**Java Web Greeter and Quote App**

**Overview**:

A simple Java application that greets the user using a given name and also displays a random motivational quote. Also using Maven and Tomcat to deploy in two different servers one for building and another for deploying in AWS EC2.

**Tech Stack:**

* **Language:** Java
* **Build Tool:** Maven
* **Server:** Apache Tomcat
* **Deployment:** Two servers – Build Server & Deploy Server

**Prerequisites:**

* **Java JDK 17** on both servers
* **Maven** installed on **Build Server**
* **Tomcat** installed on **Deploy Server**
* Both servers **must be in the same network**, typically the same **VPC** in AWS, to **copy files internally using private IPs**

**Architecture:**

# **+----------------+ +----------------+**

# **| Build Server | -----> | Deploy Server |**

# **| (Maven Build) | | (Tomcat) |**

# **+----------------+ +----------------+**

# 

# **Step-by-Step Guide: Two-Server Java Web Calculator Deployment**

## **1. Create the Servers:**

### **Build Server**

1. **Launch a Ubuntu EC2.**
   * **Amazon Machine Image: Ubuntu 24.04 LTS**
   * **Instance type: t3 micro(2 vCPU, RAM: 1 GB)**
2. **Create a Key pair(Build-Key.pem) for Secure Login.**
3. **Create a Security group(SSH-Build) or select an existing group(SSH-Build).** 
   * **Add SSH port(22) to the inbound rule (If creating a Security group).**

### **Deploy Server**

1. **Launch a Ubuntu EC2.**
   * **Amazon Machine Image: Ubuntu 24.04 LTS**
   * **Instance type: t3 micro(2 vCPU, RAM: 1 GB)**
2. **Create a Key pair(Deploy-Key.pem) for Secure Login.**
3. **Create a Security group(Tomcat-Deploy) or select an existing group(Tomcat-Deploy).** 
   * **Add SSH port(22) to the inbound rule (If creating a Security group).**
   * **Add Custom port(8080) to the inbound rule(Tomcat runs on this Port).**

**Tip: Make sure both servers can communicate (or at least Build Server can SCP files to Deploy Server).**

## **2. Install Prerequisites:**

## **a. On Both Servers:**

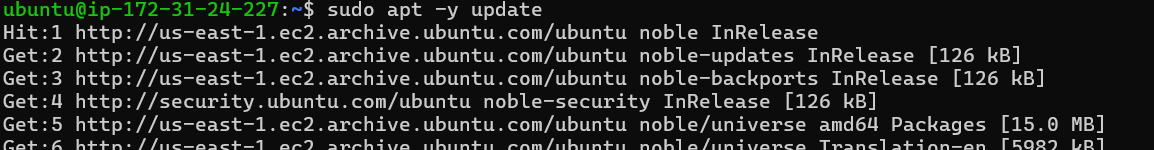
sudo apt -y update (To Update the Server)

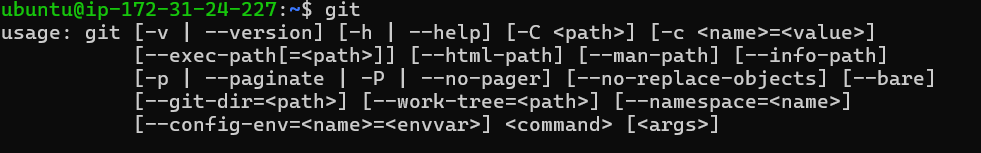
git (to check whether the git is installed or not)

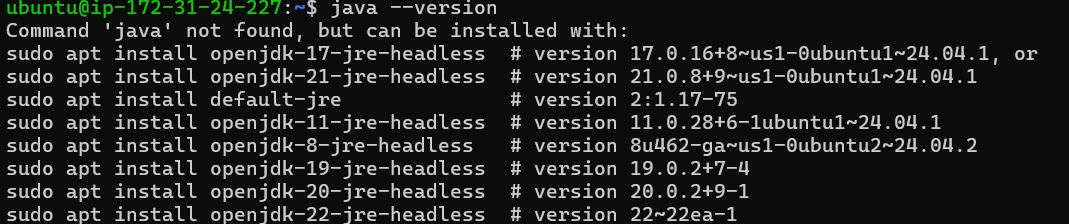
**java** –version (Shows the Java Versions to Download)

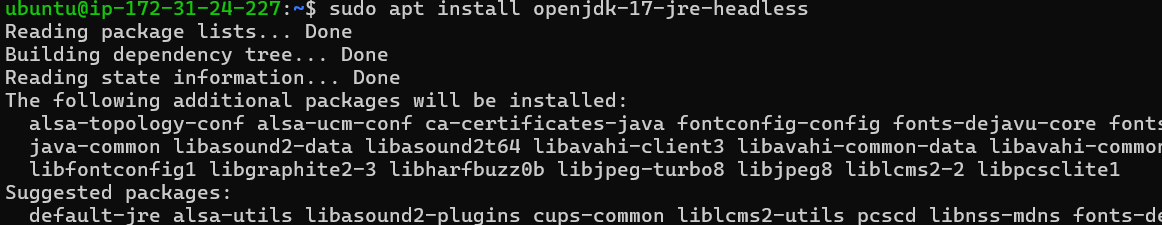
sudo apt install openjdk-17-jre-headless (To install the Java 17 version)

**Build-Server:**

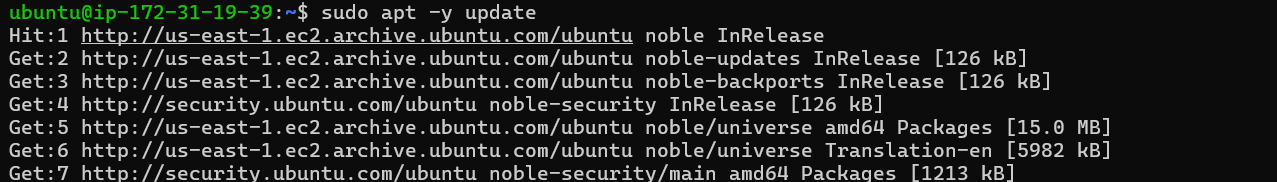
****

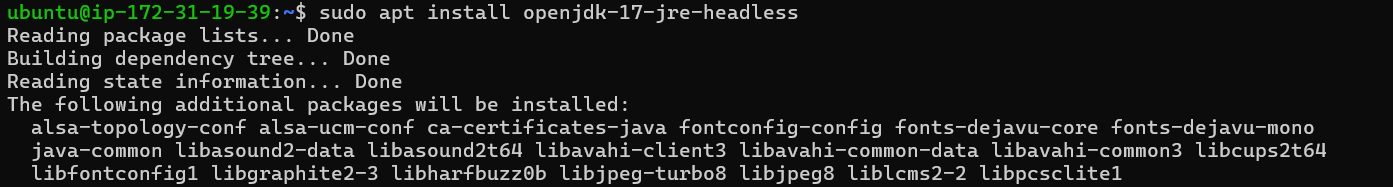






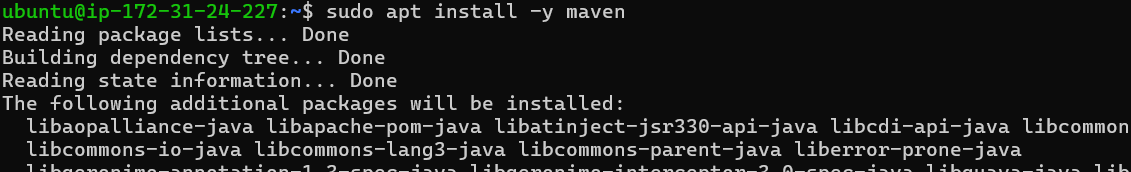
**Deploy-Server:**

****

****

## **b. Build Server:**

* Install **Maven**:
  + Sudo apt -y install **maven**



## **c. Deploy Server:**

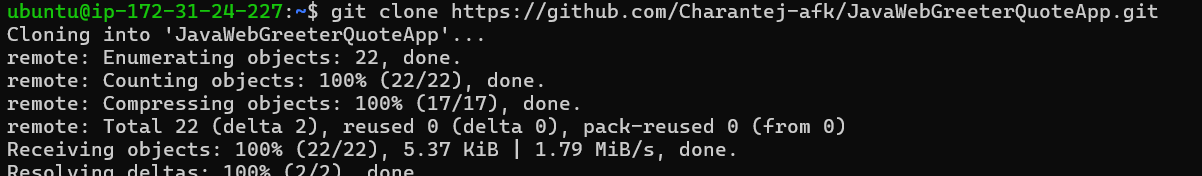
* Install **Tomcat**:
  + Using **wget** command and URL from **Tomcat** [tar.gz](http://tar.gz).

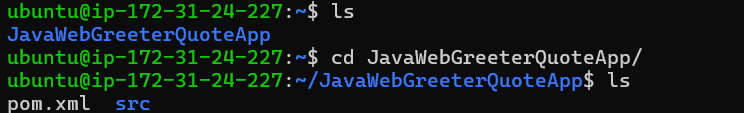


## **3. Setup Project on Build Server:**

## **Clone the repository:**

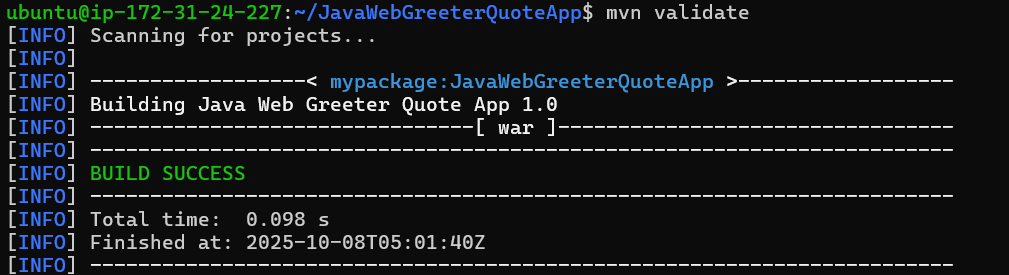
* 1. Using the command git clone URL of the repository.
  2. Change directory to JavaWebCalculator



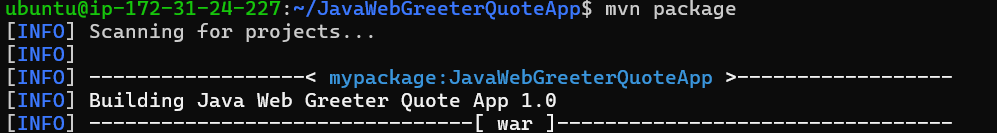


## **Validate and Build with Maven:**

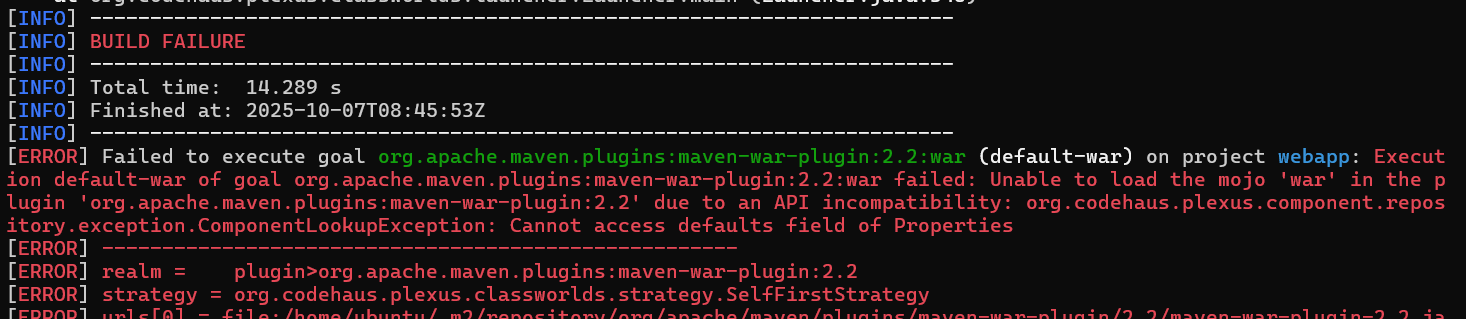
* 1. Validate with using the command - “mvn validate”



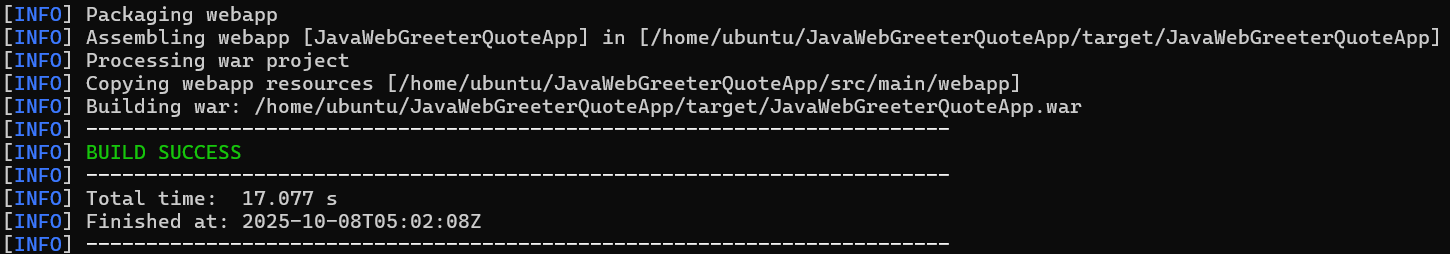
* 1. Build with using the command - “mvn package”
     1. Use the command to compile and build the java webapp at once.



* + 1. This is a situation in which we will get errors during packaging because the pom.xml file is build for the older version of Java and Maven. So we need to modify the pom.xml file to the correct version of Java and Maven. Normally this is done by the Code Developers.



* + 1. But here we will get no errors because the pom.xml is in line with the newer version or whatever the Java version we are using for building and deploying the Java webapp.

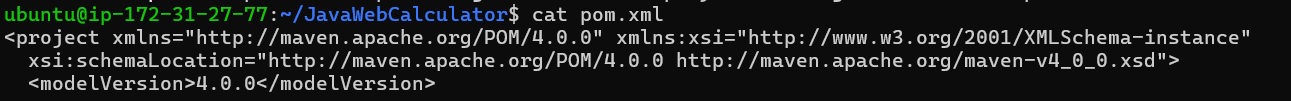


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

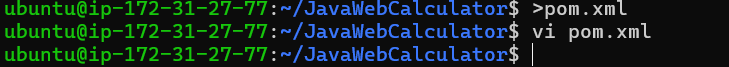
**This is a Situation where we get an error while packaging the following Steps need to be run by the “Code Developer” to edit so that the pom.xml file is in line with the Newer version of the Java and Maven.**

## **If there are errors in the pom.xml file then edit & upload it to the Build server.**

* 1. **To read the pom.xml file we use the command:**
     1. Cat pom.xml

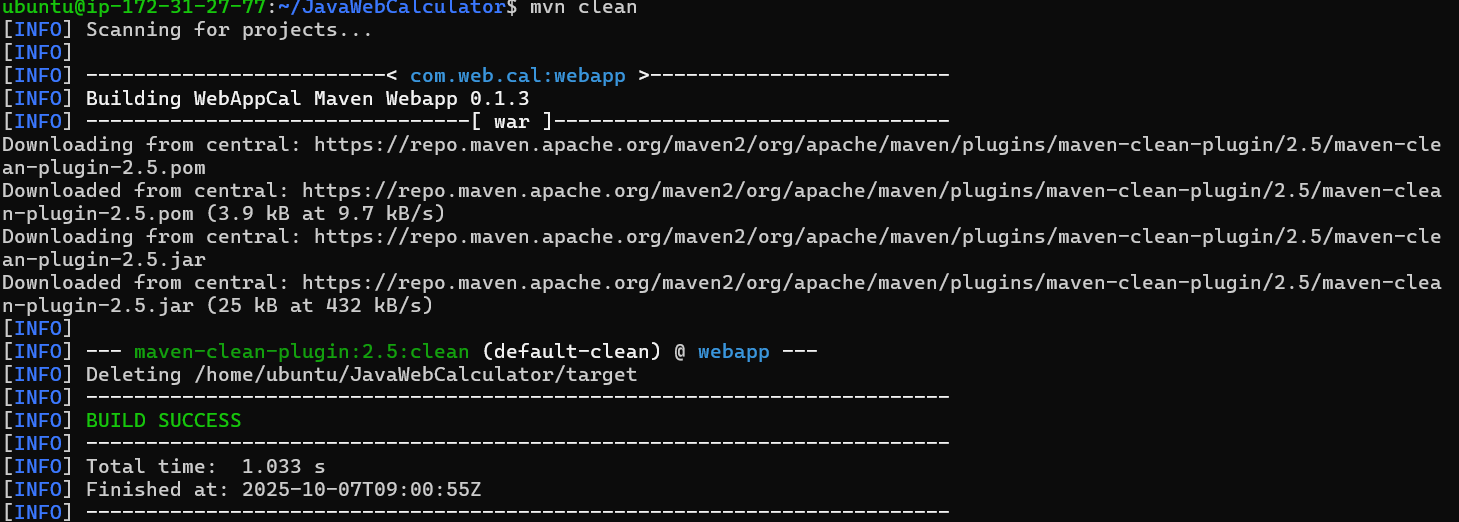


* 1. **Clear and Edit the pom.xml file using the command:**
     1. > pom.xml
     2. vi pom.xml (vi is the text editor).



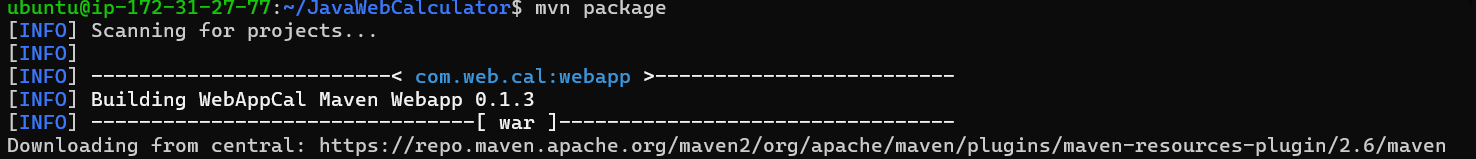
## **Clean maven package:**

* + 1. mvn clean



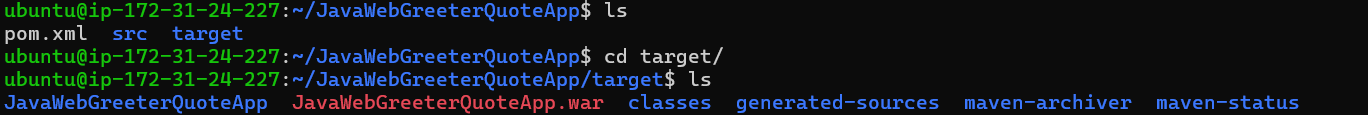
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Build the Webapp:**

* + 1. mvn package

## **Result:**

* 1. After Successful Building of webapp you will get a folder target in it you will have the Artifact file or WAR file.

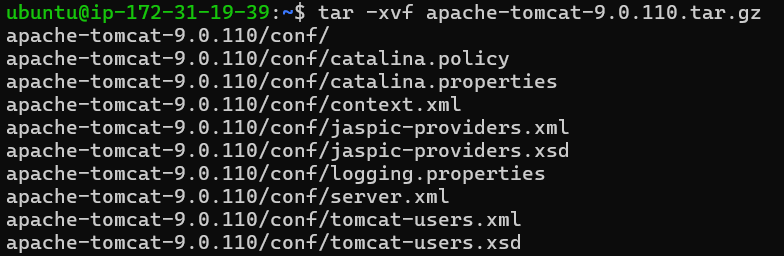


## **4.Setup project on Deploy server:**

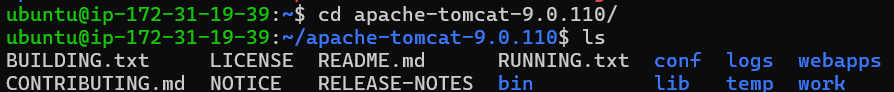
## **Tomcat Zipped Tar file:**



## **Extract the tar file content:**

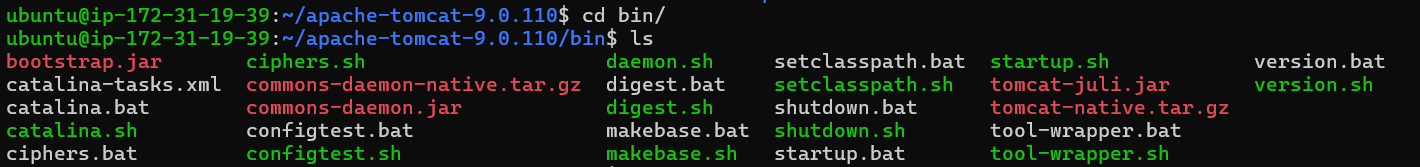
 

## **Change Directory to tomcat file:**

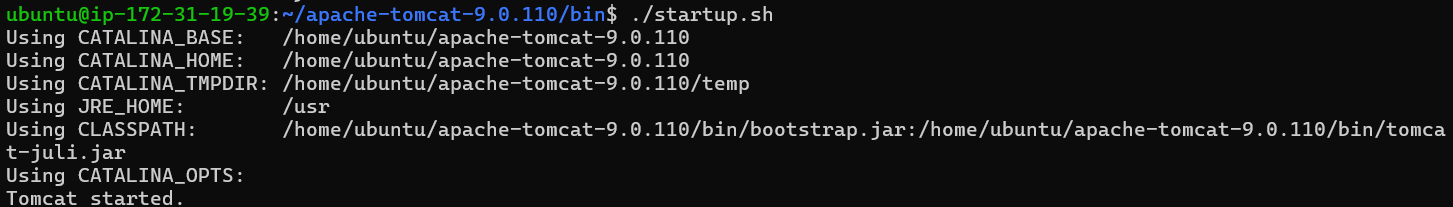


## **Start Tomcat:**

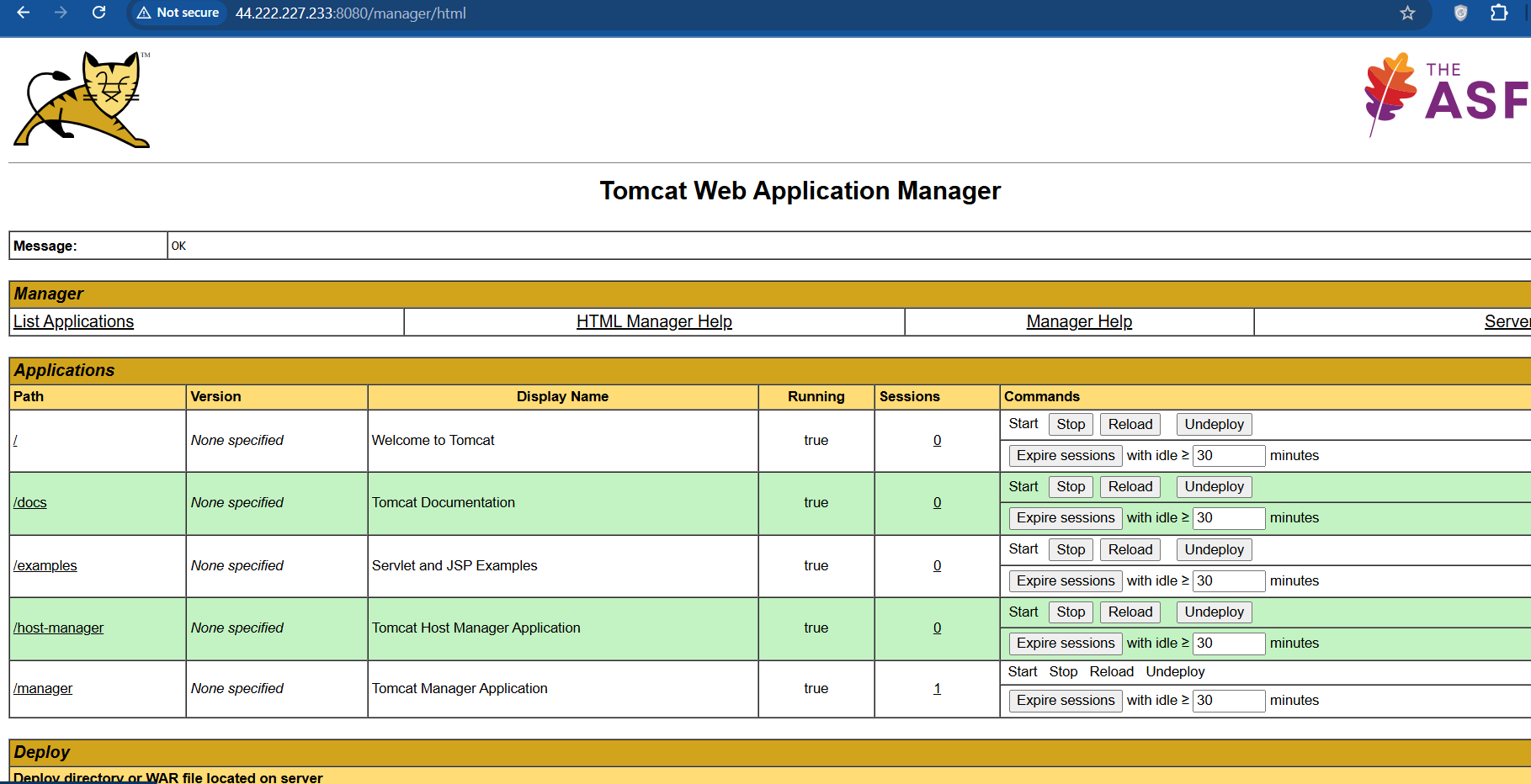
* 1. Go to bin directory using cd bin/

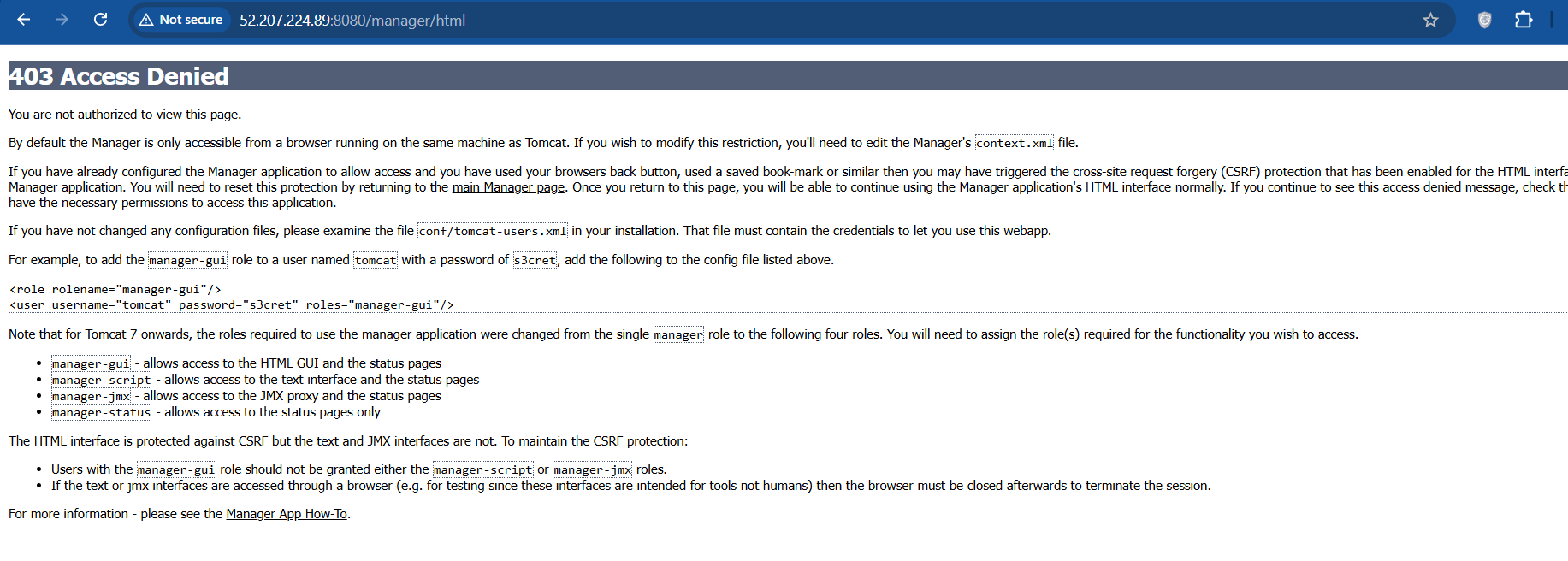


* 1. Run the Script [startup.sh](http://startup.sh): using ./[startup.sh](http://startup.sh)



* 1. Using the public ip of the Deploy server and port number of tomcat we can observe tomcat.
     1. <PUBLIC-IP-OF-THE-DEPLOY-SERVER>:8080
     2. Access Manager App on the website. We will be Access Denied.

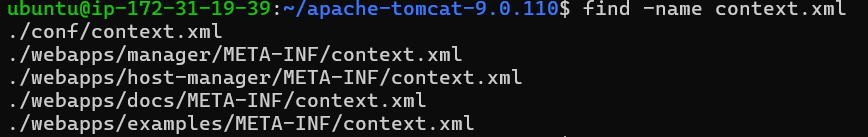




## **Edit Manager’s context.xml files:**

* 1. Search Manager’s context.xml files using command

Find -name context.xml



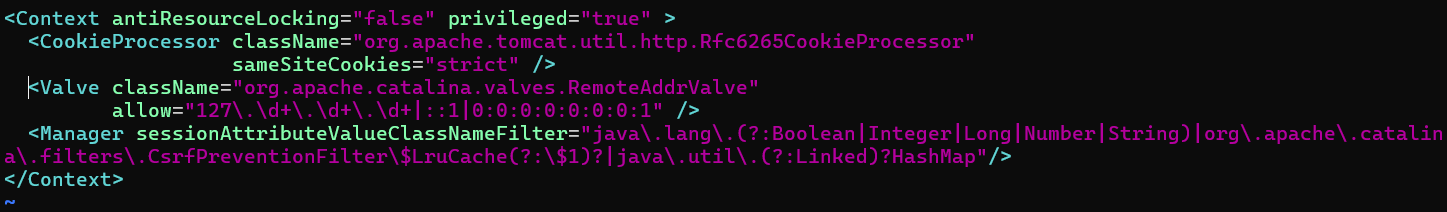
* 1. Edit the files that contain manager words or remove the highlighted lines.

vi ./webapps/manager/META\_INF/context.xml

vi ./webapps/host-manager/META-INF/context.xml



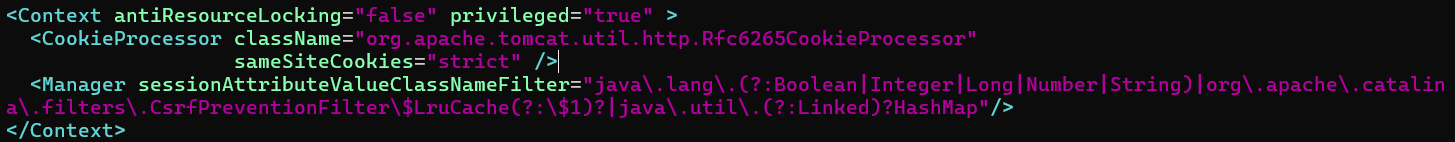
* This is the content that will be in the both file with some difference but we need to change only one line that is in the next point.



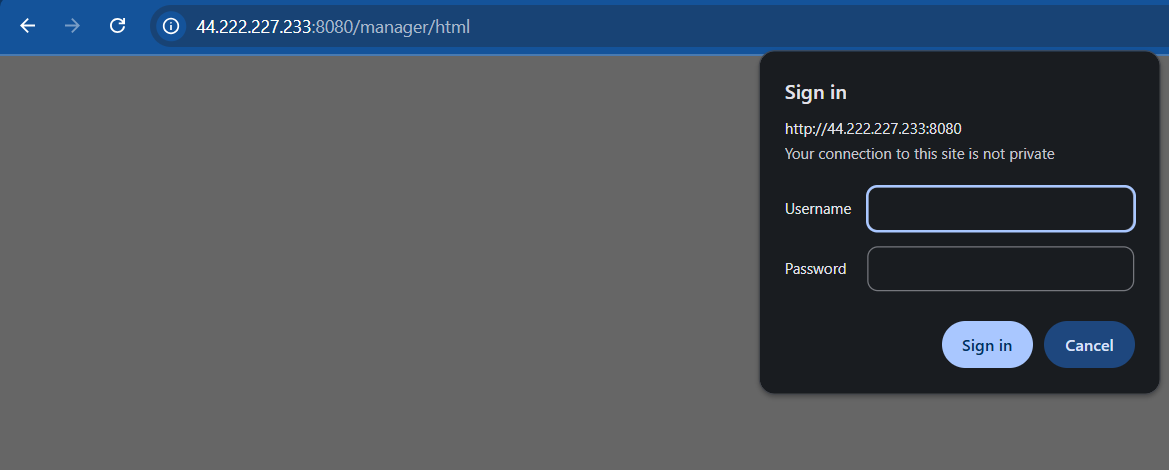
* This is the line that was commented so that when tomcat reads the file it will ignore this line.



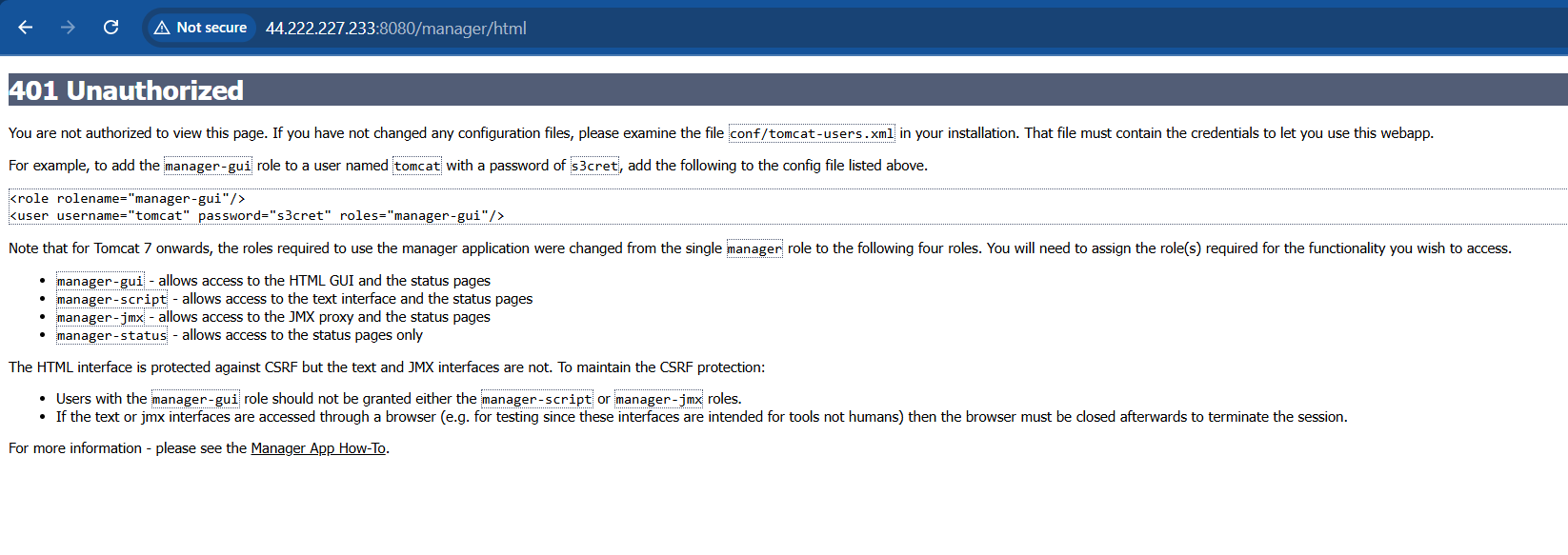
* Another option is to delete the line so that we have no need to comment on the line.



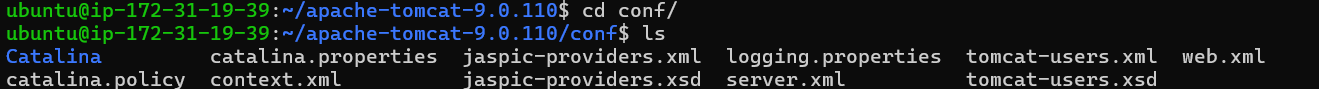
* 1. Now go to Tomcat Website again it’ll ask for username and password which we didn’t configure:



* + 1. Select Cancel which takes us to this page:

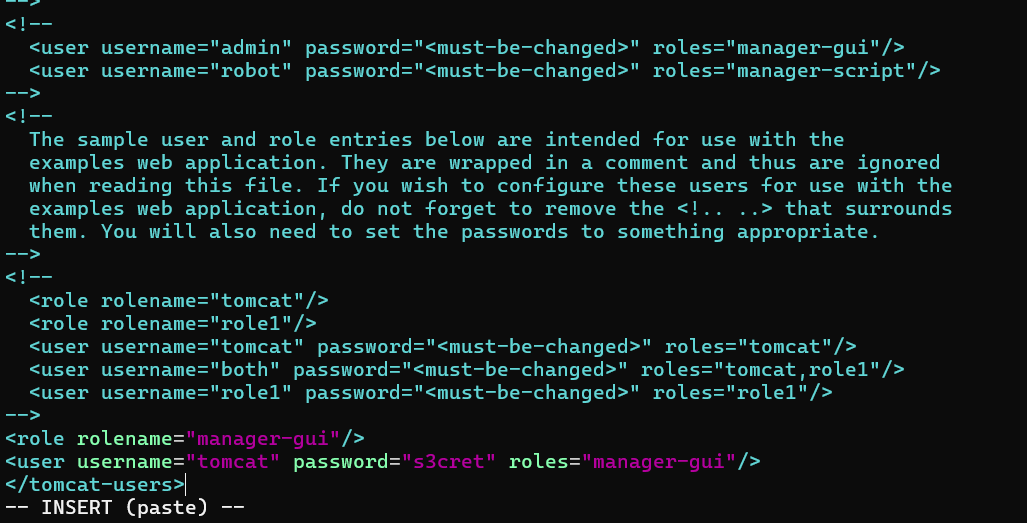


* 1. To avoid this we need to configure a username and password in the conf directory in it we need to edit the tomcat-users.xml file

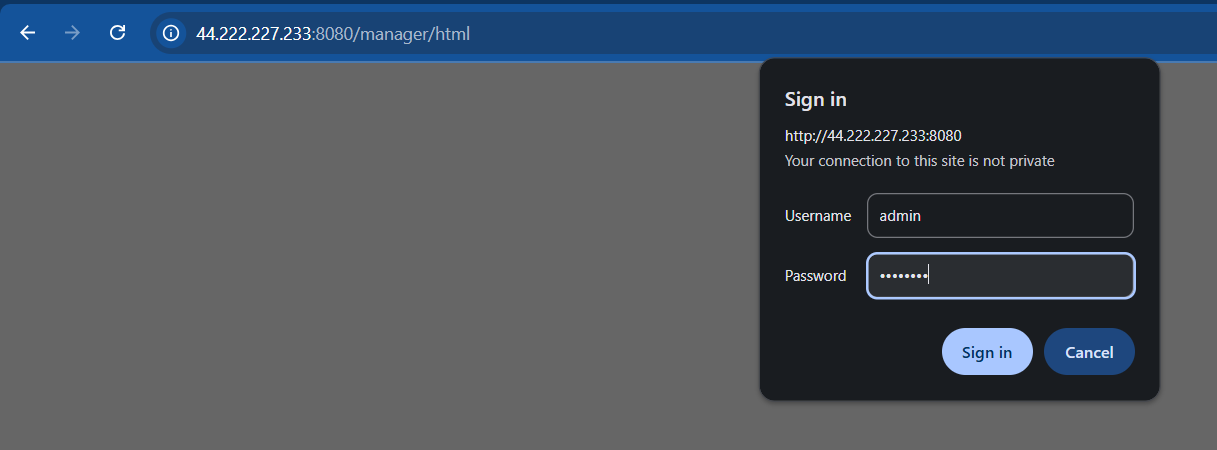




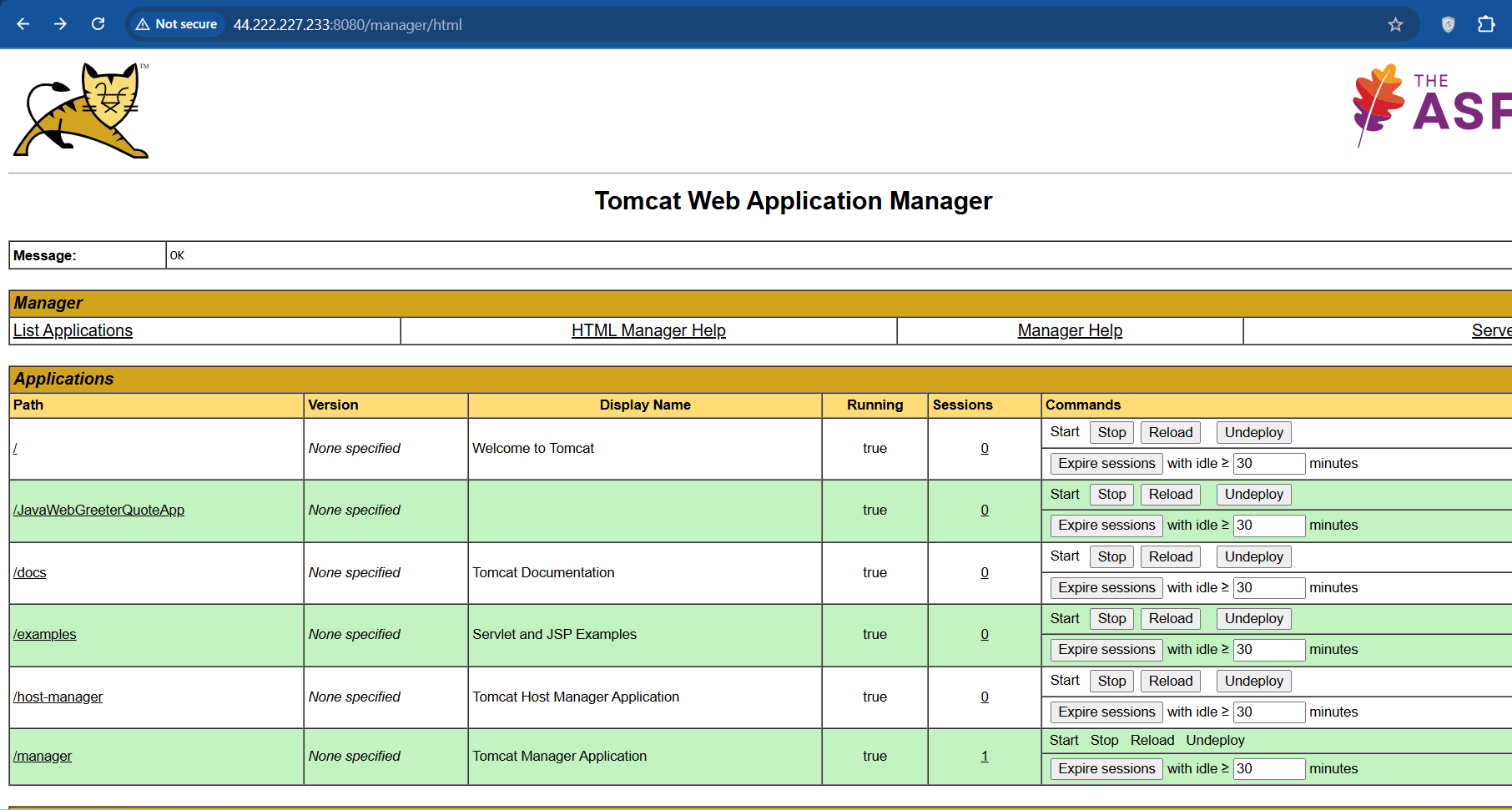
* 1. Here we need to add the user and password to this file.



* 1. Now, reload the Webpage and Enter credentials we have given in tomcat webpage.



* 1. We will see this page where all the webapps in the webapp directory contained in the tomcat file.

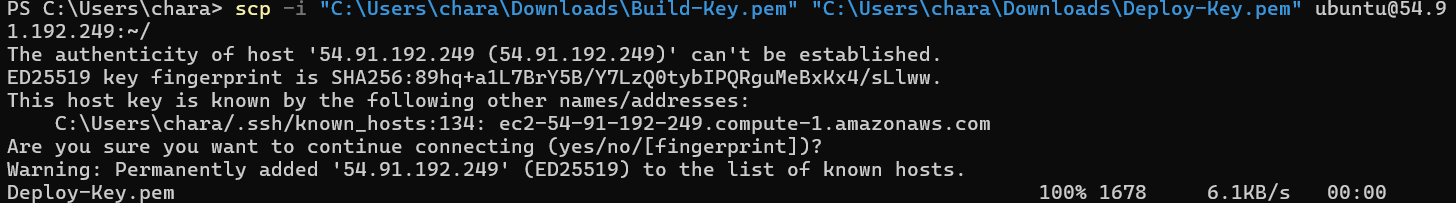


## **5. Transfer WAR File to Deploy Server:**

## **Copying pem file:**

* 1. We need to make sure the Deploy server PEM file is on the Build server.
     1. From your **local machine**, copy it to the build server and check in the Build server using:

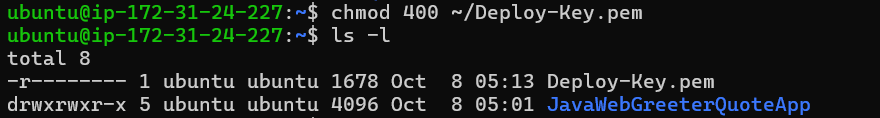
scp -i "C:\Users\chara\Downloads\Build-Key.pem" "C:\Users\chara\Downloads\Deploy-Key.pem" ubuntu@<PUBLIC\_IP\_OF\_BUILD\_SERVER>:~/





## **Give proper Permissions:**

* 1. After copying the pem file we need to give proper(read) permissions to the pem file.

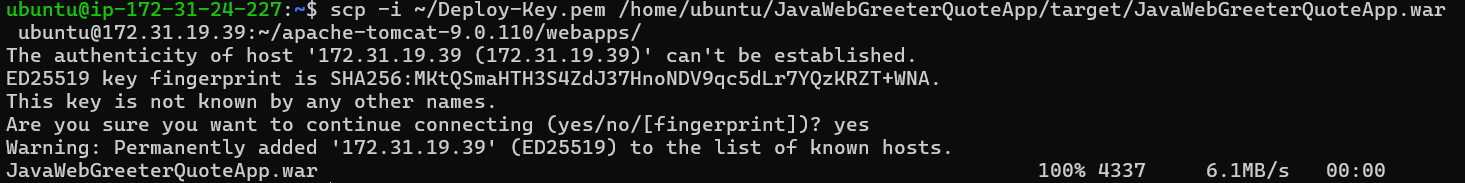


## **Copy WAR file to Deploy server:**

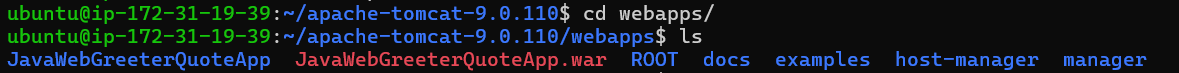
* 1. Inside the Build server we need to SCP(Secure Copy) the WAR file using:

scp -i ~/Deploy-Key.pem ~/JavaWebCalculator/target/\*.war ubuntu@172.31.20.230:~/apache-tomcat-9.0.110/webapps/

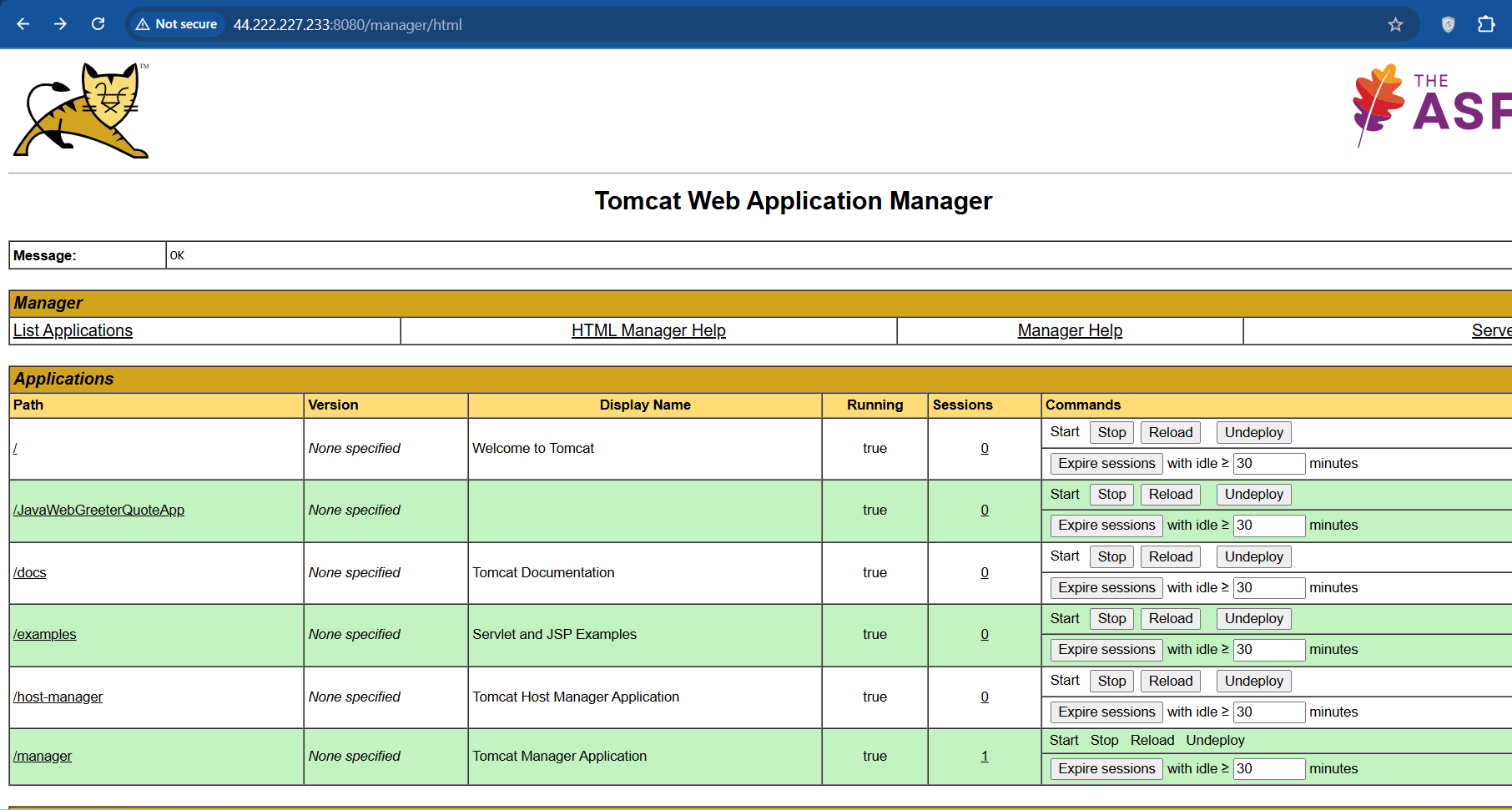
* 1. It will ask confirmation say “yes” it will send the copy to the Deploy server:



* 1. Check in the Deploy server to find the WAR file in webapps directory.

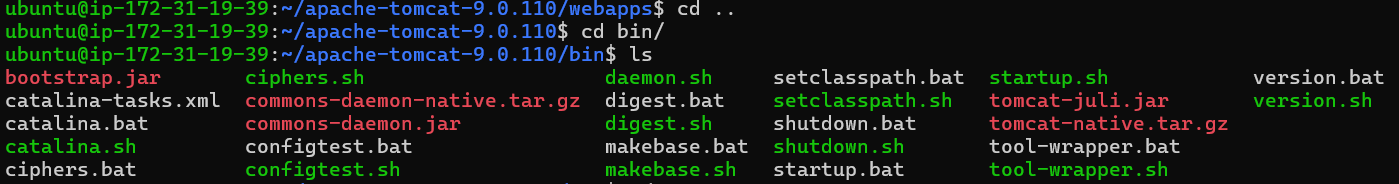


* 1. Now, check the webpage whether it is reflected in the Applications Table:

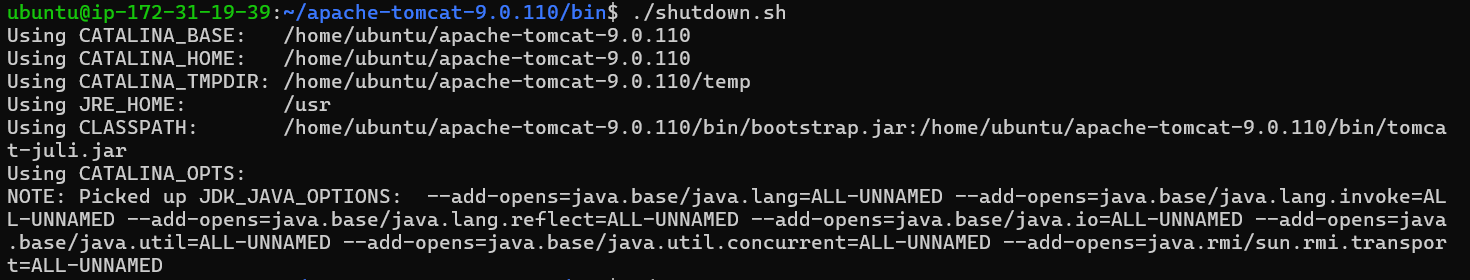


## **Restart the Tomcat or Deploy server:**

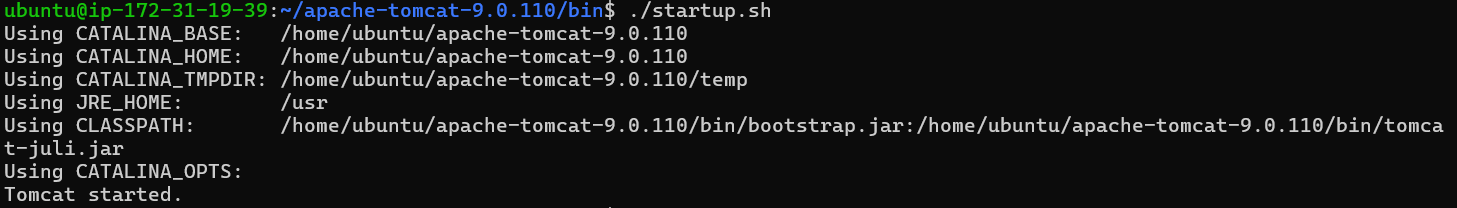
* 1. Go to bin/ directory using:
     1. cd ~/apache-tomcat-9.0.110/bin/



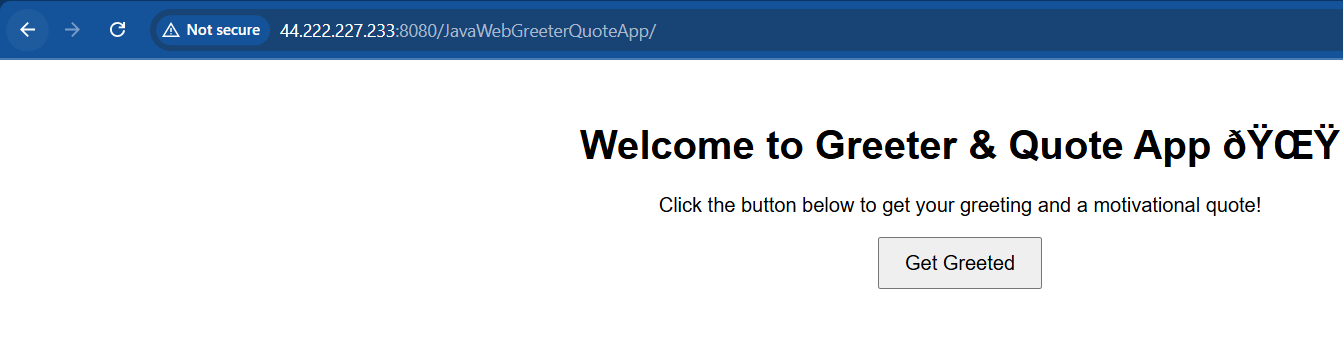
* 1. Run the Shutdown Script in the bin directory:
     1. ./[shutdown.sh](http://shutdown.sh)



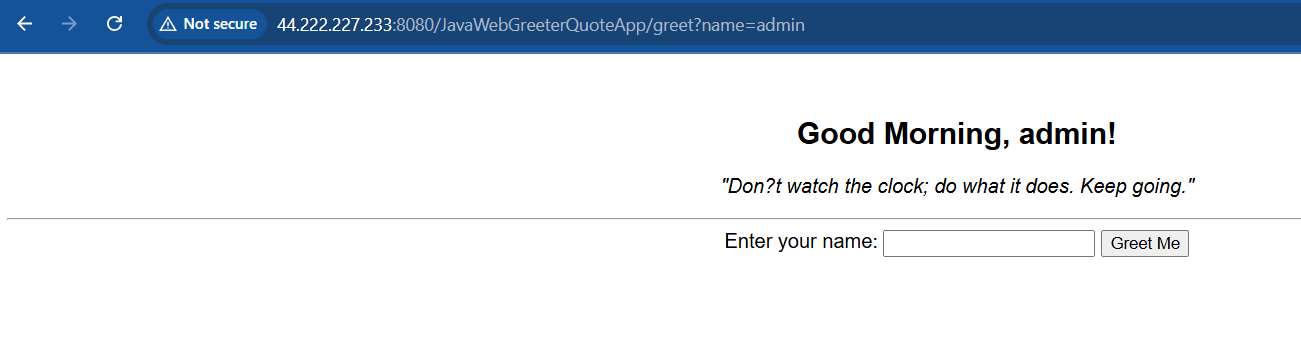
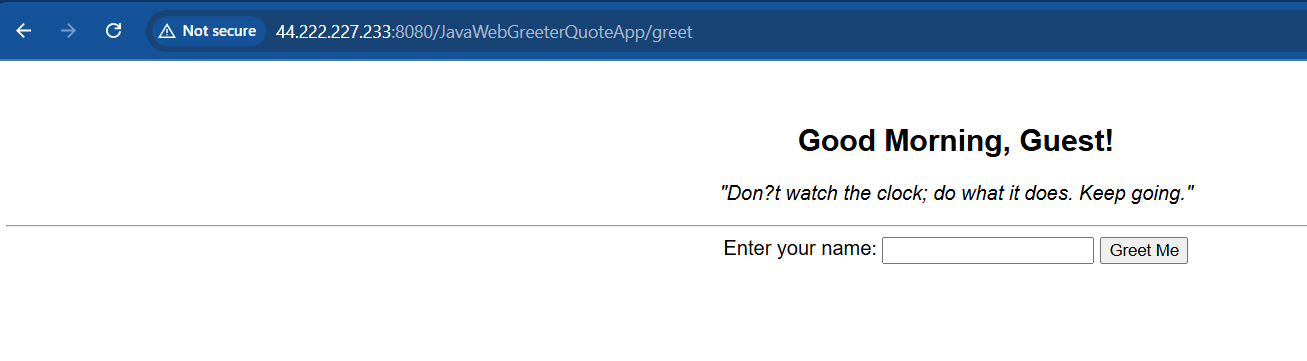
* 1. Run the Startup Script in the bin directory:
     1. ./[startup.sh](http://startup.sh)



## **Open Webapp in Tomcat Application table:**



## **Test the Application:**



## **6. Conclusion:**

This project demonstrates the complete process of building and deploying a simple **Java web calculator** using **Maven** and **Tomcat** on **two separate servers** — one for building and another for deployment.

It highlights basic **DevOps workflow concepts**, including:

* Multi-server environment setup
* Maven build management
* WAR file transfer and deployment