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 Jboss wildfly. 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 Jboss wildfly.

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-172-31-26-192:~$ git clone https://github.com/Rajnanchari/devsecopstasks.git
Cloning into 'devsecopstasks'...
remote: Enumerating objects: 434, done.
remote: Counting objects: 100% (434/434), done.
remote: Compressing objects: 100% (332/332), done.
remote: Total 434 (delta 106), reused 415 (delta 87), pack-reused 0 (from 0)
Receiving objects: 100% (434/434), 35.05 MiB | 49.29 MiB/s, done.
Resolving deltas: 100% (106/106), done.
```

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

```
ubuntu@ip-172-31-26-192:~/devsecopstasks$ ls
java_code
ubuntu@ip-172-31-26-192:~/devsecopstasks$ cd java_code/
ubuntu@ip-172-31-26-192:~/devsecopstasks/java_code$ ls
Docker.md  Dummy-Database.md  Images  Javatask.md  README.md  Screenshots  WebContent  pom.xml  src
```

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

```
[INFO] Configured Artifact: com.github.jsimone:webapp-runner:8.0.30.2:jar
Downloading from central: https://repo.maven.apache.org/maven2/com/github/jsimone/webapp-runner/8.0.30.2/webapp-runner-8.0.30.2.jar
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/jsimone/webapp-runner/8.0.30.2/webapp-runner-8.0.30.2.jar (9.1 MB at 34 MB/s)
[INFO] Copying webapp-runner-8.0.30.2.jar to /home/ubuntu/devsecopstasks/java_code/target/dependency/webapp-runner.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.784 s
[INFO] Finished at: 2025-10-07T12:36:12Z
[INFO] ------------------------------------------------------------------------
ubuntu@ip-172-31-26-192:~/devsecopstasks/java_code$
```

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

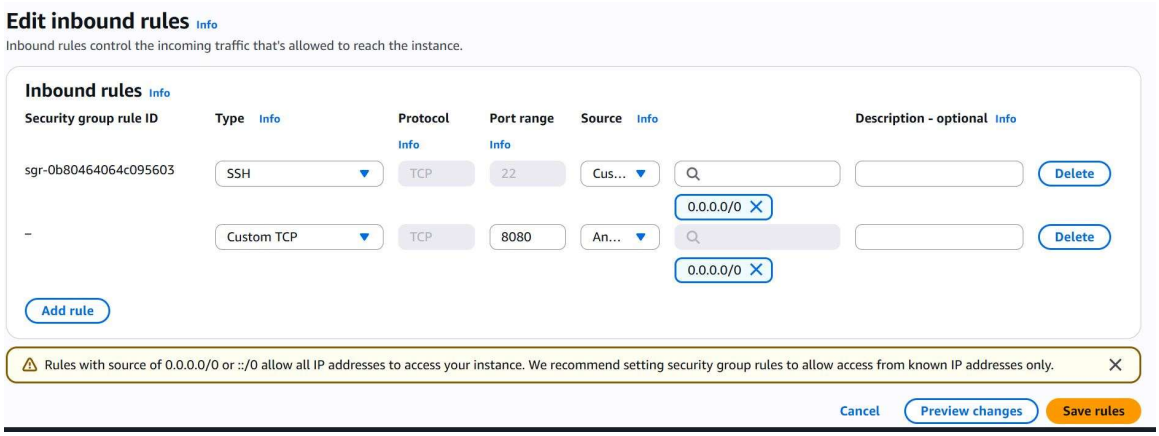
6. Install Jboss-wildfly on the Deploy Server
   - Go to Jboss-wildfly official website
   - Copy the tar extension file url link
   - Download it using wget command followed by the url link
   - After downloading tar file, extract the tar file using “tar -xvf wildfly-20.0.1.Final.tar.gz” command

```
ubuntu@ip-172-31-30-191:~$ ls
wildfly-36.0.1.Final.tar.gz
ubuntu@ip-172-31-30-191:~$ tar -xvf wildfly-36.0.1.Final.tar.gz
```

7. Start the Wildfly service on the Deploy Server by changing to “bin” directory

```
ubuntu@ip-172-31-30-191:~/wildfly-36.0.1.Final/bin$ cd
ubuntu@ip-172-31-30-191:~$ cd wildfly-36.0.1.Final/bin/
ubuntu@ip-172-31-30-191:~/wildfly-36.0.1.Final/bin$ ls
add-user.bat         domain.bat                  jboss-cli.bat    standalone.conf.bat
add-user.properties  domain.conf                 jboss-cli.ps1    standalone.conf.ps1
add-user.ps1         domain.conf.bat             jboss-cli.sh     standalone.ps1
add-user.sh          domain.conf.ps1             jboss-cli.xml    standalone.sh
appclient.bat        domain.ps1                  jconsole.bat     systemd
appclient.conf       domain.sh                   jconsole.ps1     wildfly-elytron-tool.jar
appclient.conf.bat   elytron-tool.bat            jconsole.sh      wsconsume.bat
appclient.conf.ps1   elytron-tool.ps1            jdr.bat          wsconsume.ps1
appclient.ps1        elytron-tool.sh             jdr.ps1          wsconsume.sh
appclient.sh         installation-manager.bat    jdr.sh           wsprovide.bat
client               installation-manager.properties launcher.jar wsprovide.ps1
common.bat           installation-manager.ps1    product.conf     wsprovide.sh
common.ps1           installation-manager.sh     standalone.bat
common.sh            jboss-cli-logging.properties standalone.conf
ubuntu@ip-172-31-30-191:~/wildfly-36.0.1.Final/bin$ ./standalone.sh -b=0.0.0.0
```

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



9. Access the Wildfly web interface on the browser using the public IP of the Deploy Server.

10. 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

```
ubuntu@ip-172-31-26-192:~$ cd .ssh
ubuntu@ip-172-31-26-192:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCzQsxyDNTQ7yykMQHy7qINZ6+GltiDwSOgA87KKNgy62S0LrxEEGFBbq1hmP8dSivUk0H7
DXR06VXuXS9alxxrjxbNsBlsFu7jbc58DsgxjVzSiDf5f/Ij8ef3FWpXJOogpV0QG/wZ5SsZ+NySdtqyIQeMVcKNbCkPrfchfV6NK8ag6QdY
dPtWCDvLSseLB8TKUWqbpgJQTTSXq/bGCzPNradeEfjS4w+mWWPQreaqJAQDx79zgb2Zut3XwVVHd82Hy065HoqvOccLpBV+YW1gmFNoxOUv
TYfsBRbei6H+hHrpyj3PSIVYORJ8vbcS4El2Vsrztd7d2GkS/BFu7/SlL9MqedhA5OMqB+6zyb1ZO2oFpl+KZOo6tgZjUYuv0IiJZ0ZYBbZh
AdcIbxwskV1/+vE65ULjYaXz8Ng62CZDZN67w88k5YasLSnfu0QzCrGk0EQfdwbRNAs+XT/ixKHh45Huh6UZwyfbLdYVyFG4xQinrXDqnkhc
rMyTOGwWjH8= ubuntu@ip-172-31-26-192
```

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

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDuva0VJof0Vxs990rNkqWoMX3ewX9ly4s8BqCT4TzmPmyA5Ho8+XAFjym5OzBOz8HFP5qH
C50Q+Dseg9UvI2Pb8GsEhMZp75BRCoJHo5n9fyAbBANqlwHE9TR0xt717v0NnXXbIFMGHr34sMEyCKiIJJ1rq71Gom9c/0b+RjK/6xE83c3g
8ODtKYI1JrERI5EQdrHZ3vkM3ls3+IkE3x0IOrL7HnxiBcZFRCTZvim48dUFUBQURHKz0j8tDPWl5+H/lvIggKcvKLUbFKUPd0O+GaWIIUkZ
uhOApCaCxWvgSPl9haLDQhX+wJrnRTpIUhuL+9tX2nqqxHXlznU3Pxs5 Tomcat-server

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCzQsxyDNTQ7yykMQHy7qINZ6+GltiDwSOgA87KKNgy62S0LrxEEGFBbq1hmP8dSivUk0H7
DXR06VXuXS9alxxrjxbNsBlsFu7jbc58DsgxjVzSiDf5f/Ij8ef3FWpXJOogpV0QG/wZ5SsZ+NySdtqyIQeMVcKNbCkPrfchfV6NK8ag6QdY
dPtWCDvLSseLB8TKUWqbpgJQTTSXq/bGCzPNradeEfjS4w+mWWPQreaqJAQDx79zgb2Zut3XwVVHd82Hy065HoqvOccLpBV+YW1gmFNoxOUv
TYfsBRbei6H+hHrpyj3PSIVYORJ8vbcS4El2Vsrztd7d2GkS/BFu7/SlL9MqedhA5OMqB+6zyb1ZO2oFpl+KZOo6tgZjUYuv0IiJZ0ZYBbZh
AdcIbxwskV1/+vE65ULjYaXz8Ng62CZDZN67w88k5YasLSnfu0QzCrGk0EQfdwbRNAs+XT/ixKHh45Huh6UZwyfbLdYVyFG4xQinrXDqnkhc
rMyTOGwWjH8= ubuntu@ip-172-31-26-192
```

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

```
ubuntu@ip-172-31-26-192:~/.ssh$ scp /home/ubuntu/devsecopstasks/java_code/target/TrainBook-1.0.0-SNAPSHOT.war ubuntu@23.22.131.22:/home/ubuntu/wildfly-20.0.1.Final/standalone/deployments
```

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

```
ubuntu@ip-172-31-29-247:~/wildfly-20.0.1.Final/standalone/deployments$ ls
README.txt  TrainBook-1.0.0-SNAPSHOT.war  TrainBook-1.0.0-SNAPSHOT.war.deployed
ubuntu@ip-172-31-29-247:~/wildfly-20.0.1.Final/standalone/deployments$
```

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