**DEPLOYING MANUALLY A JAVA CODE IN TOMCAT**

We will deploy a Java code using Maven as a build tool in Tomcat.

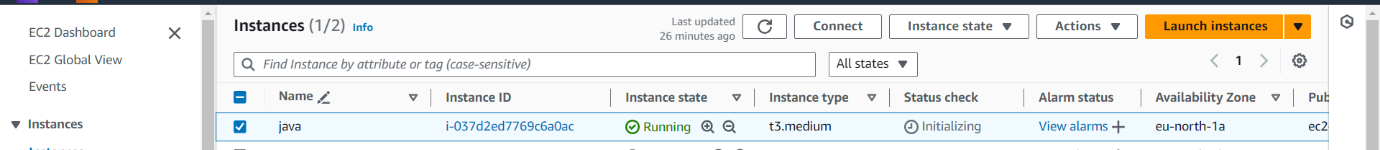
**Maven:** It is a build automation tool used primarily for Java-based projects. It provides a standardized way to manage and build software projects, ensuring consistency and efficiency.

Every java written code will consist of a pom.xml extension.

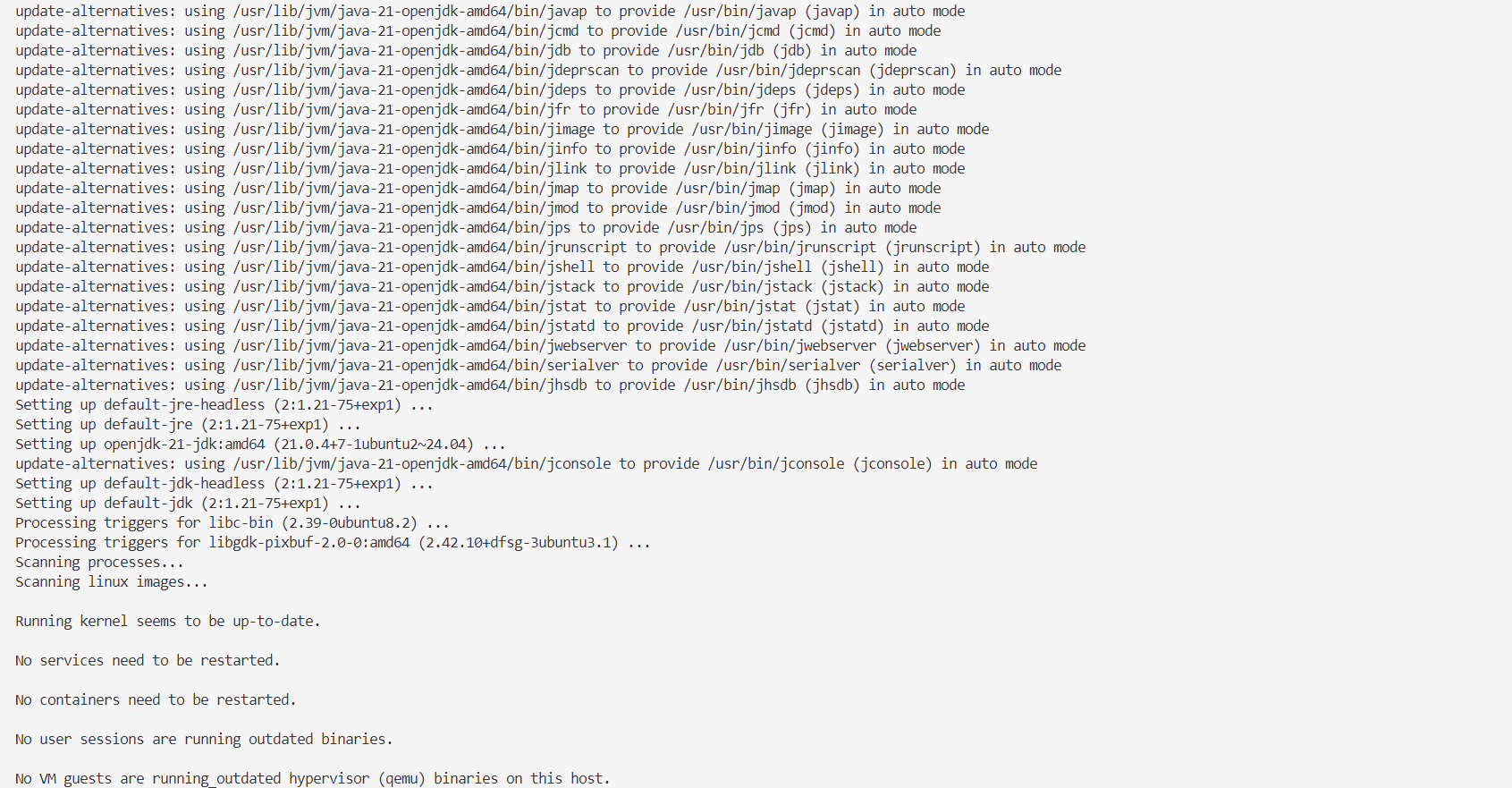
**Pom.xml:** POM is an acronym for Project Object Model. The pom.xml file is the core configuration file for a Maven project. It contains information about the project, such as its name, version, dependencies, and build settings.

**DEPLOYING APPLICATION IN TOMCAT USING EC2 INSTANCE**

* Create an instance by allowing port 8080 and connect and switch to root user.



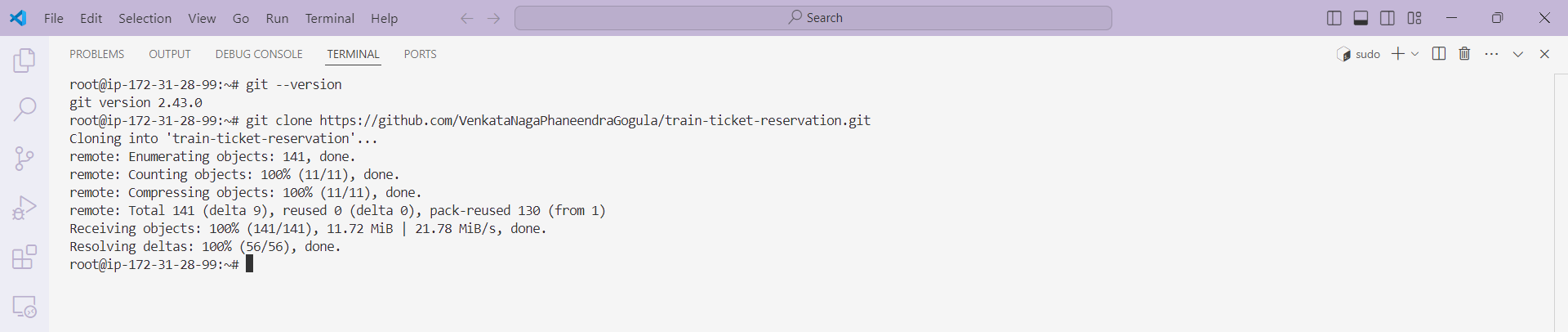
* “apt update -y” – updates the server
* “apt install default-jdk -y” – for installing java in the server



* **“**apt install mvn -y” – for installing maven in the server



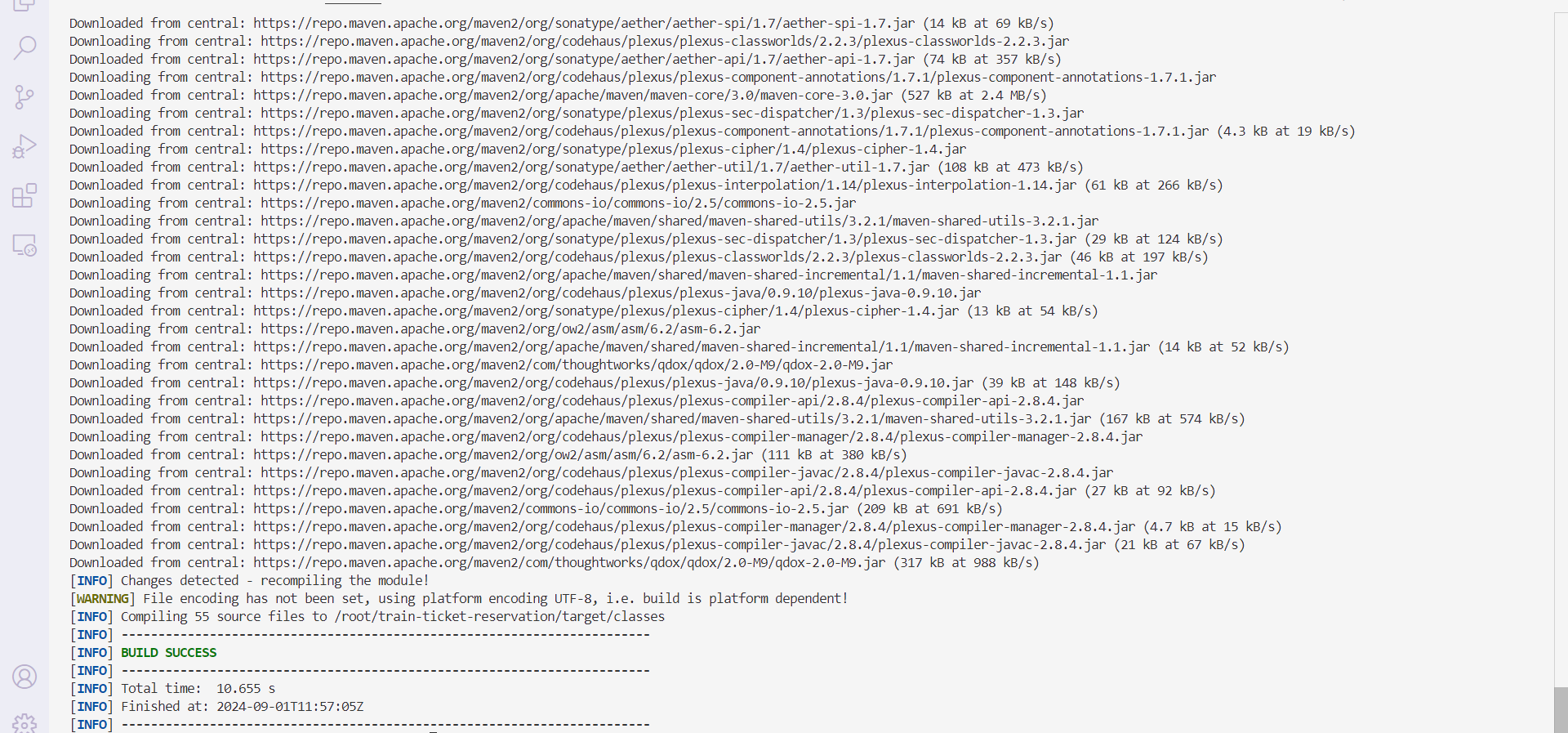
* We have to deploy a java code to take a repository in the git hub and clone it.
* “git clone <repository url>” – Cloning a remote repository to local.



* Enter into the repository using “cd <repository-name>”
* “mvn validate” – checks the project’s structure and configuration to ensure everything is set up correctly.



* “mvn compile” – compiles the source code as well as download the dependencies available in pom.xml



* **“**mvn clean” – cleans the files generated by the previous build

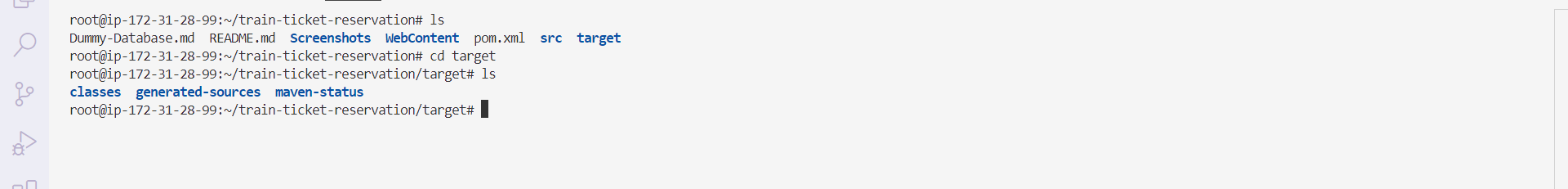


In the above picture, after compiling “target” directory is created. When we run “mvn clean” command, it cleans the target directory.

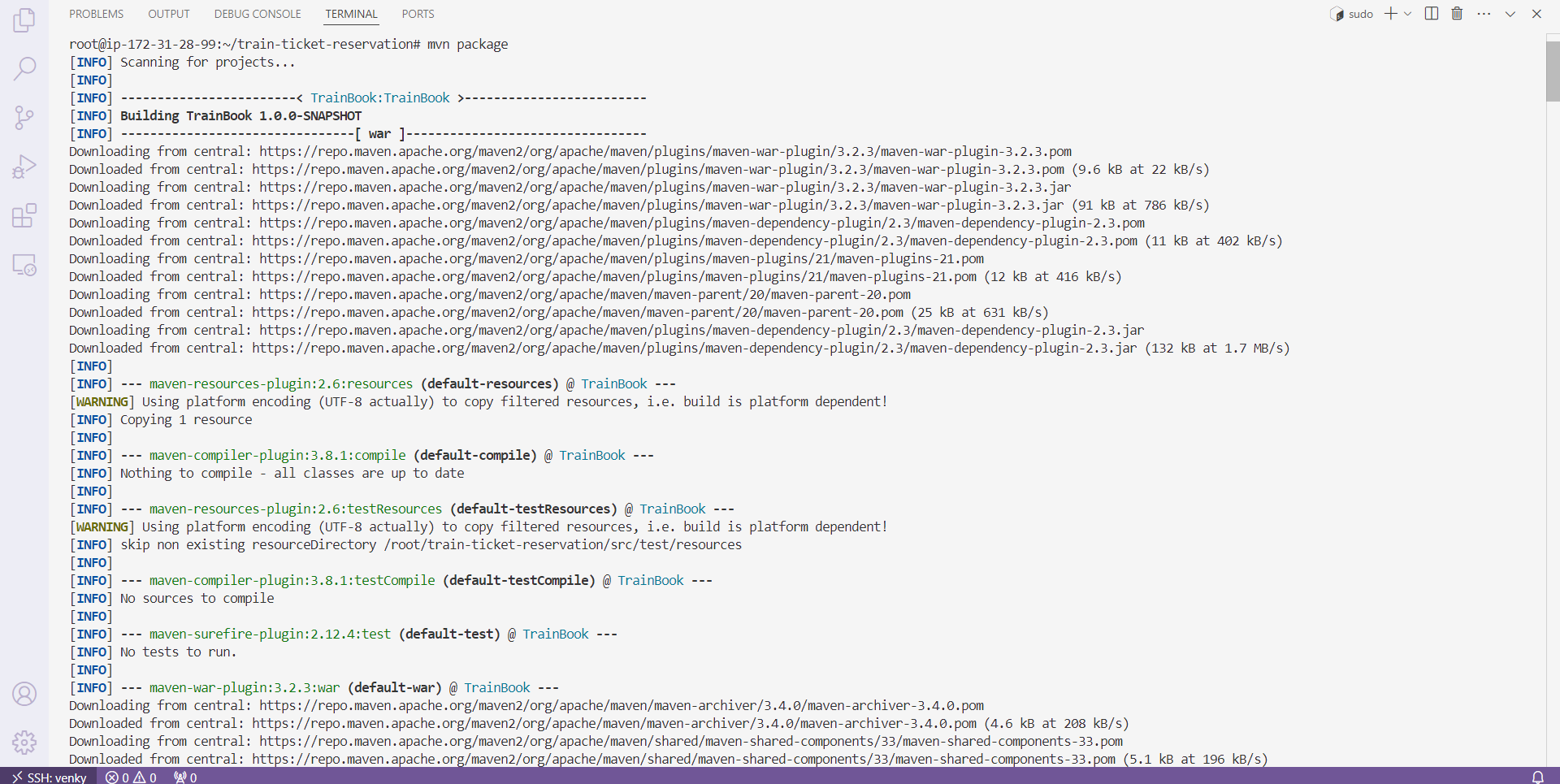
* “mvn test” – to run unit tests

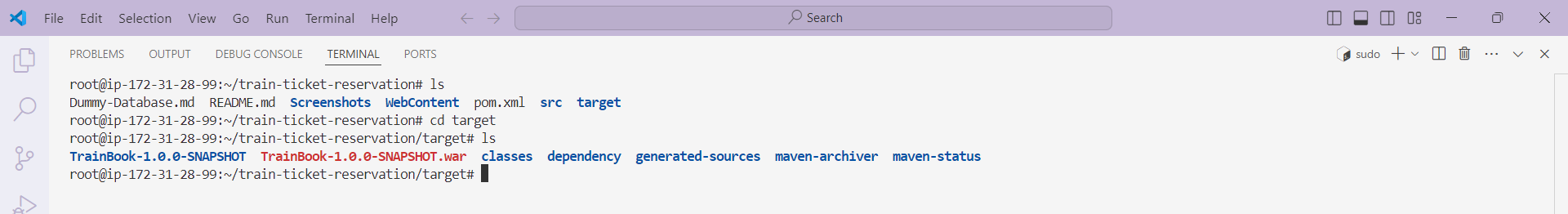


* A target folder will be generated. Enter into the target directory and say “ls”, we will not see “.WAR” file.

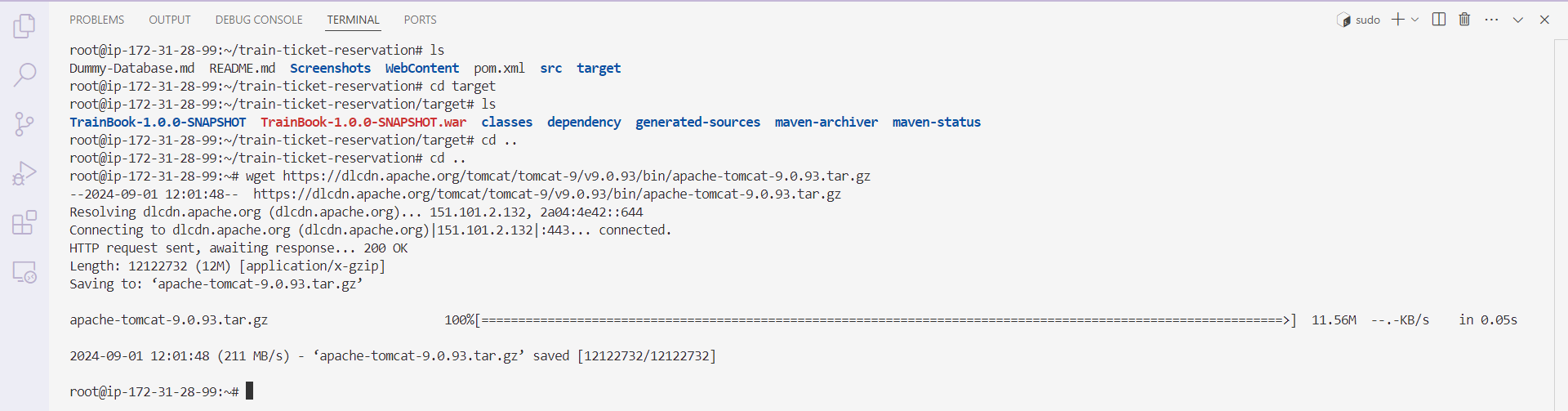


* “mvn package” – package compiled source code into distributed format. Now a “.WAR” file is generated. By using this “.WAR” file we will deploy our application in tomcat.

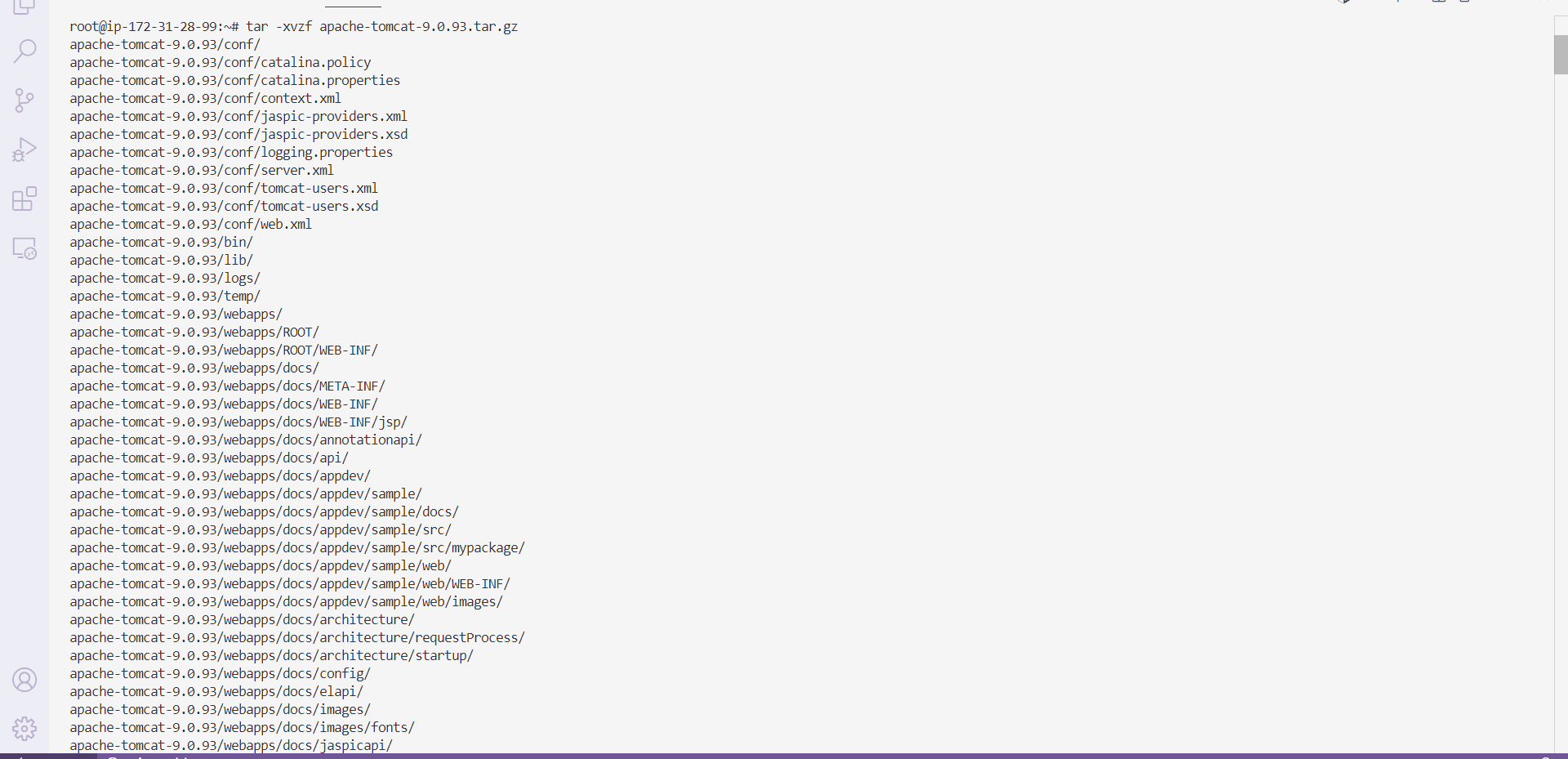




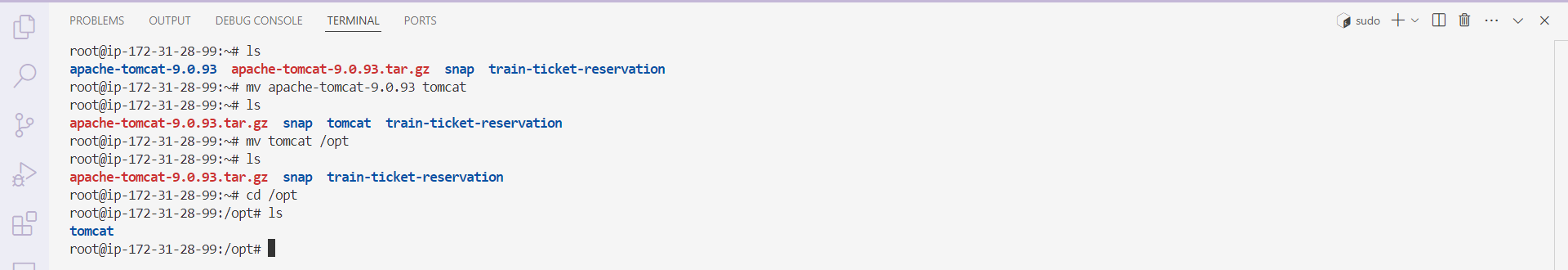
* Back to the “root” directory and install Apache Tomcat by copying the “tar.gz” extension link and paste into the terminal with “wget”.



* Now extract all files from installed Apache Tomcat using “tar -xvzf <installed file-name>”



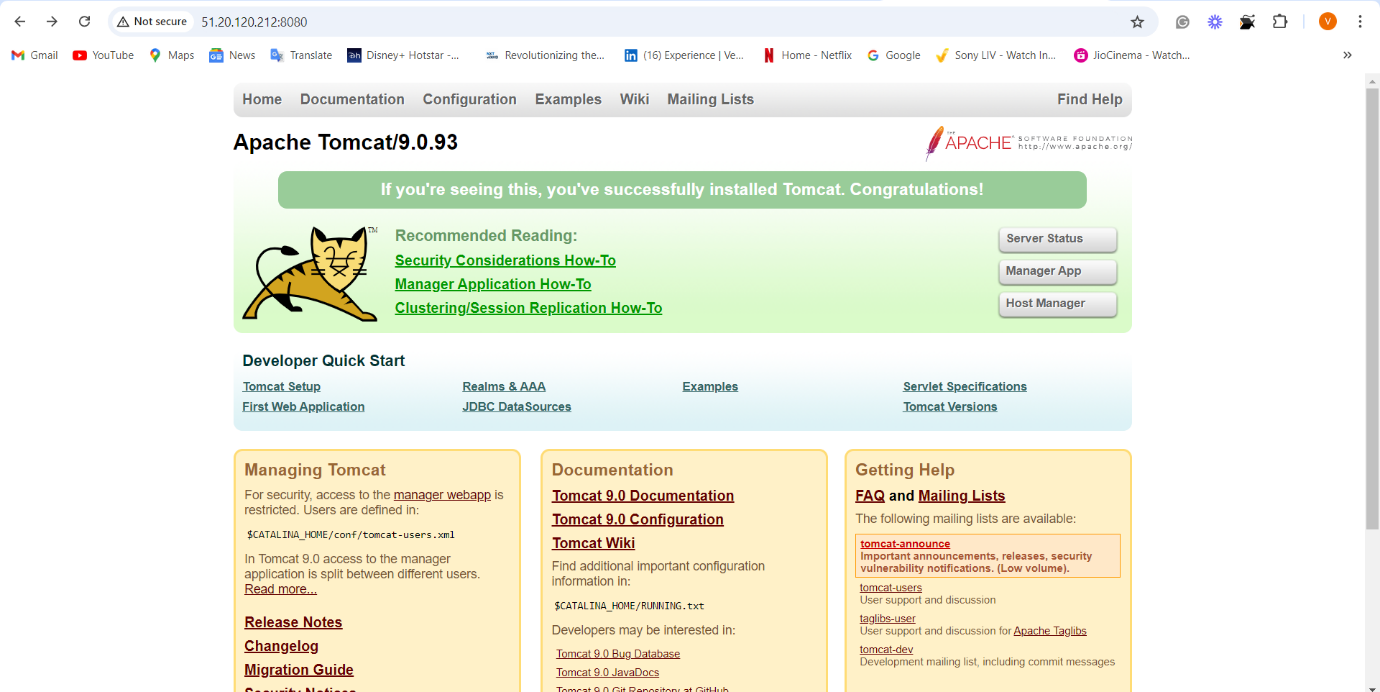
* Rename the installed file name into a short name and move it into “/opt” directory in Apache Tomcat.



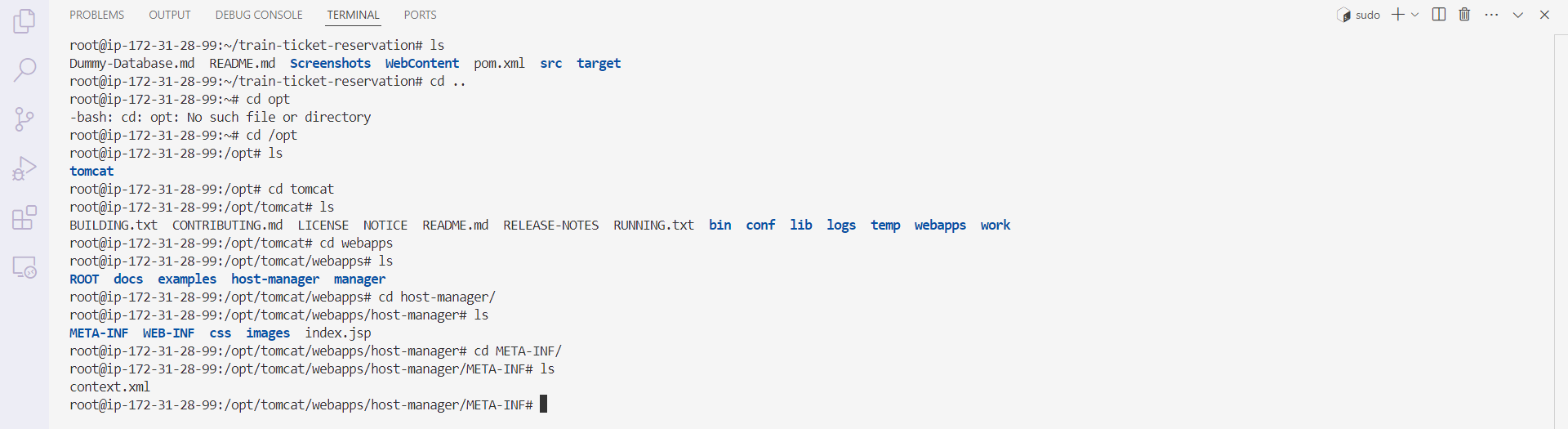
* Now start the Apache Tomcat using “./startup.sh”



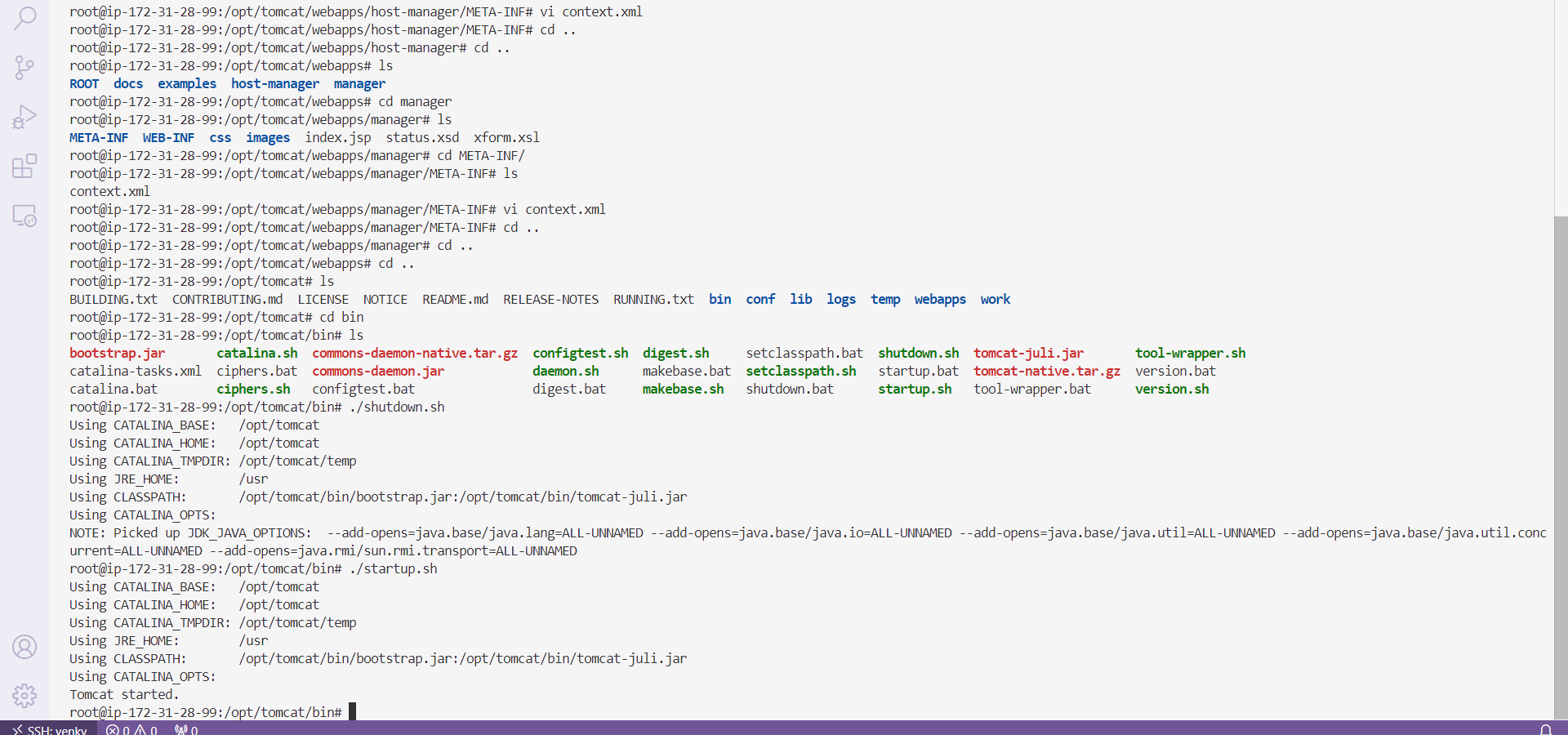
* Copy the public\_ip and paste in browser with “:8080”.



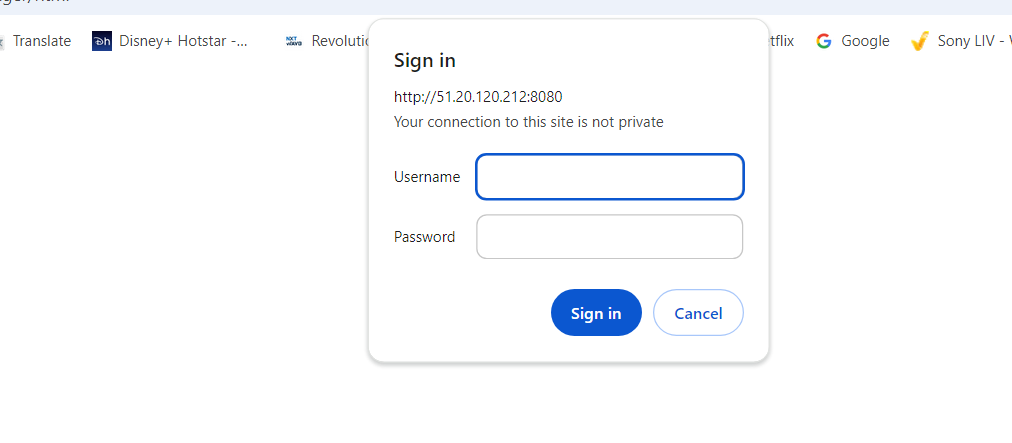
* While entering into the “manager app”, it will raise an error. So, to remove this error, we will enter into “/opt/tomcat/webapps/host-manager/META-INF” and make comments on some lines in “context.xml”.



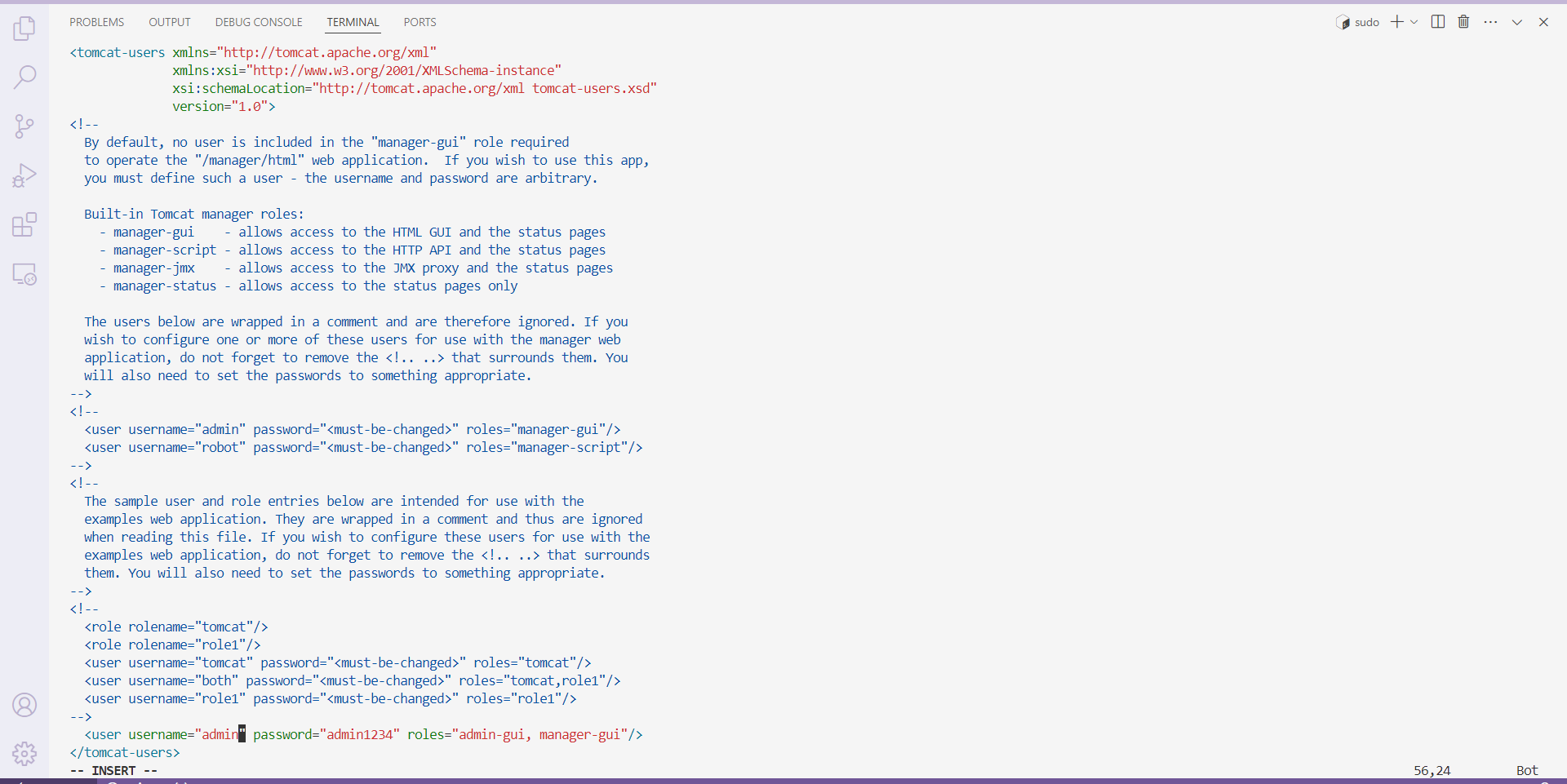
* The same process applies to the manager directory, shutdown, and startup Apache.



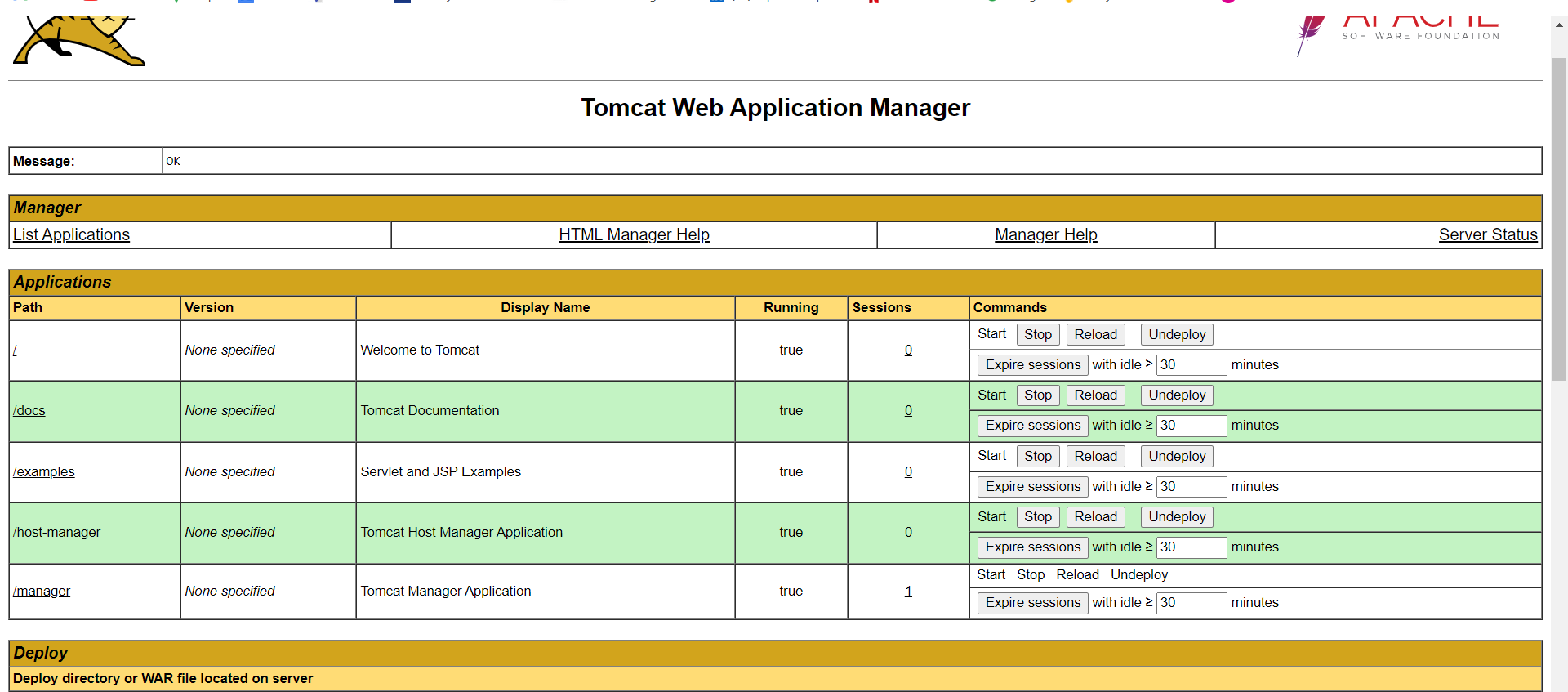
* When we click on the manager app, it will ask for username and password.



* Enter into “/opt/tomcat/conf” and open “tomcat-users.xml” to add user and password.



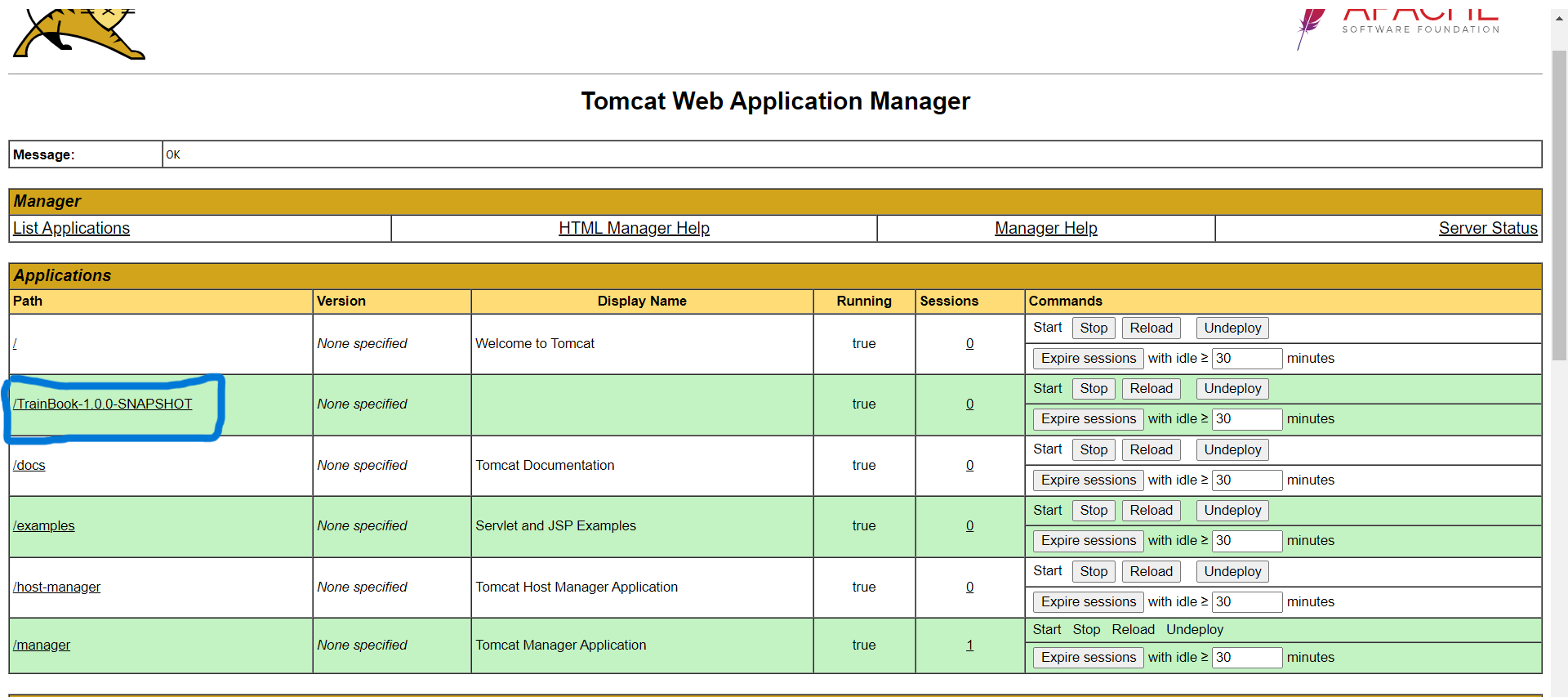
* Again shut down and startup Apache.
* Now enter your username and password in the manager app. We will see the application manager page.



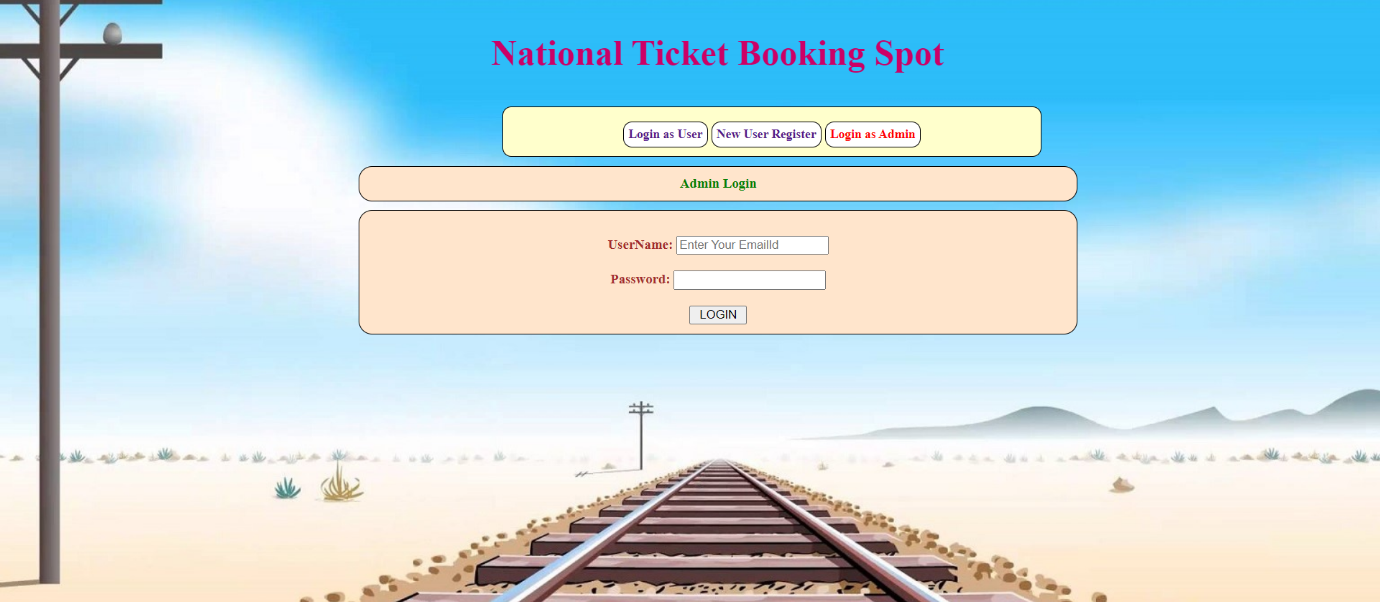
* Now, copy the “.WAR” file path and also the destination path.

cp /root/train-ticket-reservation/target/TrainBook-1.0.0-SNAPSHOT.war /opt/tomcat/webapps

* Now we will see our application in tabular columns



* Click on the above blue-circled link, and we will enter into the application.



**Conclusion:** We can conclude that the Java-based application is deployed manually in Tomcat with the help of “Maven” build tool.