# PROJECT-1 (MAVEN WEB APP)

List of steps involved in deploying a Maven Web application on Tomcat server.

|  |  |
| --- | --- |
| Description | Command |
| Install updates to linux instance | sudo yum update -y |
| Install openjdk-11 | sudo yum install java-11-openjdk -y |
| Check the java version | java -version |
| Verify the JAVAHOME link | readlink -f $(which java) |
| Install wget utility | sudo yum install git wget -y |
| Verify the git version | git --version |
| Stage 1 starts:  Downloading the latest version of Apache-Maven to /tmp/ folder. The contents of the tmp folder will be cleared on restart. | **sudo wget** [**https://dlcdn.apache.org/maven/maven-3/3.9.1/binaries/apache-maven-3.9.1-bin.tar.gz**](https://dlcdn.apache.org/maven/maven-3/3.9.1/binaries/apache-maven-3.9.1-bin.tar.gz) **-P /tmp** |
| Extract the maven tar file to ‘/opt’ folder. | sudo tar xvzf /tmp/**apache-maven-3.9.1-bin.tar.gz** -C /opt |
| Rename the extracted directory to ‘maven’ or we can also create a soft link ‘maven’ in ‘/opt’ directory | sudo mv /opt/**apache-maven-3.9.1** /opt/maven  (or)  sudo ln -s /opt/apache-maven-3.9.1 maven |
| Add the environment variables in ‘/etc/profile.d/maven.sh’ file  (The /etc directory in Linux stores system-wide configuration files that control the behaviour of the operating system and installed software. The /etc/profile.d/ directory in Linux stores shell scripts used to make custom system-wide environment changes and configurations. These scripts are executed by the /etc/profile script during user login, allowing administrators to set global environment variables and customize system behaviour without directly modifying the main /etc/profile file.) | **sudo vi /etc/profile.d/maven.sh**  **export M2\_HOME=/opt/maven**  **export PATH=${M2\_HOME}/bin:${PATH}**  **(**This tells programs (especially Apache Maven and some tools that use it) where to find the Maven installation directory, so they know where to look for Maven’s files and tools. M2\_HOME is the recommended variable for Maven 2 and newer versions. After setting this, you often also add $M2\_HOME/bin to your PATH so you can run Maven by typing mvn in the terminal.) |
| Change the permissions of ‘maven.sh’ file to executable | sudo chmod +x /etc/profile.d/maven.sh  (**This command makes the maven.sh script executable so the system can run it automatically for user sessions or when needed**.) |
| Run the ‘maven.sh’ file with source | source /etc/profile.d/maven.sh  (The source /etc/profile.d/maven.sh command reads and executes the shell script located at /etc/profile.d/maven.sh in the current shell environment. This action loads the environment variables defined within the script, specifically setting the MAVEN\_HOME and PATH variables to include the Maven binary directory, thereby making the mvn command available for use in the current terminal session.) |
| Now, maven is available and we can verify maven version.  Stage – 1 completed. | mvn -version |
| Stage – 2 Starts:  Proceeding to download Apache-tomcat latest version and saving it to ‘/tmp’ directory | sudo wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.74/bin/apache-tomcat-9.0.74.tar.gz -P /tmp |
| Extracting the contents of the tarball to ‘/opt’ directory | sudo tar xvzf /tmp/apache-tomcat-9.0.74.tar.gz -C /opt  (The /opt/ directory is primarily used for storing software and add-on packages that are not part of the default operating system installation. It serves as a standard location for third-party applications, particularly those distributed as pre-packaged binary bundles or self-contained software that does not rely on system dependencies outside its own package. Each software package is typically installed in its own subdirectory under /opt/, such as /opt/example for a package named "example".) |
| Renaming the extracted directory in ‘/opt’ to ‘tomcat’ for ease of use. | sudo mv /opt/apache-tomcat-9.0.74 /opt/tomcat |
| If we are running other application with port 8080 in the same instance we can consider to change the port for tomcat application by editing the ‘server.xml’ file | sudo nano /opt/tomcat/conf/server.xml  # update port number to 8081 , ctrl+O and ctrl+X |
| Create user and group ‘tomcat’ | sudo useradd tomcat  (Creating a dedicated user for Tomcat is recommended primarily for security reasons. Running Tomcat as the root user poses a significant risk, as a security vulnerability in the application could allow an attacker to gain full control over the entire server.  By running Tomcat under a dedicated, low-privileged user account, the potential damage from a security breach is contained; an attacker would only have the permissions associated with that specific user, limiting their ability to compromise the broader system.  This principle of least privilege is a fundamental security practice, ensuring the Tomcat process has only the minimum necessary permissions to function.  The Tomcat documentation explicitly advises against running the server as root and recommends creating a dedicated user with minimal system permissions.  This practice is standard for any service accessible over an untrusted network, such as the internet.) |
| Change ownership of ‘/opt/tomcat’ directory to ‘tomcat’ user and group | sudo chown -R tomcat:tomcat /opt/tomcat  (Changing the ownership of the Tomcat folder to the tomcat user and group is a fundamental security and operational practice. The primary reason is to ensure that the Tomcat process runs with the principle of least privilege, meaning it only has the minimum necessary permissions to function.  By setting the tomcat user as the owner of the Tomcat installation directory, the process is confined to its own environment and cannot access or modify files outside its designated scope, which significantly reduces the risk if the process is compromised.  This setup prevents the Tomcat process from being able to modify critical system configuration files, deploy unauthorized web applications, or alter existing ones, even if an attacker gains control of the Tomcat process.  The tomcat user is typically configured so that it cannot log in interactively, further enhancing security.  The tomcat group is often used to grant specific, controlled access to directories like logs, temp, and webapps where the application needs to write data, while keeping the configuration files (conf) read-only for the group to prevent accidental or malicious changes.  This separation of ownership and permissions is a standard best practice recommended by Tomcat's official security documentation.) |
| Make the binary files of tomcat as executable | sudo sh -c 'chmod +x /opt/tomcat/bin/\*.sh'  (The command sudo sh -c 'chmod +x /opt/tomcat/bin/\*.sh' is used to grant execute permission to all shell script files (those ending in .sh) located within the /opt/tomcat/bin/ directory.  This is a necessary step after installing Apache Tomcat, as the startup and shutdown scripts (like startup.sh and shutdown.sh) need to be executable for the Tomcat server to be managed properly. This step is crucial because if the scripts are not executable, attempting to start or stop Tomcat will result in a "Permission denied" error.  The command ensures that all necessary Tomcat scripts have the correct permissions to run, allowing the server to be started and managed as a service.) |
| Verify the JAVA\_HOME environment variable to be added to ‘tomcat.service’ file | readlink -f $(which java)  # copy /usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.el7\_9.x86\_64 |
| Add the following text to ‘tomcat.service’ file with proper JAVA\_HOME variable  (The JAVA\_HOME environment variable is not typically added directly to the tomcat.service file itself. Instead, the tomcat.service file, which is a systemd service configuration file on Linux systems, is used to define the environment in which the Tomcat process runs. To set JAVA\_HOME for Tomcat via this service, you need to configure it within the service file's environment section.  Setting JAVA\_HOME in the service configuration ensures that the Tomcat process starts with the correct Java environment, which is crucial for the server to function properly, especially for features like JSP compilation that require a JDK.)  (Using this script, We are creating a service file for starting and stopping the tomcat service without entering the lengthy commands every time) | sudo nano /etc/systemd/system/tomcat.service  [Unit]  Description=Apache Tomcat Web Application Container  After=network.target  [Service]  Type=oneshot  RemainAfterExit=yes  User=tomcat  Group=tomcat  Environment="JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-11.0.19.0.7-1.el9\_1.x86\_64/"  Environment="JAVA\_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true"  Environment="CATALINA\_BASE=/opt/tomcat"  Environment="CATALINA\_HOME=/opt/tomcat"  Environment="CATALINA\_PID=/opt/tomcat/temp/tomcat.pid"  Environment="CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"  ExecStart=/opt/tomcat/bin/startup.sh  ExecStop=/opt/tomcat/bin/shutdown.sh  [Install]  WantedBy=multi-user.target |
| Reload the system daemon  (The command sudo systemctl daemon-reload tells the Linux system manager (systemd) to reload its configuration files and unit files (which define services), so changes you made to service definitions will take effect without restarting your computer or stopping services. Use this command after you add or modify service files, so systemd knows about your updates.) | sudo systemctl daemon-reload  (In Linux, a daemon is a computer program that runs in the background, independent of any user interaction or controlling terminal. Daemons are essential for providing various system services and functionalities. [1]  Key characteristics of daemons include:  • **Background Operation**: They run continuously in the background, often starting automatically when the system boots up and typically only terminating when the system shuts down.  • **Independence from User Sessions**: Daemons operate independently of any user logged in or any specific terminal session. They continue running even if a user logs out.  • **System Services**: They provide crucial system-level services, such as managing network connections (e.g., sshd for SSH), handling print jobs (e.g., lpd), scheduling tasks (e.g., crond), or serving web pages (e.g., httpd or nginx).  • **No Direct User Interface**: Daemons typically do not have a graphical user interface (GUI) or require direct user input for their operation.  • **Naming Convention:** In Unix-like systems, daemons are often named with a 'd' suffix (e.g., syslogd, sshd, crond) to indicate their nature. |
| Enable tomcat service to run at startup | sudo systemctl enable tomcat  (This command is used to enable the tomcat server. In tomcat disabled mode, when we restart the system the tomcat will not start and run automatically. For tomcat to run automatically when the system is restarted, we need to enable the tomcat service.) |
| Start the tomcat service | sudo systemctl start tomcat |
| Install firewalld if not already installed | sudo yum install firewalld -y |
| Enable firewalld to run at startup (good practice) | sudo systemctl enable firewalld |
| Start firewalld service | sudo systemctl start firewalld |
| Add an exception to port 8081 which is presently being used by tomcat application in this project | sudo firewall-cmd --zone=public --permanent --add-port=8081/tcp |
| Reload the firewall to apply the exception to port 8081 | sudo firewall-cmd --reload |
| Edit the tomcat-users.xml file to add new roles ‘admin-gui’ and ‘manager-gui’ and user credentials for both the roles | sudo nano /opt/tomcat/conf/tomcat-users.xml  Add the following roles in the file:  <role rolename="admin-gui"/>  <role rolename="manager-gui"/>  <user username="admin" password="admin" roles="admin-gui,manager-gui"/> |
| Add your system IP into the allow rules for Manager and Host-manager pages of tomcat application | sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml  Add your system IP to the allow list (or .\* at the end of Value) |
| sudo nano /opt/tomcat/webapps/host-manager/META-INF/context.xml  Add your system IP to the allow list (or .\* at the end of Value) |
| Restart tomcat service to apply the updated changes | sudo systemctl restart tomcat |
| Clone the application code from github repository | git clone https://github.com/javabyraghu/maven-web-app.git |
| Switch into the application directory | cd maven-web-app |
| Build the package using maven | mvn clean package |
| Verify if the ‘.war’ file is created in ‘/target’ sub-directory of the application directory | ls -l ./target/ |
| Copy the ‘.war’ file to ‘/opt/tomcat/webapps’ directory | sudo cp ./target/01-maven-web-app.war /opt/tomcat/webapps/ |
| Now the application can be accessed by the user at the specified URL | Enter URL like:  http://<ifconfig>:8081/01-maven-web-ap |