Build and Deployment Process Using Two Servers

This document describes the step-by-step process for setting up a Build Server and a Deployment Server for Java web application deployment using Maven and Tomcat. The Build Server handles code compilation and WAR creation, while the Deployment Server hosts the application using Apache Tomcat.

Server Setup Overview

1. Build Server: Used to build and create the artifact (.war file).

   - Install Java and Maven.

2. Deploy Server: Used to deploy and host the application using Tomcat.

   - Install Java and Tomcat.

Step-by-Step Process

1. Install Java on Build Server

```
ubuntu@ip-172-31-19-96:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre              # version 2:1.11-72build2, or
sudo apt install openjdk-11-jre-headless  # version 11.0.28+6-1ubuntu1~22.04.1
sudo apt install openjdk-17-jre-headless  # version 17.0.16+8~us1-0ubuntu1~22.04.1
sudo apt install openjdk-18-jre-headless  # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless  # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-21-jre-headless  # version 21.0.8+9~us1-0ubuntu1~22.04.1
sudo apt install openjdk-8-jre-headless   # version 8u462-ga~us1-0ubuntu2~22.04.2
ubuntu@ip-172-31-19-96:~$ sudo apt install openjdk-17-jre-headless
```

2. Install Maven on Build Server

```
ubuntu@ip-172-31-19-96:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-java libcommons-cli-java
  libcommons-io-java libcommons-lang3-java libcommons-parent-java libgeronimo-annotation-1.3-spec-java
  libgeronimo-interceptor-3.0-spec-java libguava-java libguice-java libhawtjni-runtime-java libjansi-java
  libjansi-native-java libjsr305-java libmaven-parent-java libmaven-resolver-java
  libmaven-shared-utils-java libmaven3-core-java libplexus-cipher-java libplexus-classworlds-java
  libplexus-component-annotations-java libplexus-interpolation-java libplexus-sec-dispatcher-java
  libplexus-utils2-java libsisu-inject-java libsisu-plexus-java libslf4j-java libwagon-file-java
  libwagon-http-shaded-java libwagon-provider-api-java
Suggested packages:
```

3. Clone the code repository from GitHub on Build server

```
ubuntu@ip-172-31-19-96:~$ git clone https://github.com/akracad/JavaWebCal.git
Cloning into 'JavaWebCal'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 29 (delta 3), reused 29 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (29/29), 5.78 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (3/3), done.
ubuntu@ip-172-31-19-96:~$
```

Now check, whether you have cloned the code and pom.xml file exists.

```
ubuntu@ip-172-31-19-96:~$ ls
JavaWebCal
ubuntu@ip-172-31-19-96:~$ cd JavaWebCal/
ubuntu@ip-172-31-19-96:~/JavaWebCal$ ls
pom.xml  src
ubuntu@ip-172-31-19-96:~/JavaWebCal$
```

4. Build the artifact using “mvn package” command, then you can see the output as below

```
[INFO] Packaging webapp
[INFO] Assembling webapp [webapp] in [/home/ubuntu/JavaWebCal/target/webapp]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/ubuntu/JavaWebCal/src/main/webapp]
[INFO] Building war: /home/ubuntu/JavaWebCal/target/webapp.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.821 s
[INFO] Finished at: 2025-10-07T07:01:57Z
[INFO] ------------------------------------------------------------------------
ubuntu@ip-172-31-19-96:~/JavaWebCal$
```

5. Install Java on Deploy server using command “sudo apt install openjdk-17-jre-headless” (Same way we have installed java on Build server)

6. Install Tomcat on the Deploy Server
   - Go to Apache tomcat official website
   - Copy the tar extension file url link
   - Download it using wget command followed by the url link
   - After downloading tar file, extract the tar file using "tar -xvf apache-tomcat-9.0.110.tar.gz" command

```
ubuntu@ip-172-31-27-217:~$ wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.110/bin/apache-tomcat-9.0.110.tar.gz
--2025-10-07 07:13:13--  https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.110/bin/apache-tomcat-9.0.110.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13036068 (12M) [application/x-gzip]
Saving to: 'apache-tomcat-9.0.110.tar.gz'

apache-tomcat-9.0.110.tar. 100%[=====================================>]  12.43M  --.-KB/s    in 0.03s

2025-10-07 07:13:14 (357 MB/s) - 'apache-tomcat-9.0.110.tar.gz' saved [13036068/13036068]

ubuntu@ip-172-31-27-217:~$ ls
apache-tomcat-9.0.110.tar.gz
ubuntu@ip-172-31-27-217:~$ tar -xvf apache-tomcat-9.0.110.tar.gz
```

7. Configure "webapps/host-manager/META-INF/context.xml" on the Deploy Server
   - After opening the context.xml file, locate the following lines

   <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" and comment it out by enclosing it within <!-- and --> tags.

```
<!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

8. Configure webapps/manager/META-INF/context.xml on the Deploy Server.
   This step is the same as the previous one, but here you need to modify the context.xml file located in webapps/manager/META-INF/.

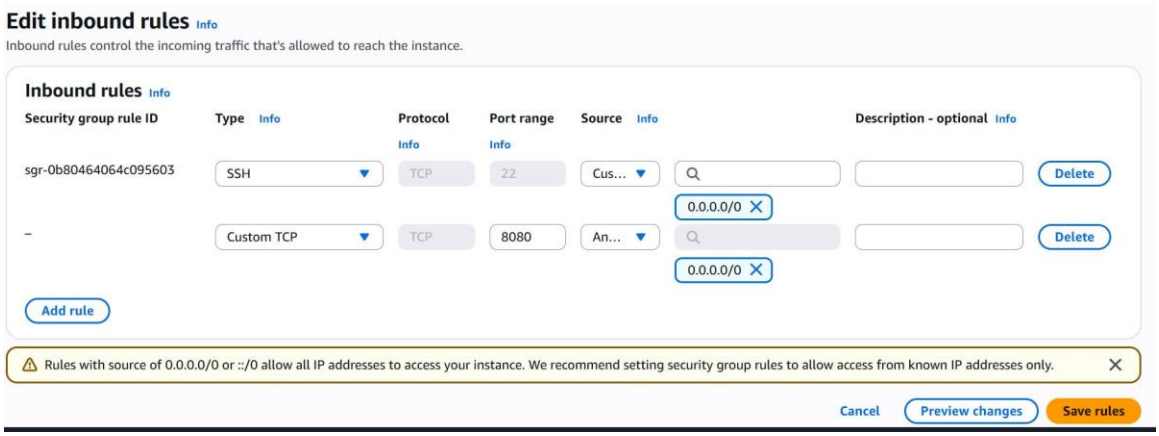
9. Configure conf/tomcat-users.xml on the Deploy Server with appropriate roles and users
   - After opening file, locate for roles lines which are pre-defined and commented, those lines should be uncommented and passwords should be changed.

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

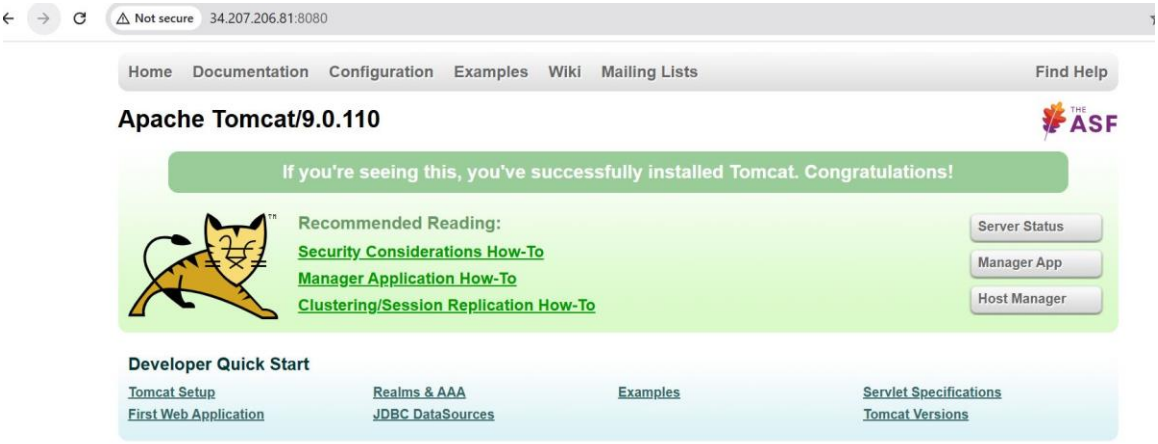
10. Start the Tomcat service on the Deploy Server by changing to “bin” directory

```
ubuntu@ip-172-31-27-217:~/apache-tomcat-9.0.110/conf$ cd ../bin/
ubuntu@ip-172-31-27-217:~/apache-tomcat-9.0.110/bin$ ls
bootstrap.jar       commons-daemon-native.tar.gz  digest.sh         shutdown.sh          tool-wrapper.sh
catalina-tasks.xml  commons-daemon.jar            makebase.bat      startup.bat          version.bat
catalina.bat        configtest.bat                makebase.sh       startup.sh           version.sh
catalina.sh         configtest.sh                 setclasspath.bat  tomcat-juli.jar
ciphers.bat         daemon.sh                     setclasspath.sh   tomcat-native.tar.gz
ciphers.sh          digest.bat                    shutdown.bat      tool-wrapper.bat
ubuntu@ip-172-31-27-217:~/apache-tomcat-9.0.110/bin$ ./startup.sh
Using CATALINA_BASE:   /home/ubuntu/apache-tomcat-9.0.110
Using CATALINA_HOME:   /home/ubuntu/apache-tomcat-9.0.110
Using CATALINA_TMPDIR: /home/ubuntu/apache-tomcat-9.0.110/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ubuntu/apache-tomcat-9.0.110/bin/bootstrap.jar:/home/ubuntu/apache-tomcat-9.0.1
10/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

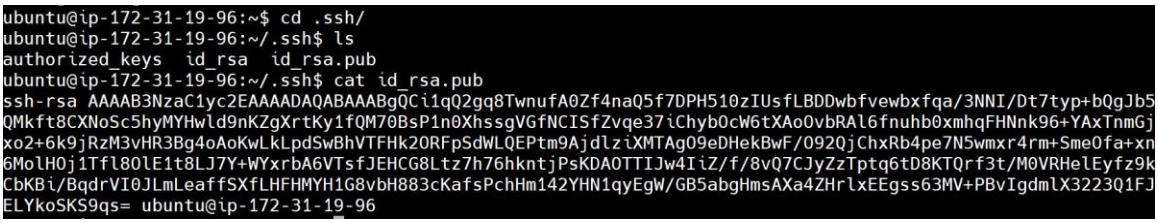
11. Allow inbound rules for port 8080 on the Deploy Server in AWS security groups



12. Access the Tomcat web interface on the browser using the public IP of the Deploy Server.

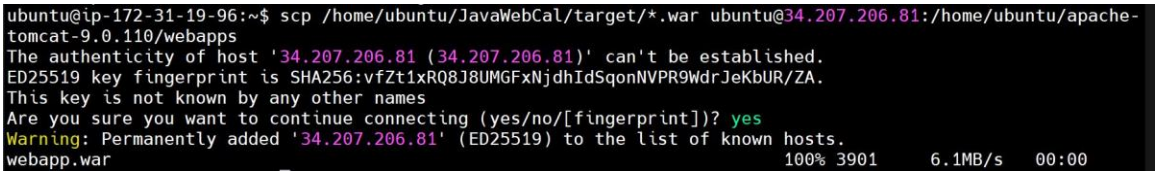


13. Now, to copy the .war artifact from Build server to Deploy server you need to connect both the servers through ssh keys.
    Create ssh keys using “ssh-keygen” command on Build server
    Copy the public keys “id_rsa.pub” of Build server



    Paste the copied public keys of build server to authorized keys (by changing to .ssh directory) of Deploy server

14. Copy the .war file from the Build Server to the Deploy Server using SCP command



15. Verify that the .war file has been copied to /opt/tomcat/webapps/ on the Deploy Server

16. Access the deployed application through the browser using the public IP of the Deploy Server.