1. create a new maven project
2. add dependencies/plugins/configurations to pom.xml
    Note : for providing the main class to be written within maven-jar-plugin below </plugin> for maven compiler plugin

```
    <configuration>
        <archive>
            <manifest>
                <!-- <addClasspath>true</addClasspath>
                <classpathPrefix>libs/</classpathPrefix> -->
                <mainClass>
                    com.ameya.tests.TestServices
                </mainClass>
            </manifest>
        </archive>
    </configuration>
```

    Note :
    <finalName>devopsproject</finalName> ---> the jar will be created by this name

++++++++++++++++++++++++++++++++ pom.xml ++++++++++++++++++++++++++++++++

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ameya</groupId>
  <artifactId>devopsproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>devopsproject</name>
  <description>Test Driven Development - Testing services</description>
  <organization>
    <name>Vinsys</name>
    <url>https://services.vinsys.com</url>
  </organization>
  <inceptionYear>2021</inceptionYear>
  <distributionManagement>
    <repository>
        <url>https://mvnrepository.com/</url>
        <id>test</id>
    </repository>
  </distributionManagement>
  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <sonar.projectKey>devopsproject-ameya</sonar.projectKey>
  <sonar.organization>ameya-org</sonar.organization>
  <sonar.host.url>https://sonarcloud.io</sonar.host.url>
  <SONAR_TOKEN>df3378fd07ea65df778ee60a3d7f1c3a7f657329</SONAR_TOKEN>
```

```
<sonar.coverage.jacoco.xmlReportPaths>../devopsproject/target/site/jacoco/jacoco.xml</sonar.coverage.jacoco.xmlReportPaths>
  </properties>
  <scm>
    <tag>dev</tag>
    <url>https://github.com/amsalways/training_material.git</url>
  </scm>
  <contributors>
    <contributor>
        <name>Avani</name>
        <email>avani@abc.com</email>
    </contributor>
  </contributors>
  <developers>
    <developer>
        <id>d1</id>
        <name>Ameya</name>
        <email>ameya.joshi@vinsys.com</email>
    </developer>
  </developers>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-all -->
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>2.28.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.30</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.30</version>
        </dependency>
  </dependencies>
  <reporting>
```

```
  <plugins>
        <plugin>
<!-- Generate the Report as Part of Project Reports ...
To generate the jacoco report as part of the site generation, add the following in the <reporting> -->
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <reportSets>
               <reportSet>
                  <reports>
                     <report>report</report>
                  </reports>
               </reportSet>
            </reportSets>
         </plugin>
   <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-report-plugin</artifactId>
            <version>3.0.0-M5</version>
         </plugin>
  </plugins>
  </reporting>
  <build>

    <plugins>
        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.7</version>
            <executions>
               <execution>
                  <goals>
<!-- Prepares a property pointing to the JaCoCo runtime agent that can be passed as a VM argument to the application under test -->
                     <goal>prepare-agent</goal>
                  </goals>
               </execution>
               <execution>
                  <id>report</id>
<!-- perform any operations necessary to prepare a package before the actual packaging. This often results in an unpacked, processed version of the package. -->
                  <phase>prepare-package</phase>
                  <goals>
<!-- It creates code coverage reports from the execution data recorded by the JaCoCo runtime agent. -->
                     <goal>report</goal>
                  </goals>
```

```
                </execution>
            </executions>
        </plugin>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.9.1</version>
        </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>3.1.2</version>
        </plugin>
     <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.7.0</version>
            <configuration>
                <release>11</release>
            </configuration>
        </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <configuration>
            <archive>
                <manifest>
                    <mainClass>
                        com.ameya.tests.TestServices
                    </mainClass>
                </manifest>
            </archive>
        </configuration>
    </plugin>
  </plugins>
  <finalName>devopsproject</finalName>
 </build>
</project>

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3. write the test cases
4. write the dao/service interfaces/impls
5. write the main method
6. run the mvn clean package
6.1. from target folder run java -jar devopsproject.jar

--- create a repo within the project devopsproject
```

--- create a remote repo devopsproject
--- gotot local repo git remote add origin https://amsalways@bitbucket.org/amsalways/test.git
--- go to local repo --- git pull origin master --allow-unrelated-histories
--- git push origin master

7. goto jenkins folder
8. start jenkins.war
9. browse to localhost:8080/
10. login to jenkins
11. click on Manage jenkins
12. click on global tools configuration
13. click on Add JDK and add Name as Java_11 and path to java home
14. Add path till git.exe under git
15. Click on Add Maven and add Maven home path
16. Click on Save
17. Click on Manage plugins
18. Available-- Maven integration plugin-- install without restart
19. From dashboard click on new item
20. enter name devopsproject
21. select Maven project
22. Click on Ok
23. In Job Configuration under general section tick github
    paste https://amsalways@bitbucket.org/amsalways/devopsproject.git

24. Under Source Code Management section click on Git radio button and provide Repository URL
        https://amsalways@bitbucket.org/amsalways/devopsproject.git
        Add credentials username/password by clicking on add --- jenkins
25. Additional behaviours add-- clean before checkout
26. under Build Trigger > tick Poll SCM and in text field * * * * *
27. Navigate to build section provide path to pom.xml just give the name as pom.xml
    Goals and options : clean install site
28. Click on Save .
29. Go to dashboard click on build history u will see the build triggered.
30.Click on console output
31. Click on completed builds and check the output and artifacts created.
32. Goto devopsproject---configure---Add post build step
33. select Execute Windows batch command
34. Enter following
        cd C:\Users\Ameya\.jenkins\workspace\devopsproject\target
        java -jar devopsproject.jar
35. Build Now and see console.

**********Jacoco**********
36. dashboard manage jenkins --- manage plugins --- available --- jacoco plugin --- install without restart
--- Post-Build Actions

--- Add Post-Build action --- Record JaCoCo Coverage ---
--- Path to exec files : **/target/jacoco.exec
--- Path to class directories : **/target/classes
--- Put in any other values if and as required
--- Apply and save
37. Click on the build and click on Coverage Report Also observe the Jacoco report on the build page

**********Sonarqube**********
38. Goto sonarcloud.io
39. Login--- Login using bitbucket
40. Click on + in right top corner
41 Create new organization
42. Click Create an organization manually
43. Provide a key -> ameya-org
44. Click continue
45. select free plan
46. Click Create organization
47. Click Create new Project
48. Provide project key -> devopsproject-ameya
49. click on create
50. select Maven
51. copy this to maven pom.xml in properties (sonar token shud be coiped from sonar cloud)

```
<sonar.projectKey>devopsproject-ameya</sonar.projectKey>
<sonar.organization>ameya-org</sonar.organization>
<sonar.host.url>https://sonarcloud.io</sonar.host.url>

<SONAR_TOKEN>4a4b4960db0b6a0abfd4373778dd2795e73eb302</SONAR_TOKEN>

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

52. Update the maven projects
53. mvn clean verify sonar:sonar -Dsonar.login=4a4b4960db0b6a0abfd4373778dd2795e73eb302
54. After build is success... Goto sonarcloud and observe the code quality analysis.

**********Jenkins Sonar integration**********
55. Goto dashboard ------ Manage Jenkins ----- Manage plugins ----- Available
56. Search Sonarqube scanner ---- Install without restart
57. Goto dashboard ------ Manage Jenkins ----- Configure system
58. SonarQube Servers
59. Add SonarQube
60. Name SonarQube
61. Server URL : https://sonarcloud.io/
62. Click on Add button under Authentication Token
63. select secret text paste the token

4a4b4960db0b6a0abfd4373778dd2795e73eb302
64. Select this secret text in place of none. ---- Click on save.
65. Goto Manage jenkins---- Global Tool Configuration
66. Scroll down click on Add SonarQube Scanner
67. Name : SonarQube --- Click Install automatically ---- Install From Maven Central. ----- Click on Save.
68. In your project in eclipse create file sonar.properties as follows
    sonar.sources=./src
    sonar.projectKey=devopsproject-ameya
    sonar.organization=ameya-org
    sonar.java.binaries=./target/classes
69. Goto jenkins ------ devopsproject ------- Configure
70. Add Post build step ---- Execute Sonar scanner
71. Path To Project properties C:\Users\Ameya\.jenkins\workspace\devopsproject\sonar.properties
72. Click on save.
73. Trigger the build.
74. Browse to sonarcloud and refresh...
75. Goto TestServices.java
76. add System.out.println("Invoking the service and dao layer"); to code.
77. from cmd git add . ---------- git commit -m "Altered Code" -------- git push bucketdevops master.
78. Goto jenkins... A build will be triggered ------ check the console logs -------- goto sonarcloud and refresh

********** Docker Integration using manual commands **********
Create a Dockerfile in eclipse project as follows
```
FROM openjdk:11
EXPOSE 9191
ADD target/devopsproject.jar devopsproject.jar
ENTRYPOINT ["java","-jar","/devopsproject.jar"]
```

79. Gotot Jenkins... Dashboard--------- devopsproject ---------- configure ----------
80. Add Post step -------- Execute windows batch command
81. Paste following
```
    cd C:\Users\Ameya\.jenkins\workspace\devopsproject
    docker build . -t devopsproject
    #docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG] -->This command just creates an alias (a reference) by the name of the TARGET_IMAGE that
    #refers to the SOURCE_IMAGE. That's all it does. It's like assigning an existing image another name to refer to it
    docker tag devopsproject ameyajoshi/devopsproject
    docker push ameyajoshi/devopsproject
```