

```

1 1. create a new maven project
2 2. add dependencies/plugins/configurations to pom.xml
3 Note : for providing the main class to be written within maven-jar-plugin below
4 <configuration>
5     <archive>
6         <manifest>
7             <!-- <addClasspath>true</addClasspath>
8             <classpathPrefix>libs/</classpathPrefix> -->
9             <mainClass>
10                 com.ameya.tests.TestServices
11             </mainClass>
12         </manifest>
13     </archive>
14 </configuration>
15 Note :
16 <finalName>devopsproject</finalName> ---> the jar will be created by this name
17 ++++++ pom.xml
18 ++++++
19 <project xmlns="http://maven.apache.org/POM/4.0.0"
20     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
21     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
22     http://maven.apache.org/xsd/maven-4.0.0.xsd">
23     <modelVersion>4.0.0</modelVersion>
24     <groupId>com.ameya</groupId>
25     <artifactId>devopsproject</artifactId>
26     <version>0.0.1-SNAPSHOT</version>
27     <name>devopsproject</name>
28     <description>Test Driven Development - Testing services</description>
29     <organization>
30         <name>Vinsys</name>
31         <url>https://services.vinsys.com</url>
32     </organization>
33     <inceptionYear>2021</inceptionYear>
34     <distributionManagement>
35         <repository>
36             <url>https://mvnrepository.com/</url>
37             <id>test</id>
38         </repository>
39     </distributionManagement>
40     <properties>
41         <maven.compiler.source>11</maven.compiler.source>
42         <maven.compiler.target>11</maven.compiler.target>
43         <sonar.projectKey>devopsproject-ameya</sonar.projectKey>
44         <sonar.organization>ameya-org</sonar.organization>
45         <sonar.host.url>https://sonarcloud.io</sonar.host.url>
46         <SONAR_TOKEN>df3378fd07ea65df778ee60a3d7f1c3a7f657329</SONAR_TOKEN>

```

44

```
<sonar.coverage.jacoco.xmlReportPaths>../devopsproject/target/site/jacoco/jacoco.xml  
|</sonar.coverage.jacoco.xmlReportPaths>
```

45 </properties>

46 <scm>

47 <tag>dev</tag>

48 <url>https://github.com/amsalways/training\_material.git</url>

49 </scm>

50 <contributors>

51 <contributor>

52 <name>Avani</name>

53 <email>avani@abc.com</email>

54 </contributor>

55 </contributors>

56 <developers>

57 <developer>

58 <id>d1</id>

59 <name>Ameya</name>

60 <email>ameya.joshi@vinsys.com</email>

61 </developer>

62 </developers>

63 <dependencies>

64 <!-- https://mvnrepository.com/artifact/junit/junit -->

65 <dependency>

66 <groupId>junit</groupId>

67 <artifactId>junit</artifactId>

68 <version>4.12</version>

69 <scope>test</scope>

70 </dependency>

71 <!-- https://mvnrepository.com/artifact/org.mockito/mockito-all -->

72 <dependency>

73 <groupId>org.mockito</groupId>

74 <artifactId>mockito-core</artifactId>

75 <version>2.28.2</version>

76 <scope>test</scope>

77 </dependency>

78 <dependency>

79 <groupId>org.slf4j</groupId>

80 <artifactId>slf4j-simple</artifactId>

81 <version>1.7.30</version>

82 </dependency>

83 <dependency>

84 <groupId>org.slf4j</groupId>

85 <artifactId>slf4j-api</artifactId>

86 <version>1.7.30</version>

87 </dependency>

88 </dependencies>

89 <reporting>

```

90     <plugins>
91         <plugin>
92     <!-- Generate the Report as Part of Project Reports ...
93     To generate the jacoco report as part of the site generation, add the following in
the <reporting> -->
94         <groupId>org.jacoco</groupId>
95         <artifactId>jacoco-maven-plugin</artifactId>
96         <reportSets>
97             <reportSet>
98                 <reports>
99                     <report>report</report>
100                 </reports>
101             </reportSet>
102         </reportSets>
103     </plugin>
104     <plugin>
105         <groupId>org.apache.maven.plugins</groupId>
106         <artifactId>maven-surefire-report-plugin</artifactId>
107         <version>3.0.0-M5</version>
108     </plugin>
109 </plugins>
110 </reporting>
111 <build>
112
113     <plugins>
114         <plugin>
115             <groupId>org.jacoco</groupId>
116             <artifactId>jacoco-maven-plugin</artifactId>
117             <version>0.8.7</version>
118             <executions>
119                 <execution>
120                     <goals>
121     <!-- Prepares a property pointing to the JaCoCo runtime agent that can be
122     passed as a VM argument to the application under test -->
123                 <goal>prepare-agent</goal>
124                     </goals>
125                 </execution>
126                 <execution>
127                     <id>report</id>
128     <!-- perform any operations necessary to prepare a package before
129     the actual packaging. This often results in an unpacked, processed version of the
package. -->
130                     <phase>prepare-package</phase>
131                     <goals>
132     <!-- It creates code coverage reports from the execution data recorded by
133     the JaCoCo runtime agent. -->
134                     <goal>report</goal>
135                     </goals>

```

```

136         </execution>
137     </executions>
138 </plugin>
139     <plugin>
140         <groupId>org.apache.maven.plugins</groupId>
141         <artifactId>maven-site-plugin</artifactId>
142         <version>3.9.1</version>
143     </plugin>
144     <plugin>
145         <groupId>org.apache.maven.plugins</groupId>
146         <artifactId>maven-project-info-reports-plugin</artifactId>
147         <version>3.1.2</version>
148     </plugin>
149 <plugin>
150     <groupId>org.apache.maven.plugins</groupId>
151     <artifactId>maven-compiler-plugin</artifactId>
152     <version>3.7.0</version>
153     <configuration>
154         <release>11</release>
155     </configuration>
156 </plugin>
157 <plugin>
158     <groupId>org.apache.maven.plugins</groupId>
159     <artifactId>maven-jar-plugin</artifactId>
160     <configuration>
161         <archive>
162             <manifest>
163                 <mainClass>
164                     com.ameya.tests.TestServices
165                 </mainClass>
166             </manifest>
167         </archive>
168     </configuration>
169 </plugin>
170 </plugins>
171 <finalName>devopsproject</finalName>
172 </build>
173 </project>
174
175 ++++++
176
176 3. write the test cases
177 4. write the dao/service interfaces/impls
178 5. write the main method
179 6. run the mvn clean package
180 6.1. from target folder run java -jar devopsproject.jar
181
182 --- create a repo within the project devopsproject

```

```
183 --- create a remote repo devopsproject
184 --- gotot local repo git remote add origin
    https://amsalways@bitbucket.org/amsalways/test.git
185 --- go to local repo --- git pull origin master --allow-unrelated-histories
186 --- git push origin master
187
188 7. goto jenkins folder
189 8. start jenkins.war
190 9. browse to localhost:8080/
191 10. login to jenkins
192 11. click on Manage jenkins
193 12. click on global tools configuration
194 13. click on Add JDK and add Name as Java_11 and path to java home
195 14. Add path till git.exe under git
196 15. Click on Add Maven and add Maven home path
197 16. Click on Save
198 17. Click on Manage plugins
199 18. Available-- Maven integration plugin-- install without restart
200 19. From dashboard click on new item
201 20. enter name devopsproject
202 21. select Maven project
203 22. Click on Ok
204 23. In Job Configuration under general section tick github
205     paste https://amsalways@bitbucket.org/amsalways/devopsproject.git
206
207 24. Under Source Code Management section click on Git radio button and provide
    Repository URL
208     https://amsalways@bitbucket.org/amsalways/devopsproject.git
209     Add credentials username/password by clicking on add --- jenkins
210 25. Additional behaviours add-- clean before checkout
211 26. under Build Trigger > tick Poll SCM and in text field * * * * *
212 27. Navigate to build section provide path to pom.xml just give the name as pom.xml
213     Goals and options : clean install site
214 28. Click on Save .
215 29. Go to dashboard click on build history u will see the build triggered.
216 30. Click on console output
217 31. Click on completed builds and check the output and artifacts created.
218 32. Goto devopsproject---configure---Add post build step
219 33. select Execute Windows batch command
220 34. Enter following
221     cd C:\Users\Ameya\.jenkins\workspace\devopsproject\target
222     java -jar devopsproject.jar
223 35. Build Now and see console.
224
225 *****Jacoco*****
226 36. dashboard manage jenkins --- manage plugins --- available --- jacoco plugin ---
    install without restart
227 --- Post-Build Actions
```

```

228 --- Add Post-Build action --- Record JaCoCo Coverage ---
229 --- Path to exec files : **/target/jacoco.exec
230 --- Path to class directories : **/target/classes
231 --- Put in any other values if and as required
232 --- Apply and save
233 37. Click on the build and click on Coverage Report Also observe the Jacoco report on
    the build page
234
235 *****Sonarqube*****
236 38. Goto sonarcloud.io
237 39. Login--- Login using bitbucket
238 40. Click on + in right top corner
239 41 Create new organization
240 42. Click Create an organization manually
241 43. Provide a key -> ameya-org
242 44. Click continue
243 45. select free plan
244 46. Click Create organization
245 47. Click Create new Project
246 48. Provide project key -> devopsproject-ameya
247 49. click on create
248 50. select Maven
249 51. copy this to maven pom.xml in properties (sonar token shud be coiped from sonar
    cloud)
250     <sonar.projectKey>devopsproject-ameya</sonar.projectKey>
251     <sonar.organization>ameya-org</sonar.organization>
252     <sonar.host.url>https://sonarcloud.io</sonar.host.url>
253
254     <SONAR_TOKEN>4a4b4960db0b6a0abfd4373778dd2795e73eb302</SONAR_TOKEN
    >
255
256     <sonar.coverage.jacoco.xmlReportPaths>../devopsproject/target/site/jacoco/jacoco.x
    ml
257 52. Update the maven projects
258 53. mvn clean verify sonar:sonar
    -Dsonar.login=4a4b4960db0b6a0abfd4373778dd2795e73eb302
259 54. After build is success... Goto sonarcloud and observe the code quality analysis.
260 *****Jenkins Sonar integration*****
261 55. Goto dashboard ----- Manage Jenkins ----- Manage plugins ----- Available
262 56. Search Sonarqube scanner ---- Install without restart
263 57. Goto dashboard ----- Manage Jenkins ----- Configure system
264 58. SonarQube Servers
265 59. Add SonarQube
266 60. Name SonarQube
267 61. Server URL : https://sonarcloud.io/
268 62. Click on Add button under Authentication Token
269 63. select secret text paste the token

```

4a4b4960db0b6a0abfd4373778dd2795e73eb302

```
269 64. Select this secret text in place of none. ---- Click on save.
270 65. Goto Manage jenkins---- Global Tool Configuration
271 66. Scroll down click on Add SonarQube Scanner
272 67. Name : SonarQube --- Click Install automatically ---- Install From Maven
    Central. ----- Click on Save.
273 68. In your project in eclipse create file sonar.properties as follows
274     sonar.sources=./src
275     sonar.projectKey=devopsproject-ameya
276     sonar.organization=ameya-org
277     sonar.java.binaries=./target/classes
278 69. Goto jenkins ----- devopsproject ----- Configure
279 70. Add Post build step ---- Execute Sonar scanner
280 71. Path To Project properties
    C:\Users\Ameya\.jenkins\workspace\devopsproject\sonar.properties
281 72. Click on save.
282 73. Trigger the build.
283 74. Browse to sonarcloud and refresh...
284 75. Goto TestServices.java
285 76. add System.out.println("Invoking the service and dao layer"); to code.
286 77. from cmd git add . ----- git commit -m "Altered Code" ----- git
    push bucketdevops master.
287 78. Goto jenkins... A build will be triggered ----- check the console logs -----
    goto sonarcloud and refresh
288
289 ***** Docker Integration using manual commands
    *****
290 Create a Dockerfile in eclipse project as follows
291 FROM openjdk:11
292 EXPOSE 9191
293 ADD target/devopsproject.jar devopsproject.jar
294 ENTRYPOINT ["java","-jar","/devopsproject.jar"]
295
296 79. Gotot Jenkins... Dashboard----- devopsproject ----- configure
    -----
297 80. Add Post step ----- Execute windows batch command
298 81. Paste following
299     cd C:\Users\Ameya\.jenkins\workspace\devopsproject
300     docker build . -t devopsproject
301     #docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG] -->This command
    just creates an alias (a reference) by the name of the TARGET_IMAGE that
302     #refers to the SOURCE_IMAGE. That's all it does. It's like assigning an
    existing image another name to refer to it
303     docker tag devopsproject ameyajoshi/devopsproject
304     docker push ameyajoshi/devopsproject
305
306 ***** Docker Integration Using plugin not
    working*****
```

307 79. Goto dashboard ----- Manage Jenkins ----- Manage plugins  
308 80. install docker plugin, cloudbees Docker build and publish plugin, Docker-build-step  
plugin ----- install without restart  
309 81. In eclipse create a Dockerfile as follows  
310 FROM openjdk:11  
311 EXPOSE 9191  
312 ADD target/devopsproject.jar devopsproject.jar (ADD source destination)  
313 ENTRYPOINT ["java","-jar","/devopsproject.jar"]  
314 82. Goto jenkins ----- devopsproject ----- configure ----- add post build step  
----- Docker build/publish  
315 83. repository name : ameyajoshi/devopsproject  
316 84. Registry credentials : username/password ameyajoshi/Avani@0811 -> dockerhub  
credentials  
317 85.  
318