

Module 5: Manage Plugins on Jenkins.

Demo Document - 1

edureka!

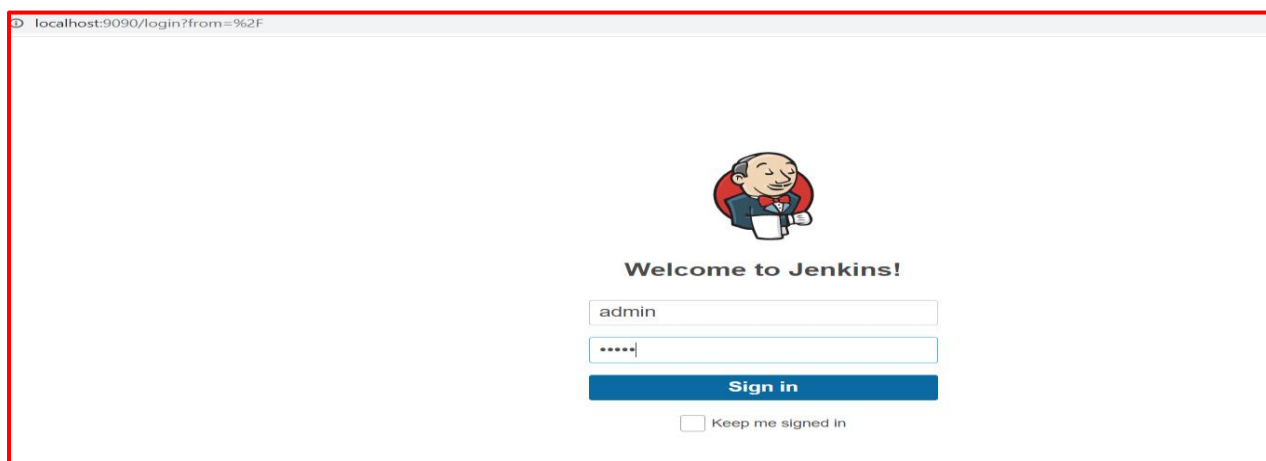
Demo: Manage Plugins on Jenkins.

Problem Statement:

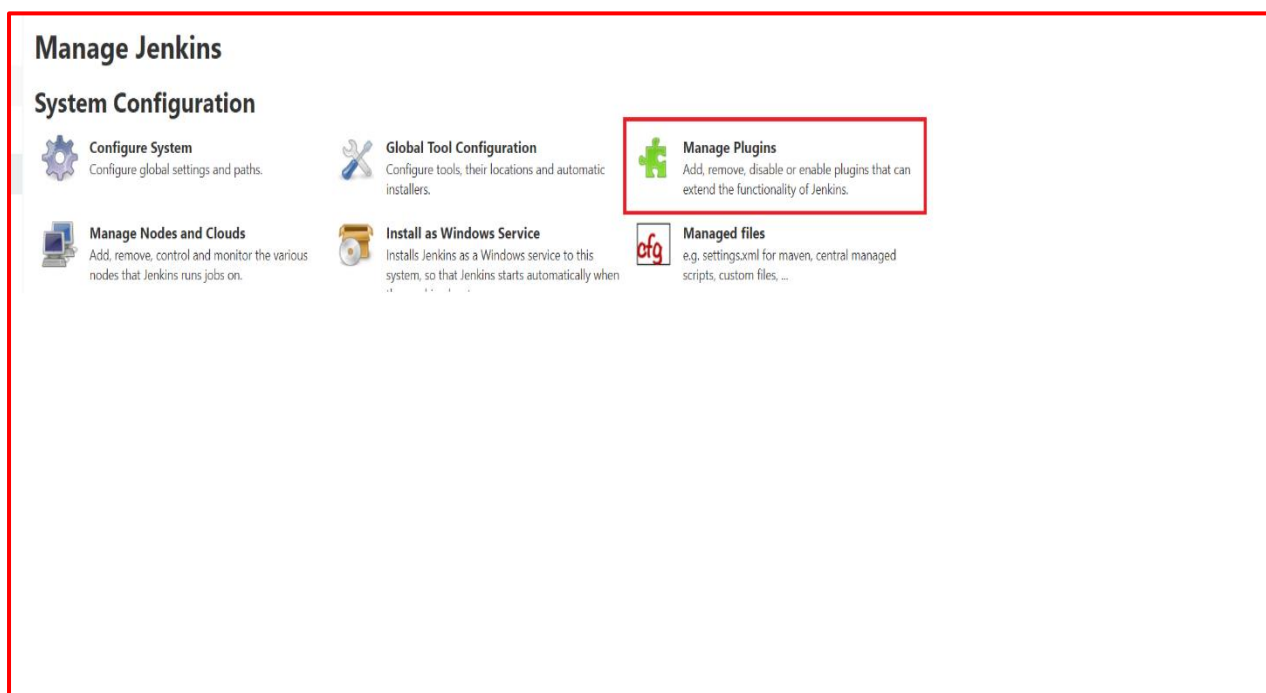
PART 1 - How to manage plugins with or without restart.

Solution Steps:

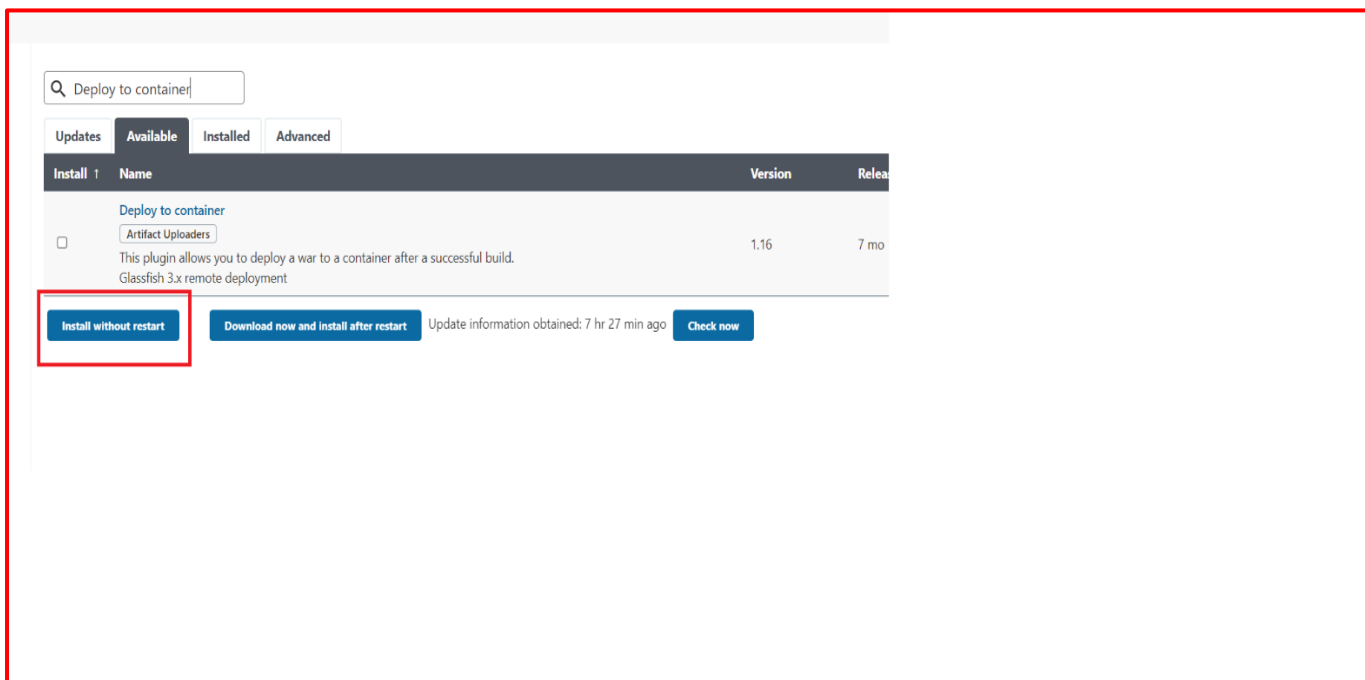
1. Login to your Jenkins console using your **'USER ID'** and **'Password'**.



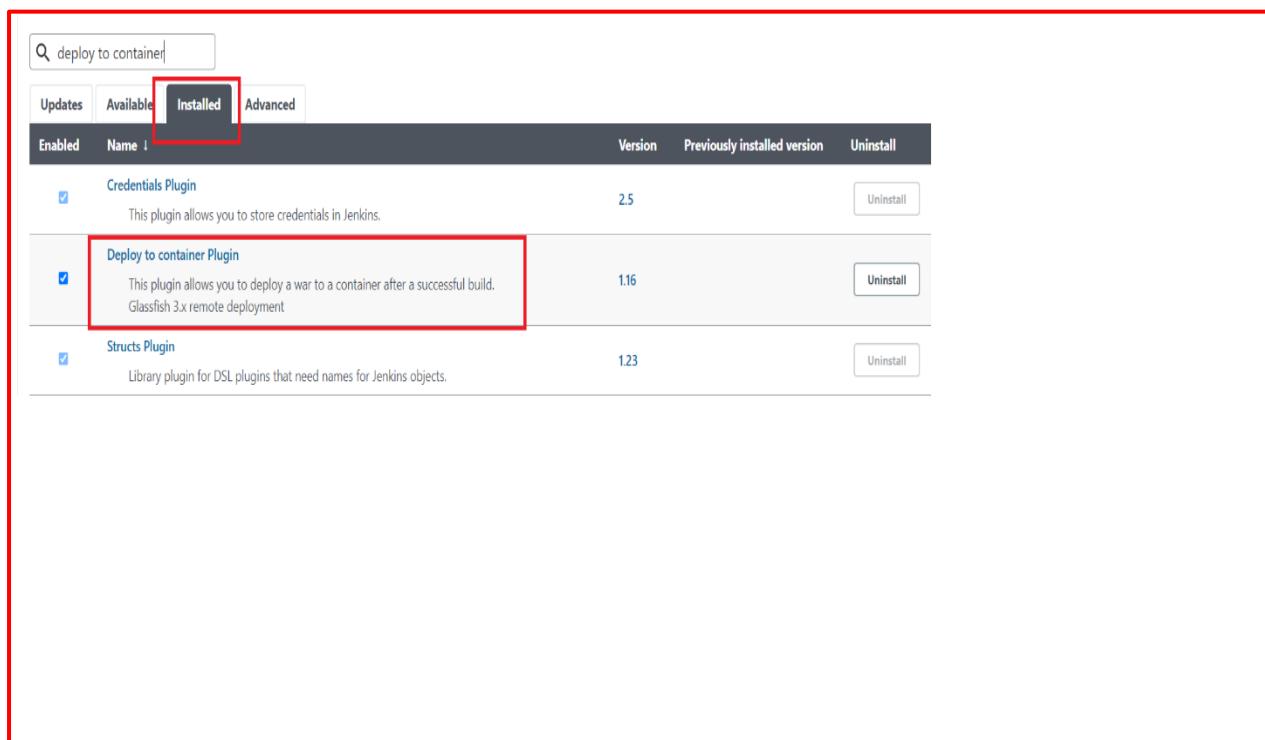
2. Now on your Jenkins Dashboard select **'Manage Jenkins'** from the left side menu. You are now landed on 'Manage Users' homepage as shown in below screenshot. Now click on **'Manage Plugins'** option.



3. Now click on **Available** option and search the plugin **Deploy to Container** and install.



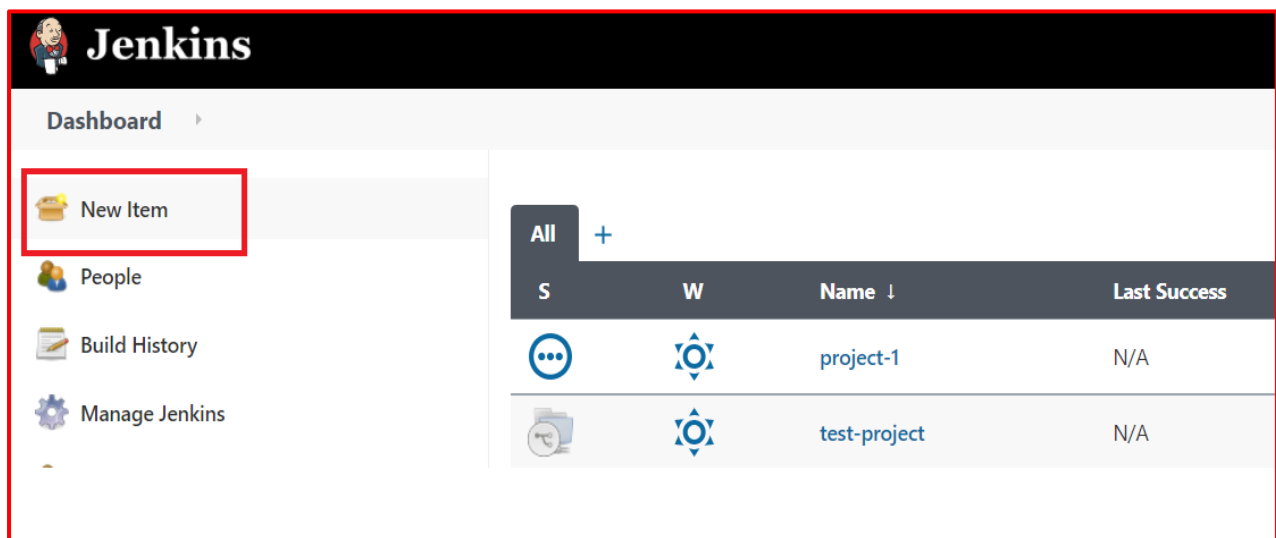
4. You can verify the plugin installed after installing under **installed** tab as shown below. You can check now the **Deploy to Container Plugin** in installed tab.



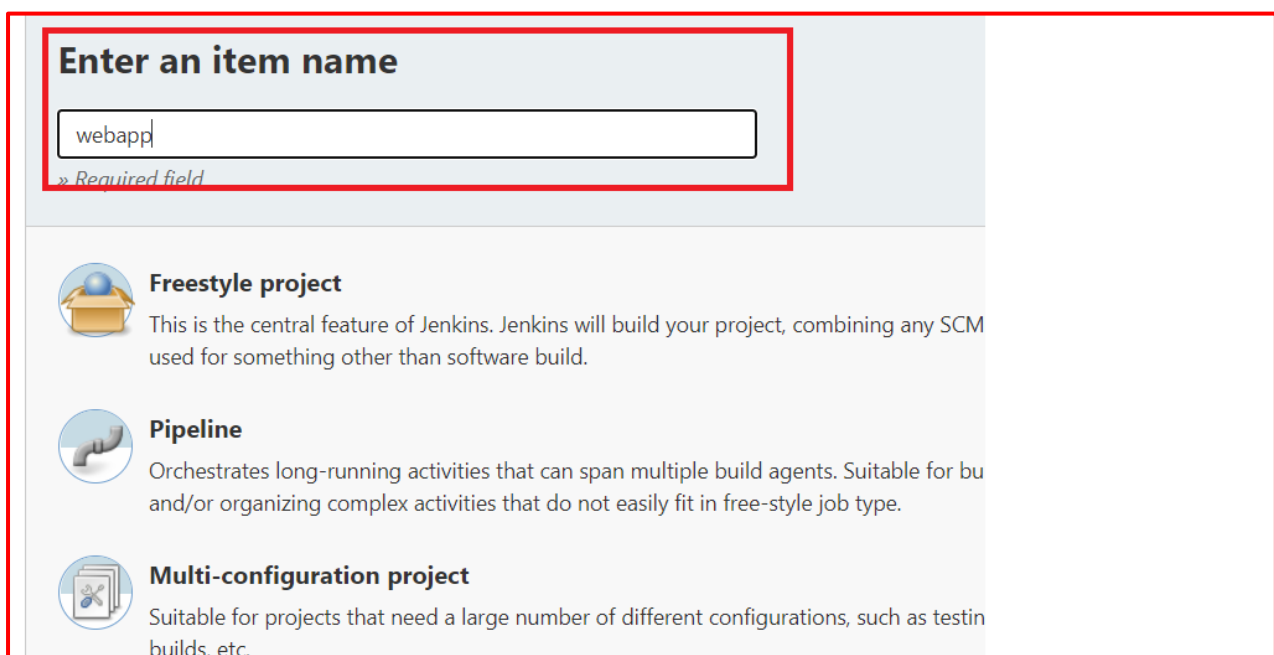
Problem Statement:

PART 2 - Create a simple Jenkins job to show Deploy to Container plugin use case.

1. Click on **New item** to create a new Jenkins job as shown below.



2. Create a sample **Freestyle** project **webapp** as shown below.



3. Click on **Source code Management** tab and select **git** and enter the **Repository URL** which has the code to build as shown.

The screenshot shows the Jenkins configuration page for a new build job. The 'General' tab is selected, and the 'Source Code Management' section is expanded. The 'Git' option is selected under 'Source Code Management'. The 'Repository URL' field is filled with 'https://github.com/kl/webAppExample.git'. The 'Credentials' dropdown is set to '- none -'.

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

[Plain text] [Preview](#)

- ☐ Discard old builds
- ☐ GitHub project
- ☐ This build requires lockable resources
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Disable this project
- ☐ Execute concurrent builds if necessary

Source Code Management

- ☐ None
- ☒ Git

Repositories

Repository URL

Credentials

[Add](#)

4. Select **BuildEnvironment** tab and click on **Add post-buit action** and select **Deploy to war/ear to container** as shown below.

The screenshot shows the Jenkins configuration page for a new build job, with the 'Build Environment' tab selected. The 'Add post-build action' button is clicked, and a dropdown menu is displayed. The 'Deploy war/ear to a container' option is selected.

General | Source Code Management | Build Triggers | Build Environment

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files

Aggregate downstream test results

Archive the artifacts

Build other projects

Publish JUnit test result report

Record fingerprints of files to track usage

Git Publisher

☒ Deploy war/ear to a container

E-mail Notification

Editable Email Notification

Set GitHub commit status (universal)

Set build status on GitHub commit [deprecated]

Delete workspace when build is done

[Add post-build action](#)

[Save](#) [Apply](#)

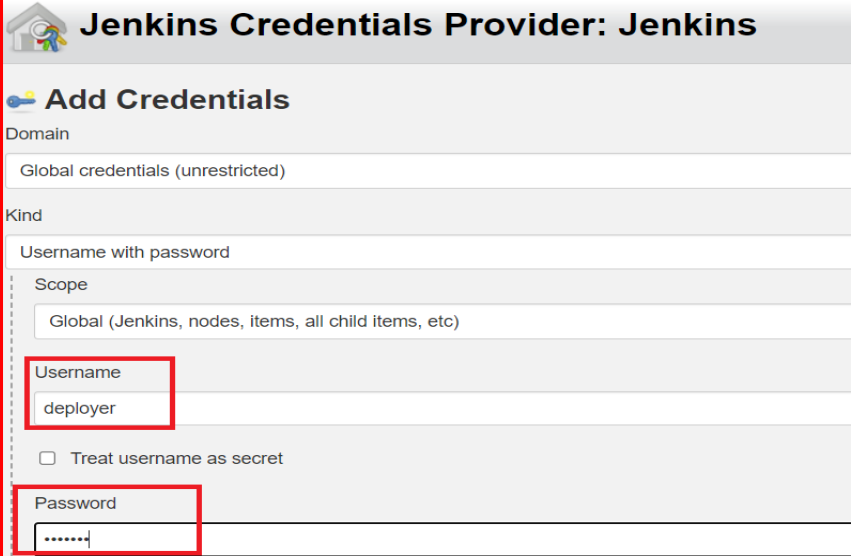
5. Select **Post-build Action** tab and enter the **WAR/EAR files**, **Context path** and **Containers** as shown below.

The screenshot shows the Jenkins 'Build' configuration page, specifically the 'Post-build Actions' tab. The 'Deploy war/ear to a container' action is selected. The 'WAR/EAR files' field is set to '**/*.war'. The 'Context path' field is set to 'webAppExample'. The 'Containers' section shows 'Tomcat 8.x Remote' with 'Credentials' set to 'deployer/*****' and 'Tomcat URL' set to 'http://localhost:9090/'. The 'Save' and 'Apply' buttons are at the bottom.

6. Create credential **deployer** for tomcat in Jenkins by clicking on **Add** in containers section as shown below.

The screenshot shows the 'Containers' section of the Jenkins configuration page, specifically the 'Tomcat 8.x Remote' configuration. The 'Add' button is highlighted, and the 'Jenkins' option is selected in the dropdown menu.

7. Enter the **username** and **password** for **deployer** username save and apply as shown below.



The screenshot shows the 'Jenkins Credentials Provider: Jenkins' interface. Under the 'Add Credentials' section, the 'Domain' is set to 'Global credentials (unrestricted)' and the 'Kind' is 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'deployer' and the 'Password' field is masked with dots. A checkbox for 'Treat username as secret' is present and unchecked. Red boxes highlight the 'Username' and 'Password' fields.

8. Build the maven project now by clicking on **Build Now** which has **Deploy to Container** plugin configured in Jenkins to build and deploy the project as shown below.

