LAB- CI/CD using Jenkins

STEP1 – Installing and configuring Jenkins

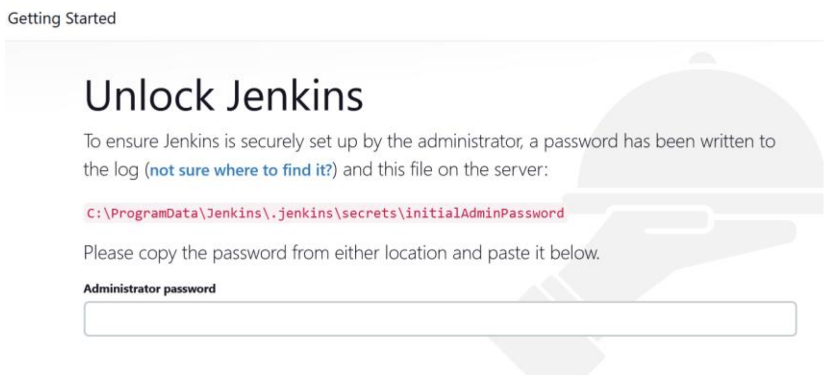
In this step, you will be installing Jenkins

Download Jenkins at https://www.jenkins.io/download/

Install Jenkins by taking all defaults

Once Jenkins starts, u can see the Jenkins UI at http://localhost:8080/

You should see a screen link below which asks for Administrator password for unlocking Jenkins


Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Open the the file `C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword`

, copy the password , paste it in the above page and click on continue

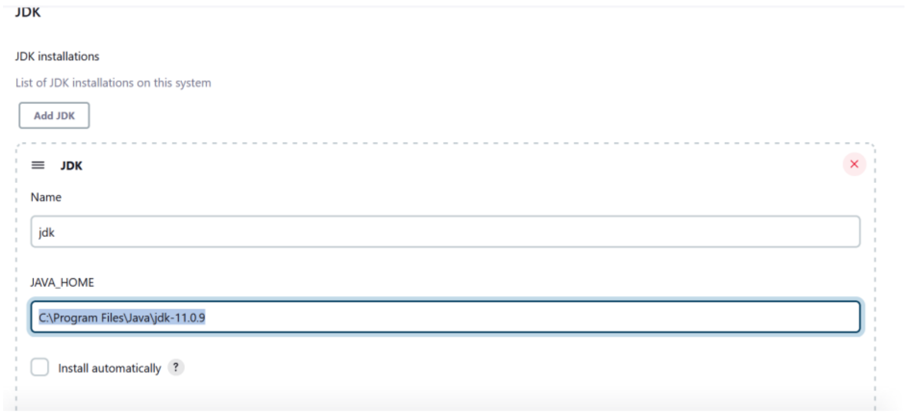
In the next page, select “Install all suggested plugins” . All the plugins will be downloaded into C:\ProgramData\Jenkins\.jenkins\plugins by default.

Once all plugins are installed, it will ask you to create a user. Create a user with name same as your name and remember the password.

Then login using your username

Click on “Manage Jenkins” and then click on Tools

Click on Add JDK and configure JDK as shown below:


JDK

JDK installations

List of JDK installations on this system

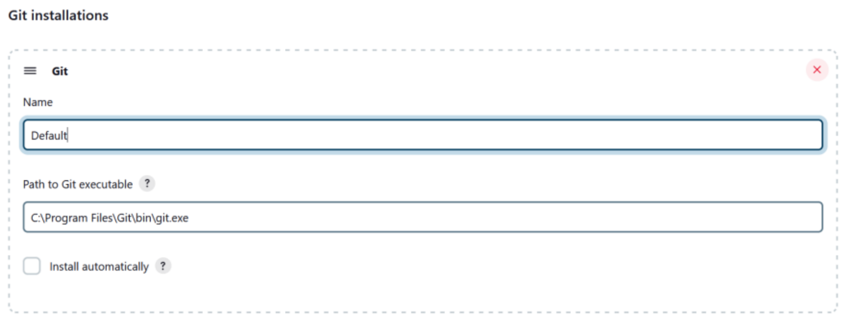
JDK

Name: jdk

JAVA_HOME: C:\Program Files\Java\jdk-11.0.9

Install automatically ?

Configure GIT as shown below


Git installations

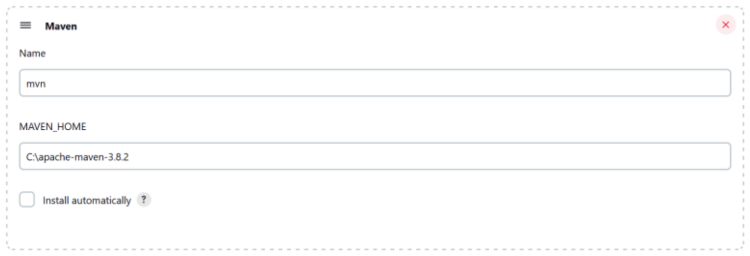
Git

Name: Default

Path to Git executable ?: C:\Program Files\Git\bin\git.exe

Install automatically ?

Configure maven as shown below:


Maven

Name: mvn

MAVEN_HOME: C:\apache-maven-3.8.2

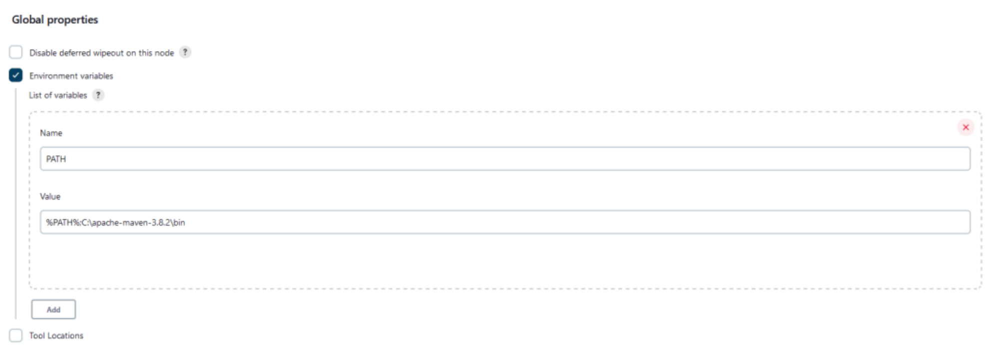
Install automatically ?

If you don’t have maven on your machine, you can google and download it.

Click on Save Button

Once you are in “Manage Jenkins page” , click on “System” link

Under Global Properties ection select environment Variables check box and configure path env variable as shown below:



Global properties

- [ ] Disable deferred wipeout on this node
- [✓] Environment variables

List of variables

Name: PATH

Value: %PATH%;C:\apache-maven-3.8.2\bin

- [ ] Tool Locations

Click on Save

STEP2 – Configuring CI/CD for your project

I am assuming that you have followed previous lab and pushed you code to your github account

I have pushed my code to https://github.com/sivamule2023/testappforcicd So, i will be using this in next step. You can use your repository

In pom.xml, I have configured mule maven plugin as shown below:

You can change your pom.xml accordingly and change your credentials

```
<plugin>
					<groupId>org.mule.tools.maven</groupId>
					<artifactId>mule-maven-plugin</artifactId>
					<version>${mule.maven.plugin.version}</version>
					<extensions>true</extensions>
```

```
                        <configuration>
                              <cloudHubDeployment>
                                    <uri>https://anypoint.mulesoft.com</uri>
                                    <muleVersion>${app.runtime}</muleVersion>
                                    <username>myusername</username>
                                    <password>mypassword</password>

<applicationName>testappbysiva</applicationName>
                                    <environment>Sandbox</environment>

                                    <workers>1</workers>
                                    <workerType>MICRO</workerType>
                                    <properties>
                                          <key>value</key>
                                    </properties>
                              </cloudHubDeployment>
                        </configuration>

                  </plugin>
```

You can create a connected app in anypoint platform and obtain client_id and client_secret.

You can replace username and password tags with below tags

```
<connectedAppClientId>your-CLIENT_ID</connectedAppClientId>
<connectedAppClientSecret>your CLIENT_SECRET</connectedAppClientSecret>

      <connectedAppGrantType>client_credentials</connectedAppGrantType>
```

Create a file with name deploy.cmd with below content:

```
mvn clean deploy -DmuleDeploy -DskipMunitTests  -U
```

Commit and push changes to Git hub.

"In the Jenkins dashboard, click on "+New Item" , enter any name ,select "FreeStyle Project" and click ok

Under source code management, select "Git" and give your github repository url.

Under Credentials, Click on Add and add your GIthub Username and password. Select the credentials u added just now

We want to poll Github for every 5 minutes.

So, Under Build Triggers, Select Poll SCM and configure schedule as H/5 * * * *

Under Build Steps, select “Execute Windows batch command” and configure the below command:

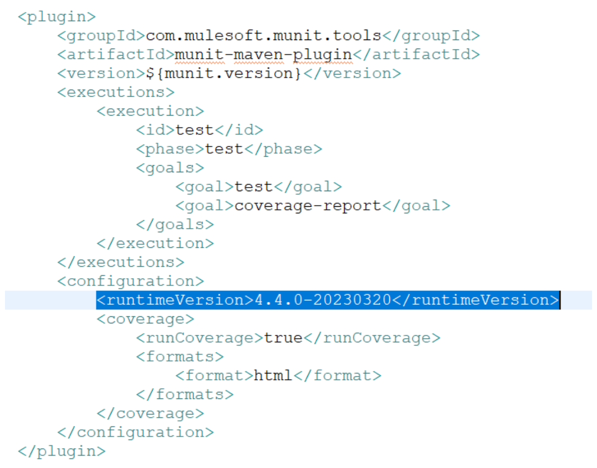
deploy.cmd (This is the file u created under your repository root folder)

Click Save.

A build will be triggered after 5 minutes. But to build immediately, click on “Build now” link on the left menu

Build should be successful and your application should be deployed in your cloudhub account.

Remember we are skipping MunitTests because maven will try to download Embedded Mule Runtime from the nexus repository. For that we need nexus repo password which comes with license. Actually, if u have executed Munit tests on studio, studio should have downloaded embedded Mule runtime. We have to add that correct version of embedded runtime which studio has downloaded under munit-maven-plugin configuration in pom.xml as shown below:



```
<plugin>
    <groupId>com.mulesoft.munit.tools</groupId>
    <artifactId>munit-maven-plugin</artifactId>
    <version>${munit.version}</version>
    <executions>
        <execution>
            <id>test</id>
            <phase>test</phase>
            <goals>
                <goal>test</goal>
                <goal>coverage-report</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <runtimeVersion>4.4.0-20230320</runtimeVersion>
        <coverage>
            <runCoverage>true</runCoverage>
            <formats>
                <format>html</format>
            </formats>
        </coverage>
    </configuration>
</plugin>
```

Also, u should copy mule folder under your ~/.m2/repository/com and ~/.m2/repository/org

Under **C:\Windows\System32\config\systemprofile\.m2\repository** folder . This is just a work around. If u have license and got nexus repo username/password, these steps are not required

If u have done like above, u can enable Munit tests also. U can try this when ever u have time later.

If you want to deploy your application to standalone server, u have to configure mule maven plugin as shown below:

```
<plugin>
  <groupId>org.mule.tools.maven</groupId>
  <artifactId>mule-maven-plugin</artifactId>
  <version>3.7.1</version>
  <extensions>true</extensions>
  <configuration>
    <armDeployment>
      <muleVersion>${app.runtime}</muleVersion>
      <uri>https://anypoint.mulesoft.com</uri>
      <target>${target}</target>
      <targetType>${target.type}</targetType>
      <username>${username}</username>
      <password>${password}</password>
```

```
        <environment>${environment}</environment>
        <properties>
          <key>value</key>
        </properties>
      </armDeployment>
    </configuration>
  </plugin>
```