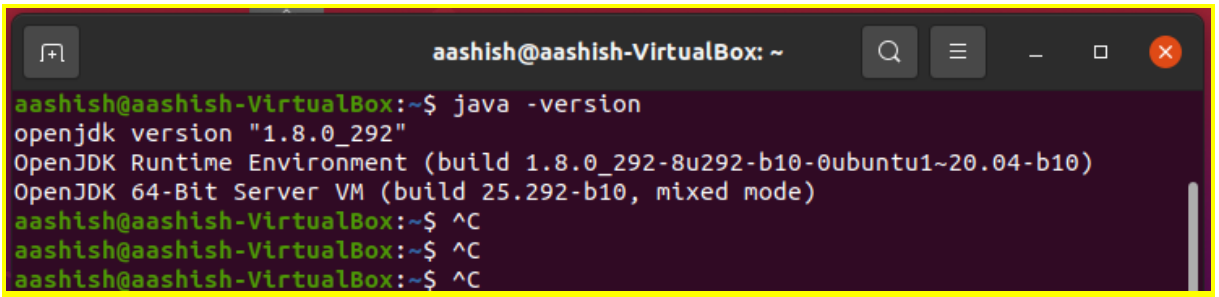
1. Glassfish

- **Install Glassfish server and change HTTP port to 8088.**
- **Create a demo Java (11) servlet application with maven.**
- **Generate war package.**
- **Deploy the war using glassfish app server.**

Answer:

To install Glassfish server, first of all we check either java is installed in the system or not, we use the following command to check;

- **java -version**
- **javac**



If it is not installed we use following command to download it;

- **sudo apt-get install openjdk-8-jre**

Now, we need to locate Java's installation directory. For that we need to edit **~/.profile** or **~/.bashrc** file and export the JAVA_HOME variable and append its path to the bin directory to our PATH variable.

We can install the Glassfish server now. It is downloaded from the site;

- **https://download.oracle.com/glassfish/4.1.1/release/glassfish-4.1.1.zip**

The downloaded file is unzipped.
Now, to start Glassfish server we need to be on Glassfish server extracted directory and we use the following commands;

- **cd glassfish4/bin**
- **sudo ./asadmin start-domain domain1**

```
aashish@aashish-VirtualBox: ~/Downloads/glassfish4/bin
Firefox Web Browser
aashish@aashish-VirtualBox:~/Downloads/glassfish4/bin$ sudo ./asadmin start-dom
ain domain1
[sudo] password for aashish:
Waiting for domain1 to start ..............
Successfully started the domain : domain1
domain  Location: /home/aashish/Downloads/glassfish4/glassfish/domains/domain1
Log File: /home/aashish/Downloads/glassfish4/glassfish/domains/domain1/logs/ser
ver.log
Admin Port: 4848
Command start-domain executed successfully.
aashish@aashish-VirtualBox:~/Downloads/glassfish4/bin$
```

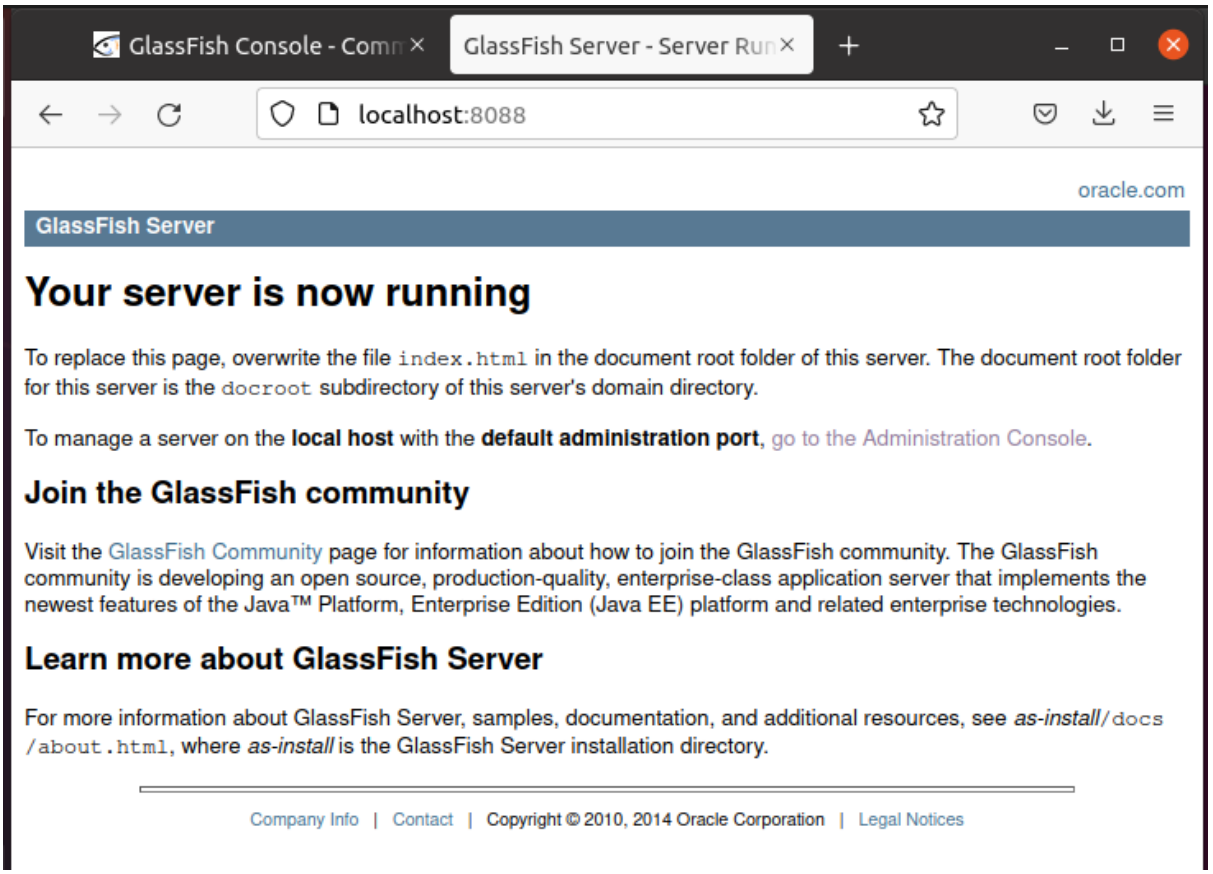
Now to change the HTTP port to **8088** of the Glassfish server we need to edit the domain.xml file.
We use following command to edit the domain.xml file;

- **sudo nano glassfish4/glassfish/domains/domain1/config/domain.xml**

```
aashish@aashish-VirtualBox: ~/Downloads/glassfish4/glass...
GNU nano 4.8                    domain.xml                    Modified
          </http>
        </protocol>
        <protocol name="http-listener-2" security-enabled="true">
          <http max-connections="250" default-virtual-server="server">
            <file-cache></file-cache>
          </http>
          <ssl classname="com.sun.enterprise.security.ssl.GlassfishSSLImpl" >
        </protocol>
        <protocol name="admin-listener">
          <http encoded-slash-enabled="true" max-connections="250" default-v>
            <file-cache></file-cache>
          </http>
        </protocol>
      </protocols>
      <network-listeners>
        <network-listener protocol="http-listener-1" port="8088" name="http->
        <network-listener protocol="http-listener-2" port="8181" name="http->
        <network-listener protocol="admin-listener" port="4848" name="admin->
      </network-listeners>
      <transports>
        <transport name="tcp"></transport>
      </transports>
    </network-config>
    <thread-pools>
      <thread-pool name="admin-thread-pool" max-queue-size="256" max-thread->
```

Then, in the network-listener divison http-listener-1 port is changed to **8088**.

To verify that the server is running at port **8088** we run **localhost:8088** on the web browser as follows;



Now, a demo Java servlet application with maven using IntelliJ IDEA is created.
For that, we check that Maven is installed or not using following command;

- **maven -version**

```
aashish@aashish-VirtualBox:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 1.8.0_292, vendor: Private Build, runtime: /usr/lib/jvm/java-8-op
enjdk-amd64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.11.0-40-generic", arch: "amd64", family: "unix"
aashish@aashish-VirtualBox:~$
```

If it is not installed we use following command to download it;

- **sudo apt install maven**

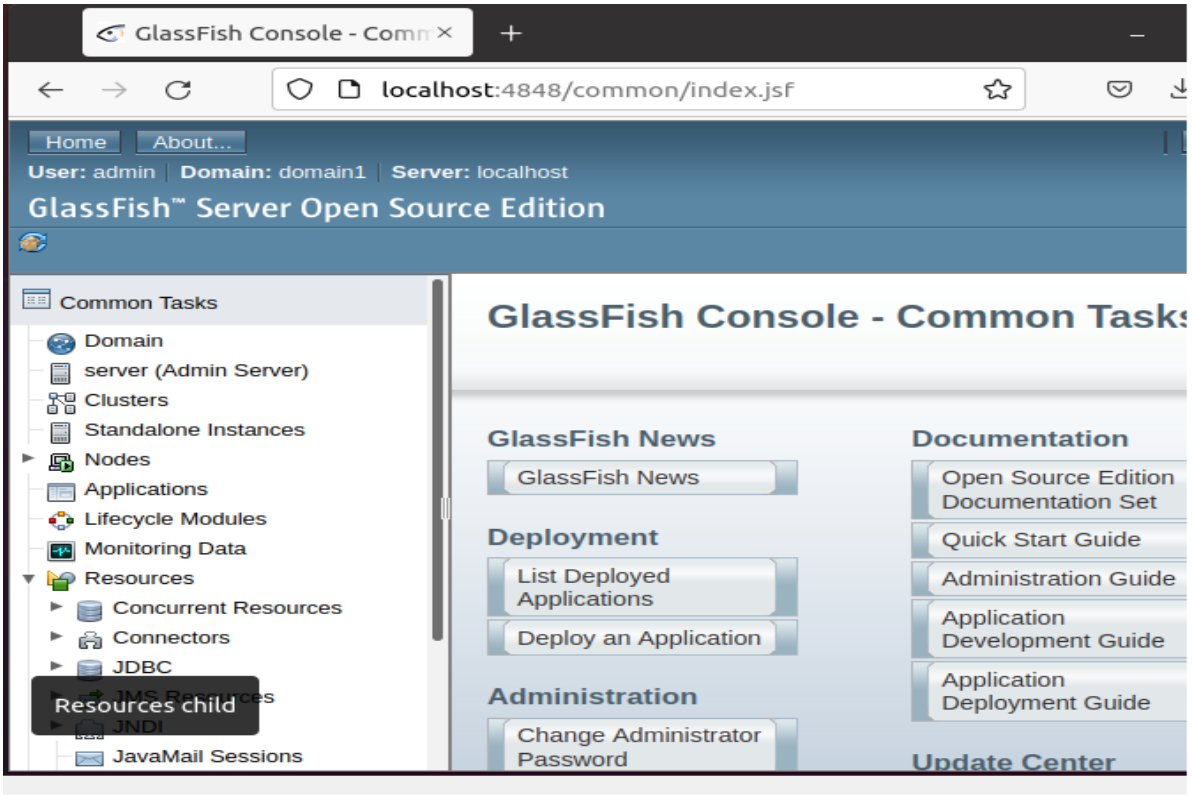
```
aashish@aashish-VirtualBox: ~
aashish@aashish-VirtualBox:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java
  libcdi-api-java 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:
  libaopalliance-java-doc libatinject-jsr330-api-java-doc libservlet3.1-java
  libcommons-io-java-doc libcommons-lang3-java-doc libasm-java libcglib-java
  libjsr305-java-doc libmaven-shared-utils-java-doc liblogback-java
  libplexus-cipher-java-doc libplexus-classworlds-java-doc
  libplexus-sec-dispatcher-java-doc libplexus-utils2-java-doc junit4 testng
  libcommons-logging-java liblog4j1.2-java
The following NEW 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
```

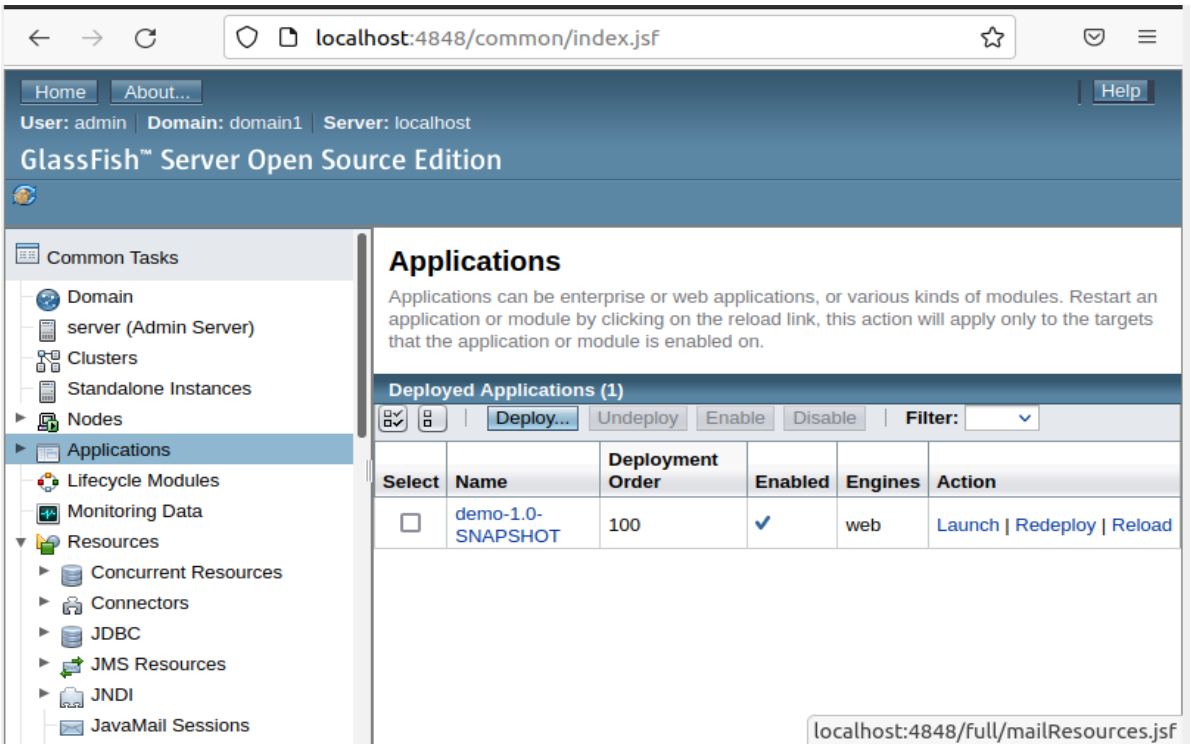
Then, the **pom.xml** file is edited to package the app into **war** file.

```
JavaProject – pom.xml (demo)
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JavaProject 〉 demo 〉 pom.xml        Add Configuration...
HelloServlet.java   index.jsp   pom.xml (demo)   demo.iml
.idea
demo
  .idea
  src
    main
      java
        com.exam
          HelloSe
      resources
      webapp
        WEB-INF
        index.jsp
    test
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>com.example</groupId>
8      <artifactId>demo</artifactId>
9      <version>1.0-SNAPSHOT</version>
10     <name>demo</name>
11     <packaging>war</packaging>
12
13     <properties>
```

Then, the **war** file is stored in the **target** folder of the project. And, we need to deploy that was file to the Glassfish server. For that, we open the Glassfish server admin page by simply typing **localhost:4848** on the browser.

After that, we click on the **application** and the **war** file is deployed on the server.





After that we launch the deployed war file to test whether the java servlet application is working or not.

Hence, the **war** file is deployed and is running successfully in the Glassfish server.