# Deployment Guide: Jenkins to Tomcat

## Overview

This document provides step-by-step instructions to deploy a Java application from GitHub to a Tomcat server using Jenkins. You can attach screenshots at each step to document the process.

---

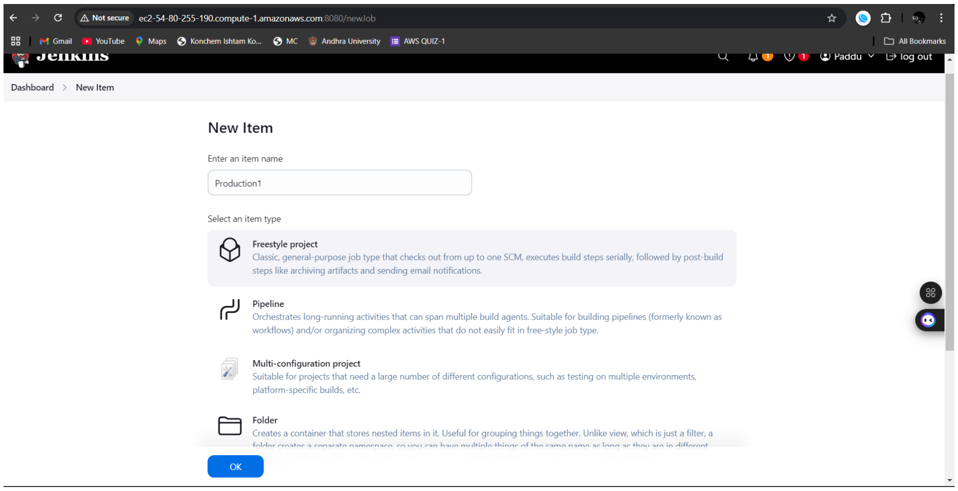
## Prerequisites

- Jenkins installed on a Linux server
- Tomcat installed on the same or a separate Linux server
- Java installed on the server
- GitHub repository containing a Java project with a `pom.xml`
- Maven installed on the Jenkins server
- Jenkins plugins:
  - Git Plugin
  - Deploy to Container Plugin

---

## Step 1: Clone GitHub Repository in Jenkins

1. **Login to Jenkins**.
2. **Create a New Job:**
   - Click on **New Item**.
   - Select **Freestyle Project** and enter a name.
   - Click **OK**.



Not secure ec2-54-80-255-190.compute-1.amazonaws.com:8080/newJob

Dashboard > New Item

New Item

Enter an item name

Production1

Select an item type

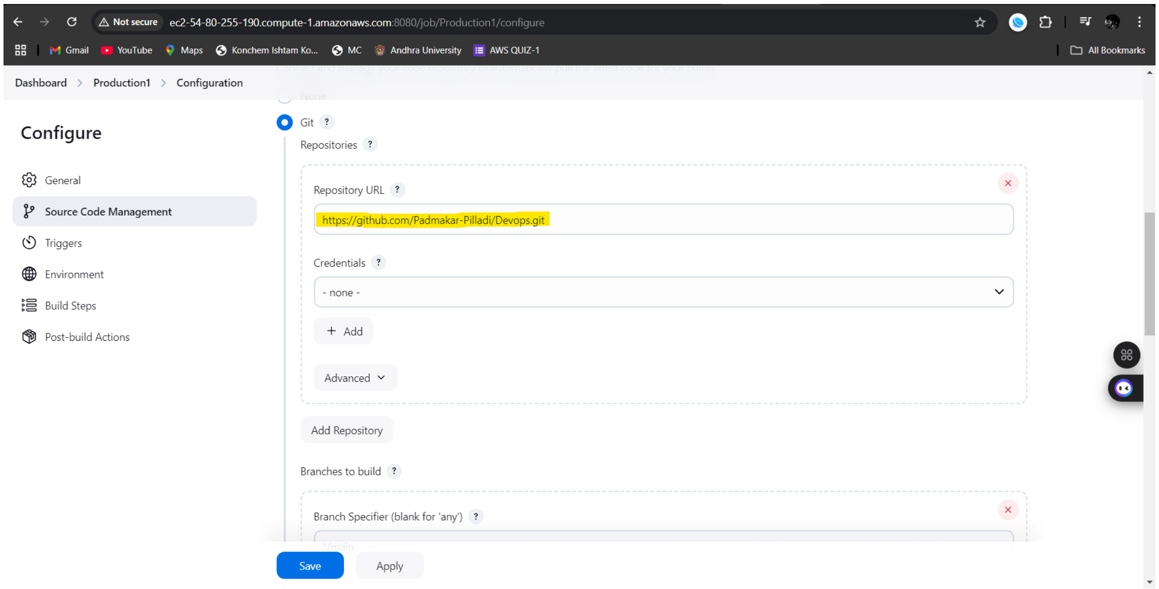
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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 different

3. **Configure GitHub Repository:**
   - Go to the **Source Code Management** section.
   - Select **Git**.
   - Enter the GitHub repository URL.
   - Provide credentials if the repository is private.
   - Click **Save**.



ec2-54-80-255-190.compute-1.amazonaws.com:8080/job/Production1/configure

Dashboard > Production1 > Configuration

● Git

Repositories

Repository URL: https://github.com/Padmakar-Pilladi/Devops.git

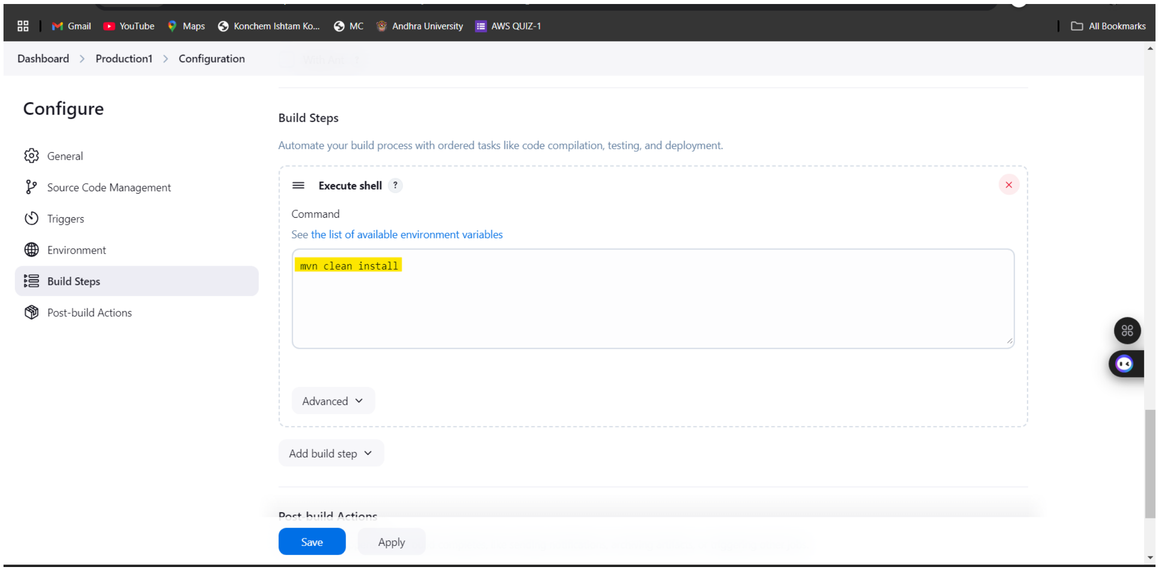
Credentials: - none -

Branches to build

Branch Specifier (blank for 'any')

# Step 2: Install Maven & Build the Artifact

1. **Ensure Maven is Installed:**
   - Run `mvn -version` to check if Maven is installed.
   - If not, install it:
   - `sudo apt update`
   - `sudo apt install maven -y`
2. **Configure Jenkins to Build Using Maven:**
   - Under **Build**, click **Add Build Step → Execute Shell**.
   - Enter the build command:
   - `mvn clean install`
   - Click **Save**.



Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Execute shell ?

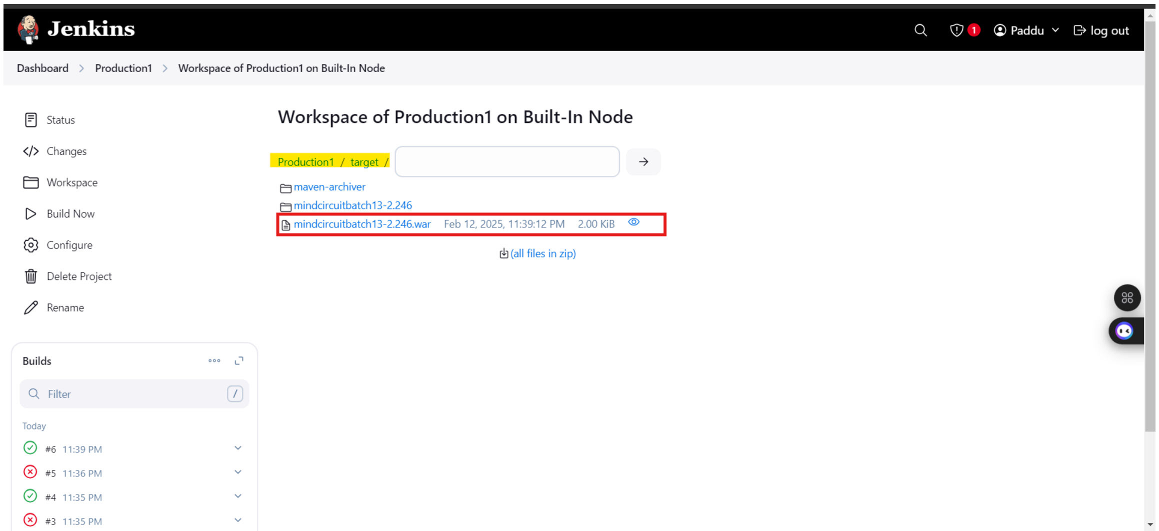
Command

See the list of available environment variables

```
mvn clean install
```

## Step 3: Verify Artifact Generation

1. After a successful build, navigate to **Workspace → target/**.
2. You should see a `.war` or `.jar` file generated.



Workspace of Production1 on Built-In Node

Production1 / target /

maven-archiver
mindcircuitbatch13-2.246
mindcircuitbatch13-2.246.war Feb 12, 2025, 11:39:12 PM 2.00 KiB

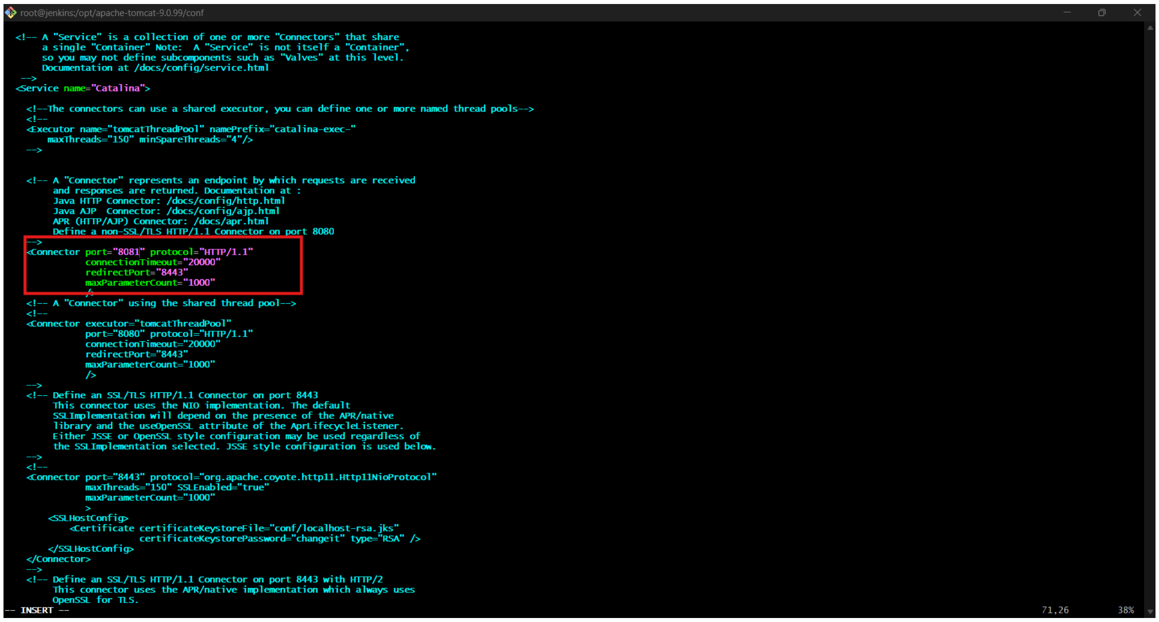
(all files in zip)

## Step 4: Install & Configure Tomcat

1. **Download Tomcat:**
2. `cd /opt`
3. `wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.99/bin/apache-tomcat-9.0.99.tar.gz`
4. `tar -xvf apache-tomcat-9.0.99.tar.gz`

## 5. Change the Default Port (Optional):

- `vim /opt/apache-tomcat-9.0.99/conf/server.xml`
    - Change `<Connector port="8080"` to `<Connector port="8081"`.



root@jenkins:/opt/apache-tomcat-9.0.99/conf

```
<!-- A "Service" is a collection of one or more "Connectors" that share
     a single "Container" Note:  A "Service" is not itself a "Container",
     so you may not define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/service.html
 -->
<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="8081" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443"
             maxParameterCount="1000"
             />
  <!-- A "Connector" using the shared thread pool-->
  <!--
  <Connector executor="tomcatThreadPool"
             port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443"
             maxParameterCount="1000"
             />
  -->
  <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
       This connector uses the NIO implementation. The default
       SSLImplementation will depend on the presence of the APR/native
       library and the useOpenSSL attribute of the AprLifecycleListener.
       Either JSSE or OpenSSL style configuration may be used regardless of
       the SSLImplementation selected. JSSE style configuration is used below.
  -->
  <!--
  <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
             maxThreads="150" SSLEnabled="true"
             maxParameterCount="1000"
             >
      <SSLHostConfig>
          <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                       certificateKeystorePassword="changeit" type="RSA" />
      </SSLHostConfig>
  </Connector>
  -->
  <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
       This connector uses the APR/native implementation which always uses
       OpenSSL for TLS.
-- INSERT --                                                              71,26          38%
```

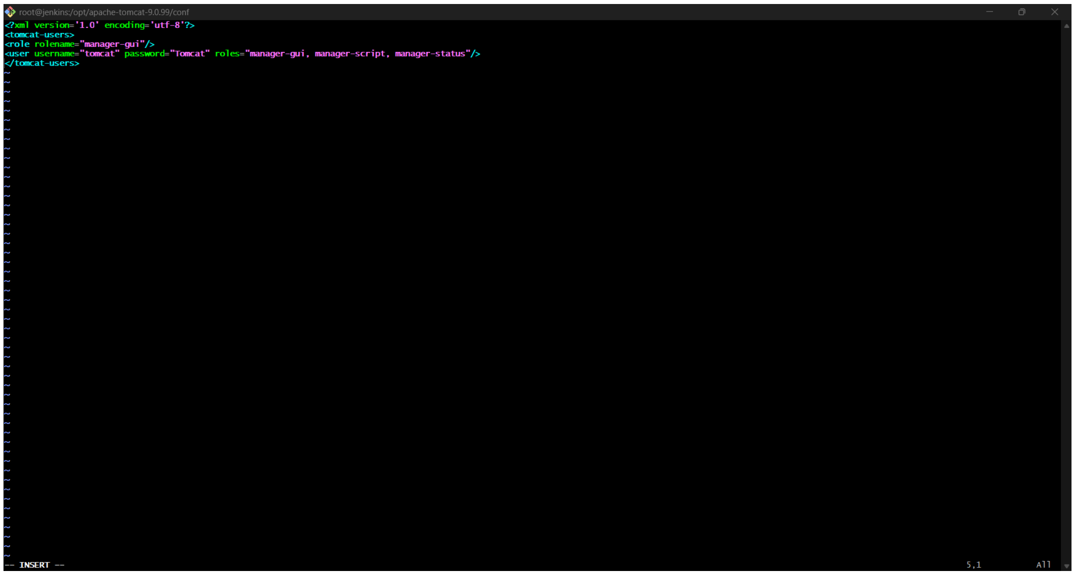
## 6. Create a Tomcat User:

- `vim /opt/apache-tomcat-9.0.99/conf/tomcat-users.xml`

Add the following:

```
<tomcat-users>
<role rolename="manager-gui"/>
<user username="tomcat" password="Tomcat" roles="manager-gui,
manager-script, manager-status"/>
```

```
</tomcat-user>
```

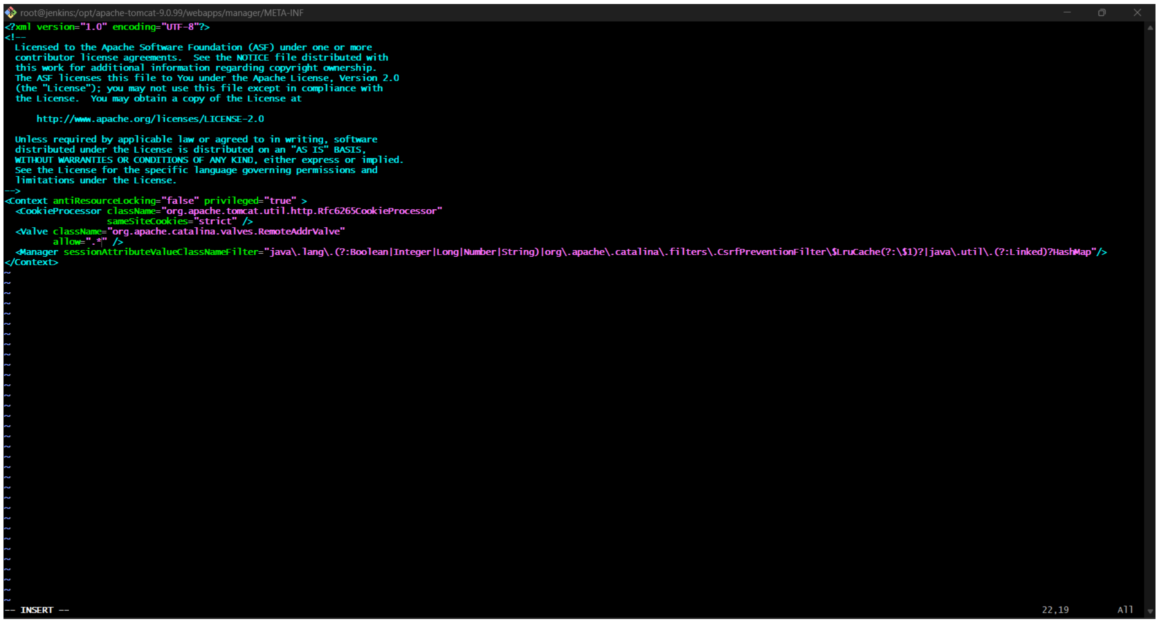


root@jenkins:/opt/apache-tomcat-9.0.99/conf

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
<role rolename="manager-gui"/>
<user username="tomcat" password="Tomcat" roles="manager-gui, manager-script, manager-status"/>
</tomcat-users>
~
-- INSERT --                                                     5,1           All
```

7. Navigate to `/opt/apache-tomcat-9.0.99/webapps/manager/META-INF`

   Need to change allowing ip to any ip by replacing ip with ".*"



root@jenkins:/opt/apache-tomcat-9.0.99/webapps/manager/META-INF

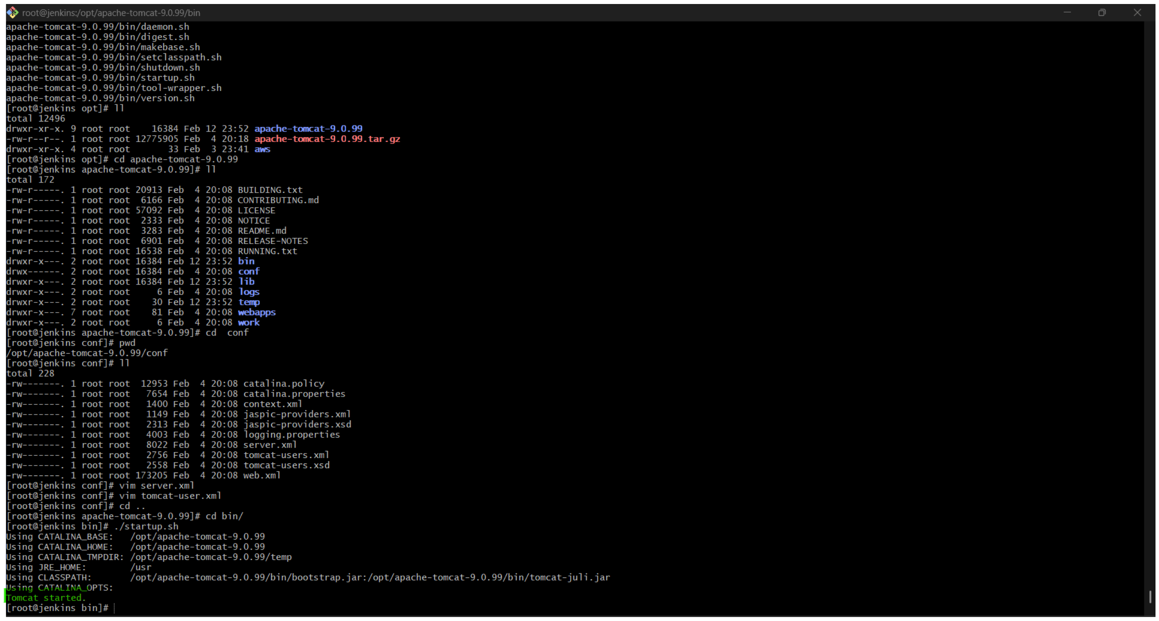
```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow=".*" />
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
~
-- INSERT --                                                     22,19         All
```

8. **Start Tomcat:**

- `cd /opt/ apache-tomcat-9.0.99/bin`
- `./startup.sh`



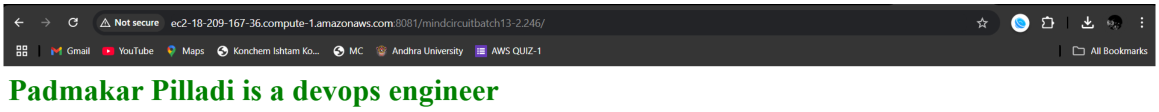
root@jenkins:/opt/apache-tomcat-9.0.99/bin

```
apache-tomcat-9.0.99/bin/daemon.sh
apache-tomcat-9.0.99/bin/digest.sh
apache-tomcat-9.0.99/bin/makebase.sh
apache-tomcat-9.0.99/bin/setclasspath.sh
apache-tomcat-9.0.99/bin/shutdown.sh
apache-tomcat-9.0.99/bin/startup.sh
apache-tomcat-9.0.99/bin/tool-wrapper.sh
apache-tomcat-9.0.99/bin/version.sh
[root@jenkins opt]# ll
total 12496
drwxr-xr-x. 9 root root    16384 Feb 12 23:52 apache-tomcat-9.0.99
-rw-r--r--. 1 root root 12775905 Feb  4 20:18 apache-tomcat-9.0.99.tar.gz
drwxr-xr-x. 4 root root       33 Feb  3 23:41 aws
[root@jenkins opt]# cd apache-tomcat-9.0.99
[root@jenkins apache-tomcat-9.0.99]# ll
total 172
-rw-r-----. 1 root root 20913 Feb  4 20:08 BUILDING.txt
-rw-r-----. 1 root root  6166 Feb  4 20:08 CONTRIBUTING.md
-rw-r-----. 1 root root 57092 Feb  4 20:08 LICENSE
-rw-r-----. 1 root root  2333 Feb  4 20:08 NOTICE
-rw-r-----. 1 root root  3283 Feb  4 20:08 README.md
-rw-r-----. 1 root root  6901 Feb  4 20:08 RELEASE-NOTES
-rw-r-----. 1 root root 16538 Feb  4 20:08 RUNNING.txt
drwxr-x---. 2 root root 16384 Feb 12 23:52 bin
drwx------. 2 root root 16384 Feb  4 20:08 conf
drwxr-x---. 2 root root 16384 Feb 12 23:52 lib
drwxr-x---. 2 root root     6 Feb  4 20:08 logs
drwxr-x---. 2 root root    30 Feb 12 23:52 temp
drwxr-x---. 7 root root    81 Feb  4 20:08 webapps
drwxr-x---. 2 root root     6 Feb  4 20:08 work
[root@jenkins apache-tomcat-9.0.99]# cd  conf
[root@jenkins conf]# pwd
/opt/apache-tomcat-9.0.99/conf
[root@jenkins conf]# ll
total 228
-rw-------. 1 root root  12953 Feb  4 20:08 catalina.policy
-rw-------. 1 root root   7654 Feb  4 20:08 catalina.properties
-rw-------. 1 root root   1400 Feb  4 20:08 context.xml
-rw-------. 1 root root   1149 Feb  4 20:08 jaspic-providers.xml
-rw-------. 1 root root   2313 Feb  4 20:08 jaspic-providers.xsd
-rw-------. 1 root root   4003 Feb  4 20:08 logging.properties
-rw-------. 1 root root   8022 Feb  4 20:08 server.xml
-rw-------. 1 root root   2756 Feb  4 20:08 tomcat-users.xml
-rw-------. 1 root root   2558 Feb  4 20:08 tomcat-users.xsd
-rw-------. 1 root root 173205 Feb  4 20:08 web.xml
[root@jenkins conf]# vim server.xml
[root@jenkins conf]# vim tomcat-user.xml
[root@jenkins conf]# cd ..
[root@jenkins apache-tomcat-9.0.99]# cd bin/
[root@jenkins bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-9.0.99
Using CATALINA_HOME:   /opt/apache-tomcat-9.0.99
Using CATALINA_TMPDIR: /opt/apache-tomcat-9.0.99/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/apache-tomcat-9.0.99/bin/bootstrap.jar:/opt/apache-tomcat-9.0.99/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[root@jenkins bin]#
```

---

# Step 5: Deploy Artifact to Tomcat Manually

1. Open Tomcat Manager: `http://<PUBLIC_IP>:8081/manager/html`
2. Login using credentials (`tomcat` / `Tomcat`).
3. Go to **WAR file to deploy** section.
4. Select the `.war` file from Jenkins workspace and deploy.



Not secure ec2-18-209-167-36.compute-1.amazonaws.com:8081/mindcircuitbatch13-2.246/

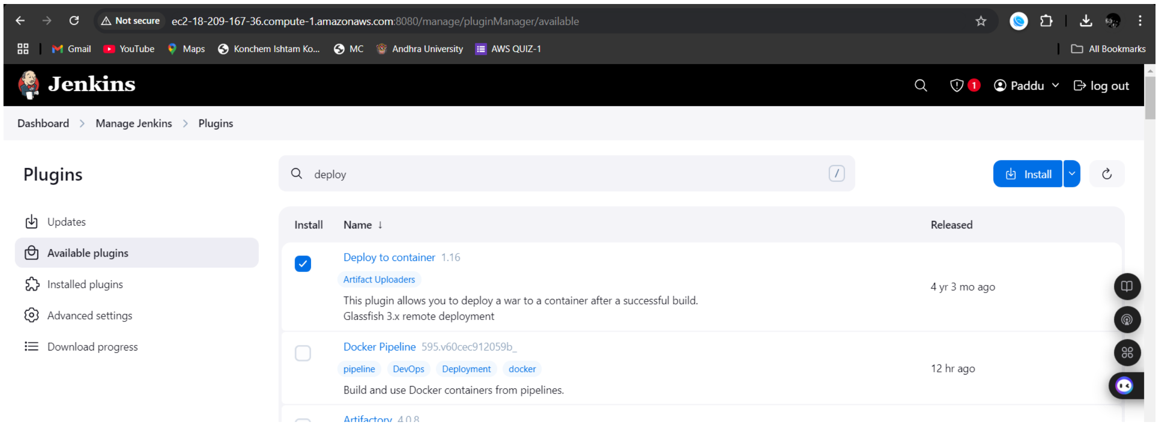
Padmakar Pilladi is a devops engineer

---

# Step 6: Automate Deployment with Jenkins

1. **Install Plugin:**

   - Go to **Manage Jenkins → Plugins**.
   - Search for **Deploy to Container** and install.



Dashboard > Manage Jenkins > Plugins

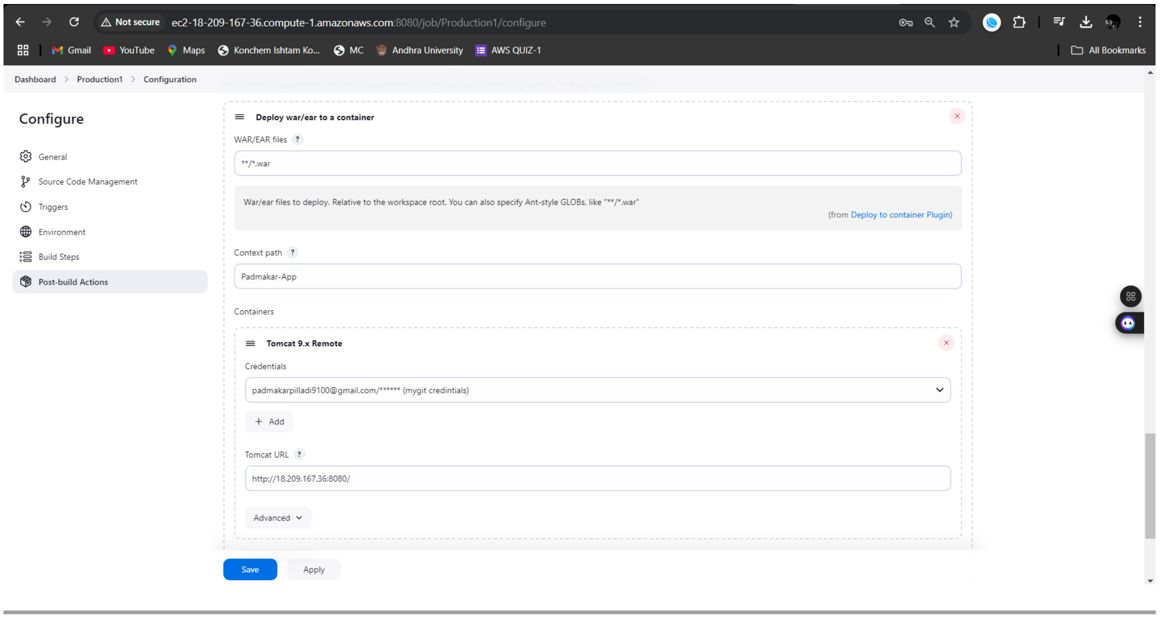
Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Search: deploy

| Install | Name ↓ | Released |
|---|---|---|
| ☑ | Deploy to container 1.16<br>Artifact Uploaders<br>This plugin allows you to deploy a war to a container after a successful build.<br>Glassfish 3.x remote deployment | 4 yr 3 mo ago |
| □ | Docker Pipeline 595.v60cec912059b_<br>pipeline DevOps Deployment docker<br>Build and use Docker containers from pipelines. | 12 hr ago |
| □ | Artifactory 4.0.8 | |

2. **Configure Post-Build Actions:**
   - Go to Jenkins Job → **Configure** → **Post Build Actions**.
   - Click **Add Post Build Action** → **Deploy WAR/EAR to a container**.
   - Enter:
     - **WAR/EAR Files:** `**/*.war`
     - **Context Path:** `MC-App`
     - **Container:** Add Tomcat credentials and URL (`http://<PUBLIC_IP>:8081`).
   - Click **Save**.



Dashboard > Production1 > Configuration

Configure

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

Deploy war/ear to a container

WAR/EAR files: **/*.war

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "**/*.war"
(from Deploy to container Plugin)

Context path: Padmakar-App

Containers

Tomcat 9.x Remote

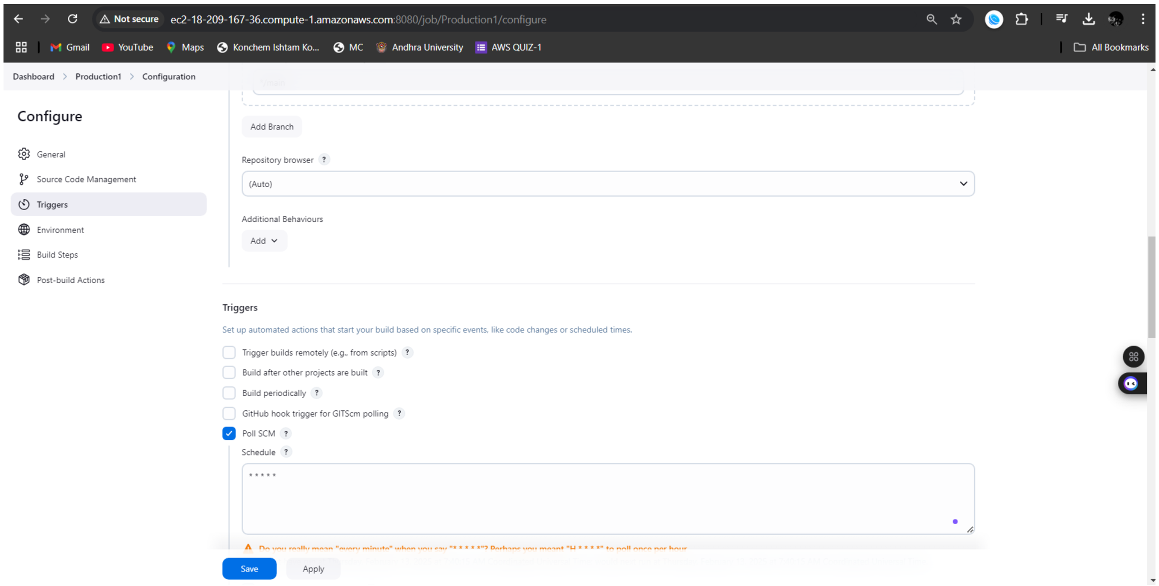
Credentials: padmakarpilladi9100@gmail.com/****** (mygit credintials)

Tomcat URL: http://18.209.167.36:8080/

# Step 7: Continuous Deployment Setup

1. **Enable Poll SCM:**

   - Go to Jenkins Job → **Configure** → **Build Triggers**.
   - Select **Poll SCM**.
   - Enter `* * * * *` (runs every minute).
   - Click **Save**.



Dashboard > Production1 > Configuration

Configure

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

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☑ Poll SCM ?

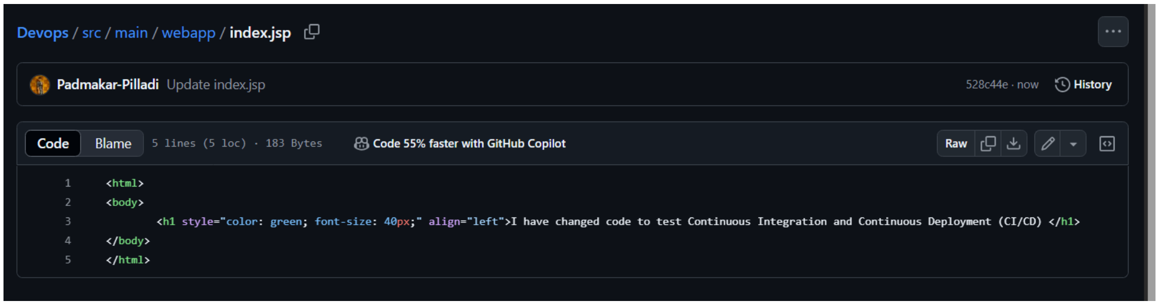
Schedule ?

```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Save Apply

2. **Modify Code in GitHub:**
   - Change any file (e.g., `index.jsp`), commit, and push.
   - Wait for Jenkins to detect the change, rebuild, and deploy.



Devops / src / main / webapp / index.jsp

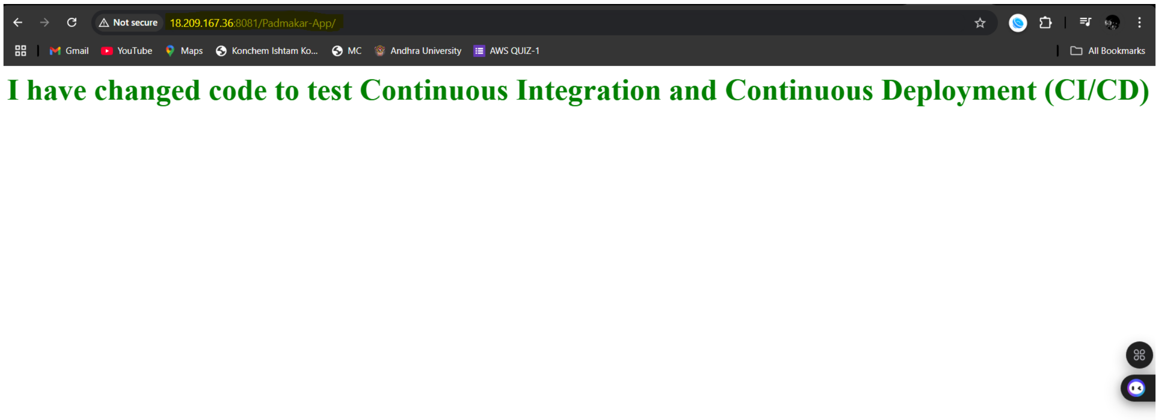
Padmakar-Pilladi Update index.jsp 528c44e · now History

Code Blame 5 lines (5 loc) · 183 Bytes

```
<html>
<body>
        <h1 style="color: green; font-size: 40px;" align="left">I have changed code to test Continuous Integration and Continuous Deployment (CI/CD) </h1>
</body>
</html>
```

## Step 8: Verify Deployment

1. Open `http://<PUBLIC_IP>:8081/MC-App` in a browser.
2. Ensure the updated application is accessible.



**I have changed code to test Continuous Integration and Continuous Deployment (CI/CD)**

---

## Conclusion

I have successfully configured Jenkins to automate deployment from GitHub to Tomcat. I can now track updates in GitHub and have Jenkins deploy the latest version automatically.

---