

Session 8. Build Deployments

Ram N Sangwan



Agenda

- Understanding Deployment.
- Tomcat installation and configuration





Automate Deployments with Jenkins



- Jenkins automated deployments are one of the most popular software development techniques in use today.
- Any organization with a need to deploy applications quickly and efficiently can benefit from its core functionality.
- But with 1K+ plug-ins and counting available (including GitHub), there is a lot it can do.
- Building complex, automated development pipelines quickly requires knowledge of automation principles and the best practices for applying them to Jenkins use.



Continuous Deployment



- **Continuous Deployment** is the practice of automating the release of the changes integrated and tested during continuous delivery.





Why Do Continuous Deployment?



- **Faster Development:** For starters, automation increases the rate changes get implemented.
- In addition, the scope of each change is much smaller than when there is a long delay between releases. That minimizes the risk of mistakes.
- **Software Quality Increases:** By increasing the iteration of small changes to the codebase, problems get fixed faster.
- In addition, mistakes in development are smaller in scope, so they can be reverted or fixed faster.
- **Easier on Users:** With small frequent changes to an application, the UI remains as consistent as possible.
- Incremental improvement means the application's functionality appears to remain, even as it is continually revamped over the long-term.



Benefits of Continuous Deployment to Developer



- **Faster Development:** With less overhead and no manual interventions, features are out being used as soon as they are ready.
- With small incremental changes testing can be hyper-focused and performed very rapidly.
- **Quality Continues to Rise:** Bugs are identified faster, and the reduced volume of change means they are often easier to remediate or roll-back.



IT VOPS practice



Benefits of Continuous Deployment to Business

- **Flexibility:** Developers making small incremental changes make it easier for an organization to change direction quickly without leaving lots of half-finished work.
- **Experimentation:** Small changes to the codebase can rapidly be A/B tested by users without a delay in data gathering to see what users prefer.
- **UI/UX:** Small rapid changes are often invisible to users, the product gradually gets better, creating a better user experience.
- **Innovation:** With experimentation, the business may be surprised by user-validated results that may lead to further changes or new opportunities in the market.



Stages of Continuous Deployment





Continuous Deployment using Jenkins



- Once you have code that you want to deploy, you need to check in the code to your version control system so that it becomes part of the codebase and the entire application can be built and tested.
- There are numerous options for version control, and most will work with Jenkins.
- Some examples that you may already be using are:
 - Git
 - GitHub
 - BitBucket
 - Mercurial



Do a Manual Build/Test/Deploy Process



- Run your current process first.
- No matter how you're currently deploying software you should run through the entire process through completion as a first step.
- The goal of this is to document everything about your process, so that when it's time to automate the deployments in Jenkins you have the information you'll need.
- Without this run you'll be guessing at information or may have to call in many different people to get the information you need.



How to Set Up Jenkins



1. Install Jenkins on a machine so that you and your team can use it first.
2. Install Plugins. Jenkins has more than 1000 plugins created by the Jenkins community so that you can customize Jenkins to do exactly what you need. The specific plugins you'll need to use will depend on your process. Some of the most popular ones are:
 - **Dashboard View** creates task dashboards for easy monitoring.
 - **Folders Plugin** organizes tasks in folders for ease of use and organization.
 - **Kubernetes Plugin** a must if you're using Kubernetes for your infrastructure.
 - **Secure Jenkins** and invite other team members. Making sure that other team members are involved in creating your Continuous Deployment in Jenkins is a necessity to ensure no important process are missed, as well as to give others visibility into the process.



Create a JenkinsFile - Pipeline



- Once a build happens in Jenkins that build artifact will move through a pipeline that will result in the deployment of the application, assuming the artifacts passes automated tests.
- These pipelines are created using JenkinsFiles.
- It's a best practice to use a JenkinsFile to create your pipelines so that these files can be stored as code in source control.



JenkinsFile - Pipeline



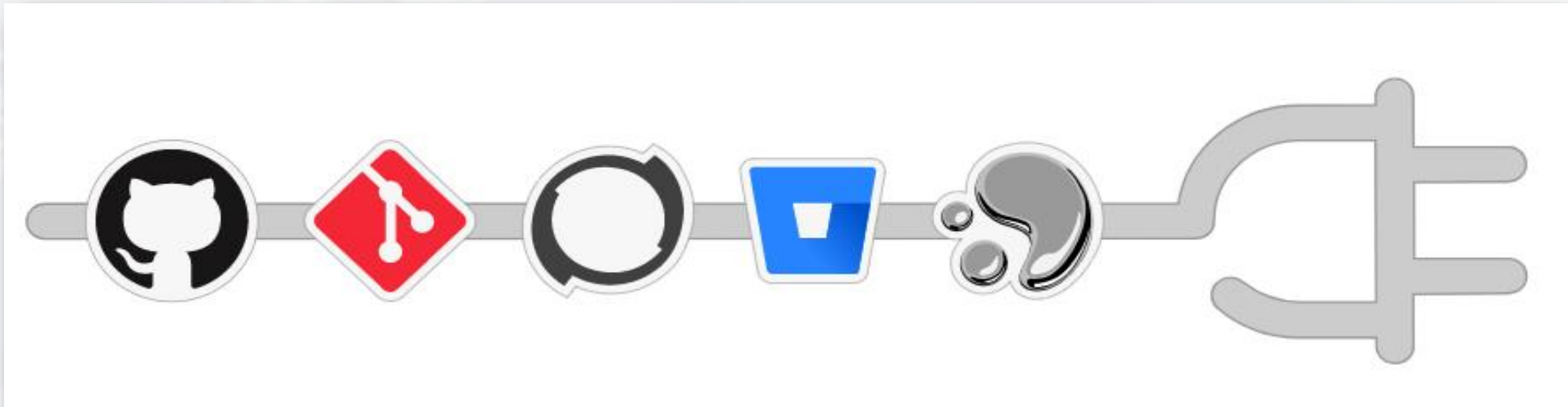
- JenkinsFiles are broken into Stages (Build, Test, Staging, etc.), and the specific stages you create will be based on the manual process that you documented.
- Each Stage in your JenkinsFile will create environments to allocate resources to use during that stage's operations.
- Secrets and credentials (user names/passwords) can be used in a JenkinsFile in a stage or environment.
- Multiple agents can be used within a stage and environment to allow access across multiple platforms.
- Each Stage will end with a true/false check to ensure that the stage was run and returned the expected results.
- Parallel executions can be created in a JenkinsFile at a stage or environment so that multiple testing setups can be run at the same time and reduce testing time needed.
- A fail at any of the parallel executions would fail the entire stage and deployment.



Connect Jenkins to Your Version Control System



- To create a build in Jenkins, you need to hook it up to your version control system.
- Once Jenkins has access to your version control, it can use that access to create builds, either on a regular basis or when triggered by a code check-in.
- All major version control systems have Jenkins plug-ins.





Install and Configure Tomcat



- Installing Tomcat on CentOS 7 requires one simple command:

yum install tomcat tomcat-admin-webapps tomcat-docs-webapp

This will install Tomcat and its dependencies, including Java and additional packages, which many users will find useful such as:

- The Tomcat Web Admin Manager (tomcat-admin-webapps)
 - The official online Tomcat documentation (tomcat-docs-webapp)
- Start Tomcat and enable Tomcat to automatically start if the server is rebooted:
systemctl start tomcat && systemctl enable tomcat



Change Tomcat Port



- Change Tomcat Port so that it does not conflict with Jenkins. Say, 8081
vi /etc/tomcat/server.xml

```
<!-- A "Connector" represents an endpoint by which requests are received
and responses are returned. Documentation at :
Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
Java AJP  Connector: /docs/config/ajp.html
APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8081" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8081" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```




Add User to Tomcat for Jenkins

- In order to use Tomcat's web management interface, you will need to create a user.
- Open the tomcat-users.xml file with the command and scroll down to below the line, which reads <tomcat-users>, and add the information for your user account.

```
# vi /usr/share/tomcat/conf/tomcat-users.xml
```

```
<tomcat-users>
```

```
<user username="tomcatmanager" password="En4EW25eI0" roles="manager-gui"/>
```

```
<user username="deployer" password="En4EW25eI0" roles="manager-script"/>
```

- Save and exit the file.

```
<!--  
<tomcat-users>  
<user username="tomcatmanager" password="En4EW25eI0" roles="manager-gui"/>  
<user username="deployer" password="En4EW25eI0" roles="manager-script"/>  
<!--  
NOTE: By default, no user is included in the "manager-gui" role required  
to operate the "/manager/html" web application. If you wish to use this app,  
you must define such a user - the username and password are arbitrary. It is  
strongly recommended that you do NOT use one of the users in the commented out
```



Purpose of the Users



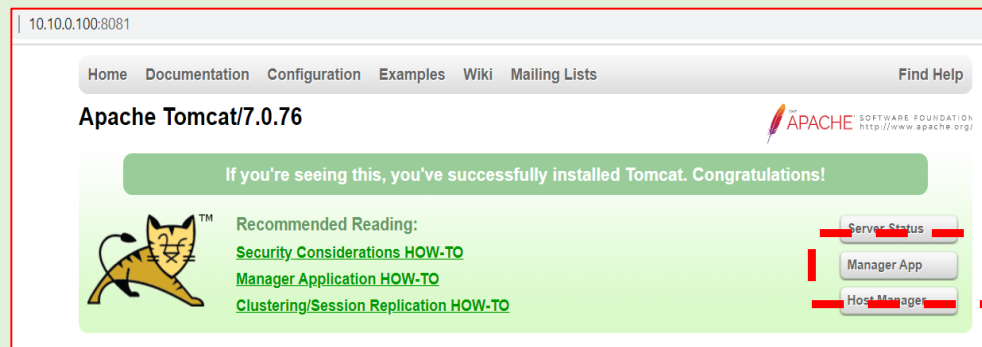
- We are creating two usernames named ***tomcatmanager*** and ***deployer***.
- The ***deployer*** account would be used to deploy the WAR file over http.
- The ***manager-gui*** based ***tomcatmanager*** user would be used to manage the manager web application at <http://10.10.0.100:8081/manager>
- Open ports in Firewall
 - # firewall-cmd --zone=public --permanent --add-port=8081/tcp*
 - # firewall-cmd --reload*
- Restart the Tomcat service for the changes to take effect:
 - # systemctl restart tomcat*



Verify Tomcat Working



- In a browser, visit the URL <http://10.10.0.100:8081> to see the Tomcat welcome page.



- Click the Manager App link. It will ask you Credentials. Enter User ID and Password as created in previous step.

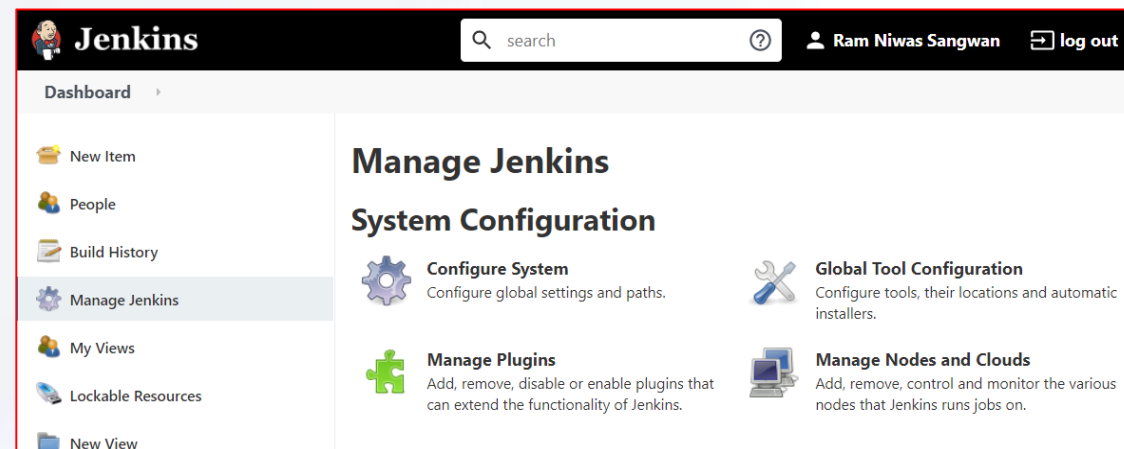


Configure Jenkins



- Now we are ready with the Tomcat Servlet Container aka Application server and it is ready to be connected from Jenkins.
- Log in to Jenkins as admin user.
- Go to **Manage Jenkins -> Global Tool Configuration** Section of Jenkins.
- Git is already installed by default.
- Maven is Manually installed in Lab 2. Check MAVEN_HOME with
`# echo $MAVEN_HOME`

```
File Edit View Search Terminal Help
[root@server ~]# echo $MAVEN_HOME
/opt/apache-maven
[root@server ~]#
```





Fill Apache Maven Details



- Change the permissions on apache-maven directory so that it is accessible by jenkins
chmod 777 /opt/apache-maven/
- Use the same as shown in the Screen and Click Apply and Save.

The screenshot shows the Jenkins configuration page for various build tools. The 'Git' section is expanded, showing fields for 'Name' (Default), 'Path to Git executable' (git), and an unchecked 'Install automatically' checkbox. There are 'Delete Git' and 'Add Git' buttons. Below this are sections for 'Gradle' and 'Ant', each with an 'Add' button and a note about the list of installations on the system. The 'Maven' section is also expanded, showing fields for 'Name' (maven3.6.3), 'MAVEN_HOME' (/opt/apache-maven), and an unchecked 'Install automatically' checkbox. There are 'Delete Maven' and 'Add Maven' buttons. At the bottom, there are 'Save' and 'Apply' buttons.

Git installations

Git

Name: Default

Path to Git executable: git

☐ Install automatically

Delete Git

Add Git

Gradle

Gradle installations

Add Gradle

List of Gradle installations on this system

Ant

Ant installations

Add Ant

List of Ant installations on this system

Maven

Maven installations

Add Maven

Maven

Name: maven3.6.3

MAVEN_HOME: /opt/apache-maven

☐ Install automatically

Delete Maven

Save Apply



Deploy to Container Plugin



- Now **Install Deploy to Container Plugin.**
- Go to **Manage Jenkins -> Manage Plugins -> Available -> Deploy to Container Plugin**

Search: Deploy to Container

Updates Available Installed Advanced

Install ↑	Name	Version	Released
<input type="checkbox"/>	Docker Pipeline Deployment DevOps docker pipeline Build and use Docker containers from pipelines. 1.25 2 mo 6 days ago		
<input checked="" type="checkbox"/>	Deploy to container Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment 1.16 2 mo 16 days ago		
<input type="checkbox"/>	Azure Container Service azure Deploy Kubernetes, DC/OS, Docker Swarm application configurations to Azure Container Service cluster. 1.0.3 2 mo 24 days ago		
<input type="checkbox"/>	OpenShift Deployer External Site/Tool Integrations openshift Other Post-Build Actions Artifact Uploaders This plugin enable Jenkins jobs to create containers(gears) on OpenShift and deploy applications to it Warning: This plugin version may not be safe to use. Please review the following security notices: • CSRF vulnerability and missing permission check • Credentials transmitted in plain text 1.2.0 5 yr 10 mo ago		

Install without restart Download now and install after restart Update information obtained: 31 min ago



Unleash Maven for Maven Jobs

- Similarly, search for "Maven plugin" and you will get search result as "Unleash Maven Plugin", enable the check-box, click on "Download now and install after restart"
- This will help us to select a Maven Job as our Choice.

This is a simple plugin to promote artifacts. The promotion is done on the repository server(s) by moving the artifact from a 'staging' repository into a 'release' repository. Currently, only Sonatype Nexus (Open Source) is supported.

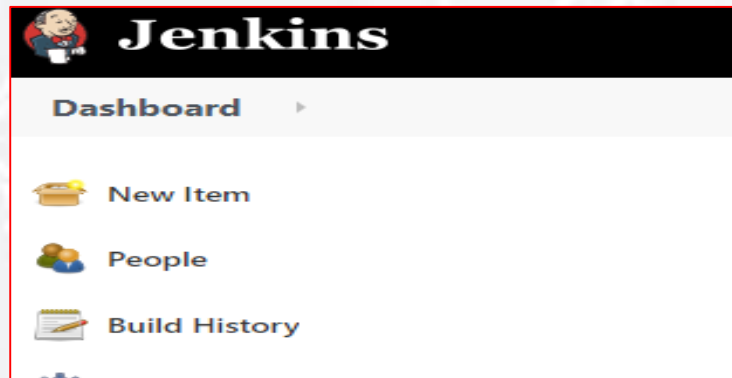
<input checked="" type="checkbox"/>	Unleash Maven Build Wrappers List view columns Maven Build Triggers Provides Maven release functionality using the Unleash Maven Plugin.
<input type="checkbox"/>	Maven SNAPSHOT Check Build Tools This plugin is used to check if pom.xml contains SNAPSHOT.



Create & Configure a Maven Job with Github



- From Menu on left select New Item - > Maven Project
- Enter Name of your Project “TomcatMaven” and Click OK



Enter an item name

» 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 e

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as work

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-sp

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder c
as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.



Fill in your GitHub Repo Details



- In the Configuration Section, Under Source Code Management Fill your Github Repository URL.
- For testing, use one of our Sample Application Named Tomcat Maven App from our Github public Repository.
- [You can use this GITHUB Repository](#)
- For private repositories, click on Add button displayed near the Credentials drop-down and enter the username and password of your SCM Repo and once it is saved, it would be available on the Dropdown for you to select.

Source Code Management

- ☐ None
- ☒ Git

Repositories

Repository URL

https://github.com/Sangwan70/tomcatmaven.git

Credentials

- none -

Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Save

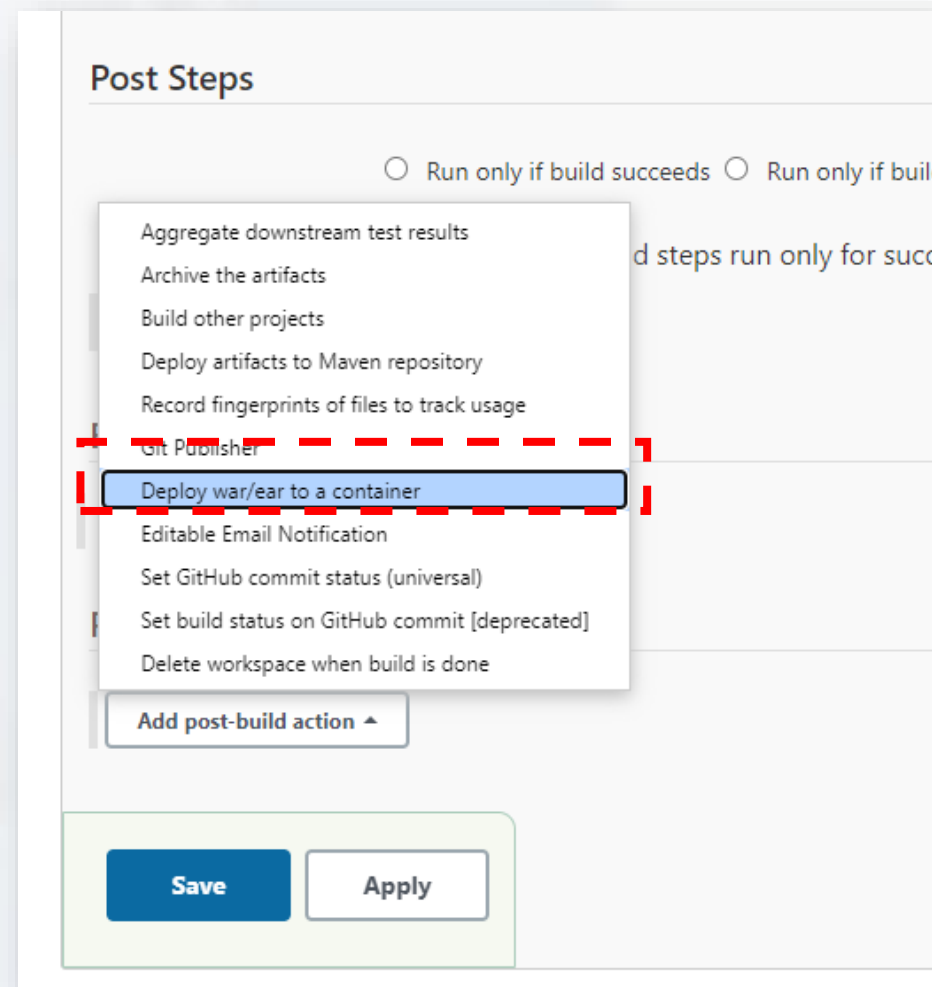
Apply



Post-build Action



- **Configure the Post-build Action and Specify the Tomcat Server Details**
 - Drag to the bottom and Go to the Post-build Actions section
 - Click on Add post-build action button
 - On the available options click on the Deploy war/ear to container





Fill the Details for WAR Deploy

- Fill the required parameters for the plugin. After entering these details, click on Add Icon in the screen shot to add tomcat Credentials.
 - WAR/EAR files: `**/*.war`
 - Context Path: `TomcatMavenApp`
 - Tomcat URL: `http://10.10.0.100:8081`

Post-build Actions

Deploy war/ear to a container

WAR/EAR files

**/*.war

Context path

TomcatMavenApp

Containers

Tomcat 7.x Remote

Credentials

- none -

Add

Tomcat URL

http://10.10.0.100:8081

Advanced...

Add Container

Deploy on failure

☐

Add post-build action

Save

Apply



Add Tomcat Credentials for Jenkins



- Enter the details as given in the screen shot.

The screenshot shows the Jenkins 'Add Credentials' form. The form is titled 'Jenkins Credentials Provider: Jenkins' and 'Add Credentials'. It contains the following fields:

- Domain:** Global credentials (unrestricted)
- Kind:** Username with password
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Username:** deployer
- Password:** (masked with dots)
- ID:** tomcat
- Description:** Apache Tomcat Credentials

At the bottom of the form are buttons for 'Add', 'Cancel', 'Save', and 'Apply'.



Use the Credentials Added

- Click on Add Button and select the credentials as given below.

Tomcat 7.x Remote

Credentials

deployer/***** (Apache Tomcat Credentials) ▼

Add ▼

Tomcat URL

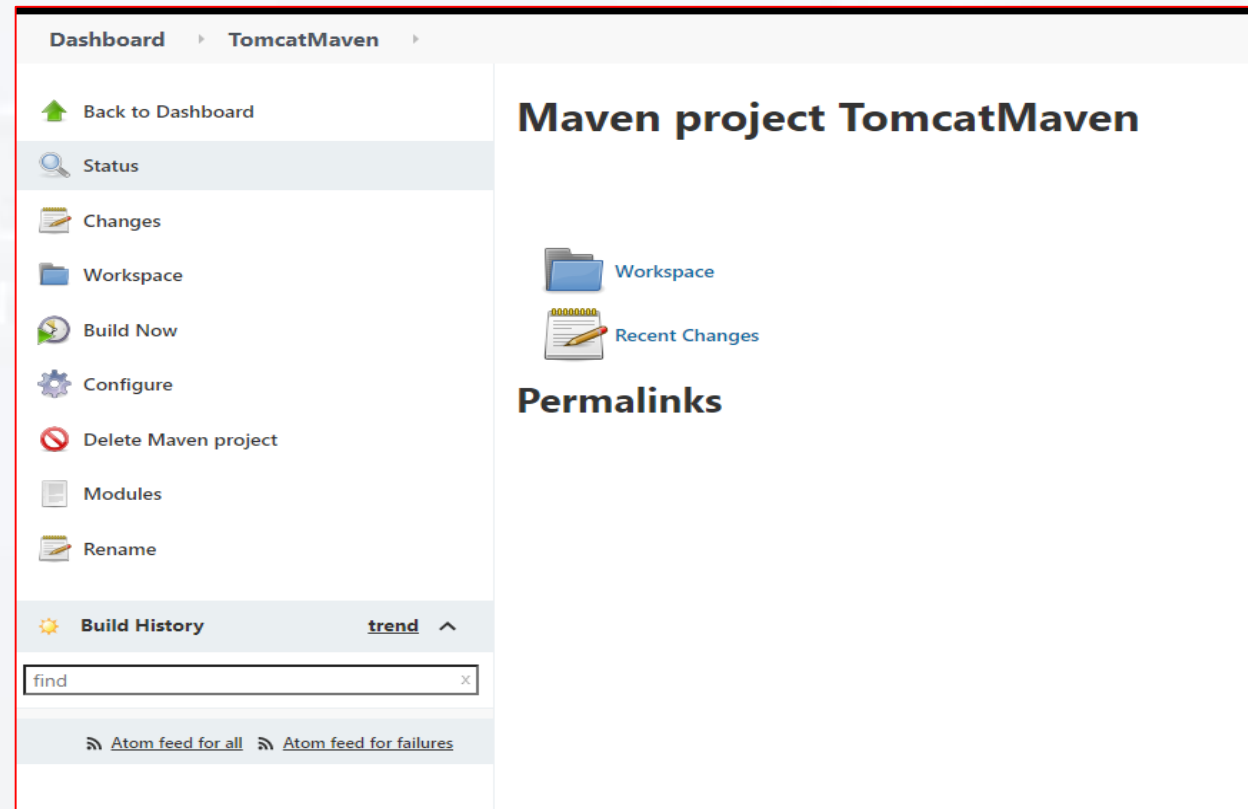
http://10.10.0.100:8081



Build the Job



- Finally Click Apply and Save.
- Execute the Job you have created by clicking on the **Build Now** button





Verify the Output in Console

- Click on Console Output after the Successful build.
- At the last line you can see that the WAR file has been generated and deployed on the remote server. in our case, `http://10.10.0.100:8081/`

```
[INFO] -----  
[INFO] Total time: 4.905 s  
[INFO] Finished at: 2021-01-18T12:10:37-05:00  
[INFO] -----  
Waiting for Jenkins to finish collecting data  
[JENKINS] Archiving /var/lib/jenkins/workspace/TomcatMaven/pom.xml to com.sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.pom  
[JENKINS] Archiving /var/lib/jenkins/workspace/TomcatMaven/target/TomcatMavenApp-2.0.war to com.sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.war  
channel stopped  
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)  
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/TomcatMaven/target/TomcatMavenApp-2.0.war to container Tomcat 7.x Remote with context TomcatMavenApp  
[ /var/lib/jenkins/workspace/TomcatMaven/target/TomcatMavenApp-2.0.war ] is not deployed. Doing a fresh deployment.  
Deploying [ /var/lib/jenkins/workspace/TomcatMaven/target/TomcatMavenApp-2.0.war ]  
Finished: SUCCESS
```




Test the Application



- As the deployment is completed and the Jenkins Job ran Successfully without issues, let us test our application. In our case, the URL should be *http://10.10.0.100:8081/TomcatMavenApp*





Thank You