

# Mulesoft Deployments Guide

Deployment

Jagadishwar Reddy  
MULEBYTES Bangalore

# Contents

Introduction-----	2
Concept of Deployment-----	2
Types of Mule Runtime -----	2
Cloudhub -----	2
Runtime Manager -----	2
Worker-----	2
Worker Size (vCore) -----	2
On-Premise -----	3
How does Mulesoft Runtime work -----	3
How to Setup Mulesoft On-Premise Runtime -----	3
How to add On-Premise Mule Runtime to Anypoint Platform (AMC Setup) -----	5
Deployment Methods -----	6
Deployment via Direct .zip or .jar Upload-----	6
CloudHub Deployment by Directly Uploading the .zip or .jar -----	6
On-Premise Deployment by Directly placing the .zip or .jar -----	8
Deployment via Anypoint Studio -----	10
CloudHub Deployment from Anypoint Studio -----	10
On-Premise Deployment from Anypoint Studio -----	11
Deployment via Jenkins CI/CD -----	12
Pre-Requisites -----	12
CI/CD Overview -----	12
Environment Setup -----	13
Create a Jenkins Build Job -----	19
Create a Jenkins Deploy Job -----	22

# Introduction

## Concept of Deployment

Deployment is a process of Bringing your Application to ACTIVE state by RUNNING it in RUNTIME. Mulesoft applications require MULE RUNTIME to bring them to working (running) state.

**Deployable Archive:** File Generated by Exporting Mule Project, Mule 3 (.zip), Mule 4 (.jar)

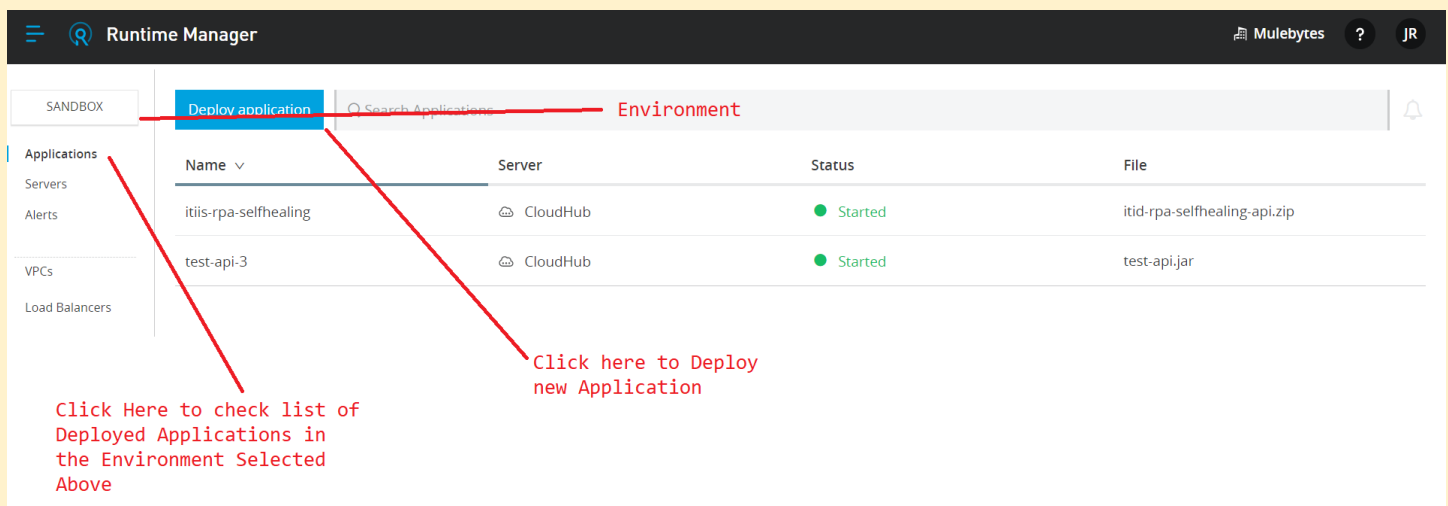
## Types of Mule Runtime

### Cloudhub

CloudHub is an Integration Platform as a Service (iPaaS). It enables you to deploy and run the application in the cloud via Runtime Manager. CloudHub is a scalable, multi-tenant, elastic, secure, and highly available iPaaS. CloudHub is managed via the Runtime Manager console in the Anypoint platform.

### Runtime Manager

Generic URL to Access Runtime Manager: <https://anypoint.mulesoft.com/cloudhub>



### Worker

Worker is a Virtual Mule Server Instance, where Application will be deployed

### Worker Size (vCore)

Worker Size or vCore is Max Memory assigned to a Worker.

Worker Type	Number of vCores for Max Use	vCore Size
MICRO	0.1 vCore	500 MB
SMALL	0.2 vCore	1 GB
MEDIUM	1 vCore	1.5 GB
LARGE	2 vCore	3.5 GB
XLARGE	4 vCore	7.5 GB
XXLARGE	8 vCore	15 GB
4XLARGE	16 vCore	32 GB

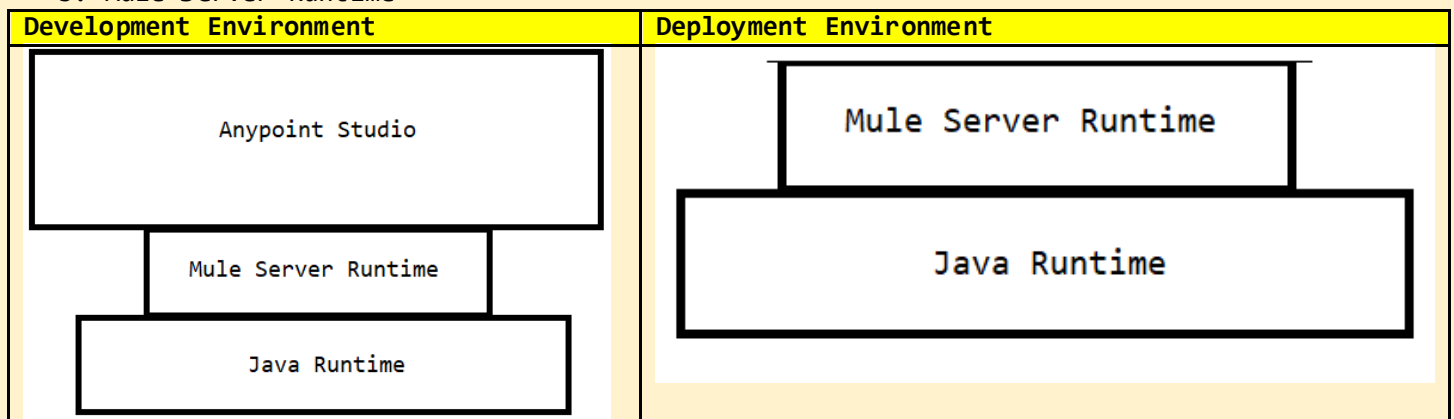
## On-Premise

On-premises software (commonly misstated as on-premise, and alternatively abbreviated "on-prem") is installed and runs on computers on the premises of the person or organization using the software, rather than at a remote facility such as a server farm or cloud.

### How does Mulesoft Runtime work

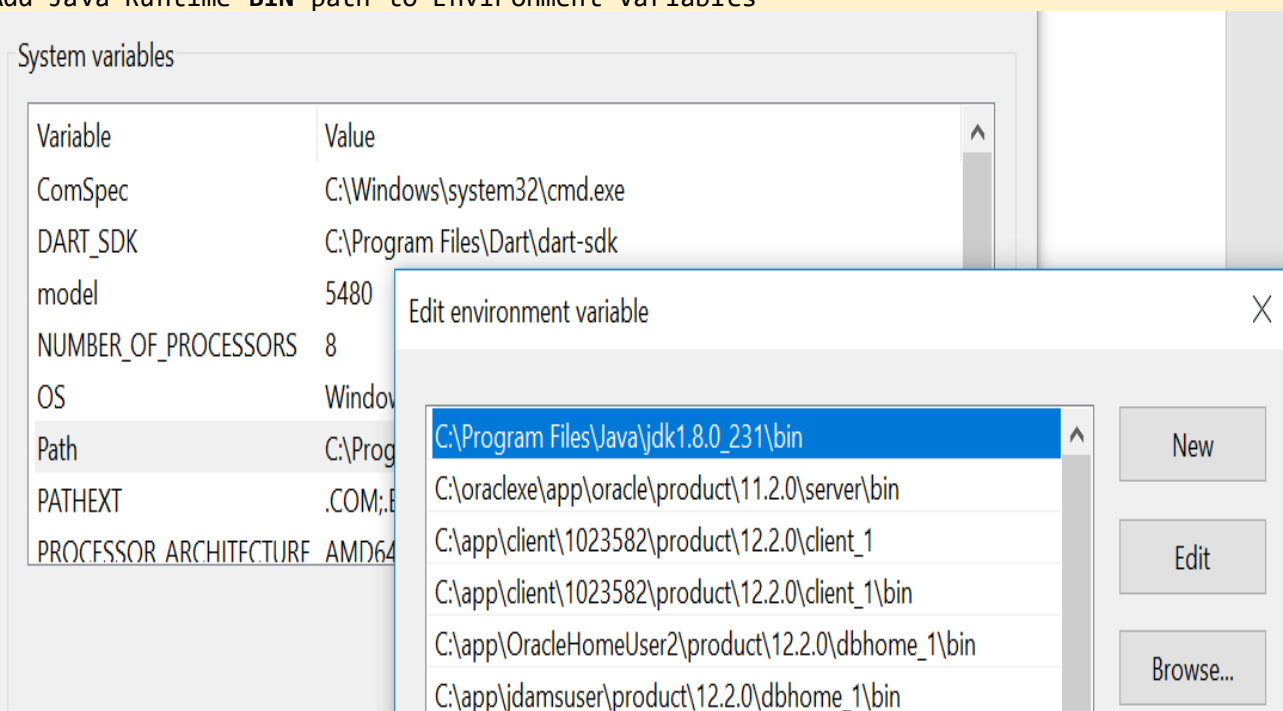
Mulesoft is a Java based integration tool, which mean it requires Java to run. Every server which needs to be called as Mule Server should have

- a. Java Runtime
- b. Mule Server Runtime



### How to Setup Mulesoft On-Premise Runtime

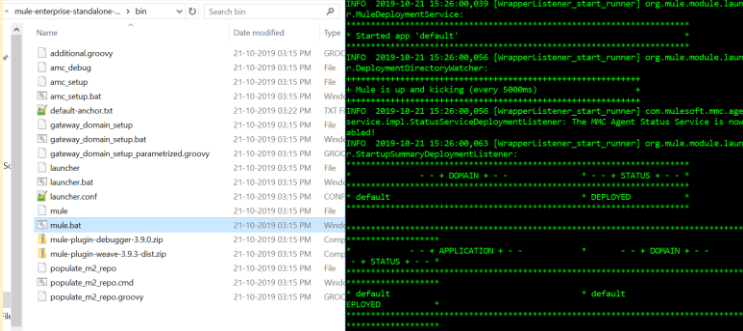
1. Setup Java Runtime
2. Add Java Runtime **BIN** path to Environment Variables



### 3. Download Mulesoft Standalone Runtime:

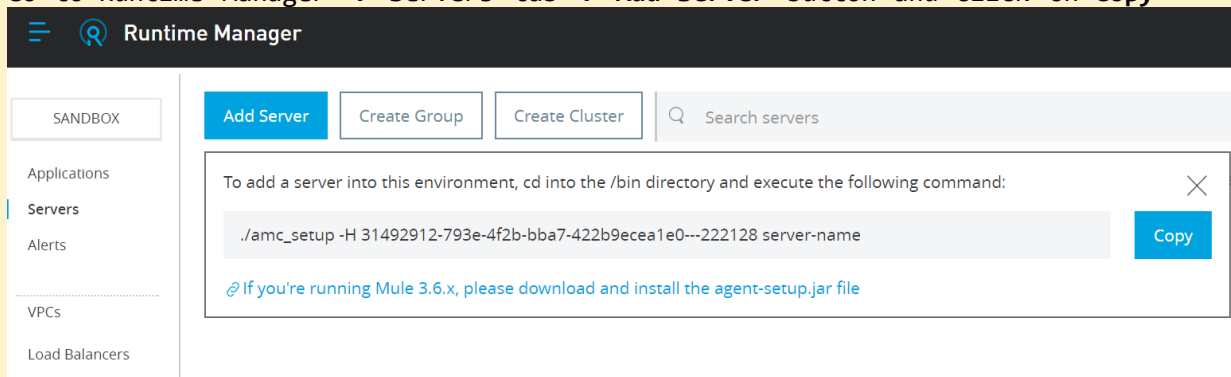
4. Extract the Downloaded Zip File and Open the Folder to see below

> This PC > Local Disk (C:) > Training Area > Mule Runtimes > mule-enterprise-standalone-3.9.3				
	Name	Date modified	Type	Size
File	apps	21-10-2019 03:15 PM	File folder	
	bin	21-10-2019 03:15 PM	File folder	
	conf	21-10-2019 03:15 PM	File folder	
	docs	21-10-2019 03:15 PM	File folder	
	domains	21-10-2019 03:15 PM	File folder	
	examples	21-10-2019 03:15 PM	File folder	
	lib	21-10-2019 03:15 PM	File folder	
	logs	21-10-2019 03:15 PM	File folder	
Softw	plugins	21-10-2019 03:15 PM	File folder	
	policies	27-05-2019 10:41 PM	File folder	
	tools	21-10-2019 03:15 PM	File folder	
	LICENSE.txt	21-10-2019 03:15 PM	TXT File	1 KB
	MIGRATION.txt	21-10-2019 03:15 PM	TXT File	38 KB
	README.txt	21-10-2019 03:15 PM	TXT File	4 KB

Folder Name	Purpose
apps	Folder where Deployed Applications will be saved with <<app-name>>-anchor.txt to verify they are deployed and Running.
bin	Contains, Commands to Start/Stop Mulesoft Runtime and to manage: 
logs	Application Log Files are saved here
conf	Contains Mulesoft Runtime License and Configuration File (wrapper.conf) for Mulesoft Runtime

## How to add On-Premise Mule Runtime to Anypoint Platform (AMC Setup)

1. Login to Anypoint Platform <https://anypoint.mulesoft.com/home/>
2. Go to Runtime Manager → Servers tab → Add Server button and click on Copy

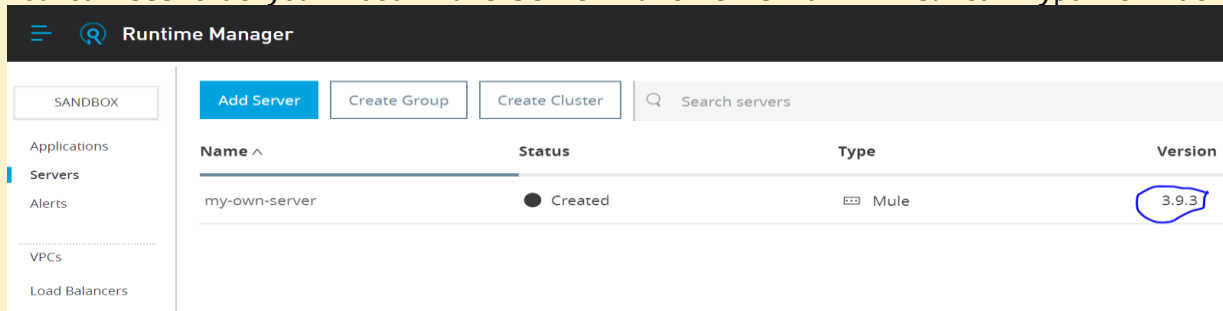


**server-name:** Give your own name to the server  
Execute following command in MULE\_HOME\bin

```
C:\Training Area\Mule Runtimes\mule-enterprise-standalone-3.9.3\bin>amc_setup -H 31492912-793e-4f2b-bba7-422b9ecea1e0---222128 my-own-server
Mule Agent Installer
-----
For help please run the script with --help option

INFO: MULE_HOME is set as C:\Training Area\Mule Runtimes\mule-enterprise-standalone-3.9.3
```

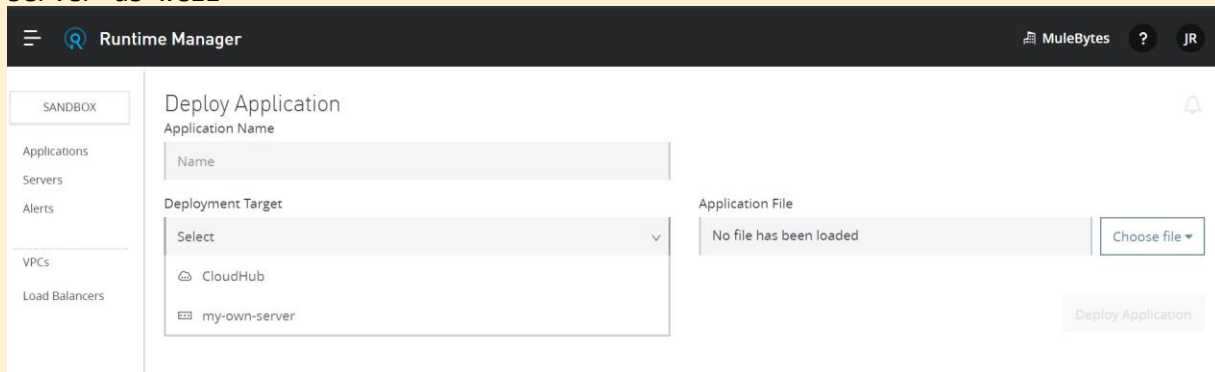
3. You can see that your local Mule Server Runtime is now Linked to Anypoint Platform:



4. Start Mule Server by going to MULE\_HOME\bin open CMD and start Mule server by executing command mule

```
C:\mule-ee-distribution-standalone-3.9.3\mule-enterprise-standalone-3.9.3\bin>mule
MULE_HOME is set to C:\mule-ee-distribution-standalone-3.9.3\mule-enterprise-standalone-3.9.3
Running in console/foreground mode by default, use Ctrl-C to exit...
--> Wrapper Started as Console
Java Service Wrapper Standard Edition 64-bit 3.5.37
Copyright (C) 1999-2018 Tanuki Software, Ltd. All Rights Reserved.
http://wrapper.tanukisoftware.com
Licensed to MuleSoft Inc. for Mule Runtime Enterprise Edition
```

5. Now try to Deploy an Application, you can see that Deployment Target now has your local server as well

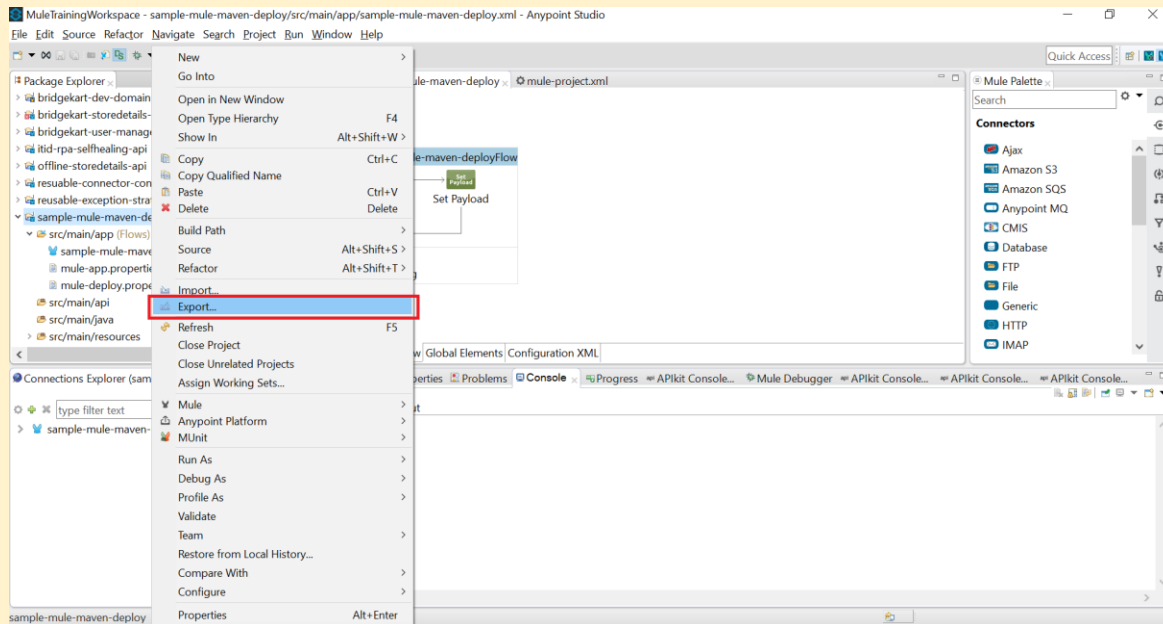


# Deployment Methods

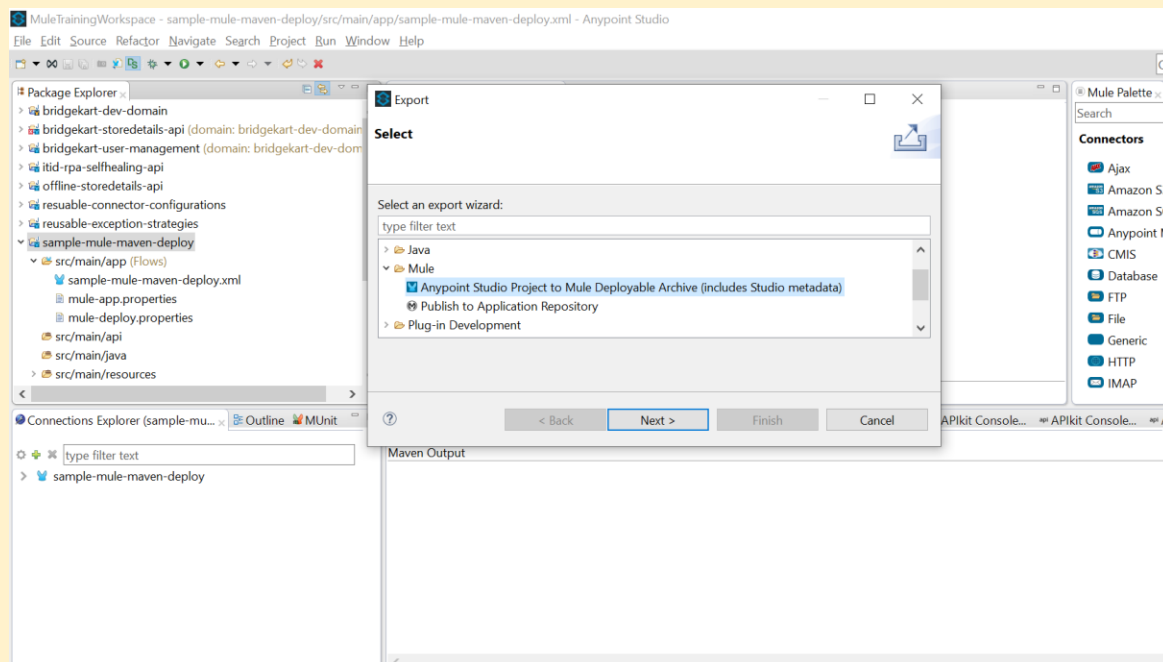
## Deployment via Direct .zip or .jar Upload

### CloudHub Deployment by Directly Uploading the .zip or .jar

#### 1. Right Click on Project and Click on Export



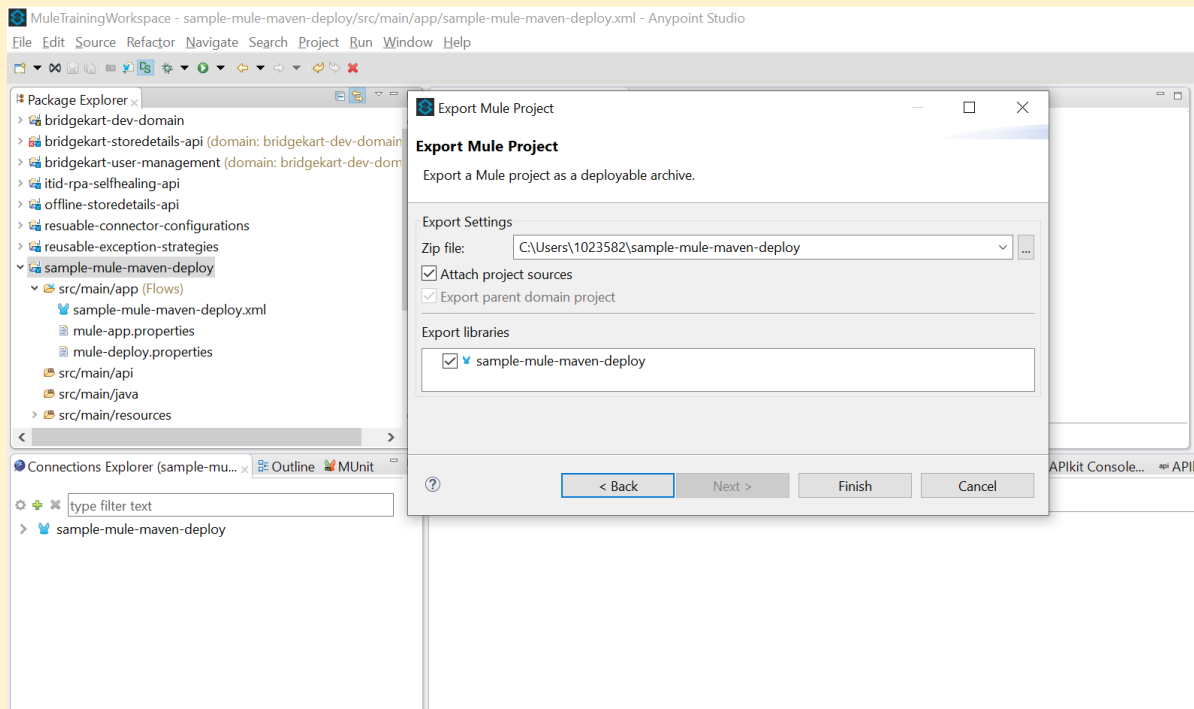
#### 2. Select Mule → Anypoint Studio Project to Mule Deployable Archive




#### 3. Select the Location, where Project will be exported

Select "Attach Project Sources" and if "Export Parent domain project" is enabled, select


that option as well and Click on “Finish”



4. Project is Exported in filesystem as below:

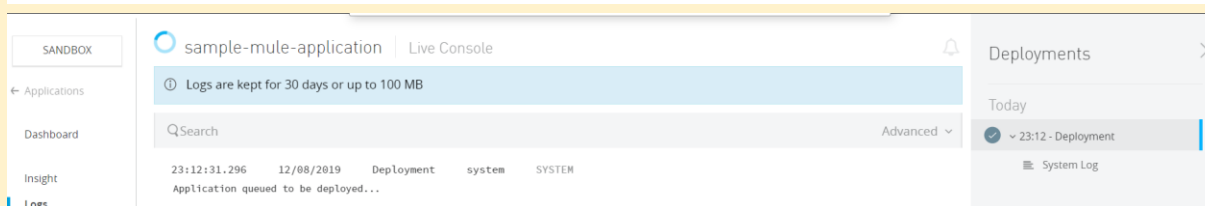
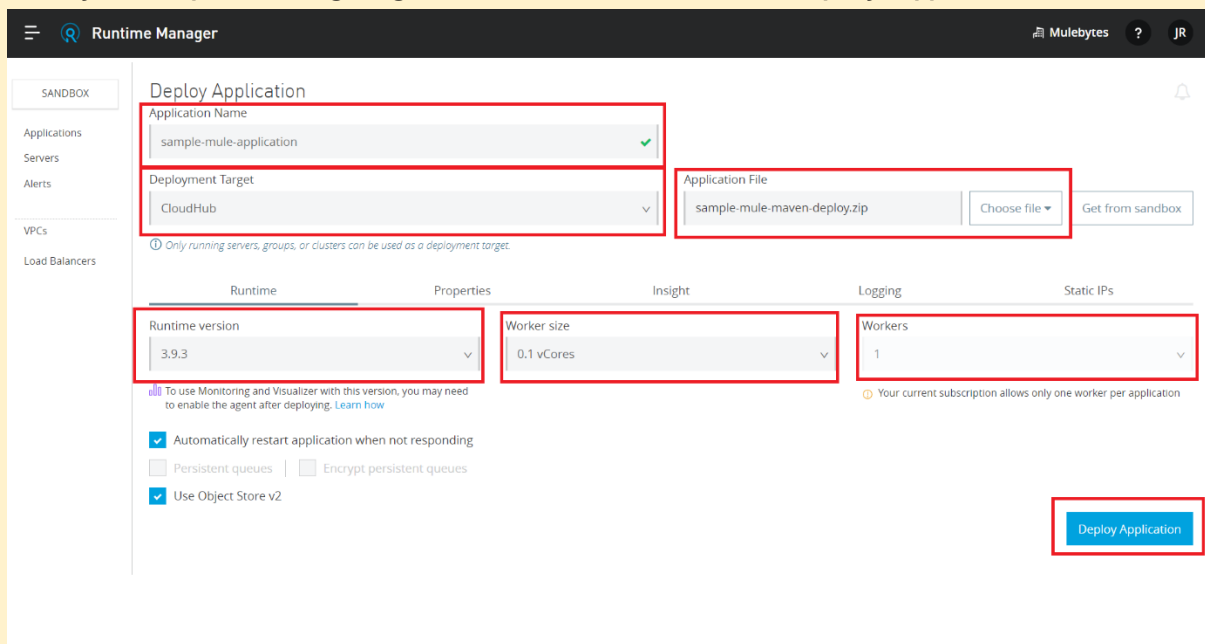
 sample-mule-maven-deploy.zip	08-12-2019 11:07 PM	Compressed (zipped)...	10 KB
--	---------------------	------------------------	-------

For Mule 4 .jar file will be generated:

 sample-mule-maven-deploy.jar	08-12-2019 11:07 PM	JAR File	10 KB
--	---------------------	----------	-------

5. Goto Anypoint Platform → Runtime Manager → Deploy Application

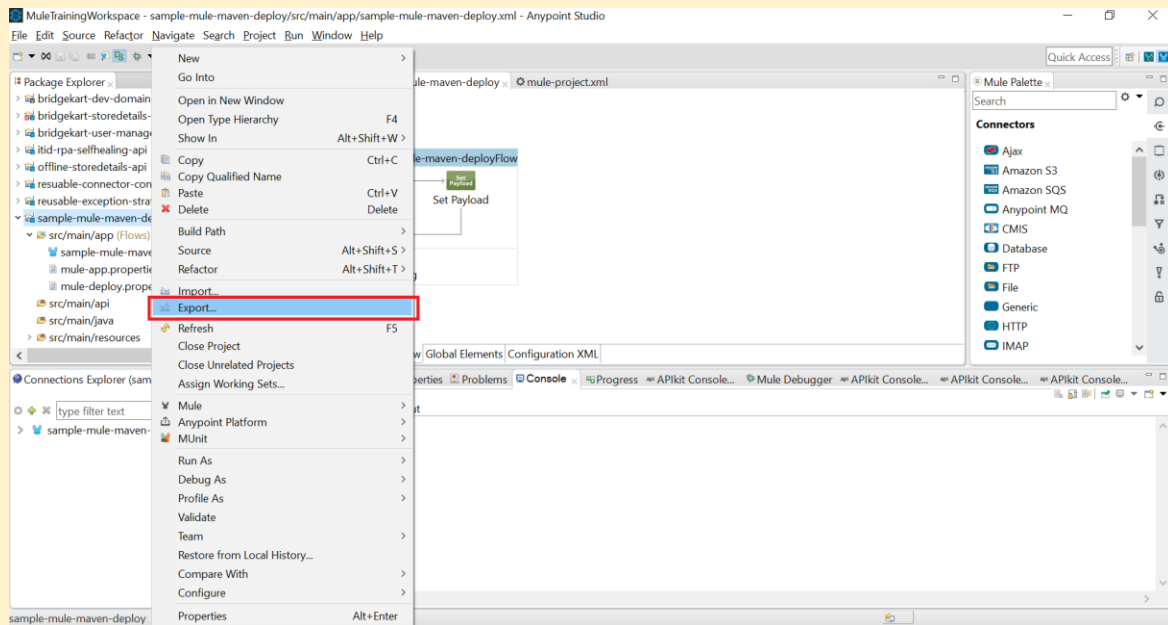
Verify the Options highlighted below and click on “Deploy Application”



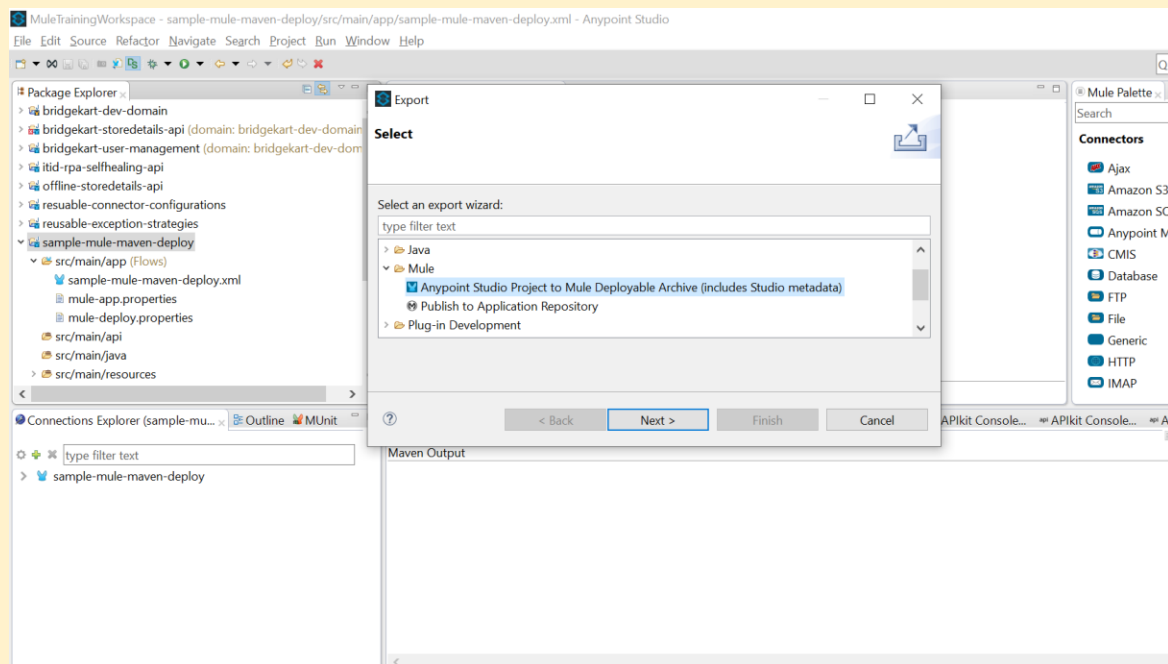


# On-Premise Deployment by Directly placing the .zip or .jar

## 1. Right Click on Project and Click on Export



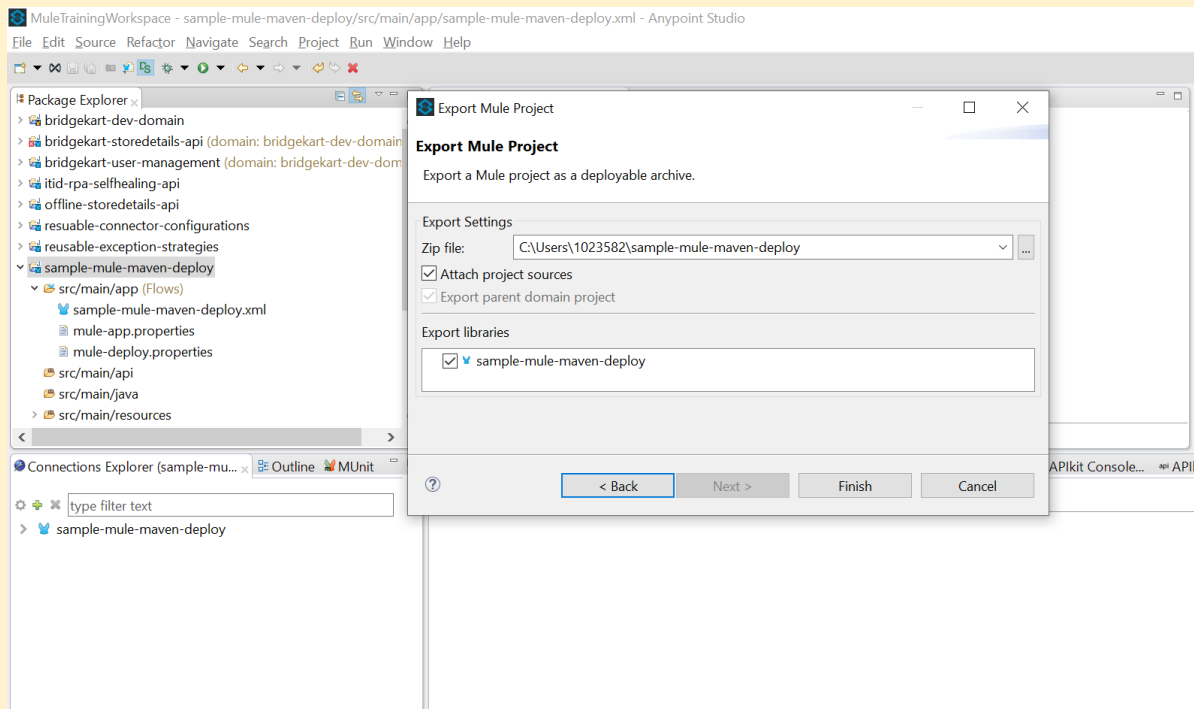
## 2. Select Mule → Anypoint Studio Project to Mule Deployable Archive




## 3. Select the Location, where Project will be exported

Select "Attach Project Sources" and if "Export Parent domain project" is enabled, select


that option as well and Click on “Finish”



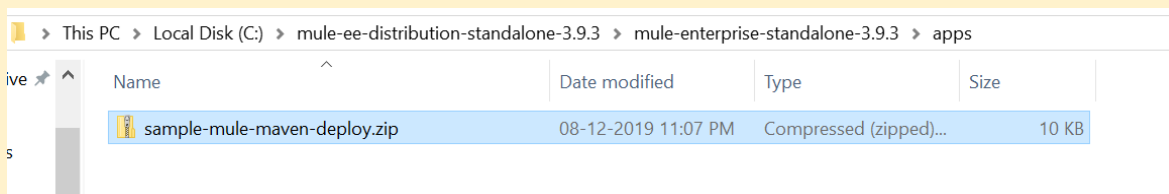
4. Project is Exported in filesystem as below:

 sample-mule-maven-deploy.zip	08-12-2019 11:07 PM	Compressed (zipped)...	10 KB
--	---------------------	------------------------	-------

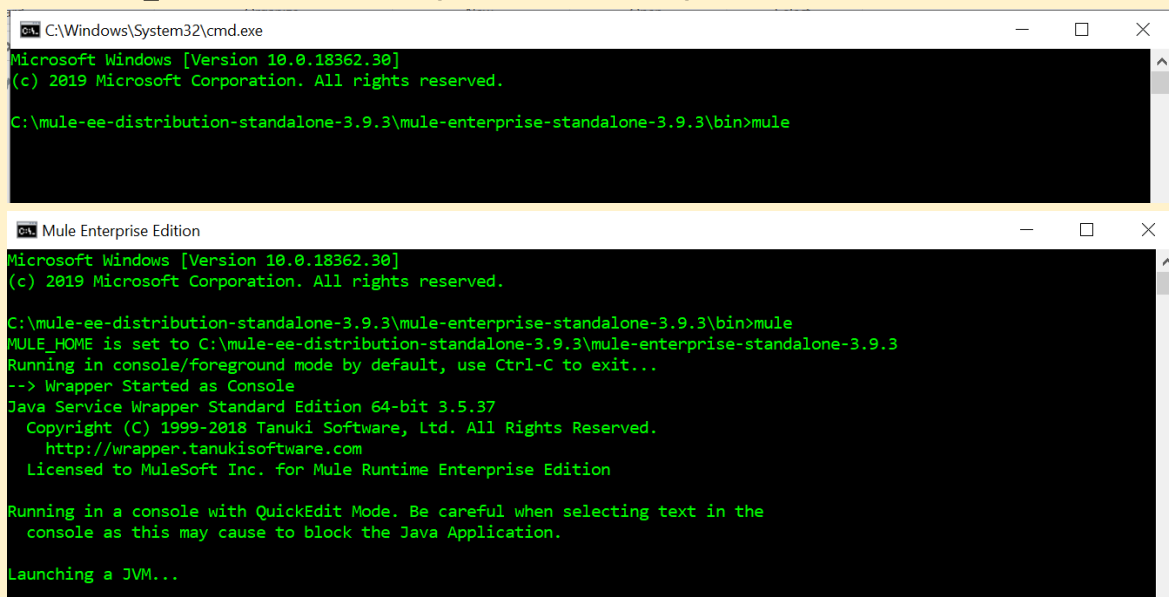
For Mule 4 .jar file will be generated:

 sample-mule-maven-deploy.jar	08-12-2019 11:07 PM	JAR File	10 KB
--	---------------------	----------	-------

5. Goto MULE\_HOME\apps folder, place the Generated Deployable Archive



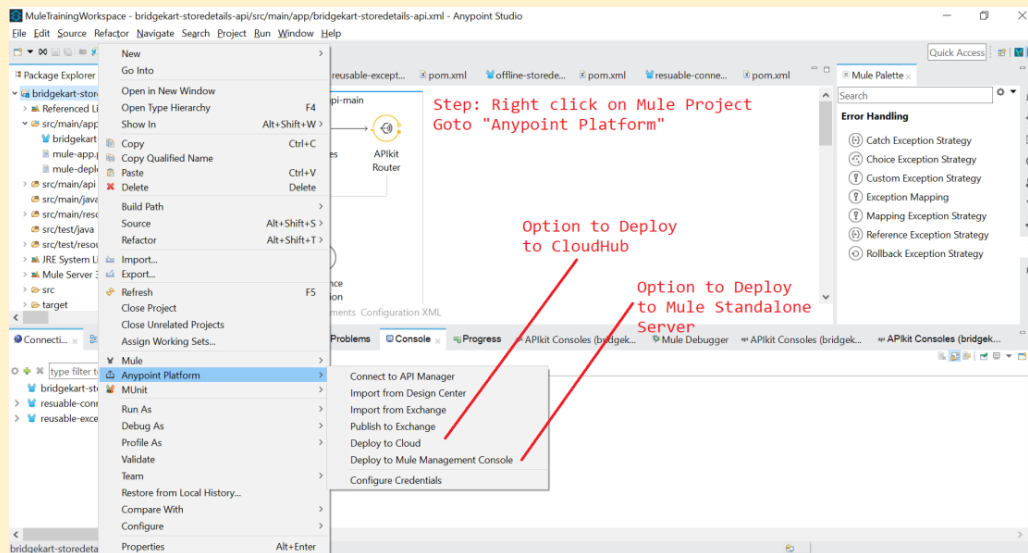
6. Goto MULE\_HOME\bin folder, open Command Prompt and enter mule command



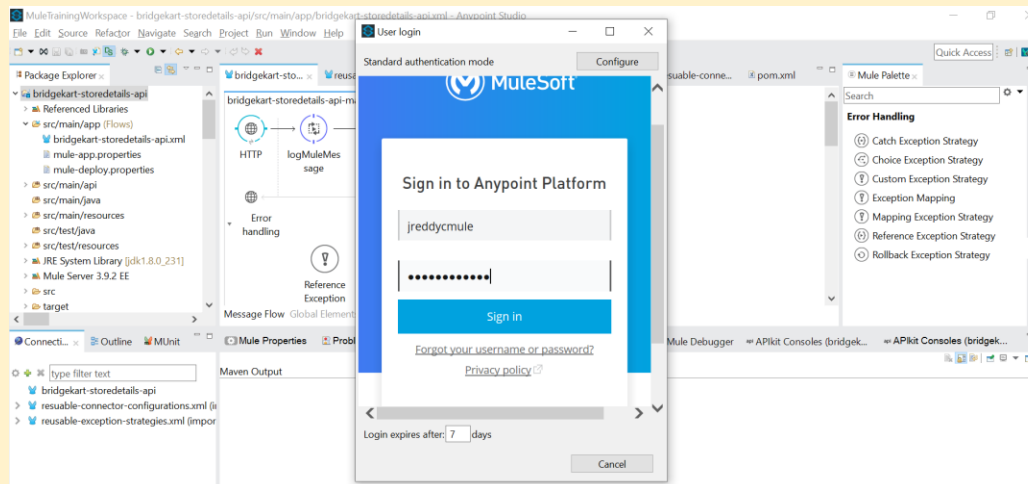
# Deployment via Anypoint Studio

## CloudHub Deployment from Anypoint Studio

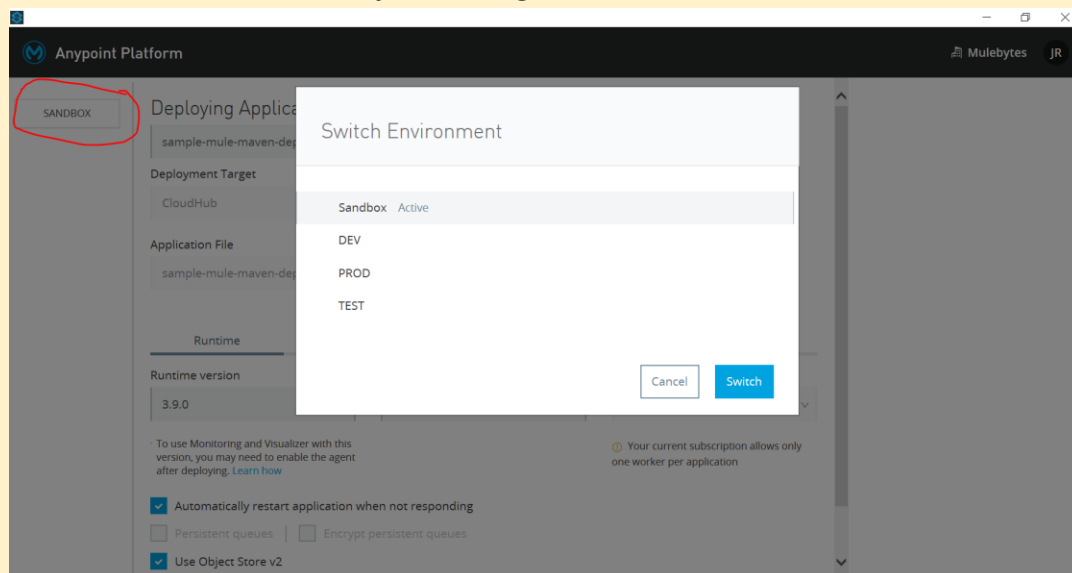
### 1. Right Click on the Project → Anypoint Platform → Deploy to Cloud



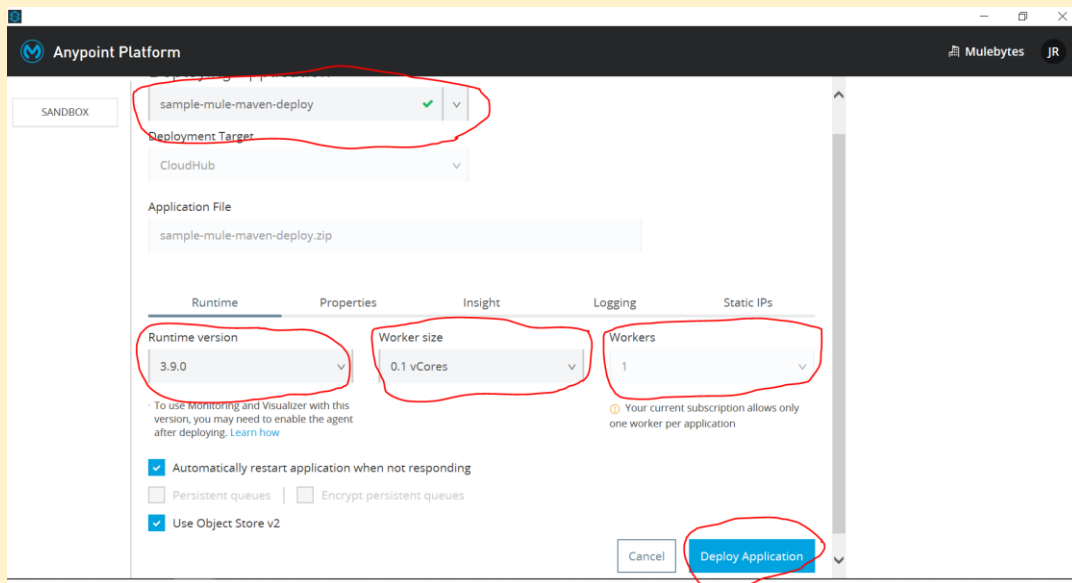
### 2. Login with Anypoint Platform Credentials



### 3. Select the Environment by clicking on the Red Circle above and click on "Switch"

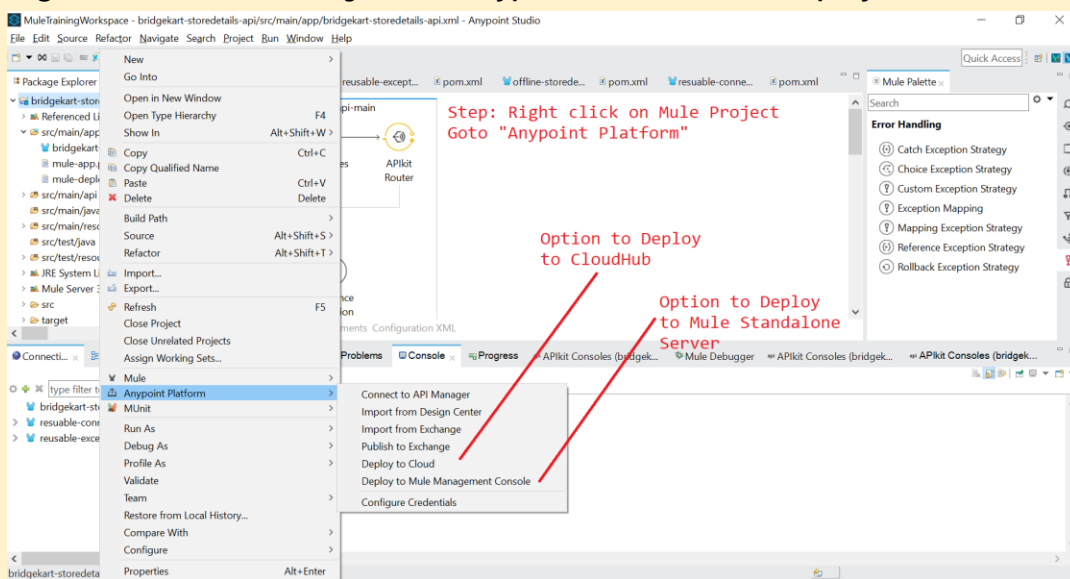


#### 4. Verify Highlighted configurations below and Click on “Deploy Application”

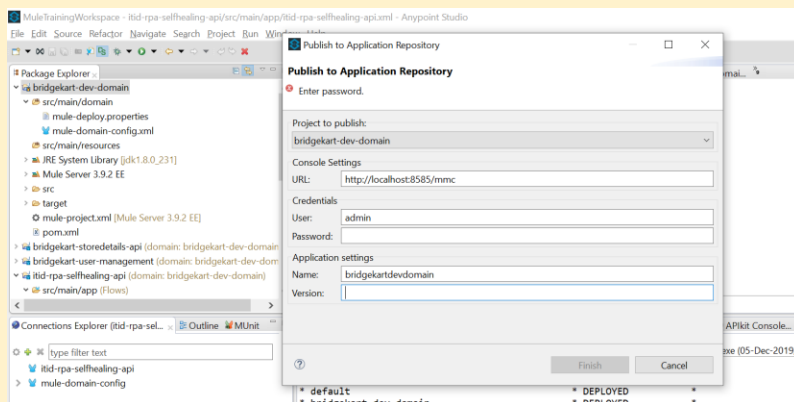


### On-Premise Deployment from Anypoint Studio

#### 1. Right Click on the Project → Anypoint Platform → Deploy to Mule Management Console



2. Enter MMC URL (Provided by your project)  
Enter Credentials (User/Password) details  
Enter Application Settings details  
Click on “Finish”



# Deployment via Jenkins CI/CD

## Pre-Requisites

1. Code Repository (GitHub, Bitbucket, Stash etc.)
2. Maven
3. Mule Runtime (On-Premise or Cloudhub)

## CI/CD Overview

CI (Continuous Integration):

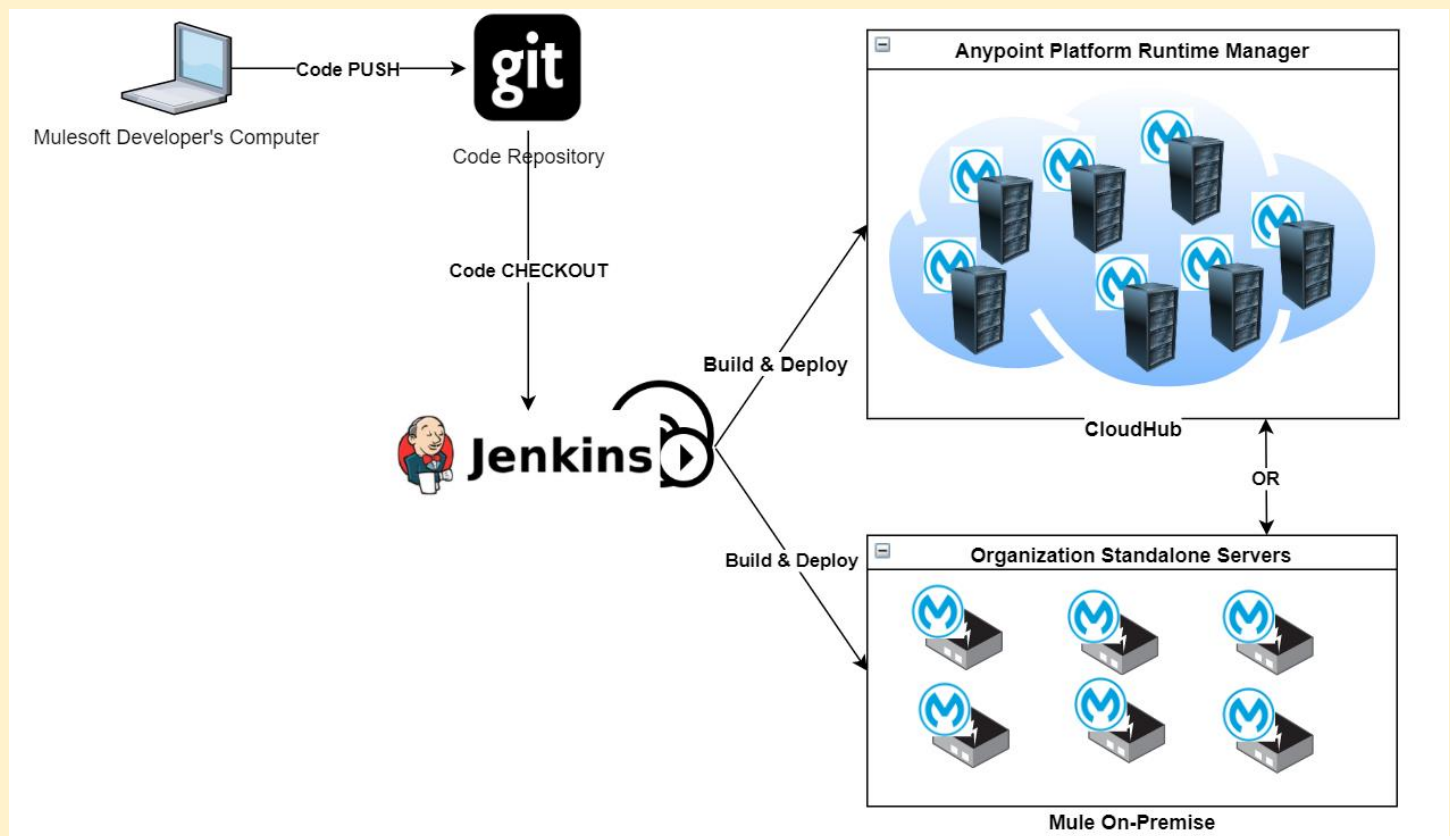
Process of building the Deployable Archive as and when Developer Check-in the code to Code repository.

CD (Continuous Deployment):

Process of Deploying the Archive Generated as part of CI process above to the desired environment as and when Developer Check-in the code to Code repository.

Using CI/CD Process you can automate the process of Deployment and can reduce the risk of human errors.

Following diagram explains, how CI/CD process works.




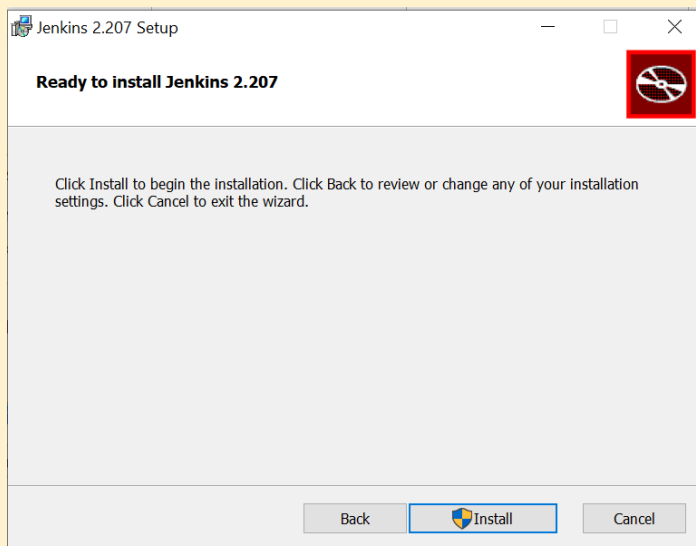
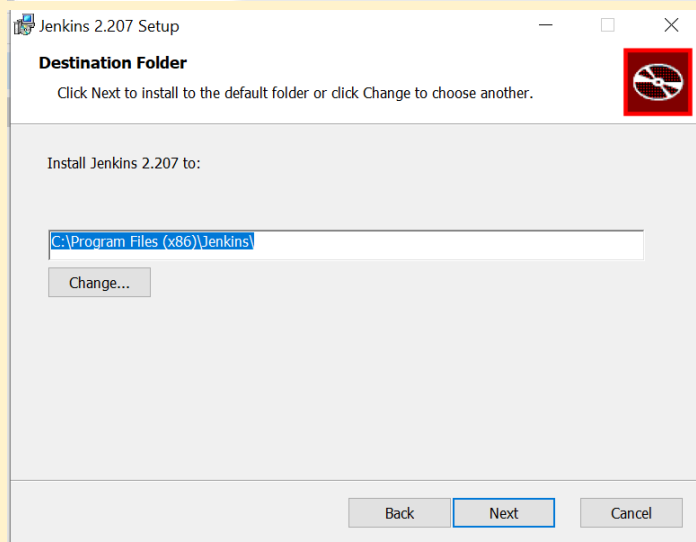
## Environment Setup

1. Download Jenkins from below link and Install it:

<https://jenkins.io/download/thank-you-downloading-windows-installer/>

Extract Windows Installer from the downloaded .zip file and Double Click on below file to install it.

Name	Date modified	Type	Size
 jenkins.msi	08-12-2019 09:11 PM	Windows Installer Pa...	1,08,208 KB



2. After Installation is complete, you can access Jenkins by using below URL:

<http://localhost:8080/>

Open the file shown in below path and copy the content and paste it in below text box and click on “Continue”

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:\Jenkins\secrets\initialAdminPassword

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

Administrator password

Continue

Click on “Install Suggested Plugins”

Getting Started

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

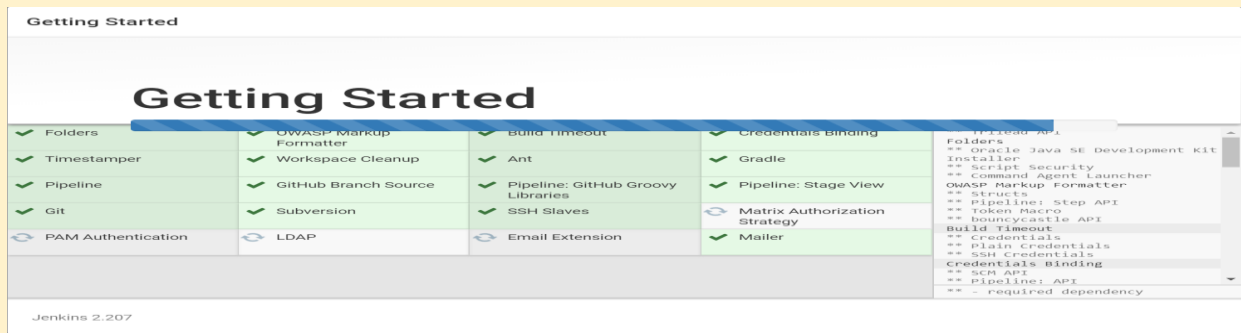
Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.207

Let all the Plugins install in the next screen



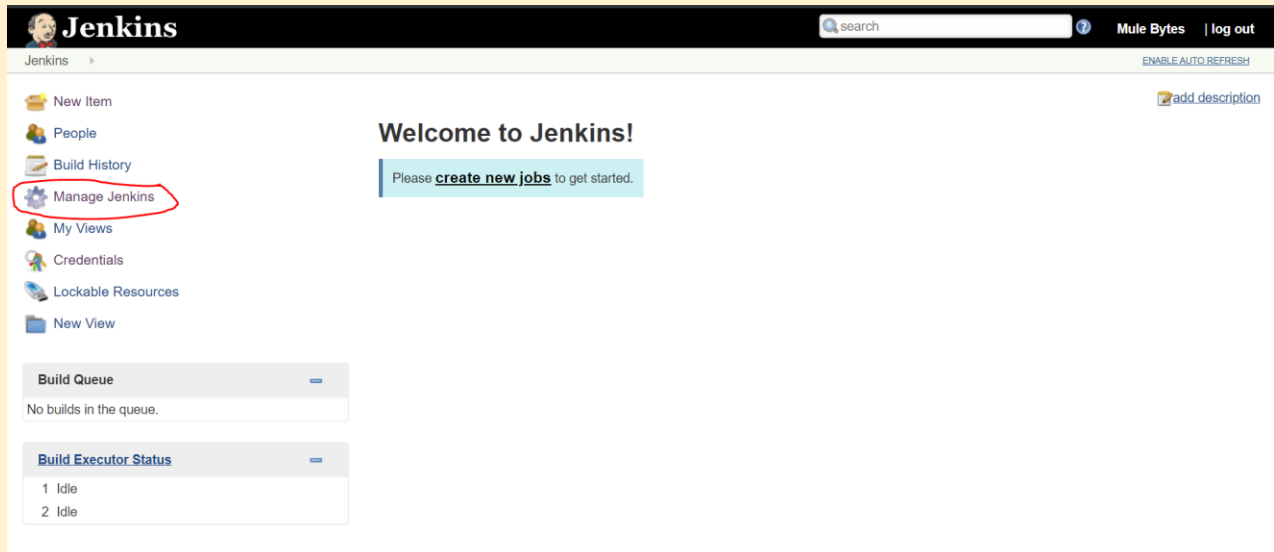
Create First Admin User and Click on “Save and Continue”

The screenshot displays the 'Create First Admin User' form in Jenkins 2.207. The form includes fields for Username (mulebytes), Password (masked with dots), Confirm password (masked with dots), Full name (Mule Bytes), and E-mail address (mulebytes@gmail.com). At the bottom right, there are two buttons: 'Continue as admin' and 'Save and Continue'. The 'Save and Continue' button is highlighted with a red circle. The bottom left corner shows 'Jenkins 2.207'.The screenshot shows the 'Instance Configuration' page in Jenkins 2.207. The 'Jenkins URL' field is set to 'http://localhost:8080/'. Below this, there is explanatory text about the Jenkins URL and its importance for various features. At the bottom right, there are two buttons: 'Not now' and 'Save and Finish'. The 'Save and Finish' button is highlighted with a red circle. The bottom left corner displays 'Jenkins 2.207'.



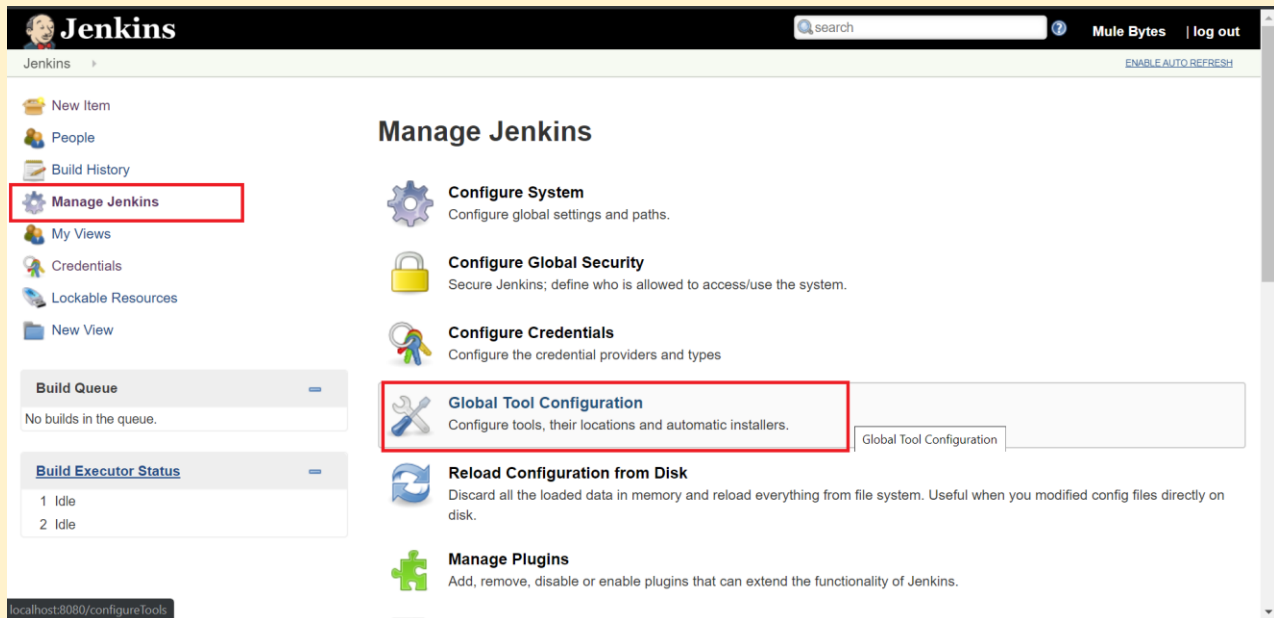
### 3. Configure Maven and Java installation Paths in Jenkins

Go to “Manage Jenkins”



The Jenkins Welcome Page shows the main dashboard. On the left sidebar, the 'Manage Jenkins' option is highlighted with a red circle. The main content area displays a 'Welcome to Jenkins!' message with a button to 'create new jobs'. Below this, there are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing 2 idle executors).

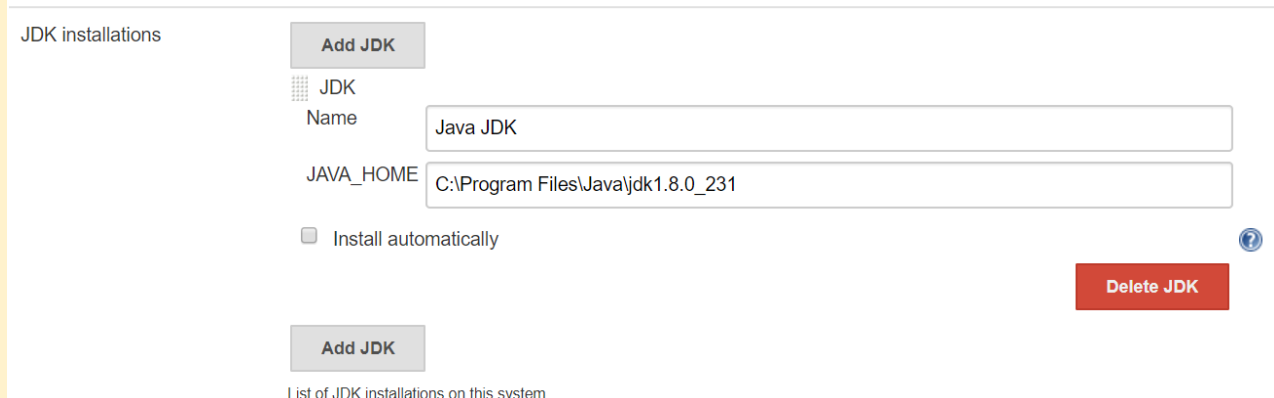
Select “Global Tool Configuration”



The 'Manage Jenkins' page is shown. On the left sidebar, 'Manage Jenkins' is highlighted with a red box. The main content area lists several configuration options: 'Configure System', 'Configure Global Security', 'Configure Credentials', 'Global Tool Configuration' (highlighted with a red box), 'Reload Configuration from Disk', and 'Manage Plugins'. The 'Global Tool Configuration' option is selected, and its sub-page is visible in the background.

### Set JDK Path

#### JDK




The 'JDK' configuration page is shown. It features a 'JDK installations' section with an 'Add JDK' button. Below this, there is a form for adding a new JDK installation. The 'Name' field is set to 'Java JDK' and the 'JAVA\_HOME' field is set to 'C:\Program Files\Java\jdk1.8.0\_231'. There is an unchecked checkbox for 'Install automatically'. At the bottom right, there is a 'Delete JDK' button. Below the form, there is another 'Add JDK' button and a note: 'List of JDK installations on this system'.

## Set GIT installation Path

**Git**

Git installations

 **Git**

Name

Path to Git executable

☐ Install automatically

**Delete Git**


**Add Git**

## Set Maven Installed Path

**Maven**

Maven installations

**Add Maven**

 **Maven**

Name

MAVEN\_HOME

☐ Install automatically

**Delete Maven**

**Add Maven**

List of Maven installations on this system

After above configurations Click on “Save” or “Apply”

Jenkins » Global Tool Configuration


**Ant**

Ant installations **Add Ant**

List of Ant installations on this system

**Maven**

Maven installations **Add Maven**

 **Maven**

Name

MAVEN\_HOME

☐ Install automatically

**Delete Maven**

**Add Maven**

List of Maven installations on this system

**Docker**

Docker installations **Add Docker**

List of Docker installations on this system

**Save** **Apply**

After above Setup is complete, Jenkins is now capable of Downloading the projects from Git Repository, Building the Mule Projects to generate “Mule Deployable Archive” and the same can be Deployed to CloudHub or On-Premise.

Jenkins is now Running in your Localhost and you can access it using <http://localhost:8080/> URL.

#### 4. Install Maven Plugin

Go to Manage Jenkins → Manage Plugins

The screenshot shows the Jenkins 'Manage Jenkins' page. On the left sidebar, the 'Manage Jenkins' link is highlighted with a red box. The main content area lists several configuration options: 'Configure System', 'Configure Global Security', 'Configure Credentials', 'Global Tool Configuration', and 'Reload Configuration from Disk'. The 'Manage Plugins' option, represented by a green puzzle piece icon, is highlighted with a red box and described as 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.'

The screenshot shows the Jenkins 'Plugin Manager' page. The 'Available' tab is selected. A search filter 'Maven' is entered in the top right. The 'Maven Integration' plugin is selected with a checkbox. The 'Install without restart' button is highlighted with a red box. The table below lists the available plugins:

Name	Version
<a href="#">View Job Filters</a>	2.1.1
<a href="#">Maven Artifact ChoiceListProvider (Nexus)</a>	1.5.1
<a href="#">Maven Metadata Plugin for Jenkins CI server</a>	2.0.0
<a href="#">Dependency Analyzer</a>	0.7
<input checked="" type="checkbox"/> <a href="#">Maven Integration</a>	3.4
<a href="#">Maven SNAPSHOT Check</a>	1.5

Available Tab in Manage Plugins page.

Search for "Maven"

Select "Maven Integration"

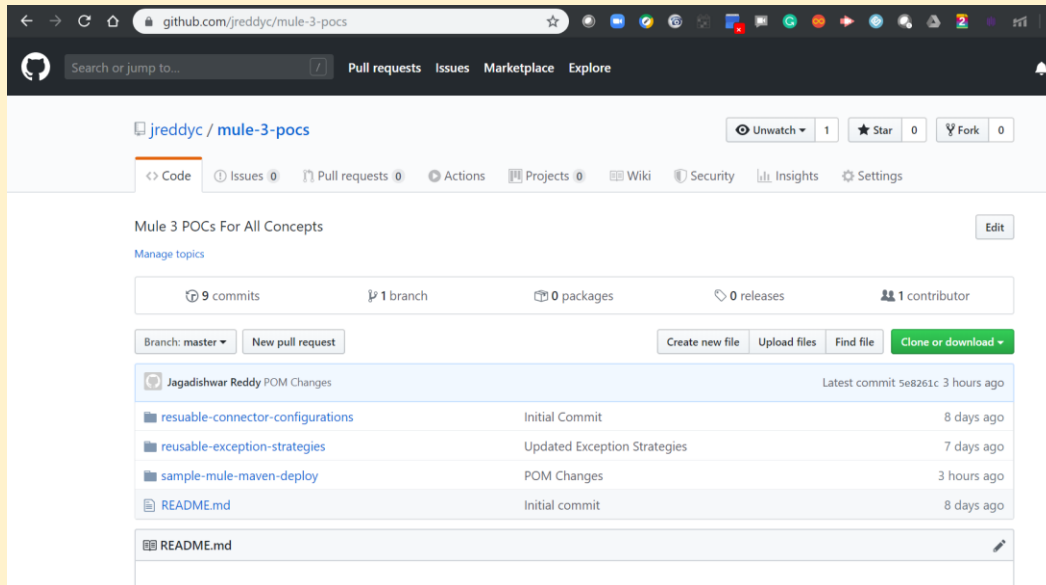
Click on "Install without restart"

After this Maven Plugin will be available in Jenkins

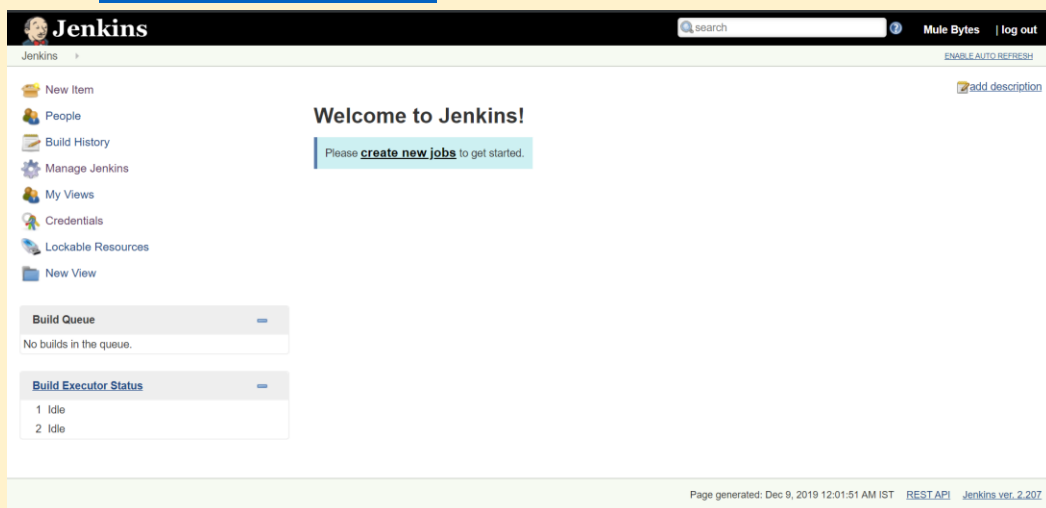
Post this Step Proceed to Create new Build Job

## Create a Jenkins Build Job

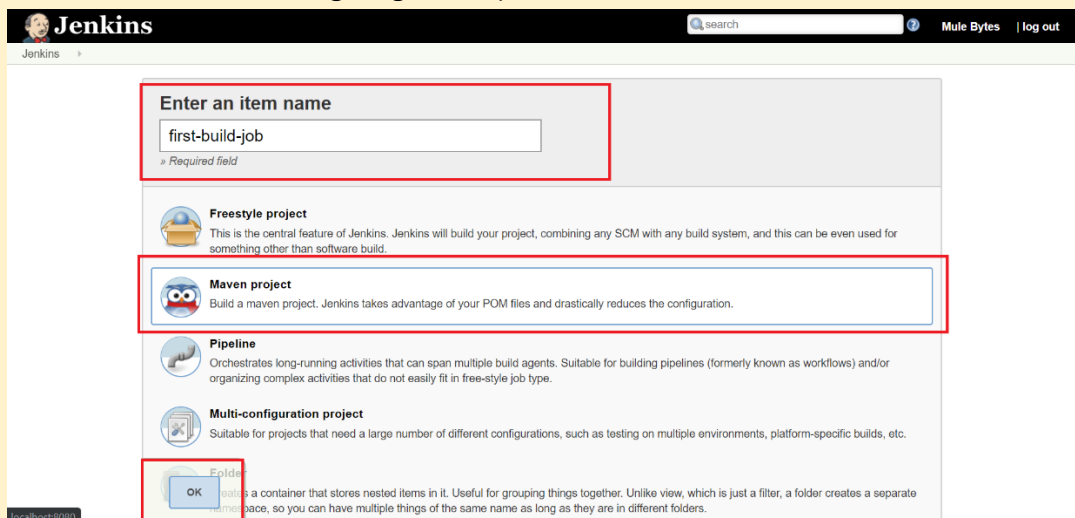
1. Make Sure Your Mulesoft Project is available in Git Repository



2. Click on Clone/Download Button and Copy the .git URL  
<https://github.com/jreddyc/mule-3-pocs.git>
3. Go to <http://localhost:8080/> Click on “Create New Jobs”



4. Enter/Select below highlighted options and click on “OK”



## 5. Configure Repository URL in “Source Code Management”

Click on “Add” to configure Github Credentials

The screenshot shows the Jenkins configuration page for Source Code Management. The 'Source Code Management' tab is selected. Under 'Post-build Actions', 'Git' is chosen. In the 'Repositories' section, the 'Repository URL' is set to 'https://github.com/jreddyc/mule-3-pocs.git'. The 'Credentials' dropdown is set to '- none -' with an 'Add' button next to it. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' set to '\*/master'. The 'Repository browser' is set to '(Auto)'. At the bottom, there are 'Save', 'Apply', and 'Add' buttons.

Configure Credentials as below and Click on “Add”

The screenshot shows the 'Add Credentials' dialog in Jenkins. The 'Domain' is set to 'Global credentials (unrestricted)'. The 'Kind' is set to 'Username with password'. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' is 'jagadishwar.cv@gmail.com'. The 'Password' is masked with dots. There are fields for 'ID' and 'Description'. At the bottom, there are 'Add' and 'Cancel' buttons.

Select the Configured Credentials in Above Step

The screenshot shows the Jenkins configuration page for Source Code Management, similar to the first image. The 'Source Code Management' tab is selected. Under 'Post-build Actions', 'Git' is chosen. In the 'Repositories' section, the 'Repository URL' is 'https://github.com/jreddyc/mule-3-pocs.git'. The 'Credentials' dropdown now shows 'jagadishwar.cv@gmail.com/\*\*\*\*\*' with an 'Add' button next to it. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' set to '\*/master'. The 'Repository browser' is set to '(Auto)'. At the bottom, there are 'Save', 'Apply', and 'Add' buttons.

6. Select the POM file location project → Configure “Goals and options” with Maven Command and Click on “Save”

General Source Code Management Build Triggers Build Environment **Pre Steps** Build Post Steps Build Settings

Post-build Actions

### Pre Steps

Add pre-build step

### Build

Root POM: sample-mule-maven-deploy/pom.xml

Goals and options: clean install -U

Advanced...

### Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step

Save Apply

7. Go to Jenkins Home Page and Select the New Job Created and Configured

All	S	W	Name	Last Success	Last Failure	Last Duration
Icon: S M L			first-build-job	47 sec - #1	N/A	33 sec

Legend: Atom feed for all Atom feed for failures Atom feed for just latest builds

8. Click on “Build Now” post which it will display the Build Number

## Maven project first-build-job

Workspace

Recent Changes

### Permalinks

- Last build (#1), 1 min 28 sec ago
- Last stable build (#1), 1 min 28 sec ago
- Last successful build (#1), 1 min 28 sec ago
- Last completed build (#1), 1 min 28 sec ago

### Build History

find

#	Dec 9, 2019 12:30 AM
#1	

```
[INFO] C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\target\META-INF does not exist, skipping
[INFO] Building zip: C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\target\sample-mule-maven-deploy-1.0.0-SNAPSHOT.zip
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ sample-mule-maven-deploy ---
[INFO] No primary artifact to install, installing attached artifacts instead.
[INFO] Installing C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\pom.xml to
C:\WINDOWS\system32\config\systemprofile\.m2\repository\com\mulebytes\sample-mule-maven-deploy\1.0.0-SNAPSHOT\sample-mule-
maven-deploy-1.0.0-SNAPSHOT.pom
[INFO] Installing C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\target\sample-mule-maven-deploy-1.0.0-
SNAPSHOT.zip to C:\WINDOWS\system32\config\systemprofile\.m2\repository\com\mulebytes\sample-mule-maven-deploy\1.0.0-
SNAPSHOT\sample-mule-maven-deploy-1.0.0-SNAPSHOT.zip
[INFO]
[INFO] --- mule-app-maven-plugin:1.7:install (default-install) @ sample-mule-maven-deploy ---
[WARNING] MULE_HOME is not set, not copying sample-mule-maven-deploy-1.0.0-SNAPSHOT.zip
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 12.977 s
[INFO] Finished at: 2019-12-09T00:30:47+05:30
[INFO]
[INFO]
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\pom.xml to com.mulebytes/sample-mule-
maven-deploy/1.0.0-SNAPSHOT/sample-mule-maven-deploy-1.0.0-SNAPSHOT.pom
[JENKINS] Archiving C:\Jenkins\workspace\first-build-job\sample-mule-maven-deploy\target\sample-mule-maven-deploy-1.0.0-
SNAPSHOT.zip to com.mulebytes/sample-mule-maven-deploy/1.0.0-SNAPSHOT/sample-mule-maven-deploy-1.0.0-SNAPSHOT.zip
channel stopped
Finished: SUCCESS
```

## Create a Jenkins Deploy Job

1. Add Below Plugin in Project POM.xml

under section → **build/plugins/plugin**

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-deploy-plugin</artifactId>
  <version>2.8.2</version>
  <configuration>
    <skip>true</skip>
  </configuration>
</plugin>
```

2. Create Deployment Profiles in POM.xml (Please find below sample file content)  
Please refer **highlighted** info in below sample POM file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mulebytes</groupId>
  <artifactId>sample-mule-maven-deploy</artifactId>
  <version>1.0.0</version>
  <packaging>mule</packaging>
  <name>Mule sample-mule-maven-deploy Application</name>
  <properties>
    <mule.tools.version>1.7</mule.tools.version>
    <!-- <mule.home>${MULE_HOME}</mule.home> -->
    <mule.version>3.9.0</mule.version>
    <anypoint.username>mulebytesme</anypoint.username>
    <anypoint.password>Jaga#mule92#</anypoint.password>
    <cloudhub.env>Sandbox</cloudhub.env>
    <anypoint.uri>https://anypoint.mulesoft.com</anypoint.uri>
    <anypoint.businessGroup>MuleBytes</anypoint.businessGroup>
    <cloudhub.workerType>MICRO</cloudhub.workerType>
    <cloudhub.workers>1</cloudhub.workers>
    <maven.build.timestamp.format>yyMMddHHmmss</maven.build.timestamp.format>
  </properties>
  <build>
```

```
<plugins>

  <plugin>
    <groupId>org.mule.tools.maven</groupId>
    <artifactId>mule-app-maven-plugin</artifactId>
    <version>${mule.tools.version}</version>
    <extensions>true</extensions>
    <configuration>
      <copyToAppsDirectory>>false</copyToAppsDirectory>
    </configuration>
  </plugin>

  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-deploy-plugin</artifactId>
    <version>2.8.2</version>
    <configuration>
      <skip>true</skip>
    </configuration>
  </plugin>

  <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <version>1.7</version>
    <executions>
      <execution>
        <id>add-resource</id>
        <phase>generate-resources</phase>
        <goals>
          <goal>add-resource</goal>
        </goals>
        <configuration>
          <resources>
            <resource>
              <directory>src/main/app/</directory>
            </resource>
          </resources>
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
```



```
        <resource>
            <directory>src/main/api/</directory>
        </resource>
        <resource>
            <directory>mappings/</directory>
        </resource>
    </resources>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
<profiles>
    <profile>
        <id>standalone</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.mule.tools.maven</groupId>
                    <artifactId>mule-maven-plugin</artifactId>
                    <version>2.3.2</version>
                    <configuration>
                        <standaloneDeployment>
                            <muleVersion>${mule.version}</muleVersion>
                            <muleHome>${mule.home}</muleHome>
                            <applicationName>${artifactId}</applicationName>
                        </standaloneDeployment>
                    </configuration>
                    <executions>
                        <execution>
                            <id>deploy</id>
                            <phase>deploy</phase>
                            <goals>
```

```

        <goal>deploy</goal>
    </goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
</profile>
<profile>
    <id>cloudhub</id>
    <build>
        <plugins>
            <plugin>
                <groupId>org.mule.tools.maven</groupId>
                <artifactId>mule-maven-plugin</artifactId>
                <version>2.3.2</version>
                <configuration>
                    <cloudHubDeployment>
                        <uri>${anypoint.uri}</uri>
                        <muleVersion>${mule.version}</muleVersion>
                        <username>${anypoint.username}</username>
                        <password>${anypoint.password}</password>
                        <applicationName>${artifactId}</applicationName>
                        <environment>${cloudhub.env}</environment>
                    </cloudHubDeployment>
                </configuration>
            </plugin>
        </plugins>
        <executions>
            <execution>
                <id>deploy</id>
                <phase>deploy</phase>
                <goals>
                    <goal>deploy</goal>
                </goals>
            </execution>
        </executions>
    </build>
</profile>

```

```
        </executions>
    </plugin>
</plugins>
</build>
</profile>
</profiles>
<!-- Mule Dependencies -->
<dependencies>
    <!-- Xml configuration -->
    <dependency>
        <groupId>org.mule.modules</groupId>
        <artifactId>mule-module-spring-config</artifactId>
        <version>${mule.version}</version>
        <scope>provided</scope>
    </dependency>
    <!-- Mule Transports -->
    <dependency>
        <groupId>org.mule.transports</groupId>
        <artifactId>mule-transport-file</artifactId>
        <version>${mule.version}</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.mule.transports</groupId>
        <artifactId>mule-transport-http</artifactId>
        <version>${mule.version}</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.mule.transports</groupId>
        <artifactId>mule-transport-jdbc</artifactId>
        <version>${mule.version}</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

```
<dependency>
  <groupId>org.mule.transports</groupId>
  <artifactId>mule-transport-jms</artifactId>
  <version>${mule.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.mule.transports</groupId>
  <artifactId>mule-transport-vm</artifactId>
  <version>${mule.version}</version>
  <scope>provided</scope>
</dependency>
<!-- Mule Modules -->
<dependency>
  <groupId>org.mule.modules</groupId>
  <artifactId>mule-module-scripting</artifactId>
  <version>${mule.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.mule.modules</groupId>
  <artifactId>mule-module-xml</artifactId>
  <version>${mule.version}</version>
  <scope>provided</scope>
</dependency>
<!-- for testing -->
<dependency>
  <groupId>org.mule.tests</groupId>
  <artifactId>mule-tests-functional</artifactId>
  <version>${mule.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.mule.modules</groupId>
```

```
<artifactId>mule-module-apikit</artifactId>

<version>${mule.version}</version>

<scope>provided</scope>

</dependency>
</dependencies>

<repositories>
  <repository>
    <id>Central</id>
    <name>Central</name>
    <url>http://repo1.maven.org/maven2/</url>
    <layout>default</layout>
  </repository>
  <repository>
    <id>mulesoft-releases</id>
    <name>MuleSoft Releases Repository</name>
    <url>http://repository.mulesoft.org/releases/</url>
    <layout>default</layout>
  </repository>
</repositories>

<pluginRepositories>
  <pluginRepository>
    <id>mulesoft-release</id>
    <name>mulesoft release repository</name>
    <layout>default</layout>
    <url>http://repository.mulesoft.org/releases/</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>
</project>
```

3. Repeat the Steps mentioned in [Build Job](#) but during Pre-Steps Build change like below:

**Build**

Root POM

sample-mule-maven-deploy/pom.xml

?

Goals and options

clean package deploy -P cloudhub

?

Advanced...

4. Click on “Build Now”

Jenkins ▾ > first-deploy-job > ENABLE AUTO REFRESH

[Back to Dashboard](#)  
[Status](#)  
[Changes](#)  
[Workspace](#)  
[Build Now](#)  
[Delete Maven project](#)  
[Configure](#)  
[Modules](#)  
[Rename](#)

## Maven project first-deploy-job

[add description](#)  
[Disable Project](#)

Workspace

Recent Changes

**Permalinks**

- Last build (#1), 13 sec ago

**Build History**trend ▾

x

#1

Dec 9, 2019 1:08 AM

5. Verify if Mule Application is deployed to CloudHub → Runtime Manager

Runtime Manager

MuleBytes ? JR

SANDBOX

[Deploy application](#)

Applications

Servers

Alerts

VPCs

Load Balancers

Name ▾	Server	Status	File
sample-mule-maven-deploy	CloudHub	Started	sample-mule-maven-deploy-1.0.0-SNAPSHOT.zip