

# Lesson-End Project

## Implementing CI/CD Pipeline for Deploying Application to Tomcat Apache

**Project agenda:** To implement a continuous integration and continuous deployment (CI/CD) pipeline in Jenkins for deploying a Java application to Tomcat Apache

**Description:** You work as a DevOps Engineer in an IT firm. Your company is undertaking a project to modernize its application deployment processes, aiming to streamline the deployment of new software updates. As part of this initiative, you are tasked with setting up a continuous integration and continuous deployment (CI/CD) pipeline to automate the deployment of applications to the Tomcat Apache server hosted on an Ubuntu VM.

**Tools required:** Jenkins and Tomcat Apache

**Prerequisites:** You need to have a Jenkins up and running.

**Expected deliverables:** A step-by-step guide for setting up and implementing a CI/CD pipeline for deploying applications to Tomcat Apache on an Ubuntu VM, including integration with Jenkins for automated builds and deployments.

Steps to be followed:

1. Install Tomcat Apache 9 on Ubuntu VM
2. Log in to Jenkins CI tool and install the Deploy to Container plugin
3. Configure the deployment stage in Jenkins pipeline

### Step 1: Install Tomcat Apache 9 on Ubuntu VM

- 1.1 Open the terminal in your lab and use the following command to switch to the root user:
- sudo su**

```
sakshiguptasimp@ip-172-31-25-100:~$ sudo su
root@ip-172-31-25-100:/home/sakshiguptasimp#
```

1.2 Install Tomcat Apache and other required packages using the following command:

**apt update**

**apt install tomcat9 tomcat9-admin**

```
root@ip-172-31-25-100:/home/sakshiguptasimp# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:9 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb InRelease [1189 B]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1638 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [307 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1864 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1421 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [316 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1073 kB]
Get:16 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [31.5 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [245 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [42.7 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.2 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.2 kB]
Get:24 https://pkg.jenkins.io/debian-stable binary/ Packages [26.7 kB]
Get:25 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [246 kB]
Get:26 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb Packages [13.9 kB]
```

```
root@ip-172-31-25-100:/home/sakshiguptasimp# apt install tomcat9 tomcat9-admin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapr1 libeclipse-jdt-core-java libtcnative-1 libtomcat9-java
  tomcat9-common
Suggested packages:
  tomcat9-docs tomcat9-examples tomcat9-user
The following NEW packages will be installed:
  libapr1 libeclipse-jdt-core-java libtcnative-1 libtomcat9-java tomcat9
  tomcat9-admin tomcat9-common
0 upgraded, 7 newly installed, 0 to remove and 160 not upgraded.
Need to get 12.7 MB of archives.
After this operation, 16.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libeclipse-jdt-core-java all 3.27.0+eclipse4.21-1 [6240 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libtomcat9-java all 9.0.58-1ubuntu0.1 [6047 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9-common all 9.0.58-1ubuntu0.1 [60.9 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9 all 9.0.58-1ubuntu0.1 [37.0 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 tomcat9-admin all 9.0.58-1ubuntu0.1 [68.8 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libtcnative-1 amd64 1.2.31-1build1 [95.1 kB]
Fetched 12.7 MB in 1s (11.3 MB/s)
```

1.3 Once the installation is done, open the **tomcat-users.xml** file using the following command:

**vi /etc/tomcat9/tomcat-users.xml**

```
root@ip-172-31-25-100:/home/sakshiguptasimp# vi /etc/tomcat9/tomcat-users.xml
```

1.4 Add the following content in **tomcat-users.xml** file:

```
<user username="tomcat" password="password" roles="admin-gui,manager-  
gui,manager-script"/>
```

```
<!--  
  <role rolename="tomcat"/>  
  <role rolename="role1"/>  
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>  
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>  
  <user username="role1" password="<must-be-changed>" roles="role1"/>  
-->  
  <user username="tomcat" password="password" roles="admin-gui,manager-gui,manager-script"/>  
</tomcat-users>  
"/etc/tomcat9/tomcat-users.xml" 58L, 2850B
```

**Note:** To save the file and exit, press **Esc**, type **:wq**, and then press **Enter**

1.5 Open the **server.xml** file using the following command and scroll down to change the connector port number of Tomcat to **9090**:

```
vim /etc/tomcat9/server.xml
```

```
root@ip-172-31-25-100:/home/sakshiguptasimp# vim /etc/tomcat9/server.xml
```

```
<Service name="Catalina">  
  <!--The connectors can use a shared executor, you can define one or more named thread pools-->  
  <!--  
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"  
      maxThreads="150" minSpareThreads="4"/>  
  -->  
  
  <!-- A "Connector" represents an endpoint by which requests are received  
    and responses are returned. Documentation at :  
    Java HTTP Connector: /docs/config/http.html  
    Java AJP  Connector: /docs/config/ajp.html  
    APR (HTTP/AJP) Connector: /docs/apr.html  
    Define a non-SSL/TLS HTTP/1.1 Connector on port 8080  
  -->  
  <Connector port="9090" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443" />  
  <!-- A "Connector" using the shared thread pool-->  
  <!--  
    <Connector executor="tomcatThreadPool"  
      port="8080" protocol="HTTP/1.1"  
      connectionTimeout="20000"  
      redirectPort="8443" />  
  -->
```

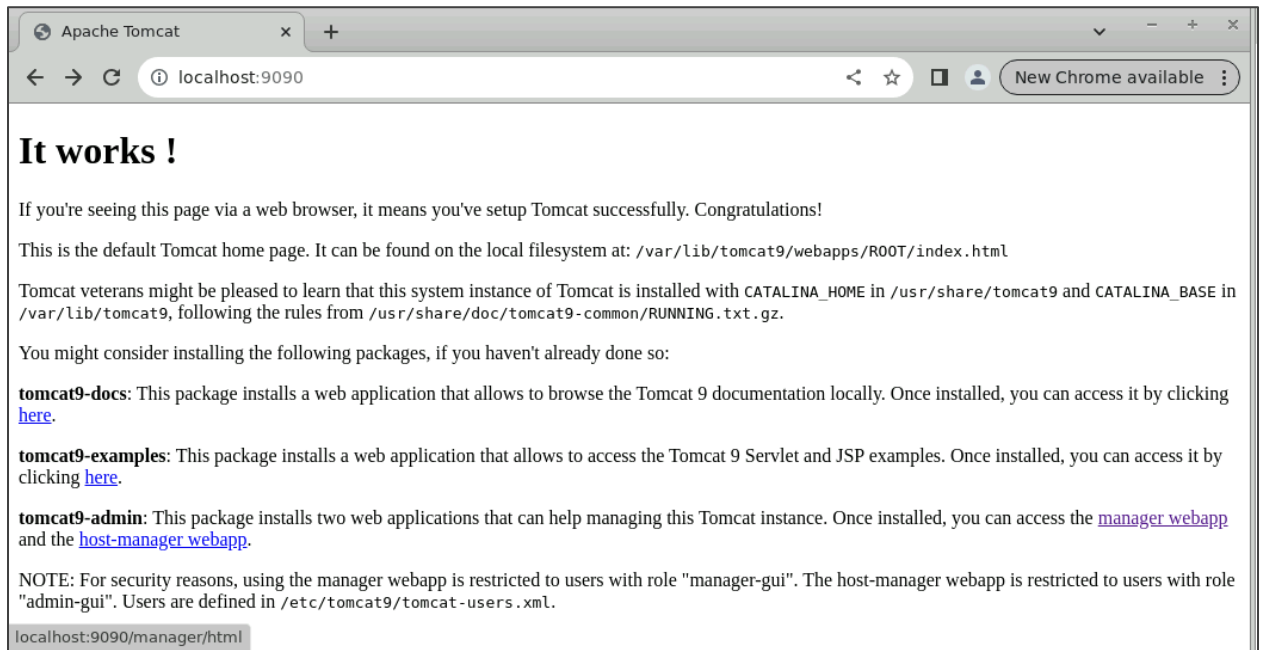
**Note:** To save the file and exit, press **Esc**, type **:wq**, and then press **Enter**

1.6 Restart Tomcat using the following command:

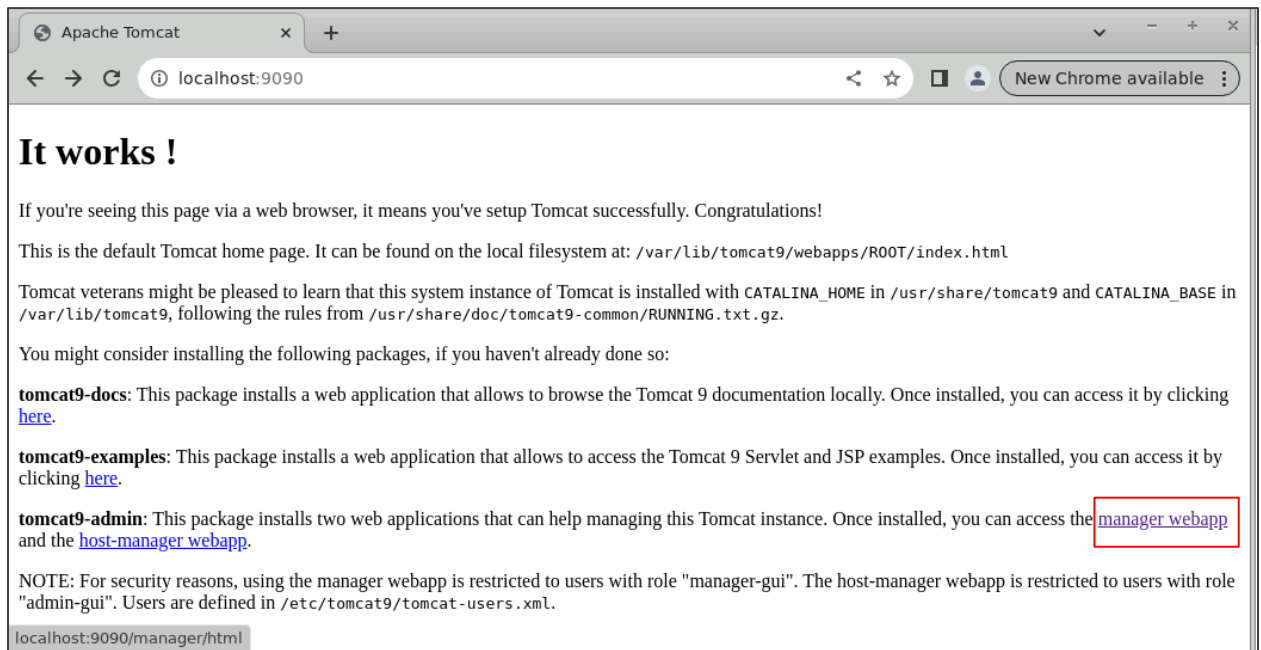
**systemctl restart tomcat9**

```
root@ip-172-31-25-100:/home/sakshiguptasimp# systemctl restart tomcat9
root@ip-172-31-25-100:/home/sakshiguptasimp# █
```

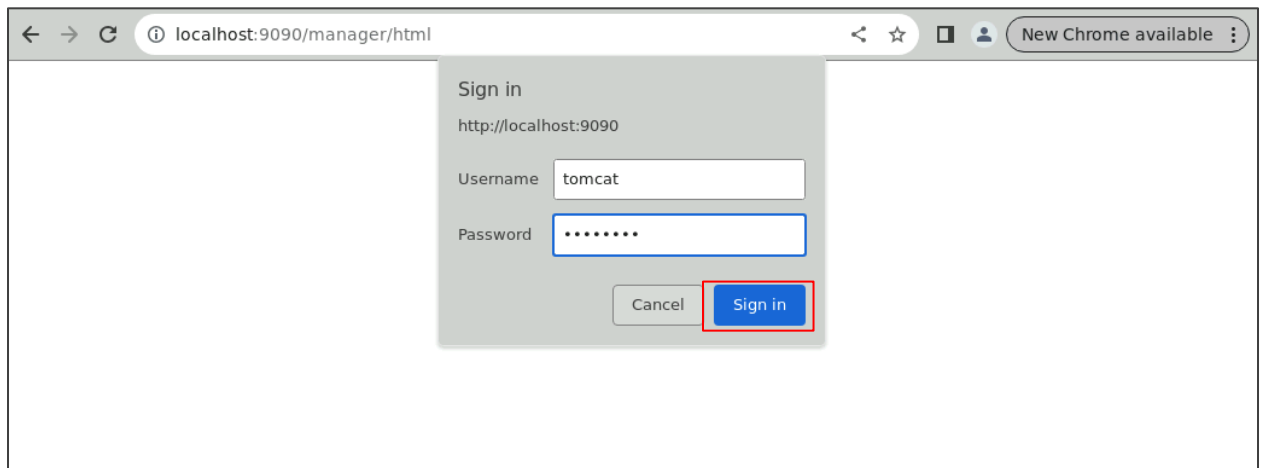
1.7 Navigate to **localhost:9090** in your web browser and access Tomcat





## 1.8 Click and access the **manager webapp** to make sure the Tomcat setup is complete



## 1.9 Enter the credentials and click on **Sign in**



**Note:** The credentials for accessing Tomcat manager web app are  
Username: **tomcat** and Password: **password**.

## Tomcat Web Application Manager

Message: OK

**Manager**

[List Applications](#)
[HTML Manager Help](#)
[Manager Help](#)
[Server Status](#)


**Applications**

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/host-manager	None specified	Tomcat Host Manager Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>

## Step 2: Log in to Jenkins CI tool and install the Deploy to Container plugin

2.1 Navigate to **localhost:8080** in your web browser, enter your credentials, and click on **Sign in**

← → ↻ localhost:8080/login?from=%2F
🔍 ☆ 📱 👤 Finish update ⋮



### Sign in to Jenkins

Username

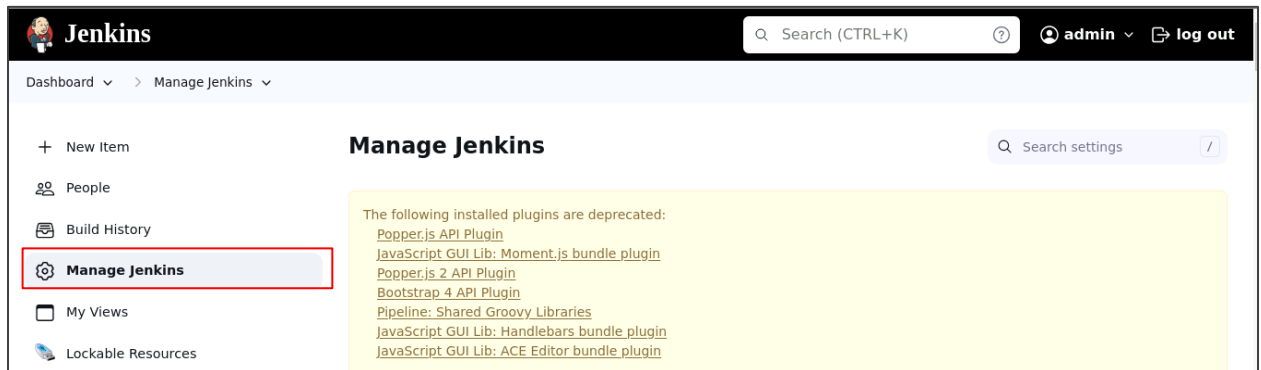
Password

☐ Keep me signed in

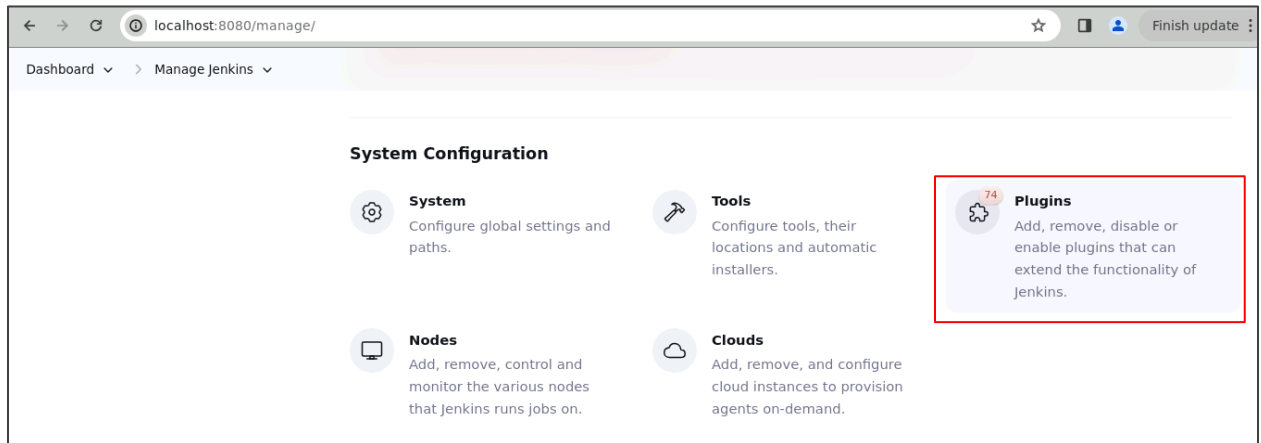
Sign in

**Note:** The credentials for accessing Jenkins in the lab are Username: **admin** and Password: **admin**.

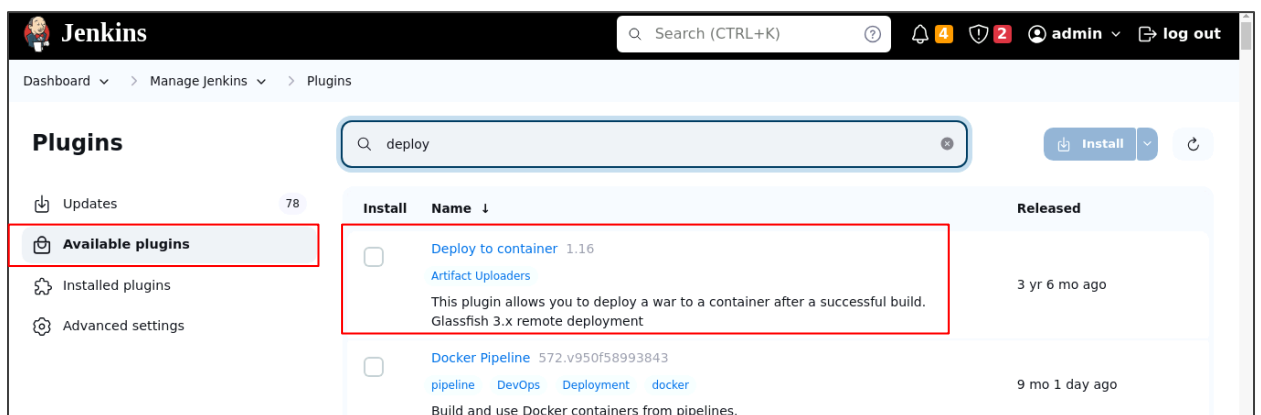
## 2.2 Click on **Manage Jenkins** on the Jenkins dashboard



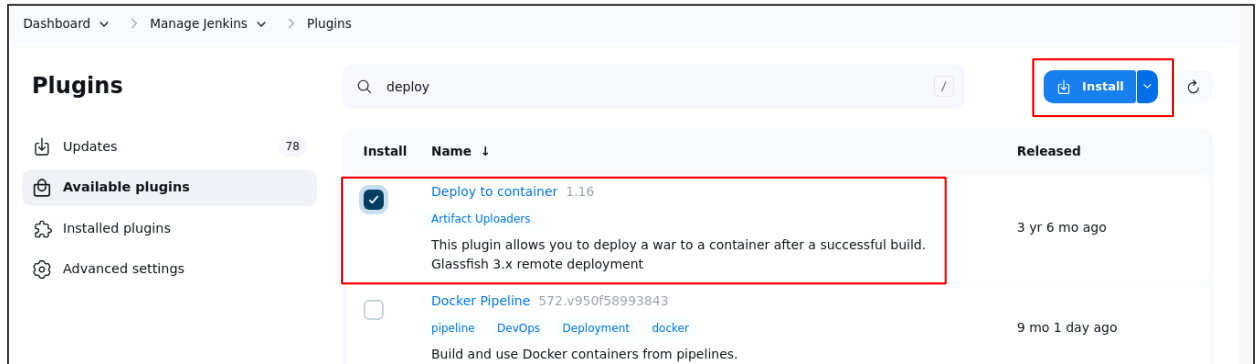
## 2.3 Scroll down and click on **Plugins** under **System Configuration**



## 2.4 Navigate to **Available plugins** and search for **Deploy to container** plugin



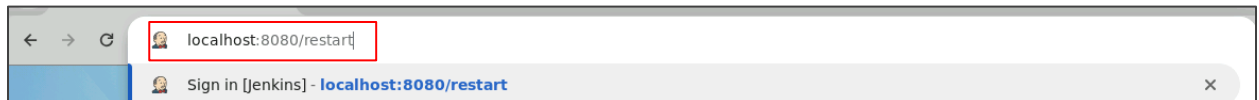
## 2.5 Select the **Deploy to container** plugin and click on **Install**



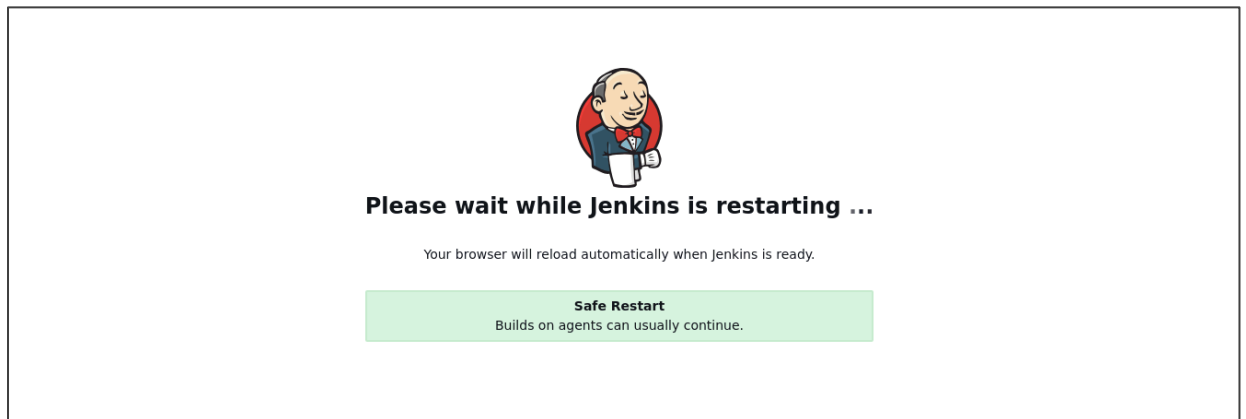
The screenshot shows the Jenkins 'Plugins' page. The breadcrumb navigation at the top reads 'Dashboard > Manage Jenkins > Plugins'. On the left sidebar, there are links for 'Updates' (78), 'Available plugins', 'Installed plugins', and 'Advanced settings'. The main area has a search bar with 'deploy' entered. In the top right corner, an 'Install' button is highlighted with a red box. Below the search bar, a table lists plugins. The first row, 'Deploy to container' version 1.16, is highlighted with a red box and has its checkbox checked. The description for this plugin states: 'This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment'. The second row, 'Docker Pipeline' version 572.v950f58993843, has its checkbox unchecked. The table has columns for 'Install', 'Name', and 'Released'.

Install	Name	Released
<input checked="" type="checkbox"/>	<b>Deploy to container</b> 1.16 <a href="#">Artifact Uploaders</a> This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	3 yr 6 mo ago
<input type="checkbox"/>	<b>Docker Pipeline</b> 572.v950f58993843 <a href="#">pipeline</a> <a href="#">DevOps</a> <a href="#">Deployment</a> <a href="#">docker</a> Build and use Docker containers from pipelines.	9 mo 1 day ago

## 2.6 After installation, navigate to the following URL to restart Jenkins: **http://localhost:8080/restart**



The screenshot shows a web browser window. The address bar contains the URL 'localhost:8080/restart', which is highlighted with a red box. Below the address bar, there is a tab labeled 'Sign in [Jenkins] - localhost:8080/restart'.

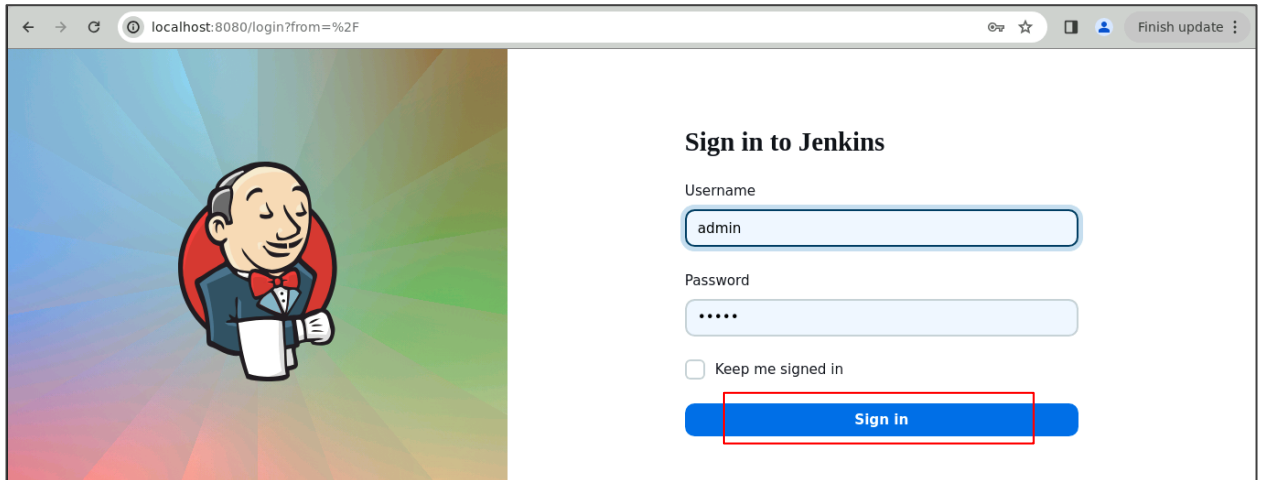


The screenshot shows the Jenkins restart page. At the top center is the Jenkins logo (a cartoon man in a suit). Below the logo, the text reads 'Please wait while Jenkins is restarting ...'. Underneath this, a smaller line of text says 'Your browser will reload automatically when Jenkins is ready.' At the bottom, there is a green button labeled 'Safe Restart' with the text 'Builds on agents can usually continue.' below it.



## Step 3: Configure the deployment stage in Jenkins pipeline

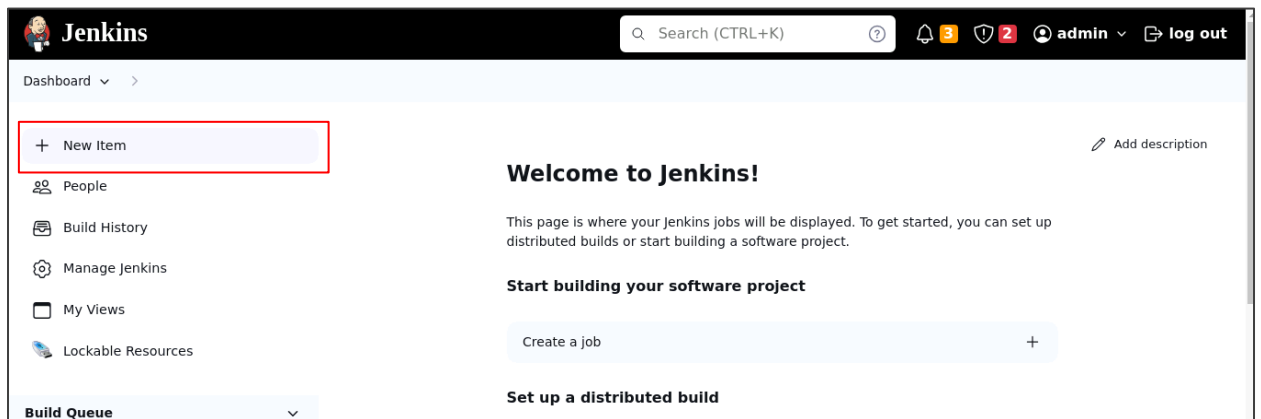
### 3.1 Enter your credentials and **Sign in** to the Jenkins CI tool



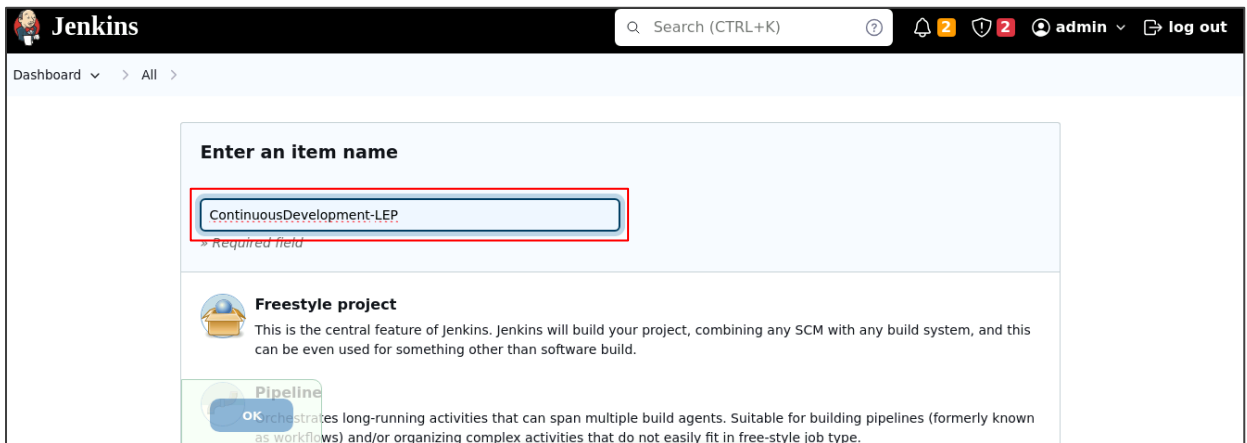
A screenshot of a web browser showing the Jenkins login page. The address bar indicates the URL is `localhost:8080/login?from=%2F`. On the left, there is a large illustration of the Jenkins mascot, a man in a tuxedo with a bow tie, holding a white cup. On the right, the heading "Sign in to Jenkins" is displayed. Below it are two input fields: "Username" with the text "admin" and "Password" with masked characters ".....". There is an unchecked checkbox labeled "Keep me signed in". At the bottom, a blue "Sign In" button is highlighted with a red rectangular box.

**Note:** The credentials for accessing Jenkins in the lab are Username: **admin** and Password: **admin**.

### 3.2 Click on **New Item** to create a new Jenkins job



### 3.3 Select **Pipeline** when creating a Jenkins job, provide a custom job name, and then click **OK** to continue



The screenshot shows the Jenkins job creation interface. At the top, the Jenkins logo and a search bar are visible. Below the breadcrumb 'Dashboard > All', there is a section titled 'Enter an item name'. A text input field contains 'ContinuousDevelopment-LEP' and is highlighted with a red border. Below this, there are three options: 'Freestyle project', 'Pipeline', and 'Folder'. The 'Pipeline' option is highlighted with a red border and has an 'OK' button next to it.

**Enter an item name**

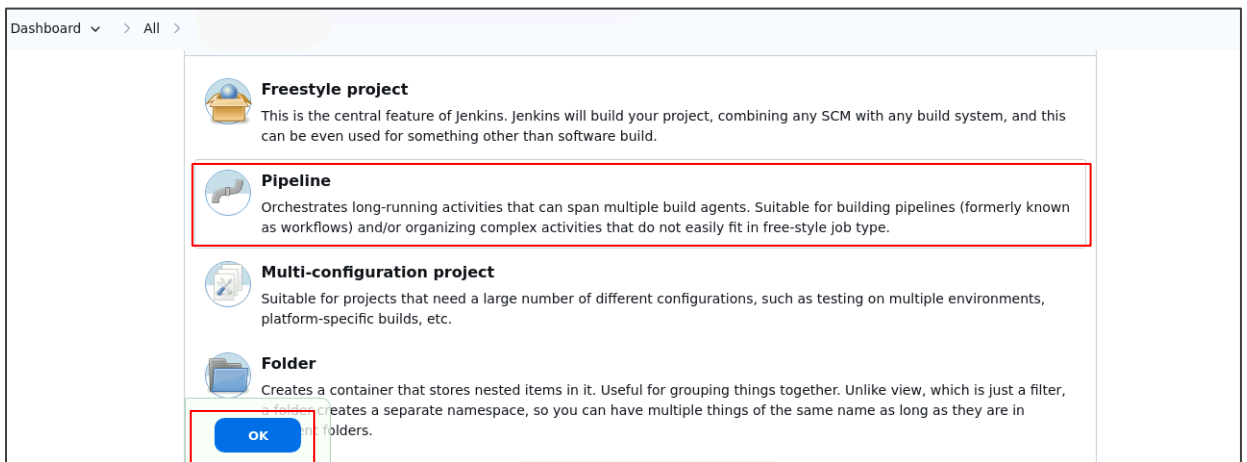
ContinuousDevelopment-LEP

\* Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**  
OK Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in folders.



The screenshot shows the Jenkins job creation interface. At the top, the Jenkins logo and a search bar are visible. Below the breadcrumb 'Dashboard > All', there is a section titled 'Enter an item name'. A text input field contains 'ContinuousDevelopment-LEP' and is highlighted with a red border. Below this, there are three options: 'Freestyle project', 'Pipeline', and 'Folder'. The 'Pipeline' option is highlighted with a red border and has an 'OK' button next to it.

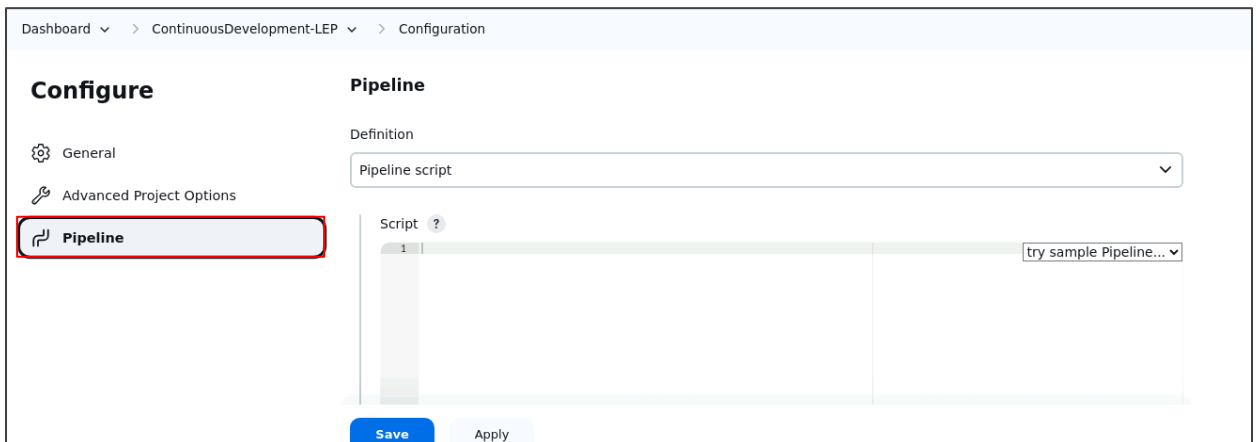
**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in folders.

### 3.4 Go to the **Pipeline** section



The screenshot shows the Jenkins Pipeline configuration screen. At the top, the breadcrumb 'Dashboard > ContinuousDevelopment-LEP > Configuration' is visible. On the left, there is a 'Configure' section with three tabs: 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is highlighted with a red border. On the right, there is a 'Pipeline' section with a 'Definition' dropdown menu set to 'Pipeline script'. Below this, there is a 'Script' section with a text area for the pipeline script. A 'try sample Pipeline...' button is visible in the top right corner of the script area. At the bottom, there are 'Save' and 'Apply' buttons.

**Configure**

General

Advanced Project Options

**Pipeline**

**Pipeline**

Definition

Pipeline script

Script

try sample Pipeline...

Save Apply

3.5 Enter the following code under **Script** and **Save** the job:

```
pipeline{

    tools{

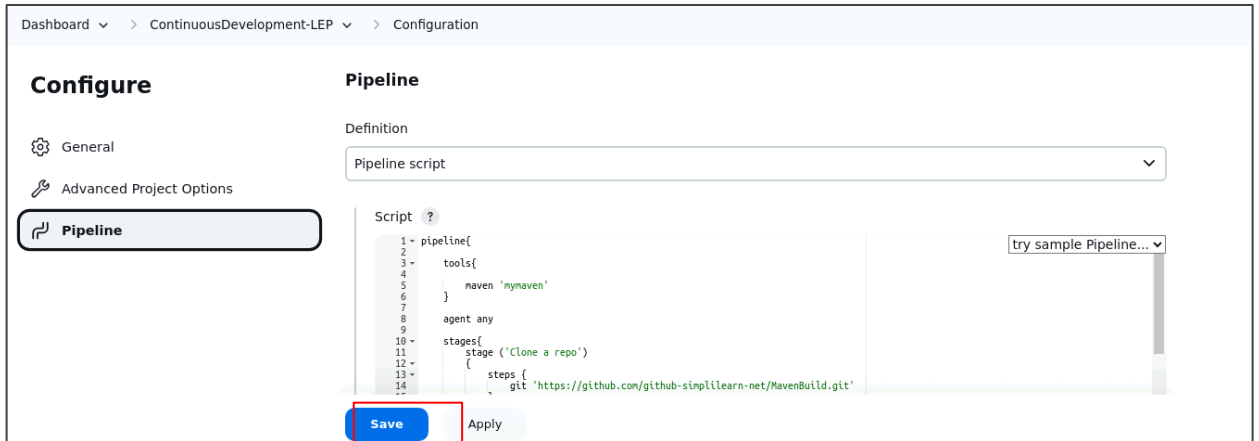
        maven 'mymaven'
    }

    agent any

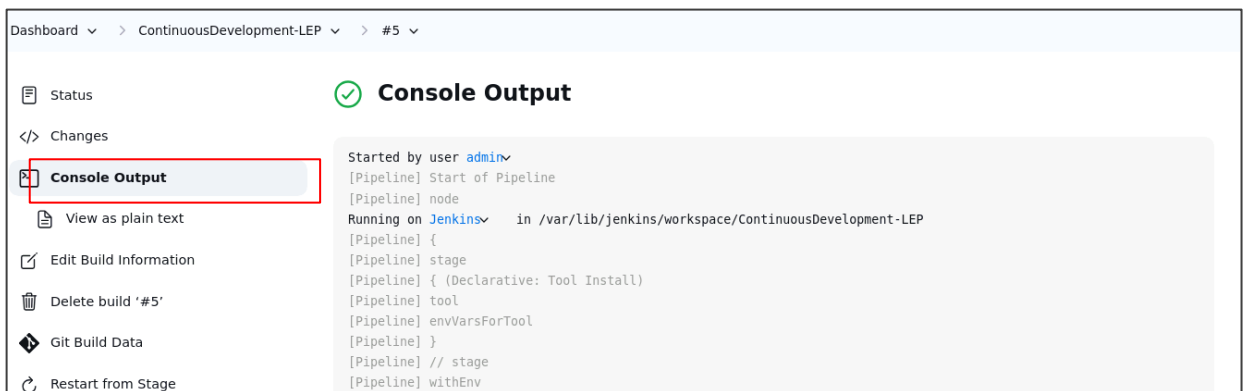
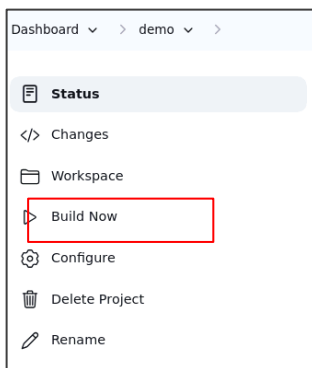
    stages{
        stage ('Clone a repo')
        {
            steps {
                git 'https://github.com/github-simplilearn-net/MavenBuild.git'
            }
        }

        stage ('Package the code')
        {
            steps{
                sh 'mvn package'
            }
        }

        stage ('Deploy the code')
        {
            steps{
                deploy adapters: [tomcat9(credentialsId: 'tomcat-id', path: '', url:
'http://localhost:9090/']], contextPath: null, war: '**/*.war'
            }
        }
    }
}
```

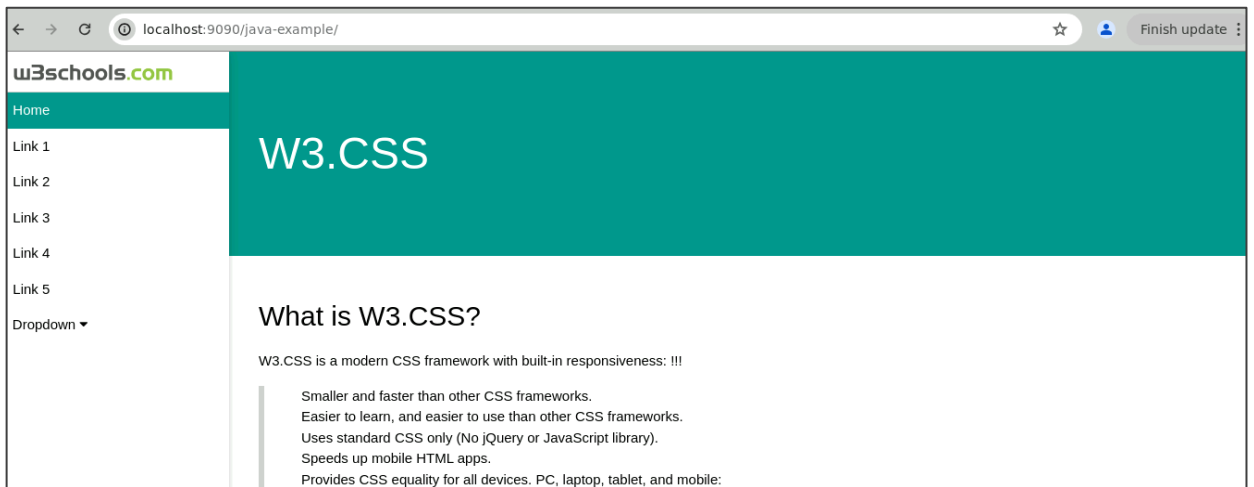


3.6 Click on **Build Now** and then select **Console Output** to check the output



```
Dashboard > ContinuousDevelopment-LEP > #5  
container Tomcat 9.x Remote with context null  
  Redeploying [/var/lib/jenkins/workspace/ContinuousDevelopment-LEP/target/java-example.war]  
  Undeploying [/var/lib/jenkins/workspace/ContinuousDevelopment-LEP/target/java-example.war]  
  Deploying [/var/lib/jenkins/workspace/ContinuousDevelopment-LEP/target/java-example.war]  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

3.7 Access the deployed application in your web browser with the following URL:  
**<http://localhost:9090/java-example/>**



By following these steps, you have successfully implemented a continuous integration and continuous deployment (CI/CD) pipeline in Jenkins for deploying a Java application to Tomcat Apache.