JAR) into AWS Beanstalk |
|  | - name: Deploy to EB |
|  | uses: einaregilsson/beanstalk-deploy@v13 |
|  | with: |
|  | aws\_access\_key: ${{ secrets.AWS\_ACCESS\_KEY\_ID }} |
|  | aws\_secret\_key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }} |
|  | aws\_session\_token: ${{ secrets.AWS\_SESSION\_TOKEN }} |
|  | use\_existing\_version\_if\_available: false |
|  | application\_name: spring-boot-deploy |
|  | environment\_name: Springbootdeploy-env |
|  | version\_label: ${{github.SHA}} |
|  | region: us-east-1 |
|  | deployment\_package: data-0.0.1-SNAPSHOT.jar |

----------------------------------------------------------------------



---------------------------------------------------------------------------------------------------