Mutation tests

Add PITest Mutation Plugin in pom.xml

```
    <!--              PITest Mutation Plugin              -->
    <plugin>
    <groupId>org.pitest</groupId>
    <artifactId>pitest-maven</artifactId>
    <version>1.5.0</version>
    <dependencies>
      <dependency>
        <groupId>org.pitest</groupId>
        <artifactId>pitest-junit5-plugin</artifactId>
        <version>0.12</version>
      </dependency>
    </dependencies>
    <configuration>
      <mutationThreshold>70</mutationThreshold> ## test fail if output less than 70%
      <outputFormats>
        <outputFormat>XML</outputFormat> ## export report at XML
        <outputFormat>HTML</outputFormat>
      </outputFormats>
    </configuration>
    </plugin>
  </build>
```

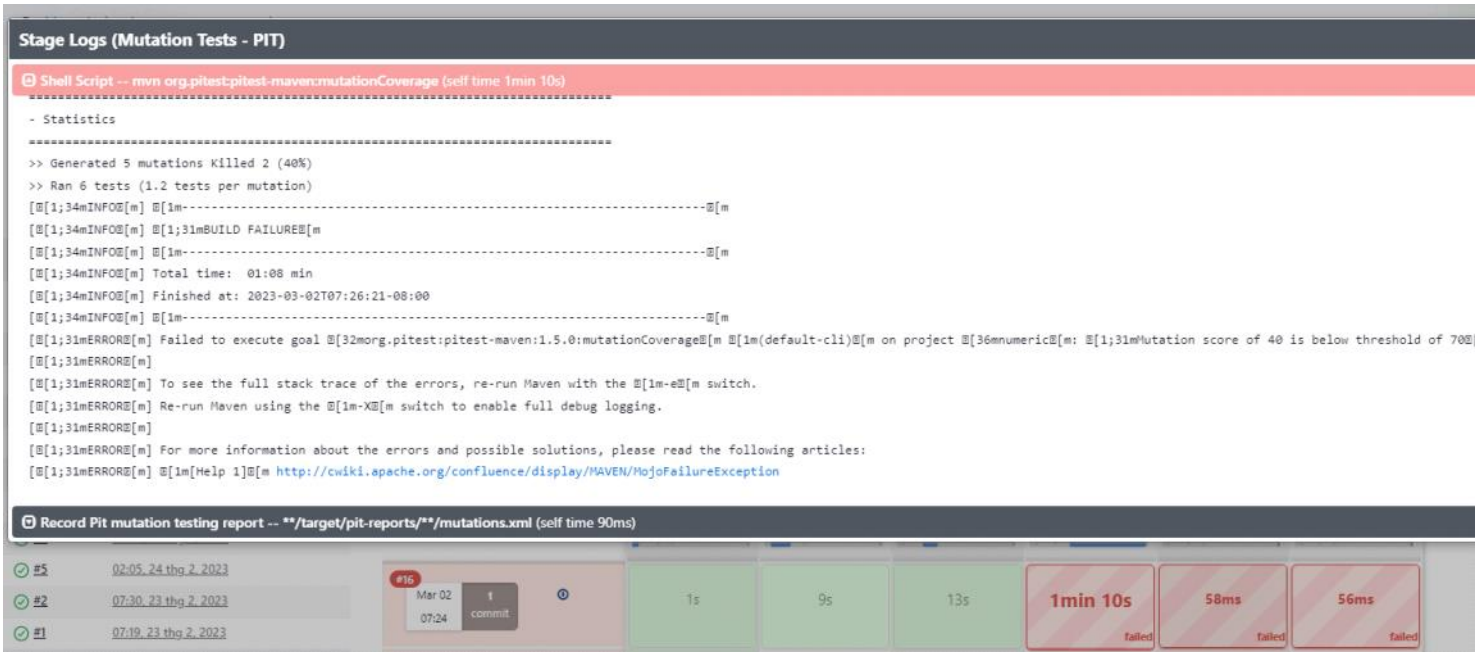
add code below before build stage:

```
stage('Mutation Tests - PIT') {
        steps {
          sh "mvn org.pitest:pitest-maven:mutationCoverage"
        }
        post {
          always {
            pitmutation mutationStatsFile: '**/target/pit-reports/**/mutations.xml'
          }
        }
```

This is a Jenkins pipeline code snippet that defines a stage called "Mutation Tests - PIT" which runs a mutation testing tool called Pitest.
The **`steps`** block describes the task to be performed in this stage. In this case, it runs the **`mvn`** command which executes the Pitest maven plugin to generate mutation coverage report.
`org.pitest:pitest-maven:mutationCoverage`: This is the specific goal being executed using the Pitest Maven plugin. The `mutationCoverage` goal is used to run the mutation tests and generate the mutation coverage report.
The **`post`** block defines a post-build action to be executed after the completion of the **`steps`** block. The **`always`** block is a post-build action that runs whether the previous steps succeeded or not.
The **`pitmutation`** command within the **`always`** block is a Jenkins plugin that parses the **`mutations.xml`** file generated by Pitest to calculate the mutation coverage statistics. The **`mutationStatsFile`** argument tells the plugin where to find the XML file.
Overall, this stage runs a mutation testing tool and generates a report using the Pitest maven plugin, followed by calculating and displaying the mutation coverage statistics using the Jenkins **`pitmutation`** plugin.

push new source code, jenkins have logs:

```
[ERROR] Failed to execute goal org.pitest:pitest-maven:1.5.0:mutationCoverage(default-cli) on project numeric: Mutation score of 40 is below threshold of 70-> [Help 1]
```


Stage Logs (Mutation Tests - PIT)
Shell Script -- mvn org.pitest:pitest-maven:mutationCoverage (self time 1min 10s)
- Statistics
>> Generated 5 mutations Killed 2 (40%)
>> Ran 6 tests (1.2 tests per mutation)
[ERROR] Failed to execute goal org.pitest:pitest-maven:1.5.0:mutationCoverage (default-cli) on project devsecops-app-num: Mutation score of 40 is below threshold of 70
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
Record Pit mutation testing report -- **/target/pit-reports/**/mutations.xml (self time 90ms)
#5 02:05, 24 thg 2, 2023
#2 07:30, 23 thg 2, 2023
#1 07:19, 23 thg 2, 2023
#16 Mar 02 07:24 1 commit
1s
9s
13s
1min 10s failed
58ms failed
56ms failed


Dashboard > devsecops-app-num > #16
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#16'
Polling Log
Git Build Data
Test Result
Coverage Report
PIT Mutation Report
Modules
Mutation Statistics
Mutations Undetected Coverage
5 (+5) 3 (+3) 40.0% (+40.0%)
Components
Name Mutations +/- Undetected +/- Coverage +/-
Module: null 5 +5 3 +3 40.0% +40.0%

```
30 1 1. welcome : replaced return value with "" for
     com/devsecops/NumericController$compare::welcome → SURVIVED
                        return "Kubernetes DevSecOps";
31                  }
32
33                  @GetMapping("/compare/{value}")
34                  public String compareToFifty(@PathVariable int value) {
35                          String message = "Could not determine comparison";
36 2 1. compareToFifty : changed conditional boundary → SURVIVED
     2. compareToFifty : negated conditional → KILLED
                          if (value > 50) {
37                                message = "Greater than 50";
38                          } else {
39                                message = "Smaller than or equal to 50";
40                          }
```

Change 3 test case become like below:

```
@Test
  public void smallerThanOrEqualToFiftyMessage() throws Exception {
    this.mockMvc.perform(get("/compare/50")).andDo(print()).andExpect(status().isOk())
        .andExpect(content().string("Smaller than or equal to 50"));
  }
  @Test
  public void greaterThanFiftyMessage() throws Exception {
    this.mockMvc.perform(get("/compare/51")).andDo(print()).andExpect(status().isOk())
        .andExpect(content().string("Greater than 50"));
  }

  @Test
  public void welcomeMessage() throws Exception {
    this.mockMvc.perform(get("/")).andDo(print()).andExpect(status().isOk())
        .andExpect(content().string("Kubernetes DevSecOps"));
  }
```

This is a JUnit test case written in Java that uses the Spring MVC Test framework.
The **`@Test`** annotation indicates that this method is a test case.
The name of the method **`welcomeMessage()`** is meant to describe the behavior being tested.
The **`throws Exception`** in the signature of the method indicate that this test may throw a generic **`Exception`**.
The **`this.mockMvc.perform(get("/"))`** is a method call in which we send an HTTP GET request to the application's root URL **`/`**.
The **`andDo(print())`** method is used to print the result of the request in the console.
The **`andExpect(status().isOk())`** method verifies that the response status code is 200 (OK).
The **`andExpect(content().string("Kubernetes DevSecOps"))`** method verifies that the response body contains the exact string "Kubernetes DevSecOps".
Overall, this test case is checking that the application's Welcome page displays the correct message "Kubernetes DevSecOps".

```
ll /var/lib/jenkins/workspace/devsecops-app-num/target/pit-reports/
total 12
drwxr-xr-x  3 jenkins jenkins 4096 Mar  2 21:49 ./
drwxr-xr-x 11 jenkins jenkins 4096 Mar  2 21:49 ../
```

```
drwxr-xr-x  3 jenkins jenkins 4096 Mar  2 21:49 202303022149/
```