

BlackJack: Develop and Deploy Guidelines

Table of Contents

BlackJack: Develop and Deploy Guidelines	1
Overview	3
Software Download List	4
Installing JDK	5
Windows 7 – Setting Environment Variables	6
Setting Up JAVA_HOME, PATH, and CLASSPATH Environment Variables	6
Windows 10 – Setting Environment Variables	7
Setting Up JAVA_HOME, PATH, and CLASSPATH Environment Variables	7
Verifying the JDK Installation	9
Installing Netbeans.....	10
Verifying the Netbeans Installation.....	10
Installing GIT	11
Verifying the GIT Installation	12
Installing Maven	13
Windows 7 – Setting Environment Variables	14
Setting Up the M2_HOME, M2, and PATH Environment Variables	14
Windows 10 – Setting Environment Variables	15
Setting Up the M2_HOME, M2, and PATH Environment Variables	15
Verifying the Maven Installation	17
Creating a GIT Repository	18
Configuring a GIT Repository.....	18
Proxy Settings for Maven	19
Creating a Project with Maven Archetypes	20
Checking the Helloworld-Example Project into a GIT Repository.....	29
Creating a Developer Cloud Service Project.....	31
Activating Developer Cloud Service	31
Creating a GIT Repository in Developer Cloud Service	40
Cloning a GIT Repository.....	42
Downloading BlackJack Project from the Repository	44
Deploying the BlackJack Application on a Local Server.....	45
Testing the Locally Deployed BlackJack Application.....	49
Generating Application Archive Files for the BlackJack Application.....	51
Activating Oracle Application Container Cloud Service (OACCS).....	53
Deploying the BlackJack Application on OACCS.....	55
Testing the BlackJack Application Deployed on OACCS	60

Overview

This document helps students understand the process to setup the Java development environment on their computer, create project using “Maven” and “Netbeans” and deploy it on Developer Cloud Service.

Here students deploy their existing “Blackjack” project on a local Application Server, followed by deploying it on Oracle Application Cloud Container Service and finally accessing it from an HTML-5 client.

Important Note: Login credentials like Identity Domain Name, User Name and Password are required to work with Developer Cloud Service and Oracle Application Container Cloud Service. Gather this information from the email you have received from Oracle and keep it handy

Software Download List

Name and Version	Download Link
JDK 8 or higher	http://www.oracle.com/technetwork/java/javase/overview/index.html
Netbeans 8.1 or higher	https://netbeans.org/downloads/
GIT 2.11.0.3 or higher	https://git-scm.com/downloads
Maven 3.3.9 or higher	http://maven.apache.org/download.cgi

Note: List of software mentioned in the above table can be downloaded and stored onto your computer before you get started to save download time.

Or

Software can be downloaded as you go along with the exercises. Each exercise contains detailed steps for downloading and installing required software.

It is assumed that you will be working on a 64bit setup and provided instructions accordingly to download and install software. If you are not working on 64bit setup then download the software compatible to with your setup.

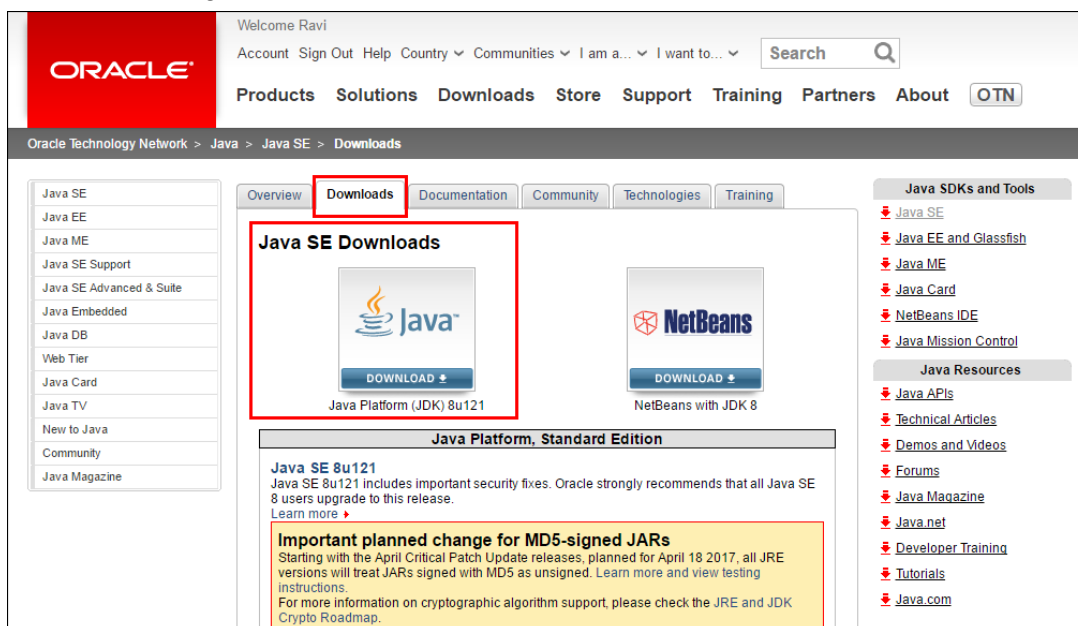
Installing JDK

Use the following instructions to download, install, and configure Java Development Kit on your computer.

Note: **JDK-8U121** is the latest version of JDK available at the time of creating this document. It is highly recommended that you download the newer version of JDK (if available) and perform these lab activities.

If you already have the JDK 8 or higher version installed on your computer then skip **Installing JDK** step and proceed with **Setting Up JAVA_HOME, PATH, and CLASSPATH Environment Variables** to setup/verify environment variables.

1. In the Firefox browser, navigate to <http://www.oracle.com/technetwork/java/javase/overview/index.html>
2. Click the **Downloads** tab and download the latest version of JDK available. In this case, we are downloading JDK-8U121.



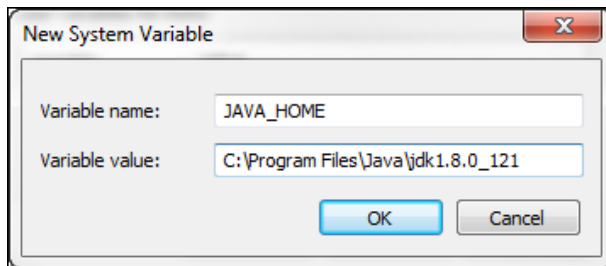
3. You must accept the “Oracle Binary Code License Agreement for Java SE” to download the software. Click the **Accept License Agreement** button.
4. Download the **jdk-8u121-windows-x64.exe** installer file on to your computer. The download may take some time. Wait for the download to complete before proceeding to the next step.
5. Double-click the **jdk-8u121-windows-x64.exe** file to start the installation.
Note: If you receive a security warning such as “Do you want to allow the following program to make changes to this computer?” click **Yes**.
6. When the installer opens, click the **Next** button.
7. Accept the default installation locations and click **Next** twice.
8. Wait until the installer installs the JDK successfully and displays a “Java SE Development Kit 8 Update 121 (64-bit)” message. Click the **Close** button.

Windows 7 – Setting Environment Variables

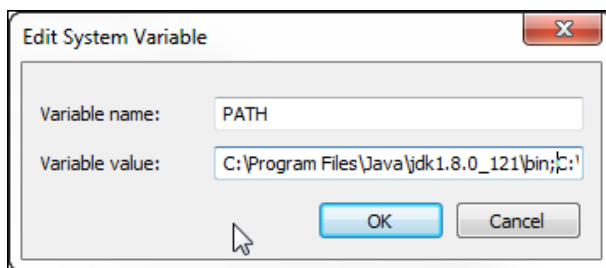
Setting Up JAVA_HOME, PATH, and CLASSPATH Environment Variables

Note: You must be logged on to your computer as the Admin user.

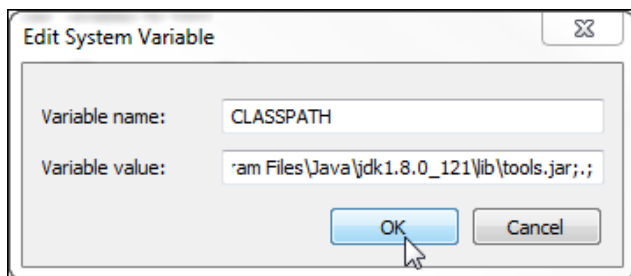
9. Click the Windows **Start** button. Right-click **Computer** and select **Properties**. Click **Advanced system settings**.
10. Click **Environment Variables**.
11. In the Environment Variables window, under **System Variables**, click the **New** button.
12. In the New System Variable window, enter the Variable name **JAVA_HOME**, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121**, and then click the **OK** button.



13. Select **PATH** system variable and click the **Edit** button (If PATH system variable is not available, click the **New** button to create PATH variable, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121**, and then click the **OK** button).
14. In the Edit System Variable window, in the Variable value field, place the cursor at the starting position and enter **C:\Program Files\Java\jdk1.8.0_121\bin**; Then click the **OK** button.



15. Click the **New** button to create another System Variable.
16. In the New System Variable window, enter the Variable name **CLASSPATH**, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121\lib\tools.jar;.** (this has a semicolon, a period, and a semicolon at the end), and then click the **OK** button.



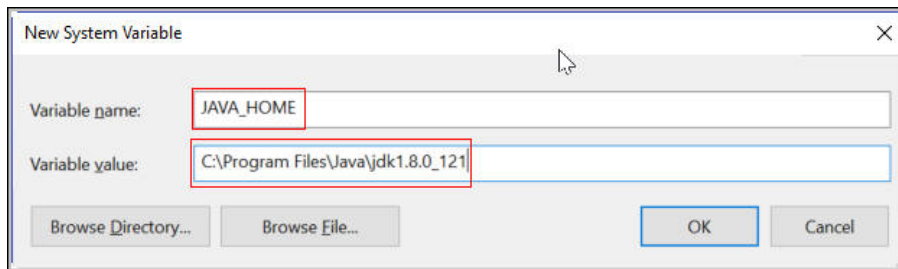
17. You have created/updated three system variables. Click the **OK** button to close the Environment Variables and System Properties windows. Close the Control Panel window.

Windows 10 – Setting Environment Variables

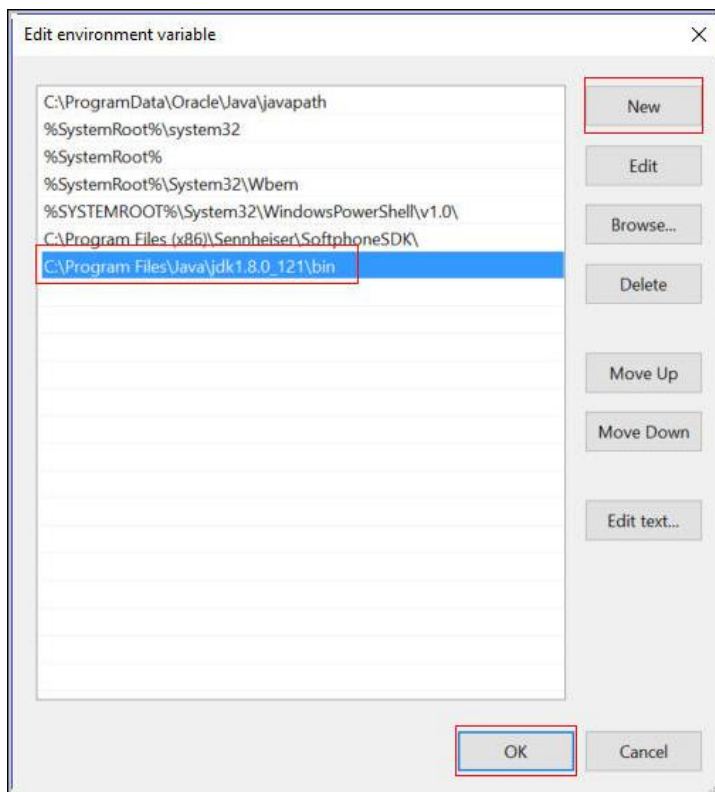
Setting Up JAVA_HOME, PATH, and CLASSPATH Environment Variables

Note: You must be logged on to your computer as the Admin user.

1. In Windows Desktop, right-click on **This PC** and select **Properties**. Click **Advanced system settings**.
2. Click **Environment Variables**.
3. In the Environment Variables window, under **System Variables**, click the **New** button.
4. In the New System Variable window, enter the Variable name **JAVA_HOME**, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121** and then click the **OK** button.



5. Select **PATH** system variable and click the **Edit** button (If PATH system variable is not available, click the **New** button to create PATH variable, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121**; and then click the **OK** button).
6. In the Edit Environment Variable window, click **New** button and enter **C:\Program Files\Java\jdk1.8.0_121\bin** then click the **OK** button.



7. Click the **New** button to create another System Variable.

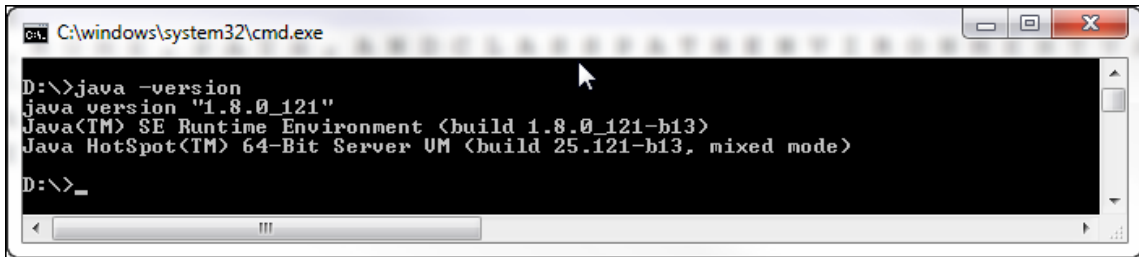
8. In the New System Variable window, enter the Variable name **CLASSPATH**, enter the Variable value **C:\Program Files\Java\jdk1.8.0_121\lib\tools.jar,;** (this has a semicolon, a period, and a semicolon at the end), and then click the **OK** button.



9. You have created/updated three system variables. Click the **OK** button to close the Environment Variables and System Properties windows.

Verifying the JDK Installation

1. **Verify the Java version:** Open a Command Prompt window and run the `java -version` command. This verifies that a JRE is installed but does not verify that the JDK is installed. Verify that the output of the `java -version` command shows “1.8.0_121” or higher.



```
C:\windows\system32\cmd.exe

D:\>java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)

D:\>_
```

Installing Netbeans

Use the following instructions to download, install, and configure Netbeans IDE on your computer.

Note: **Netbeans 8.1** is the latest version available at the time of creating this document. It is highly recommended that you download the newer version of the IDE (if available) and perform these lab activities.

If you already have the Netbeans 8.1 or higher version installed on your computer then skip **Installing Netbeans** step and proceed with **Verifying the Netbeans Installation** step.

1. In the Firefox browser, navigate to <https://netbeans.org/downloads/>.
2. Download Netbeans 8.1 version which supports **All** technologies from the last column.
3. Download the **netbeans-8.1-windows.exe** installer file on to your computer. The download may take some time. Wait for the download to complete before proceeding to the next step.
4. Double-click the **netbeans-8.1-windows.exe** file to start the installation.
Note: If you receive a security warning such as “Do you want to allow the following program to make changes to this computer?” click **Yes**.
5. When the installer opens, click the **Customize...** button, click the check box to select **Apache Tomcat 8.0.27** under the **Runtimes** section, and click the **OK** button.
6. Click the **Next** button on the Welcome screen to proceed with the installation.
7. Accept the terms in the license agreement and click the **Next** button.
8. Accept the default **Install the NetBeans IDE to:** path for NetBeans, make sure the correct installation path of JDK (jdk1.8.0_121) is selected in the **JDK for the NetBeans IDE:** field, and click the **Next** button.
9. Accept the default installation path for **Glassfish** and **Apache Tomcat** and click the **Next** button. Click the **Install** button on the Summary window.
10. Wait until the installer installs the Netbeans and displays a “**Setup Complete**” message. Click the **Finish** button.

Verifying the Netbeans Installation

1. **Verify Netbeans:** To start the Netbeans IDE and verify the version number of the JDK used by the IDE, double-click the **Netbeans 8.1** shortcut on the desktop. Netbeans opens to a “Start Page.” Open the **Help** menu and select **About**. The Netbeans and Java versions should be **Netbeans IDE 8.1** and **Java 1.8.0_121**. When done, **Close** the **About** window.

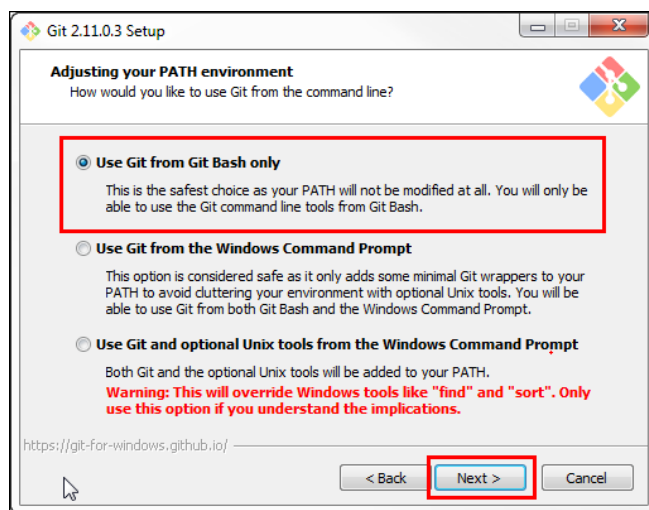
Installing GIT

Use the following instructions to download, install, and configure GIT Tool on your computer.

Note: **GIT 2.11.0.3** is the latest version of the tool available at the time of creating this document. It is highly recommended that you download the newer version of this tool (if available) and perform these lab activities.

If you already have the **GIT 2.11.0.3** or higher version installed on your computer then skip **Installing GIT** step and proceed with **Verifying the GIT Installation** step.

1. In the Firefox browser, navigate to <https://git-scm.com/downloads> and click the **Downloads for Windows** button.
2. Download the **Git-2.11.0.3-64-bit.exe** installer file to your computer. The download may take some time. Wait for the download to complete before proceeding to the next step.
3. Double-click the **Git-2.11.0.3-64-bit.exe** file to start the installation.
Note: If you receive a security warning such as “Do you want to allow the following program to make changes to this computer?” click **Yes**.
4. When the installer opens, click the **Next** button.
5. Accept the default installation path for **GIT** and click the **Next** button.
6. Accept the default selection on the **Select Components** screen and click the **Next** button.
7. Accept the default value on the **Select Start Menu Folder** screen and click the **Next** button.
8. Select the **Use Git from Git Bash only** option on the **Adjusting your PATH environment** screen and click the **Next** button.



9. Accept the default selection on the **Configuring the line ending conversions** screen and click the **Next** button.
10. Accept the default selection on the **Configuring the terminal emulator to use with Git Bash** screen and click the **Next** button.
11. Accept the default selection on the **Configuring extra options** screen and click the **Next** button.

12. Accept the default selection on the **Configuring experimental options** screen and click the **Install** button. Wait until the installer installs the **Git 2.11.0.3** and displays a “**Setup has finished installing Git on your computer**” message. Click the **Finish** button.

Verifying the GIT Installation

1. **Verify GIT:** Open **Git Bash** from the Windows **Start** menu and run the `git --version` command. Verify that the output of the `git --version` command shows “git version 2.11.0.windows.3.”



```
MINGW64:/c/Users/RAVI
$ git --version
git version 2.11.0.windows.3
$
```

Installing Maven

Use the following instructions to download, install, and configure Maven on your computer.

Note: Maven 3.3.9 is the latest version of the tool available at the time of creating this document. It is highly recommended that you download the newer version of this tool (if available) and perform these lab activities.

If you already have the Maven 3.3.9 or higher version installed on your computer then skip **Installing Maven** step and proceed with **Setting Up the M2_HOME, M2, and PATH Environment Variables** step to setup/verify the required environment variables.

1. In the Firefox browser, navigate to <http://maven.apache.org/download.cgi>.
2. Download the Binary ZIP archive, **apache-maven-3.3.9-bin.zip** file on to your computer. The download may take some time. Wait for the download to complete before proceeding to the next step.

Downloading Apache Maven 3.3.9

Apache Maven 3.3.9 is the latest release and recommended version for all users.

The currently selected download mirror is <http://redrockdigimark.com/apache-mirror/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are [backup mirrors](#) (at the end of the mirrors list) that should be available. You may also consult the [complete list of mirrors](#).

Other mirrors:

System Requirements

Java Development Kit (JDK)	Maven 3.3 requires JDK 1.7 or above to execute - it still allows you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

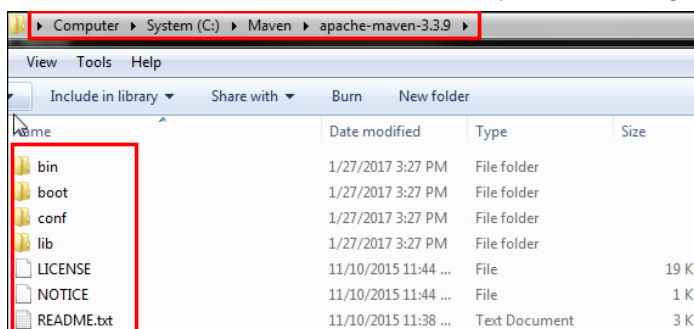
Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.9-bin.tar.gz	apache-maven-3.3.9-bin.tar.gz.md5	apache-maven-3.3.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.9-bin.zip	apache-maven-3.3.9-bin.zip.md5	apache-maven-3.3.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.9-src.tar.gz	apache-maven-3.3.9-src.tar.gz.md5	apache-maven-3.3.9-src.tar.gz.asc
Source zip archive	apache-maven-3.3.9-src.zip	apache-maven-3.3.9-src.zip.md5	apache-maven-3.3.9-src.zip.asc

3. Create a directory named **Maven** in C:\ and unzip the distribution archive to **C:\Maven** directory.

Note: You should achieve the directory structure highlighted in the screenshot



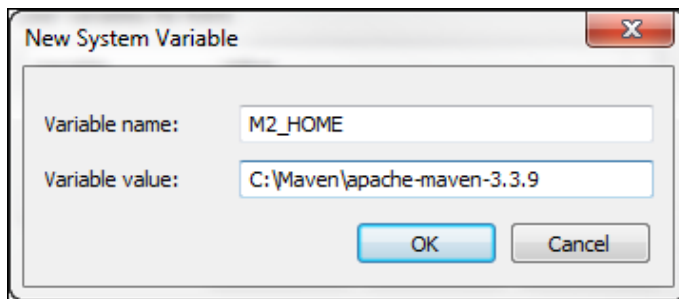
4. Copy the complete path (**C:\Maven\apache-maven-3.3.9**) once the extraction is completed; this is required to create environment variables.

Windows 7 – Setting Environment Variables

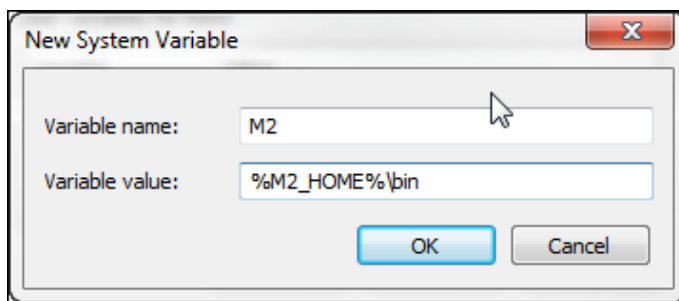
Setting Up the M2_HOME, M2, and PATH Environment Variables

Note: You must be logged on to your computer as the Admin user.

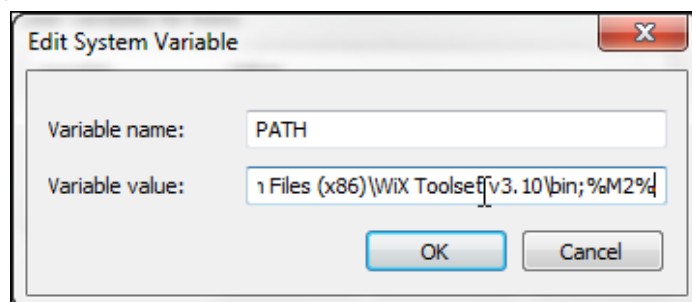
- Click the Windows **Start** button. Right-click **Computer** and select **Properties**. Click **Advanced system settings**.
- Click **Environment Variables**.
- In the Environment Variables window, under **System Variables**, click the **New** button.
- In the New System Variable window, enter the Variable name **M2_HOME**, enter the Variable value **C:\Maven\apache-maven-3.3.9**, and then click the **OK** button.



- In the Environment Variables window, under **System Variables**, click the **New** button. In the New System Variable window, enter the Variable name **M2**, enter the Variable value **%M2_HOME%\bin**, and then click the **OK** button.



- Select the **PATH** system variable and click the **Edit** button.
- In the Edit System Variable window, in the Variable value field, place the cursor at the last position and enter **;%M2%**, and then click the **OK** button.



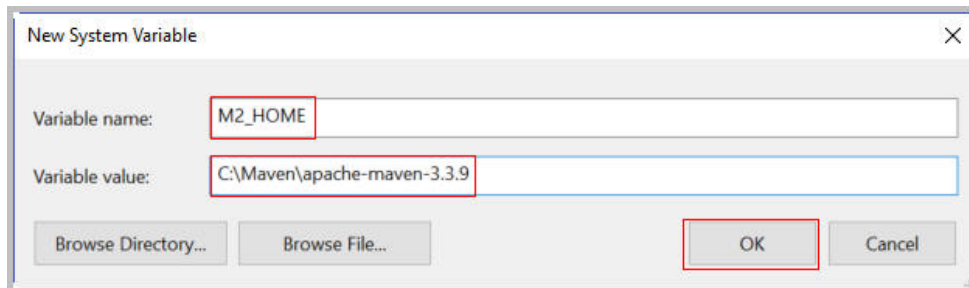
- Click **OK** button twice to close Edit System Variable and System Property windows

Windows 10 – Setting Environment Variables

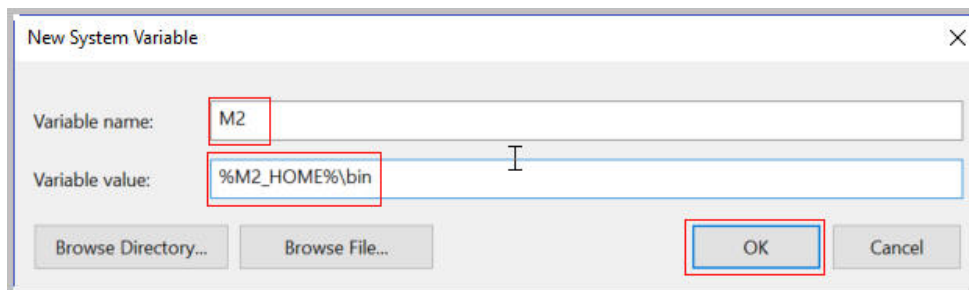
Setting Up the M2_HOME, M2, and PATH Environment Variables

Note: You must be logged on to your computer as the Admin user.

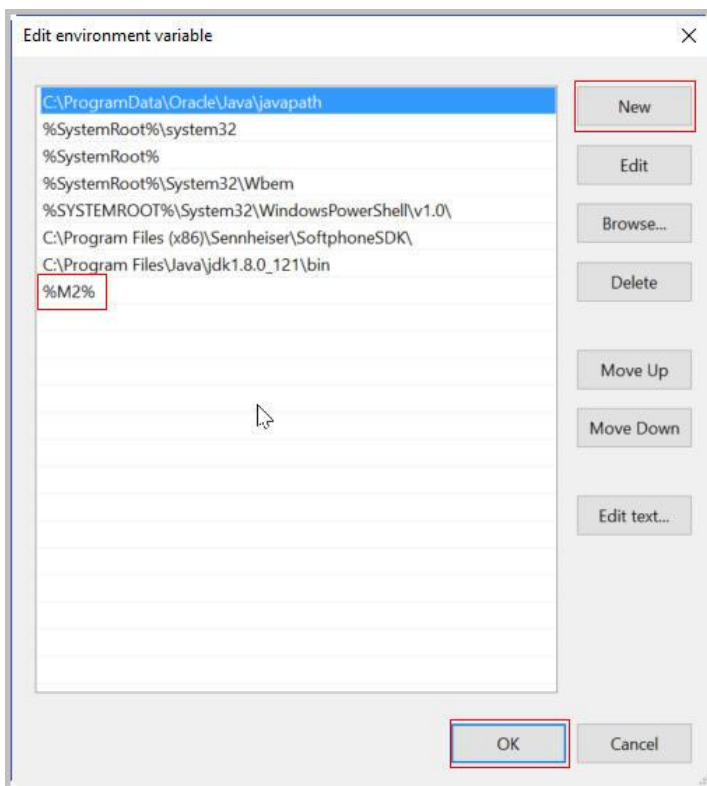
1. In Windows Desktop, right-click **This PC** and select **Properties**. Click **Advanced system settings**.
2. Click **Environment Variables**.
3. In the Environment Variables window, under **System Variables**, click the **New** button.
4. In the New System Variable window, enter the Variable name **M2_HOME**, enter the Variable value **C:\Maven\apache-maven-3.3.9** and then click the **OK** button.



5. In the Environment Variables window, under **System Variables**, click the **New** button. In the New System Variable window, enter the Variable name **M2**, enter the Variable value **%M2_HOME%\bin**, and then click the **OK** button.



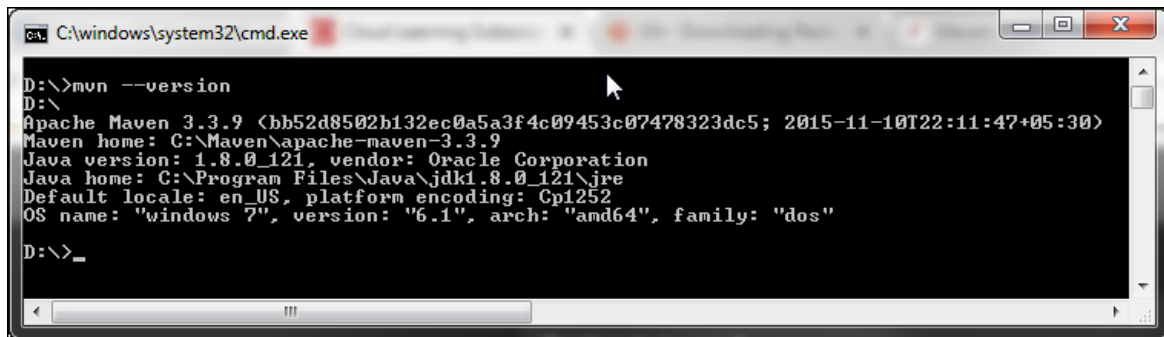
6. Select the **PATH** system variable and click the **Edit** button.
7. In the Edit System Variable window, in the Variable value field, place the cursor at the last position and enter **;%M2%**, and then click the **OK** button.



8. Click **OK** button twice to to close Edit System Variable and System Property windows

Verifying the Maven Installation

1. **Verify the Maven version:** Open a Command Prompt window and run the `mvn --version` command. Verify that the output of the `mvn --version` command matches with the following screenshot:



```

C:\windows\system32\cmd.exe
D:\>mvn --version
D:\
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T22:11:47+05:30)
Maven home: C:\Maven\apache-maven-3.3.9
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_121\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
D:\>_

```

Proxy Settings for Maven in Netbeans

Note: Use the following instructions to change your proxy settings for Maven if you are part of the secured network and behind a firewall **ONLY**.

2. Open the `C:\Program Files\NetBeans 8.1\java\maven\conf\settings.xml` file with a text editor like Notepad++.
3. Add the following lines under the `<proxies>` tag:

```

<proxy>

    <id>Oracle</id>

    <active>true</active>

    <protocol>http</protocol>

    <host>ENTER YOUR PROXY ADDRESS</host>

    <port>80</port>

    <nonProxyHosts>localhost|oracle.com</nonProxyHosts>

</proxy>

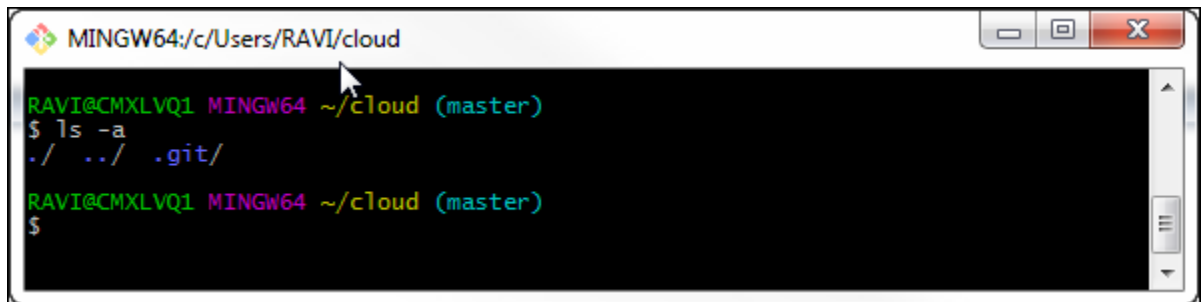
```

4. Replace **ENTER YOUR PROXY ADDRESS** within the `<host>` tag with your proxy and save the file.

Note: If you are facing problems in editing the `settings.xml` file, save a copy of the `settings.xml` file to some other location, modify it, and then put it back in to `C:\Program Files\NetBeans 8.1\java\maven\conf\` directory.

Creating a GIT Repository

1. Open Git Bash from the Windows **Start** menu.
2. In your home directory, create a **cloud** directory.
`mkdir cloud`
3. Change the directory to **cloud** directory.
`cd cloud`
4. Create a Git repository type.
`git init`
5. The cloud directory is now a Git repository. Execute the `ls -a` command to confirm the same. The output of the `ls -a` command must match the output in the following screenshot:



The screenshot shows a Git Bash terminal window with the title bar 'MINGW64:/c/Users/RAVI/cloud'. The terminal output is as follows:

```
RAVI@CMXLVQ1 MINGW64 ~/cloud (master)
$ ls -a
./ ../ .git/

RAVI@CMXLVQ1 MINGW64 ~/cloud (master)
$
```

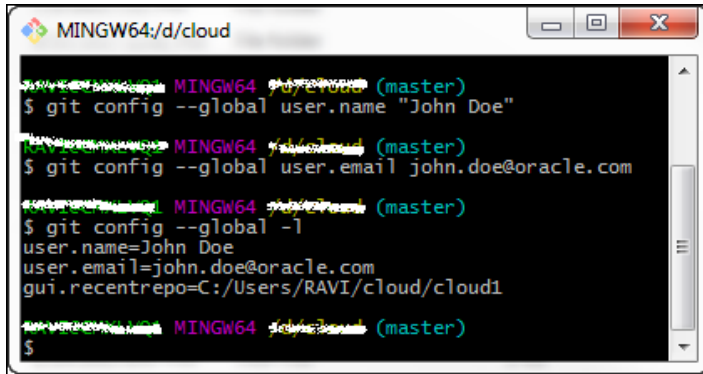
Note: Now you should see that a `.git` directory has been created inside the cloud directory and your repository is ready.

Configuring a GIT Repository

Before you commit changes to Git, you must configure your name and email address to identify your commits in the repository.

1. Execute the following commands to configure your name:
`git config --global user.name "Your Name"`
Example: `git config --global user.name "John Doe"`
2. Execute the following commands to configure your email address:
`git config --global user.email your-email@address`
Example: `git config --global user.email john.doe@oracle.com`
3. To confirm that the values have been set, execute the following command:
`git config --global -l`

The output of these commands should match the output in the following screenshot:



```
MINGW64/d/cloud
$ git config --global user.name "John Doe"
$ git config --global user.email john.doe@oracle.com
$ git config --global -l
user.name=John Doe
user.email=john.doe@oracle.com
gui.recentrepo=C:/Users/RAVI/cloud/cloud1
$
```

Note: This sets your name and email address for all Git projects on this system.

Proxy Settings for Maven

Note: Use the following instructions to change your proxy settings for Maven if you are part of the secured network and behind a firewall **ONLY**.

4. Open the C:\Maven\apache-maven-3.3.9\conf\settings.xml file with a text editor like Notepad++.
5. Add the following lines under the <proxies> tag:

```
<proxy>

    <id>Oracle</id>

    <active>true</active>

    <protocol>http</protocol>

    <host>ENTER YOUR PROXY ADDRESS</host>

    <port>80</port>

    <nonProxyHosts>localhost|oracle.com</nonProxyHosts>

</proxy>
```

6. Replace **ENTER YOUR PROXY ADDRESS** within the <host> tag with your proxy and save the file.

Creating a Project with Maven Archetypes

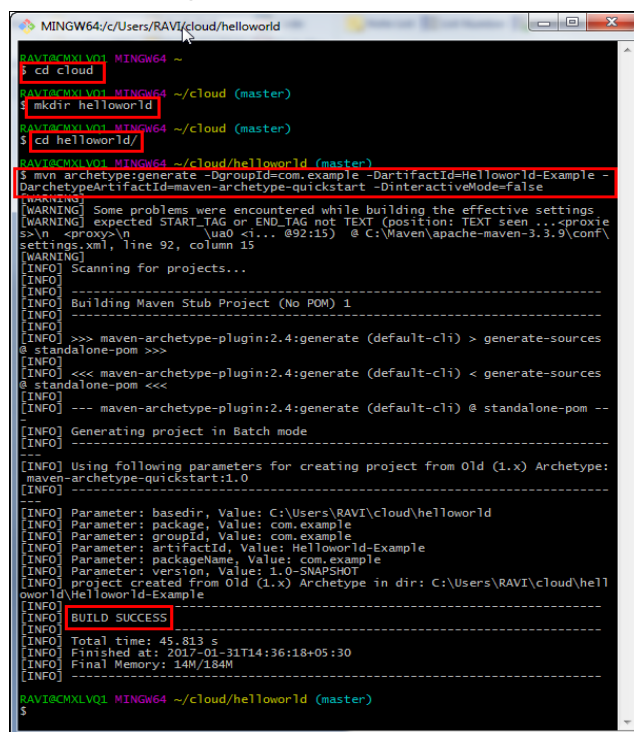
Use the following steps to create a Maven project using Archetypes from Git Bash or Command Prompt.

As part of this exercise, we are going to create a simple Maven project named, **HelloWorld-Example** to print “Hello World!” message on the console. This project will be used in the following practices to store in a local GIT repository, create project on Developer Cloud Service and then Clone it to cloud GIT repository.

1. Open Git Bash from the Windows Start menu.
2. Change to the `cloud` directory where your Git repository is stored.
`cd cloud`
3. Create a directory named `helloworld`.
`mkdir helloworld`
4. Change to the `helloworld` directory.
`cd helloworld`
5. Create an empty Maven project using the `maven-archetype-quickstart` archetype. Enter the following command:

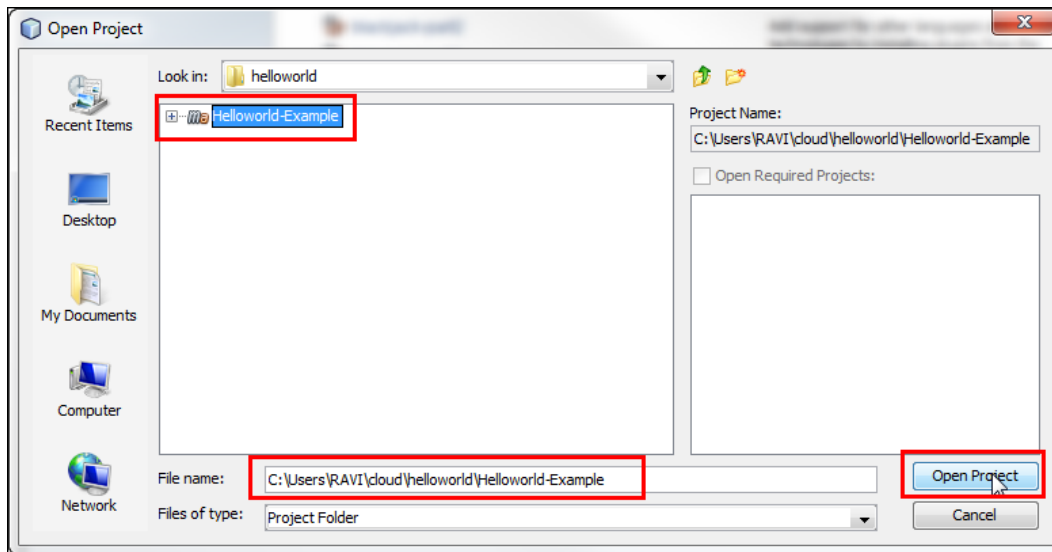
```
mvn archetype:generate -DgroupId=com.example -DartifactId=Helloworld-Example -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Note: The output of this command must be similar to the output in the following screenshot:

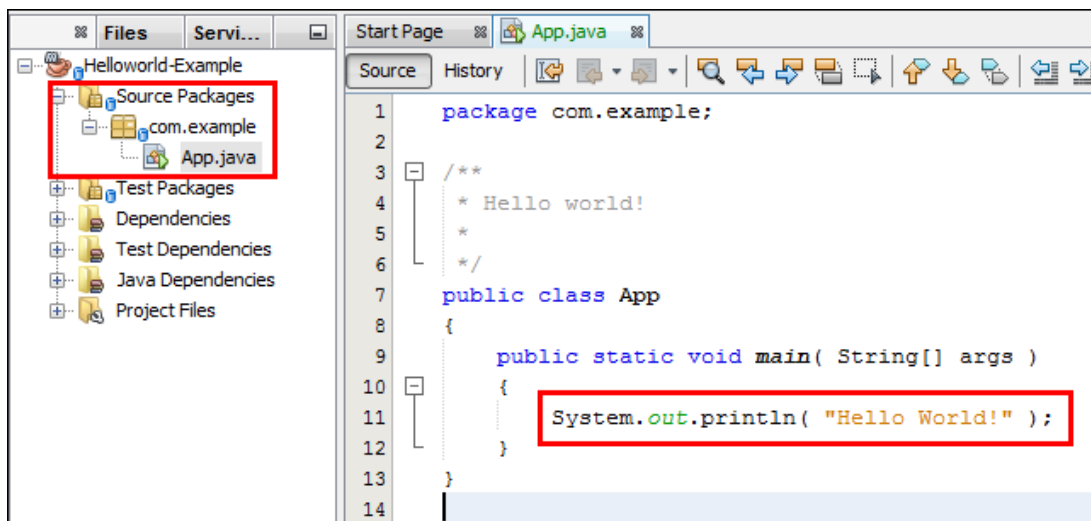


```
MINGW64/c/Users/RAVI/cloud/helloworld
RAVI@OMKLVQ1 MINGW64 ~
$ cd cloud
RAVI@OMKLVQ1 MINGW64 ~/cloud (master)
$ mkdir helloworld
RAVI@OMKLVQ1 MINGW64 ~/cloud (master)
$ cd helloworld/
RAVI@OMKLVQ1 MINGW64 ~/cloud/helloworld (master)
$ mvn archetype:generate -DgroupId=com.example -DartifactId=Helloworld-Example -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[WARNING] Some problems were encountered while building the effective settings
[WARNING] expected START_TAG or END_TAG not TEXT (position: TEXT seen ...<proxie
>>n <proxy>n      \ua0 <!-- 092:15) @ C:\Maven\apache-maven-3.3.9\conf\
settings.xml, line 92, column 15
[WARNING]
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----
[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources
@ standalone-pom >>>
[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources
@ standalone-pom <<<
[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO]
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: C:\Users\RAVI\cloud\helloworld
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: Helloworld-Example
[INFO] Parameter: packageName, Value: com.example
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Old (1.x) Archetype in dir: C:\Users\RAVI\cloud\hell
oworld\Helloworld-Example
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 45.813 s
[INFO] Finished at: 2017-01-31T14:36:18+05:30
[INFO] Final Memory: 14M/184M
[INFO]
RAVI@OMKLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

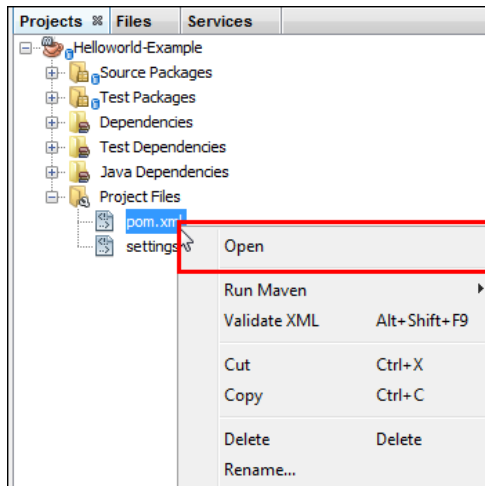
6. The command creates an empty Maven project named **Helloworld-Example**. Examine the directory structure and note that an executable class is located at `com.example.App`. Now the `pom.xml` file must be configured for plug-ins.
7. Launch Netbeans using the shortcut on the desktop.
8. Open the **Helloworld-Example** Maven project created under **cloud/helloworld** directory in Netbeans.



9. Examine the directory structure of the project, open **com.example.App** executable class, and review the code.



10. Right-click the **Project Files > pom.xml** file and click Open.



11. Add the following properties settings to the file before the dependencies section. This sets the Java version and encoding for the project.

```
<properties>
<java.version>1.8</java.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

12. After the dependencies element, add elements for build and plug-ins.

```
<build>
  <plugins>
    <!-- Your plugins go here -->
  </plugins>
</build>
```

13. Add the configuration for the compiler plug-in to the plug-ins section.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>2.3.2</version>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
```

14. Add the exec plug-in to the pom.xml file.

```

<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>1.4.0</version>
  <executions>
    <execution>
      <goals>
        <goal>exec</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <mainClass>com.example.App</mainClass>
  </configuration>
</plugin>

```

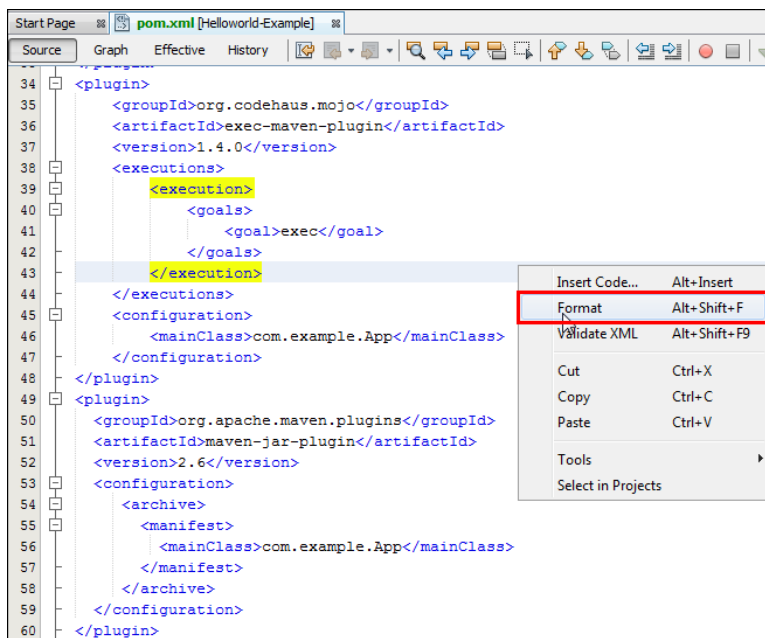
15. Add the JAR plug-in to the pom.xml file.

```

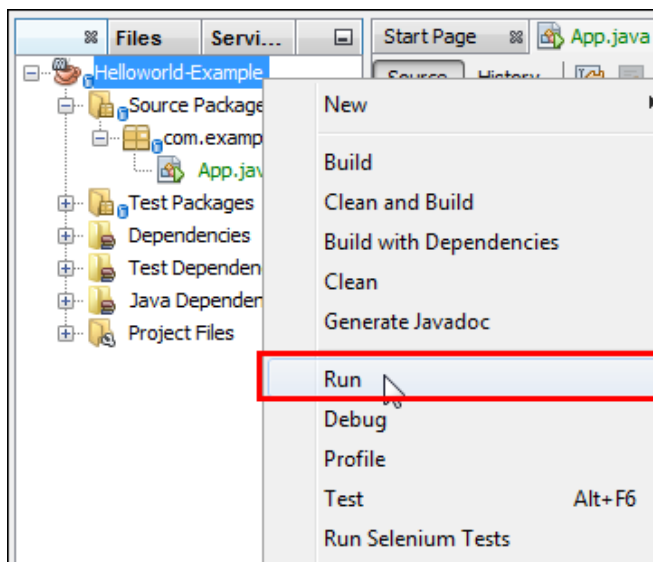
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>2.6</version>
  <configuration>
    <archive>
      <manifest>
        <mainClass>com.example.App</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>

```

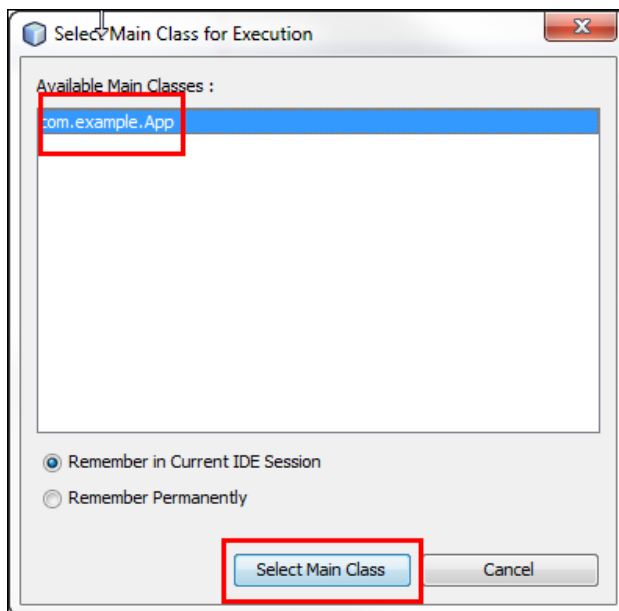
16. In the source window, right-click the **pom.xml** file and select **Format** to fix the indentation for the file.



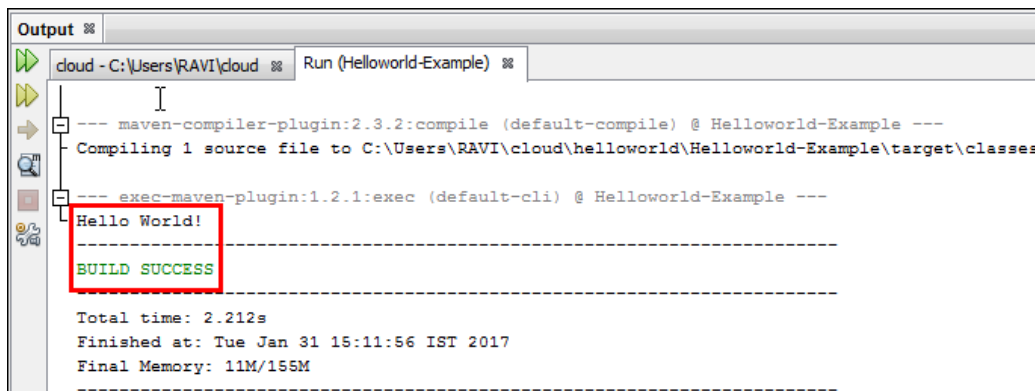
17. Save the **pom.xml** file.
18. Right-click the **Helloworld-Example** project and click **Run**.



19. Select **com.example.App** from the Available Main Classes list and click the **Select Main Class** button.

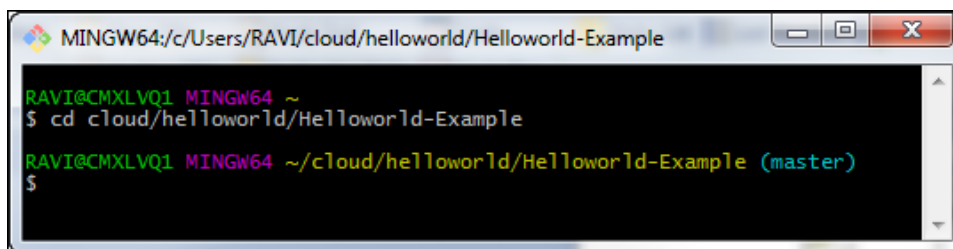


20. You should see **Hello World!** Output with a **BUILD SUCCESS** message.

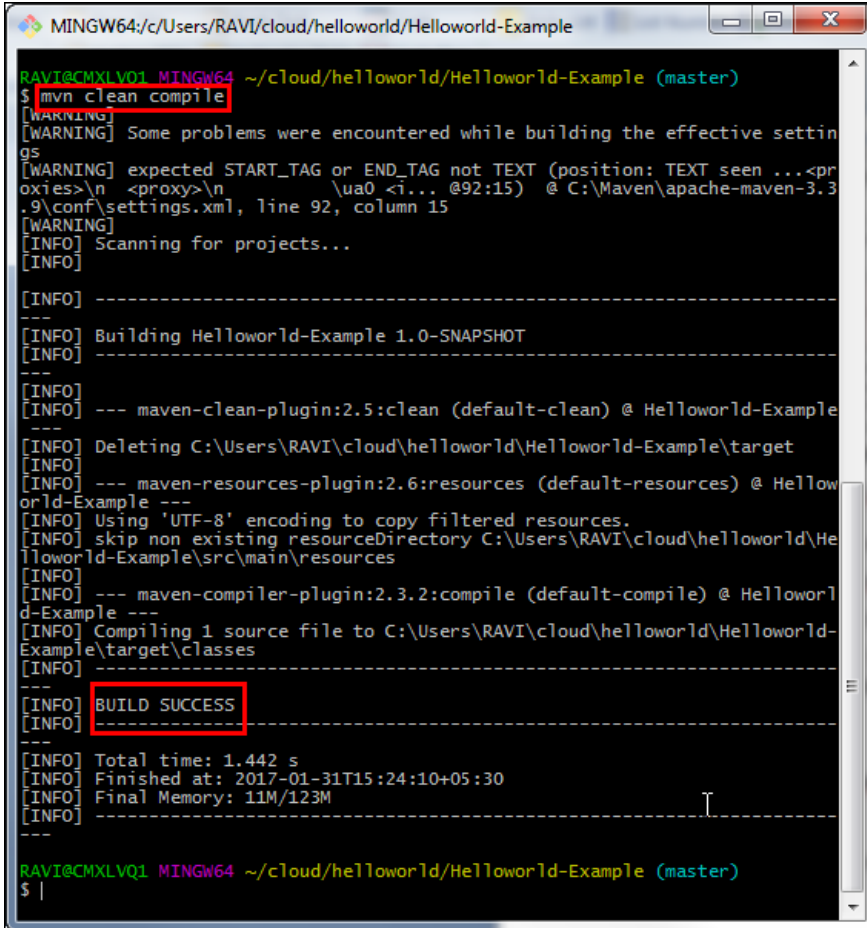


21. Switch to **Git Bash** and change the directory to Helloworld-Example.

```
cd Helloworld-Example
```

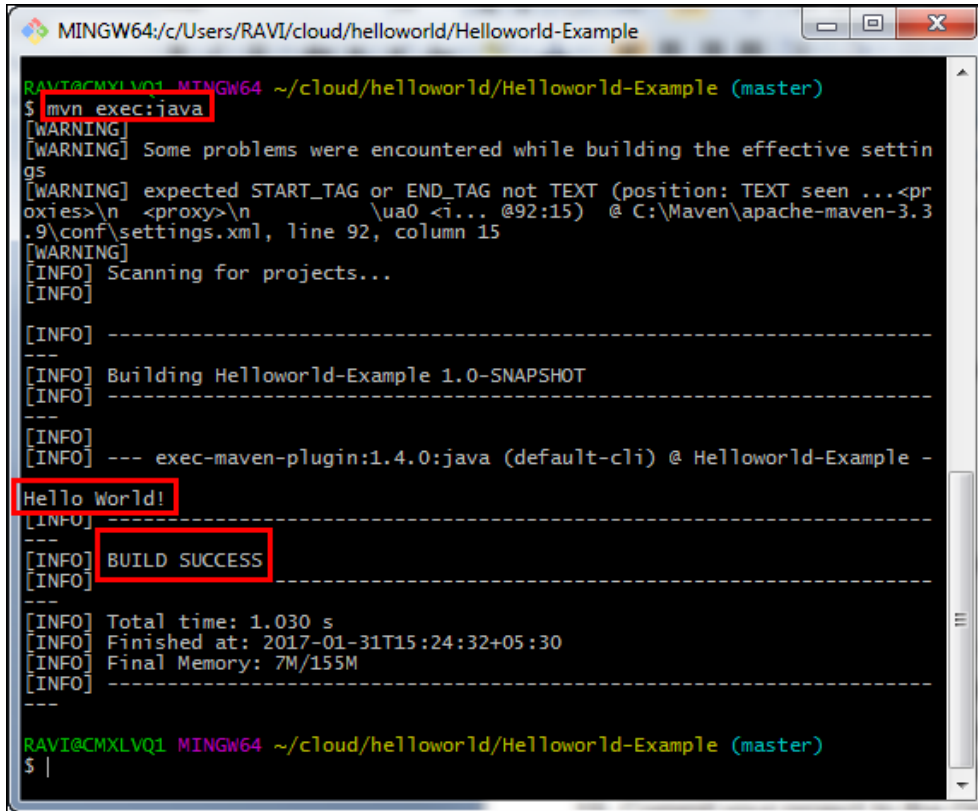


22. Execute the `mvn clean compile` command to clean and compile the project.



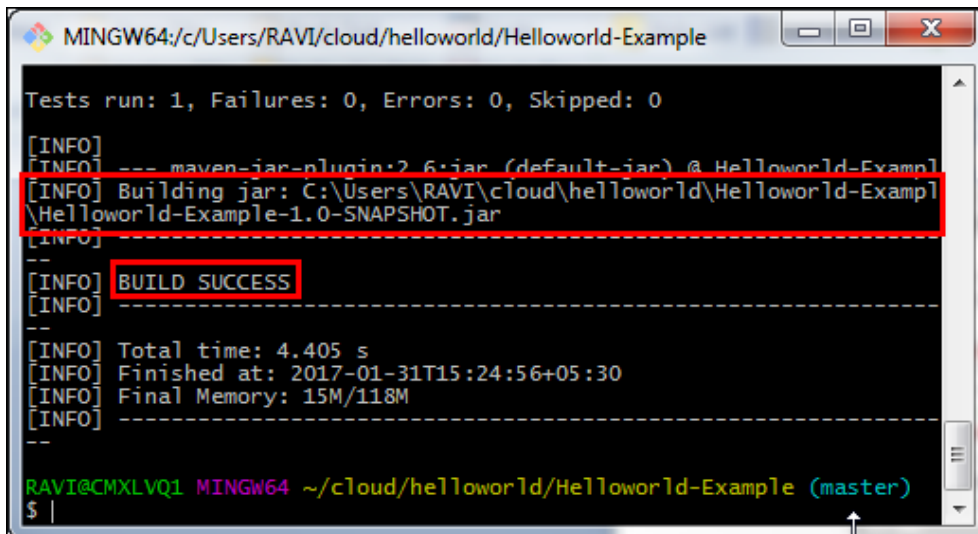
```
MINGW64: c:/Users/RAVI/cloud/helloworld/Helloworld-Example
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ mvn clean compile
[WARNING]
[WARNING] Some problems were encountered while building the effective settings
[WARNING] expected START_TAG or END_TAG not TEXT (position: TEXT seen ...<pr
oxies>\n <proxy>\n \ua0 <!-- @92:15) @ C:\Maven\apache-maven-3.3
.9\conf\settings.xml, line 92, column 15
[WARNING]
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Helloworld-Example 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Helloworld-Example ---
[INFO] Deleting C:\Users\RAVI\cloud\helloworld\Helloworld-Example\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Hellow
orld-Example ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\RAVI\cloud\helloworld\He
lloworld-Example\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ Helloworl
d-Example ---
[INFO] Compiling 1 source file to C:\Users\RAVI\cloud\helloworld\Helloworld-
Example\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 1.442 s
[INFO] Finished at: 2017-01-31T15:24:10+05:30
[INFO] Final Memory: 11M/123M
[INFO] -----
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ |
```

23. Execute the `mvn exec:java` command to execute the application.



```
MINGW64/c/Users/RAVI/cloud/helloworld/Helloworld-Example
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ mvn exec:java
[WARNING]
[WARNING] Some problems were encountered while building the effective settings
[WARNING] expected START_TAG or END_TAG not TEXT (position: TEXT seen ...<pr
oxies>\n <proxy>\n \ua0 <i... @92:15) @ C:\Maven\apache-maven-3.3
.9\conf\settings.xml, line 92, column 15
[WARNING]
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Helloworld-Example 1.0-SNAPSHOT
[INFO] -----
[INFO] --- exec-maven-plugin:1.4.0:java (default-cli) @ Helloworld-Example -
Hello World!
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 1.030 s
[INFO] Finished at: 2017-01-31T15:24:32+05:30
[INFO] Final Memory: 7M/155M
[INFO] -----
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ |
```

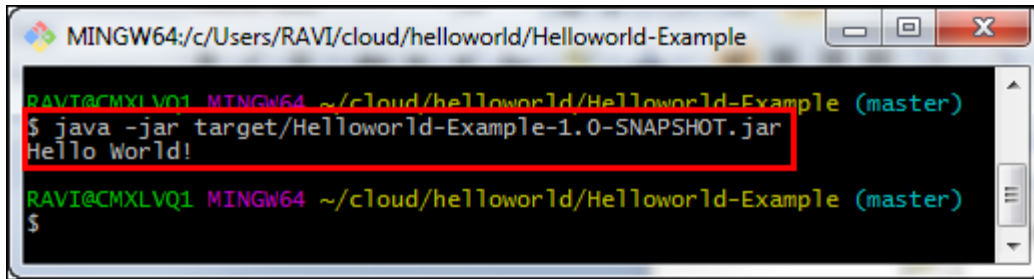
24. Execute the `mvn package` command to package the application.



```
MINGW64/c/Users/RAVI/cloud/helloworld/Helloworld-Example
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ Helloworld-Examp
[INFO] Building jar: C:\Users\RAVI\cloud\helloworld\Helloworld-Examp
\Helloworld-Example-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 4.405 s
[INFO] Finished at: 2017-01-31T15:24:56+05:30
[INFO] Final Memory: 15M/118M
[INFO] -----
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ |
```

Note: Examine the `Helloworld-Example-1.0-SNAPSHOT.jar` file created inside `cloud/helloworld/Helloworld-Example/target` directory.

25. Execute the `java -jar target/Helloworld-Example-1.0-SNAPSHOT.jar` command to run the packaged application.



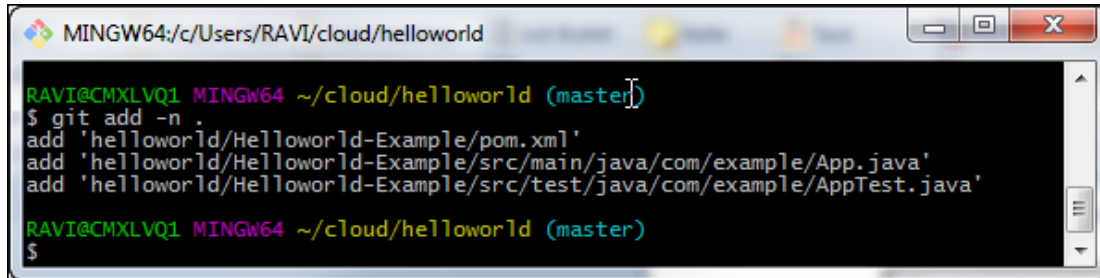
A screenshot of a MINGW64 terminal window. The title bar shows the path `MINGW64:/c/Users/RAVI/cloud/helloworld/Helloworld-Example`. The terminal content shows a user prompt `RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)`, followed by the command `$ java -jar target/Helloworld-Example-1.0-SNAPSHOT.jar` and its output `Hello World!`. The command and output are highlighted with a red rectangle. Below this, the prompt `RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)` and a new prompt `$` are visible.

```
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$ java -jar target/Helloworld-Example-1.0-SNAPSHOT.jar
Hello World!
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld/Helloworld-Example (master)
$
```

Checking the Helloworld-Example Project into a GIT Repository

Execute the following commands to commit the Helloworld-Example project to a GIT repository.

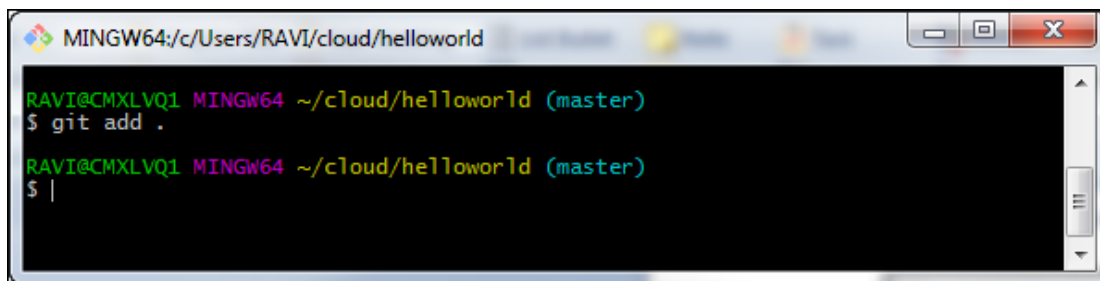
1. Change into the cloud/helloworld directory.
2. Execute the `git add -n .` command to see the list of files that are ready to be added to the repository.



```
MINGW64/c/Users/RAVI/cloud/helloworld
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ git add -n .
add 'helloworld/Helloworld-Example/pom.xml'
add 'helloworld/Helloworld-Example/src/main/java/com/example/App.java'
add 'helloworld/Helloworld-Example/src/test/java/com/example/AppTest.java'
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

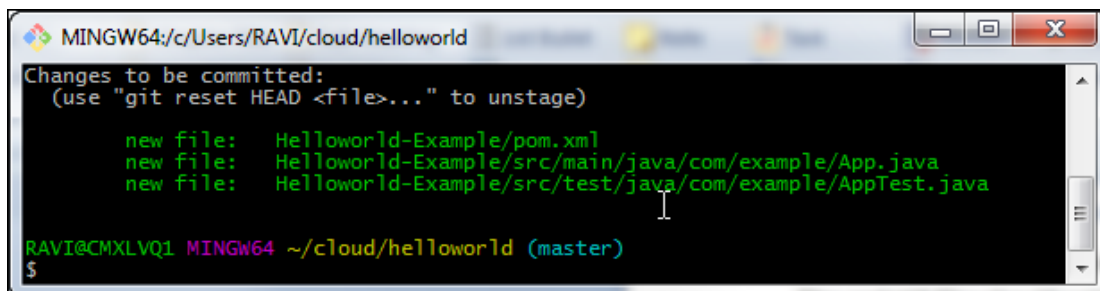
Note: Please notice that there is `.` at the end of the command.

3. Execute the `git add .` command to add the files to the repository.



```
MINGW64/c/Users/RAVI/cloud/helloworld
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ git add .
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ |
```

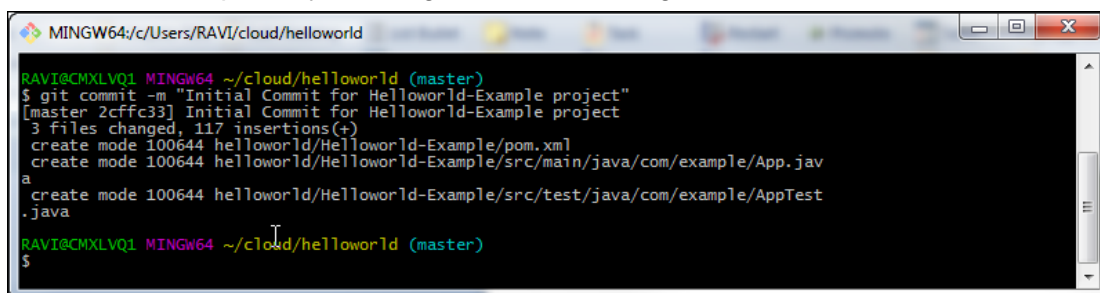
4. Execute the `git status` command to check the files that are added.



```
MINGW64/c/Users/RAVI/cloud/helloworld
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   Helloworld-Example/pom.xml
    new file:   Helloworld-Example/src/main/java/com/example/App.java
    new file:   Helloworld-Example/src/test/java/com/example/AppTest.java
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

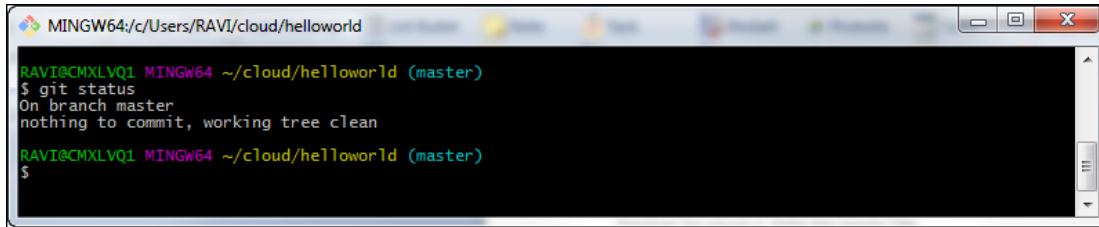
5. Execute the `git commit -m "Initial Commit for Helloworld-Example Project"` to commit the files to the repository and begin version tracking.



```
MINGW64/c/Users/RAVI/cloud/helloworld
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ git commit -m "Initial Commit for Helloworld-Example project"
[master 2cffc33] Initial Commit for Helloworld-Example project
3 files changed, 117 insertions(+)
create mode 100644 helloworld/Helloworld-Example/pom.xml
create mode 100644 helloworld/Helloworld-Example/src/main/java/com/example/App.java
create mode 100644 helloworld/Helloworld-Example/src/test/java/com/example/AppTest.java
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

6. Your files are now checked in for version tracking.

7. Check the status of the repository by executing the `git status` command.



```
MINGW64; c:/Users/RAVI/cloud/helloworld
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ git status
On branch master
nothing to commit, working tree clean
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

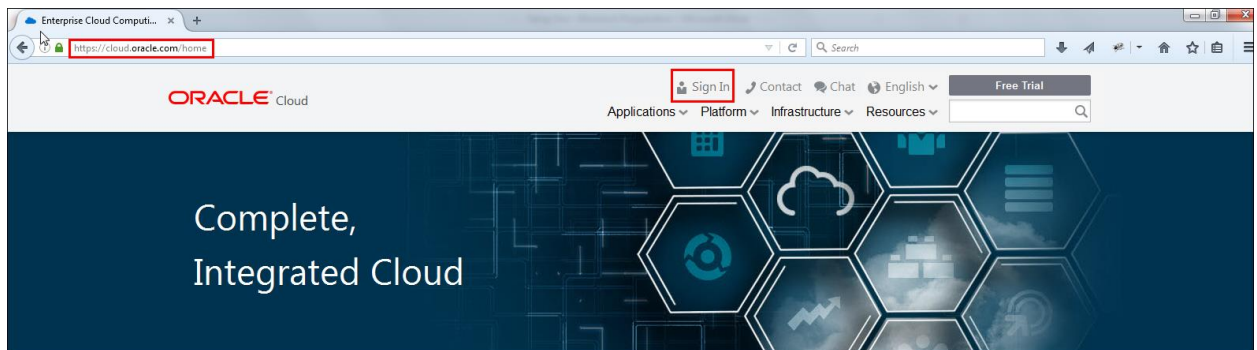
Note: You should get a response similar to the one in the screenshot.

Creating a Developer Cloud Service Project

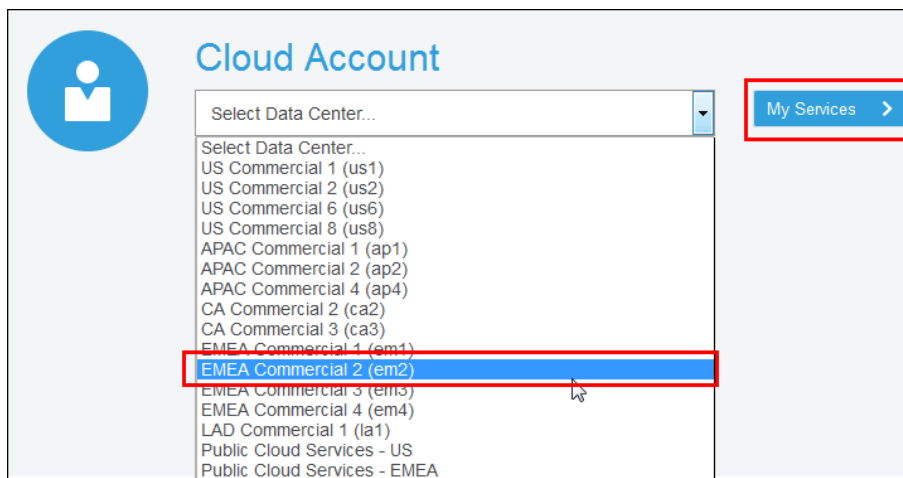
Selecting a Replication Policy for Your Service Instance

Note: The cloud login credentials and link are required to perform this part of the exercise. Gather this information from the email you have received from Oracle and keep it handy.

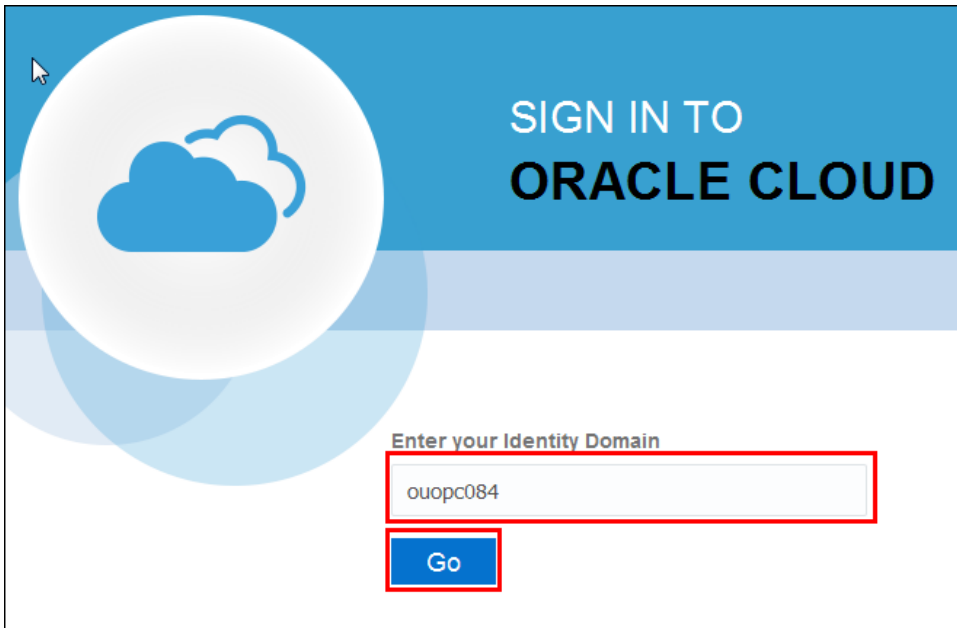
1. In a Firefox browser, navigate to <https://cloud.oracle.com/home>.
2. Click the **Sign In** button.



3. Select the Data Center and click the **My Services** button (the Data Center name is available in the email sent by Oracle).

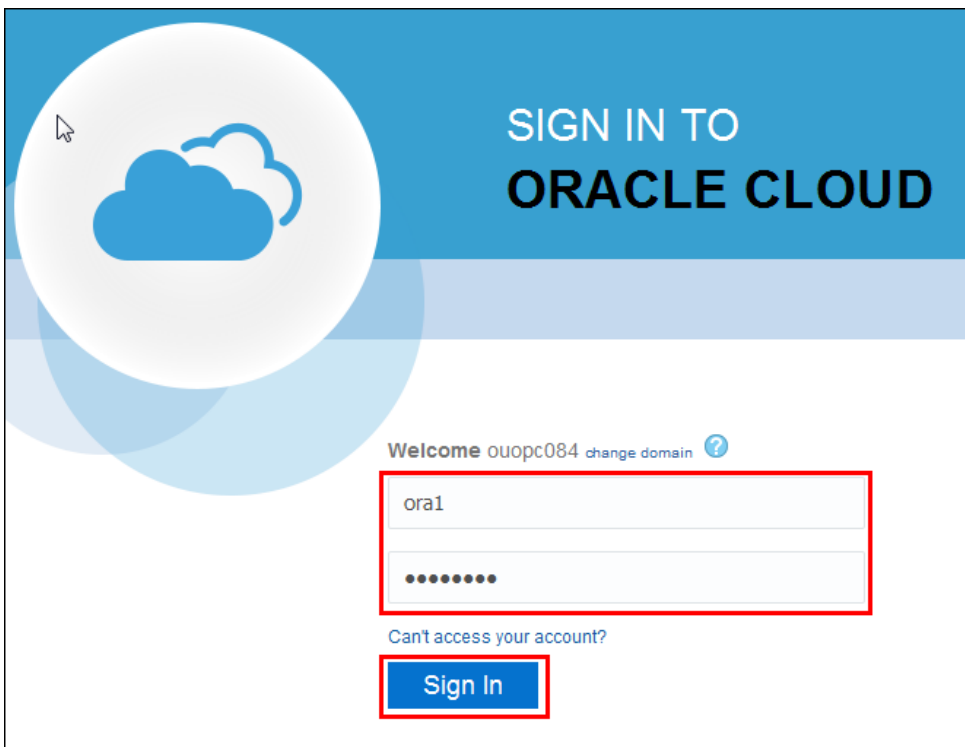


4. Enter the Identity Domain name and click the **Go** button (the Identity Domain name is available in the email sent by Oracle).



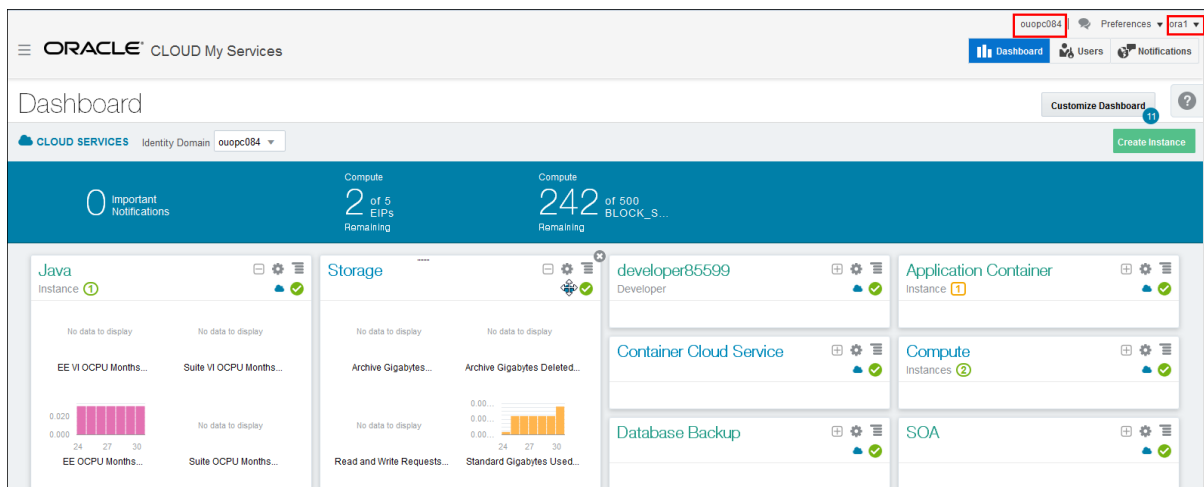
The screenshot shows the Oracle Cloud Sign In page. On the left is a large circular icon with a blue cloud. The header text reads "SIGN IN TO ORACLE CLOUD". Below the header, there is a text input field labeled "Enter your Identity Domain" containing the text "ouopc084". Below the input field is a blue button labeled "Go". Red rectangular boxes highlight the input field and the "Go" button.


5. In the next screen, enter the username and the password and click the **Sign In** button (login credentials are available in the email sent by Oracle).

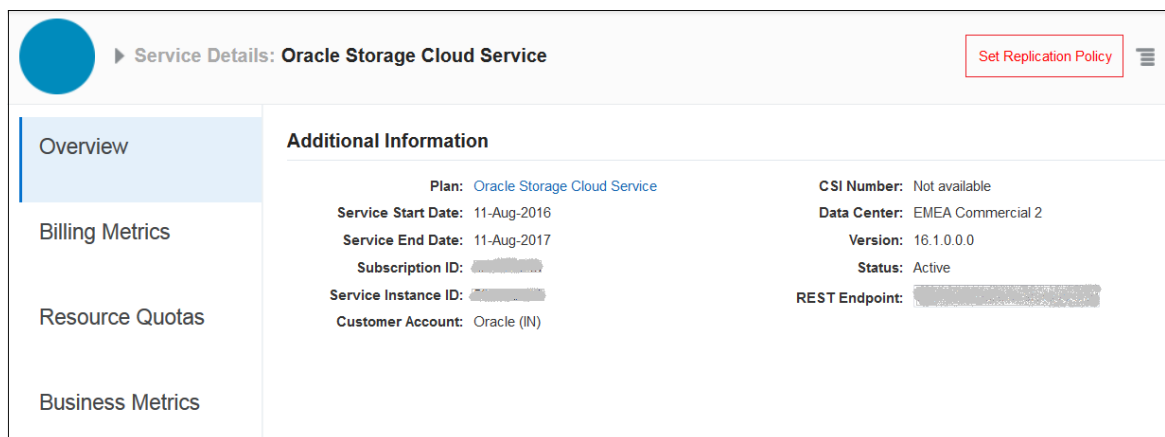


The screenshot shows the Oracle Cloud Sign In page. On the left is a large circular icon with a blue cloud. The header text reads "SIGN IN TO ORACLE CLOUD". Below the header, there is a "Welcome ouopc084" message with a "change domain" link and a help icon. Below this, there are two input fields: the first contains the username "ora1" and the second contains a masked password represented by dots. Below the input fields is a blue button labeled "Sign In". Red rectangular boxes highlight the username and password input fields and the "Sign In" button.

6. On successfully logging in, we can see the **Identity Domain Name** and the **Username** on the Welcome page.



7. Look for **Storage**
8. Click **Storage**. Alternatively, select **View Details** from the **Actions**  menu. The **Service Details** page appears. You can see the details of your Oracle Storage Cloud Service account here.
- If you see the warning, **Set Replication Policy**, you must select a replication policy as described here.



- If you don't see the **See Replication Policy** warning, skip this procedure and proceed with **Activating Developer Cloud Service**.
9. From the Actions  menu, select Set Replication Policy. The **Set Replication Policy** dialog box appears. It displays the available data centers and replication policies for your Oracle Storage Cloud Service instance.

Set Replication Policy

Select the data center (DC) and georeplication policy for your service instance.

- ☒ Primary DC: Amsterdam (em2); Georeplication DC: None
- ☐ Primary DC: Slough (em3); Georeplication DC: None
- ☐ Primary DC: Amsterdam (em2); Georeplication DC: Slough (em3)
- ☐ Primary DC: Slough (em3); Georeplication DC: Amsterdam (em2)

Caution: Once set, the replication policy cannot be changed for the service instance.

Set
Cancel

Note: For your service instance, you may see a list of georeplication policies that's different from the list displayed in the example screenshot.

10. Select a replication policy for your service instance as per your requirement.
11. After selecting a replication policy, click **Set**.

The **Confirm Replication Policy Selection** dialog box appears.

Confirm Replication Policy Selection

You selected **Primary DC: Amsterdam (em2); Georeplication DC: Slough (em3)** as the replication policy for your service instance.

Are you sure you want to set this policy? Note that once set, the replication policy cannot be changed for the service instance.


Confirm
Cancel

12. Verify the selected replication policy details in the **Confirm Replication Policy Selection** dialog box. Click **Confirm**.
13. The following message is displayed in the **Service Details** page:
Set replication policy successfully.

Verifying the Replication Policy Selected for Your Service Instance

Through the My Services Portal

To find out the replication policy that's selected for your Oracle Storage Cloud Service instance, click the **Storage** link in the **Dashboard** page. On the resulting page, expand **Service Details: Oracle Storage Cloud Service**, the details of your Oracle Storage Cloud Service instance is displayed. Look for the Replication Policy field, as highlighted in the following screenshot.

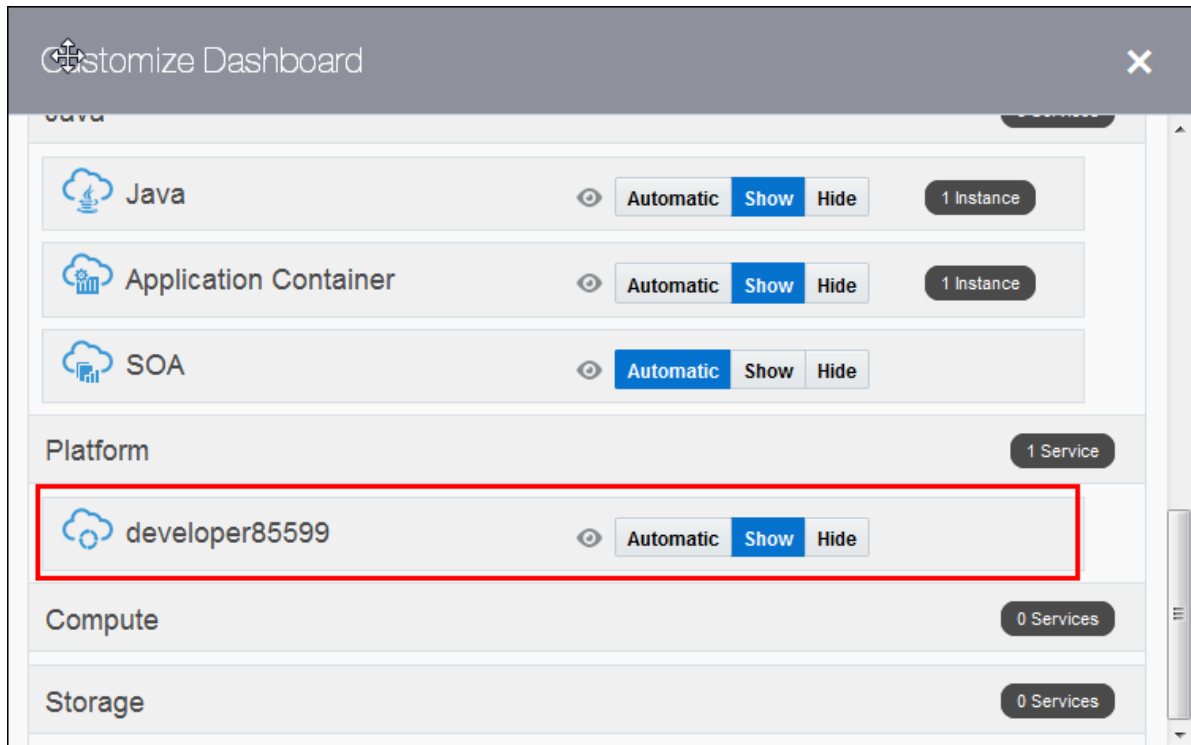


Service Details: Oracle Storage Cloud Service

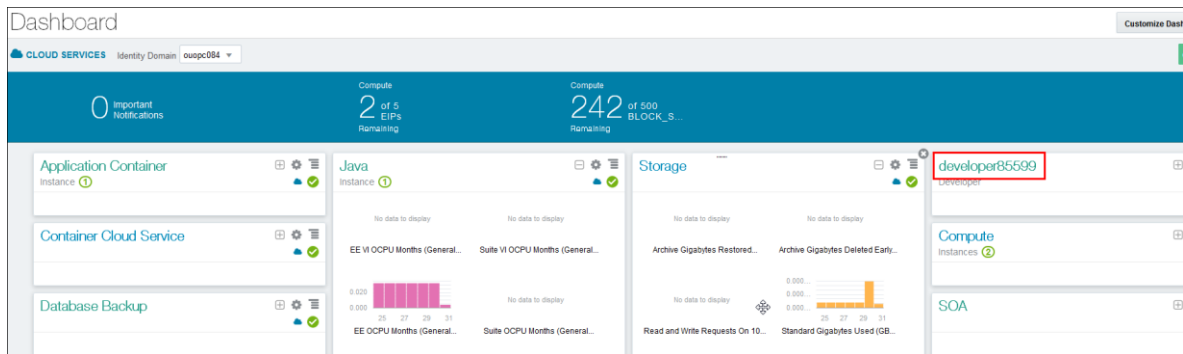
Category:	Oracle IaaS Public Cloud Services
Identity Domain Name:	ouopc084
Identity Domain Id:	ouopc084
Subscription:	Trial (Expires: 15-Dec-2017 3:47 PM JST)
Replication Policy:	Amsterdam, Archive Amsterdam

Activating Developer Cloud Service

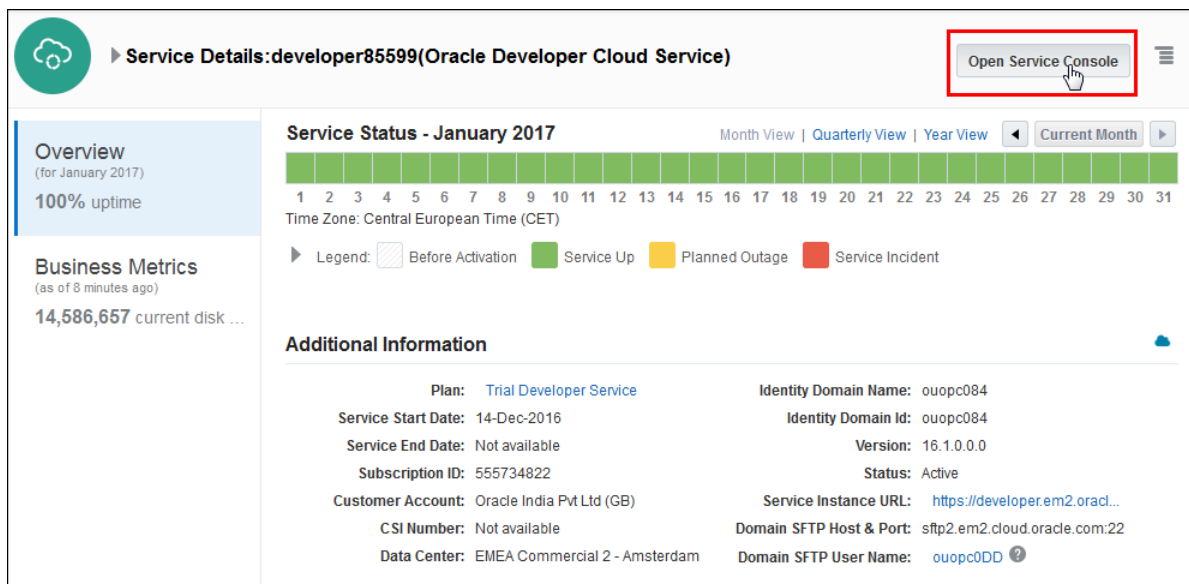
1. Services that are assigned to your account will be visible on the Dashboard. If the **Developer** service is not visible, click the **Customize Dashboard** button and the **Show** button for **Application Container** to make it visible on the Dashboard.



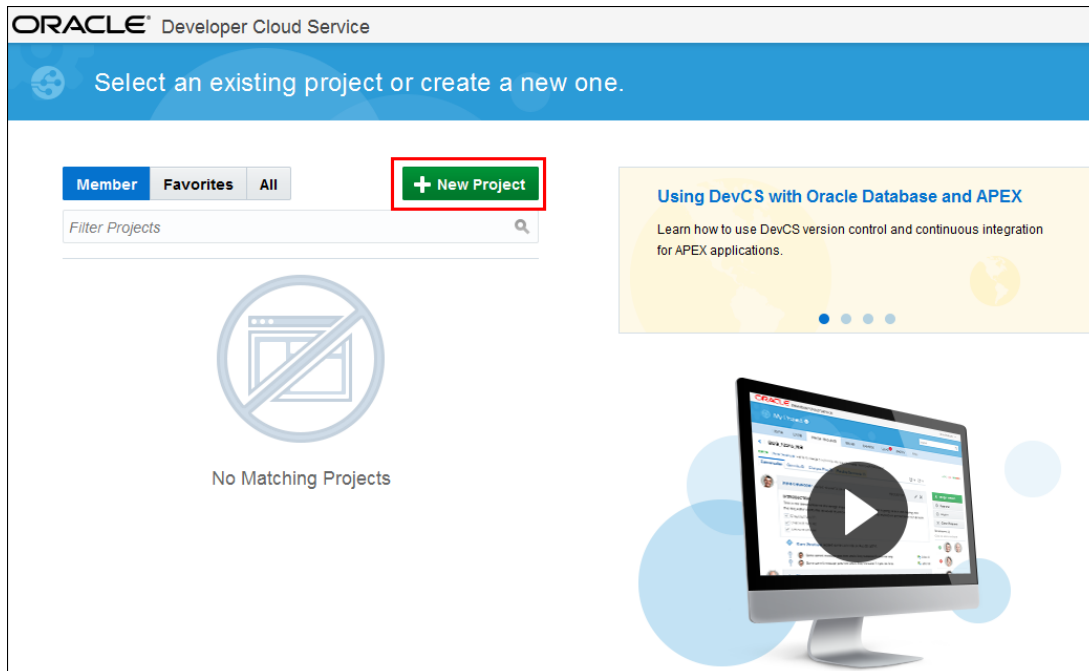
- Click **Developer Cloud Service** on the Dashboard to go to the **ServiceDetails:developer85599 (Oracle Developer Cloud Service)** page.



- Click the **Open Service Console** button.



4. Click **New Project**.

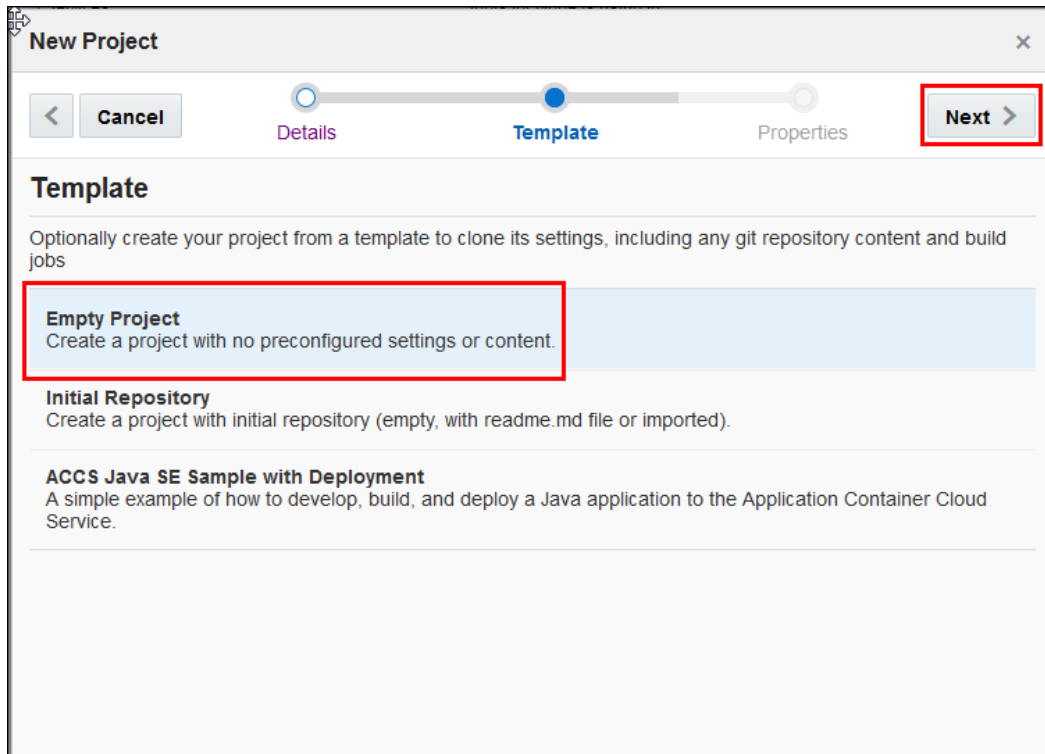


5. Enter the Project Name and Description as shown in the following screenshot and click **Next**.

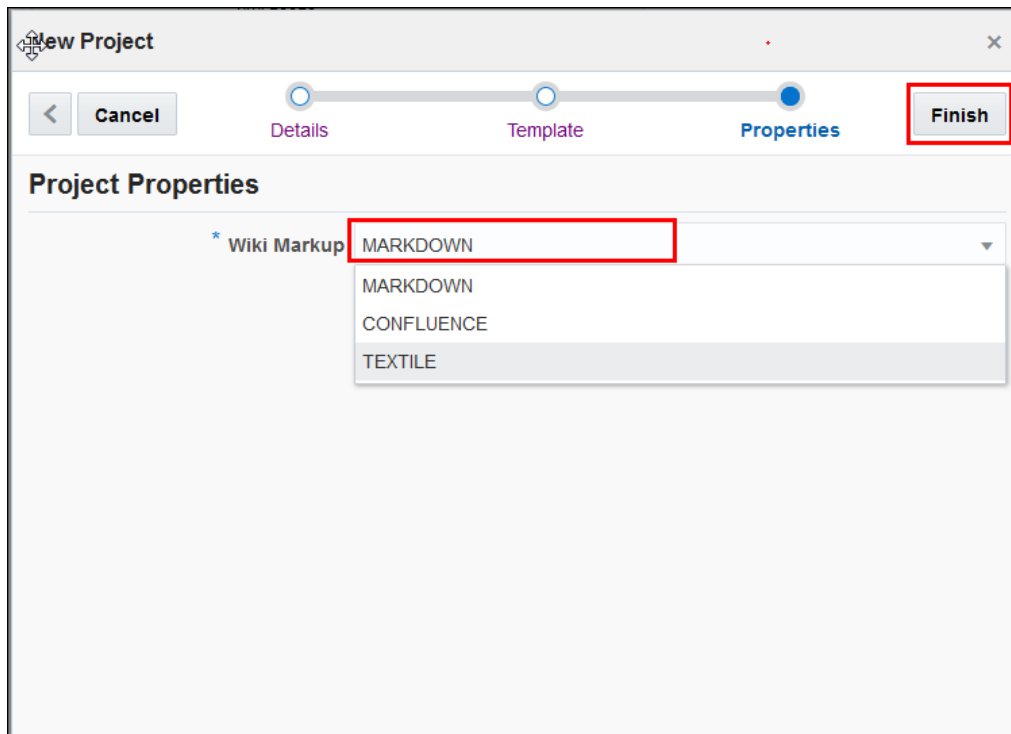
The screenshot shows the 'New Project' dialog box. At the top, there is a progress bar with three steps: 'Details', 'Template', and 'Properties'. The 'Details' step is active. There are buttons for '< Cancel' and 'Next >', with the 'Next >' button highlighted by a red box. The 'Project Details' section contains the following fields:

- Name:** A text input field containing 'HelloworldProject'.
- Description:** A text input field containing 'HelloworldProject'.
- Security:** A section with a question mark icon and two radio buttons: 'Private' (selected) and 'Shared'.
- Preferred Language:** A dropdown menu showing 'English - English'.

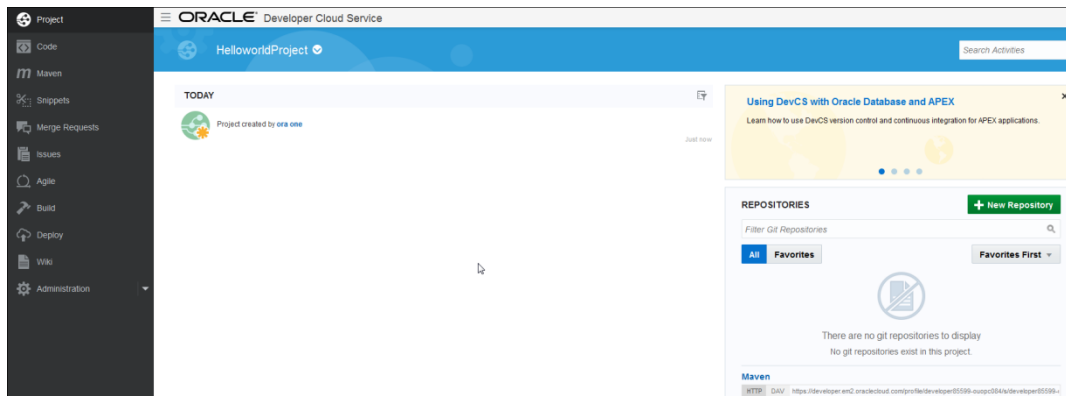
6. Click the **Empty Project** template and **Next**.



7. Select **MARKDOWN** from the Wiki Markup drop-down list and click **Finish**.



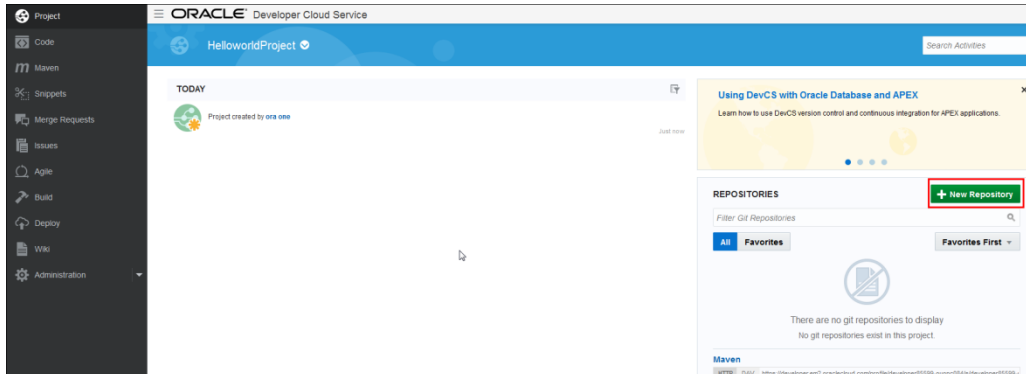
8. Provisioning HelloworldProject may take several minutes. Wait until all the modules are provisioned and redirected to the HelloworldProject home screen.



Creating a GIT Repository in Developer Cloud Service

Use the following instructions to create an empty GIT repository on Developer Cloud Service.

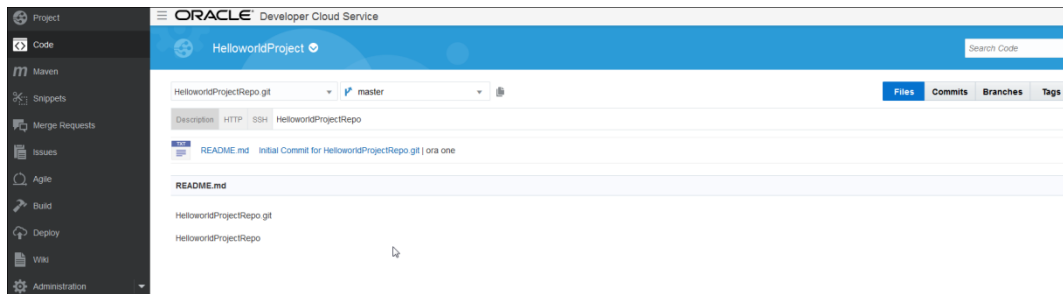
1. Click the **New Repository** button in the **REPOSITORIES** section.



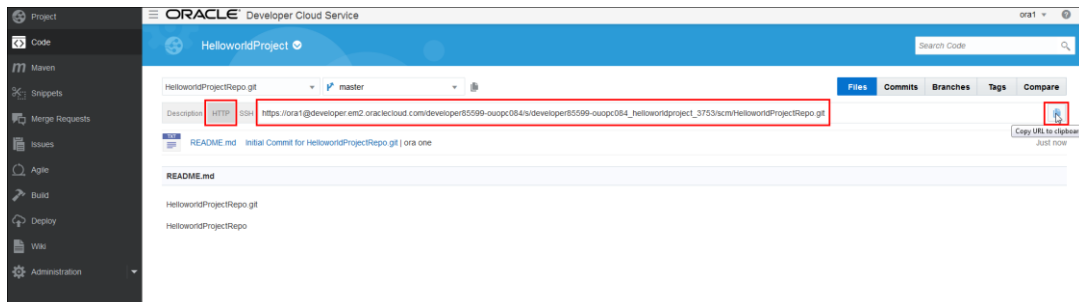
2. In the New Repository window, enter the repository name and description as shown in the following screenshot and click **Create**.

A screenshot of the 'New Repository' dialog box. It has a title bar with 'New Repository' and a close button. The form contains: a 'Name' field with the value 'HelloworldProjectRepo'; a 'Description' field with the value 'HelloworldProjectRepo'; 'Initial content' options with radio buttons for 'Empty Repository', 'Initialize repository with README file' (which is selected), and 'Import existing repository'; a text input field containing 'https://github.com/myname/myrepo.git'; a 'Credentials for non-public repos' section with a right-pointing arrow; and at the bottom, 'Create' and 'Cancel' buttons, with the 'Create' button highlighted by a red box.

- It may take a few minutes to create a repository. Wait until the HelloworldProjectRepo repository is created and redirected to the HelloworldProjectRepo home page.



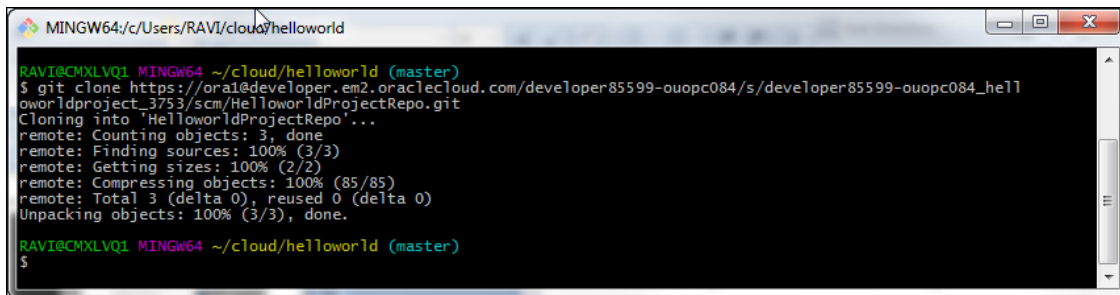
- Click the HTTP tab in the HelloworldProjectRepo home page and copy the URL.



Cloning a GIT Repository

Use the following instructions to clone the Helloworld-Example project to a GIT repository on Developer Cloud Service.

1. To clone a GIT repository, first change to the `cloud/helloworld` directory that is the root directory for your repository.
2. Execute `git clone`
https://ora1@developer.em2.oraclecloud.com/developer85599-ouopc084/s/developer85599-ouopc084_helloworldproject_3753/scm/HelloworldProjectRepo.git



```
MINGW64/c:/Users/RAVI/cloud/helloworld
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$ git clone https://ora1@developer.em2.oraclecloud.com/developer85599-ouopc084/s/developer85599-ouopc084_helloworldproject_3753/scm/HelloworldProjectRepo.git
Cloning into 'HelloworldProjectRepo'...
remote: Counting objects: 3, done
remote: Finding sources: 100% (3/3)
remote: Getting sizes: 100% (2/2)
remote: Compressing objects: 100% (85/85)
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
RAVI@CMXLVQ1 MINGW64 ~/cloud/helloworld (master)
$
```

Notes:

- Enter your cloud account username and password, if you are prompted.
 - The output of this command should be similar to the output in the screenshot.
3. Notice that there is a new directory named **HelloworldProjectRepo** created inside **cloud/helloworld** directory.
 4. Copy and paste **Helloworld-Example** project directory from **cloud/helloworld** directory to **HelloworldProjectRepo** directory

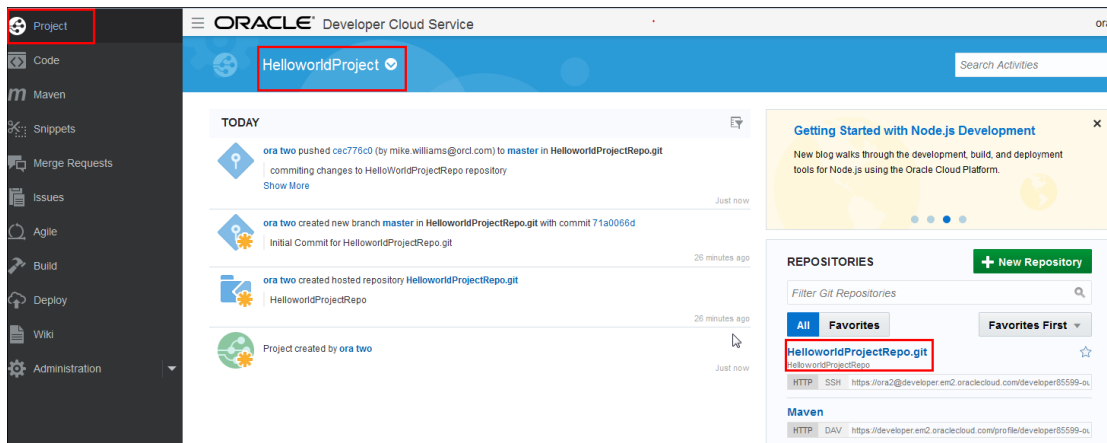
Note: Content of the **HelloworldProjectRepo** directory should match with the contents listed below screenshot.

	.git	2/3/2017 1:41 PM	File folder	
	Helloworld-Example	2/3/2017 1:51 PM	File folder	
	README.md	2/3/2017 1:41 PM	MD File	1 KB

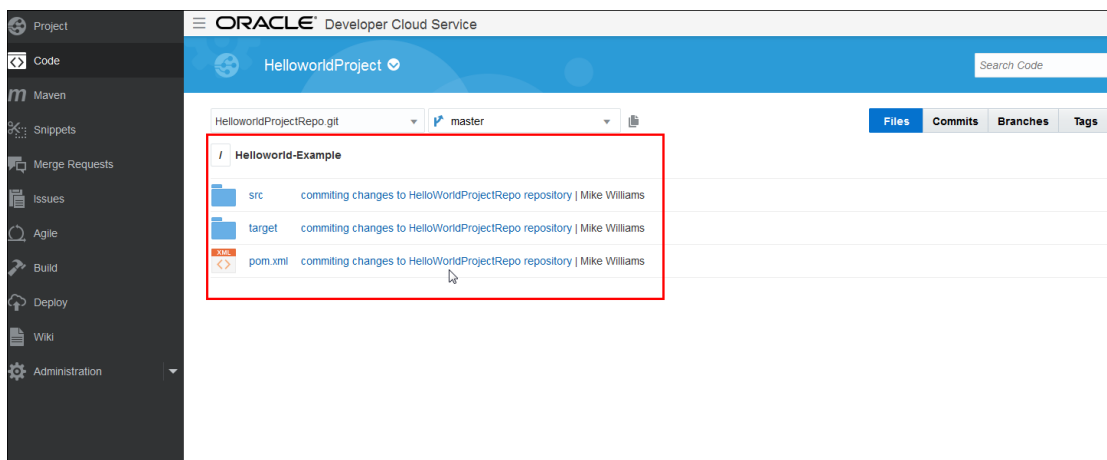
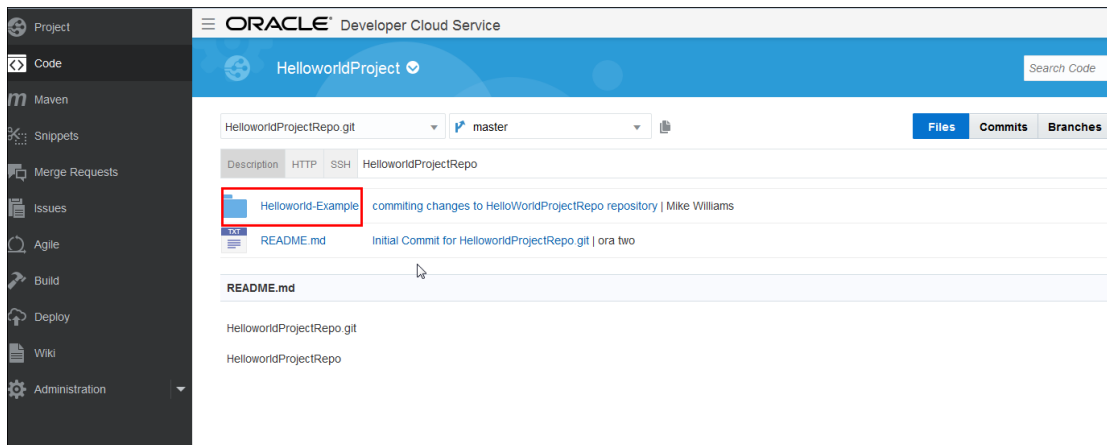
5. Change to the **HelloworldProjectRepo** directory
`cd HelloworldProjectRepo`
6. Add the source files to GIT from project root directory
`git add .`
7. Commit the changes
`git commit -m "committing changes to HelloworldProjectRepo repository"`
8. Push the files to the repository on Developer Cloud Service

```
git push origin master
```

9. Switch to Developer Cloud Service to verify the files pushed to the repository
10. In the **HelloworldProject** home page, click on **HelloworldProjectRepo.git**



11. Notice that **Helloworld-Example** project directory has been pushed to repository on Developer Cloud Service. Click on it and verify its contents.



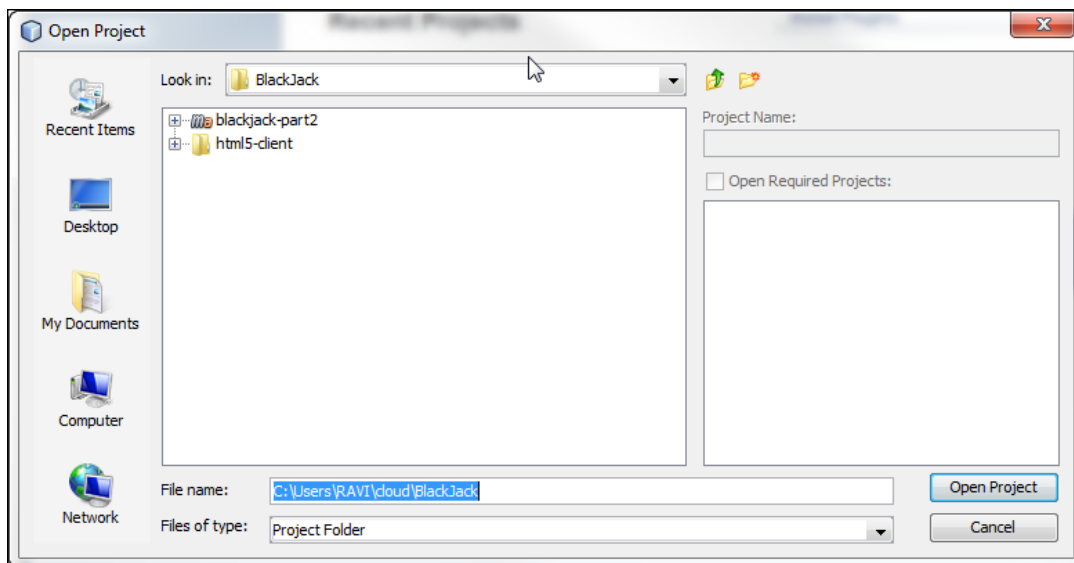
Downloading BlackJack Project from the Repository

<< Instructions for downloading the BlackJack project will go here, Diana or Peter to provide the link>>

Deploying the BlackJack Application on a Local Server

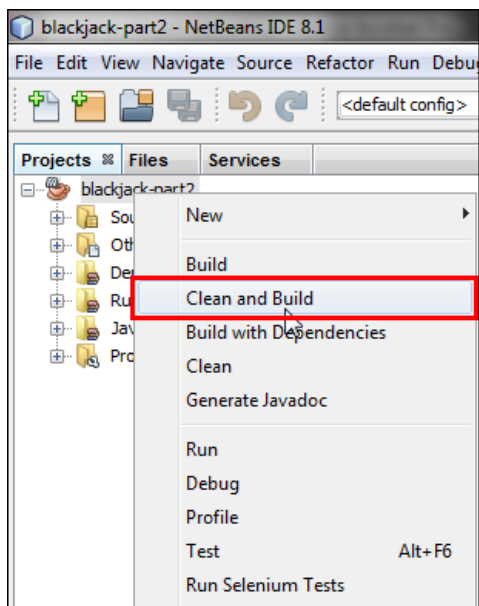
Use the following instructions to deploy the BlackJack application to Apache Tomcat Server bundled with the project.

1. Open Windows Explorer and navigate to the **cloud** directory.
2. Inside the **cloud** directory, create a directory named **BlackJack** and copy the **BlackJack.zip** file that you downloaded in the previous exercise.
3. Unzip the **BlackJack.zip** file to the **cloud > BlackJack** directory.
4. Launch Netbeans using the shortcut on the desktop.
5. Open the **blackjack-part2** project in Netbeans.

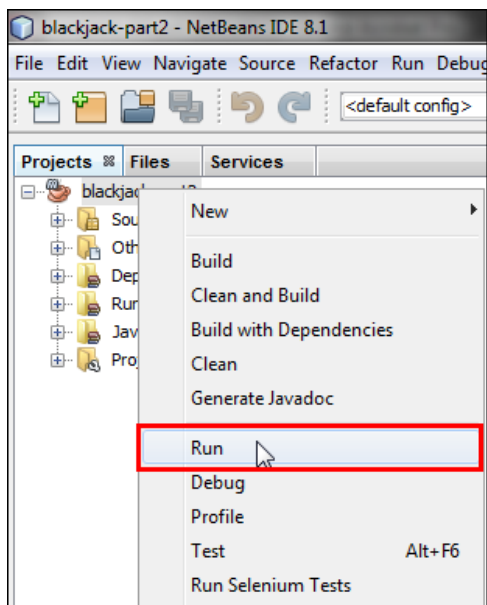


Note: If you see a [Unloaded] tag against the project name then right-click on the project and select, **Resolve Project Problems** and then click on Resolve button. Please wait until Netbeans downloads the Maven related files then click on close button.

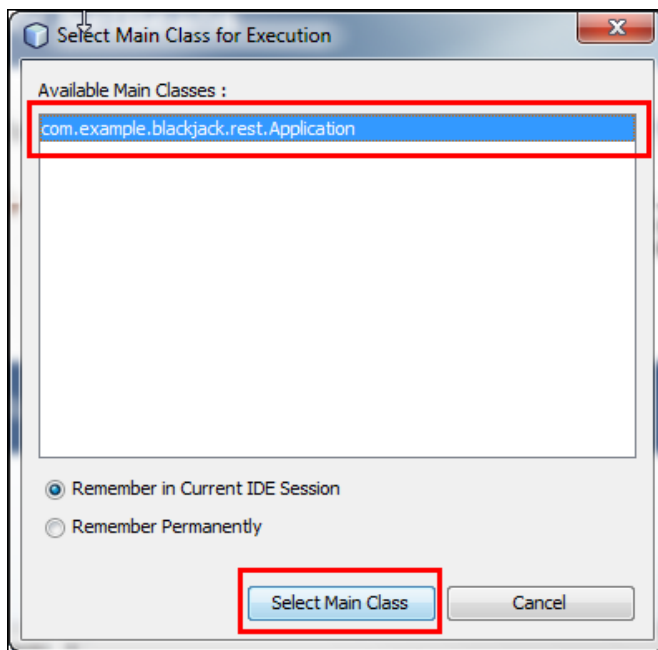
6. Right-click the project and select the **Clean and Build** option.



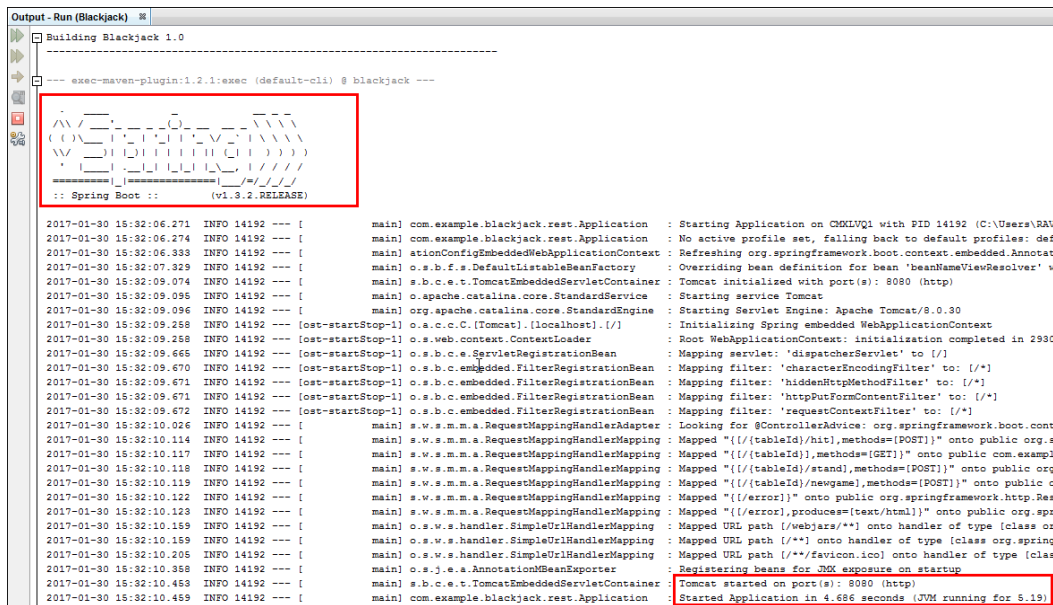
7. Right-click the project and select **Run** to deploy the project on Tomcat Server.



8. Select **com.example.blackjack.rest.Application** from the Available Main Classes list and click the **Select Main Class** button.



9. You should receive a “*Started Application in <<seconds>> seconds (JVM running for 5.19)*” message in the Output window.



```
Output - Run (Blackjack)
Building Blackjack 1.0
--- exec-maven-plugin:1.2.1:exec (default-cli) @ blackjack ---
:: Spring Boot :: (v1.3.2.RELEASE)

2017-01-30 15:32:06.271 INFO 14192 --- [main] com.example.blackjack.rest.Application : Starting Application on C:\Users\RAY
2017-01-30 15:32:06.274 INFO 14192 --- [main] com.example.blackjack.rest.Application : No active profile set, falling back to default profiles: def
2017-01-30 15:32:06.333 INFO 14192 --- [main] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext
2017-01-30 15:32:07.329 INFO 14192 --- [main] org.springframework.beans.factory.support.DefaultListableBeanFactory : Overriding bean definition for bean 'beanNameViewResolver' w
2017-01-30 15:32:09.074 INFO 14192 --- [main] org.springframework.boot.context.embedded.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2017-01-30 15:32:09.095 INFO 14192 --- [main] org.apache.catalina.core.StandardService : Starting service Tomcat
2017-01-30 15:32:09.096 INFO 14192 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.0.30
2017-01-30 15:32:09.258 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Initializing Spring embedded WebApplicationContext
2017-01-30 15:32:09.258 INFO 14192 --- [ost-startStop-1] org.springframework.context.support.AnnotationConfigApplicationContext : Root WebApplicationContext: Initialization completed in 2930
2017-01-30 15:32:09.665 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Mapping servlet: 'dispatcherServlet' to [/]
2017-01-30 15:32:09.670 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Mapping filter: 'characterEncodingFilter' to: [/]
2017-01-30 15:32:09.671 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Mapping filter: 'hiddenHttpMethodFilter' to: [/]
2017-01-30 15:32:09.671 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Mapping filter: 'httpPutFormContentFilter' to: [/]
2017-01-30 15:32:09.672 INFO 14192 --- [ost-startStop-1] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Mapping filter: 'requestContextFilter' to: [/]
2017-01-30 15:32:10.026 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.cont
2017-01-30 15:32:10.114 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{tableId}/hit],methods=[POST])" onto public org.s
2017-01-30 15:32:10.117 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{tableId}],methods=[GET])" onto public com.exempl
2017-01-30 15:32:10.118 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{tableId}/stand],methods=[POST])" onto public org
2017-01-30 15:32:10.119 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{tableId}/newgame],methods=[POST])" onto public o
2017-01-30 15:32:10.122 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{error}]" onto public org.springframework.http.Res
2017-01-30 15:32:10.123 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped "([/{error}],produces=[text/html])" onto public org.spr
2017-01-30 15:32:10.159 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped URL path [/webjars/**] onto handler of type [class org.spring
2017-01-30 15:32:10.159 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped URL path [/**] onto handler of type [class org.spring
2017-01-30 15:32:10.205 INFO 14192 --- [main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter : Mapped URL path [/**/favicon.ico] onto handler of type [class
2017-01-30 15:32:10.358 INFO 14192 --- [main] org.springframework.jersey.support.Jersey2BeanExporter : Registering beans for JAX-WS exposure on startup
2017-01-30 15:32:10.453 INFO 14192 --- [main] org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Tomcat started on port(s): 8080 (http)
2017-01-30 15:32:10.459 INFO 14192 --- [main] com.example.blackjack.rest.Application : Started Application in 4.686 seconds (JVM running for 5.19)
```

Note: You may encounter a problem in running this project due to a port conflict issue. This application will be deployed to Apache Tomcat Server and it requires **8080** local port number to listen to the client request. Make sure you stop the services running on **8080** local port number.

TCPView tool can be used to identify and terminate the process using this port number.

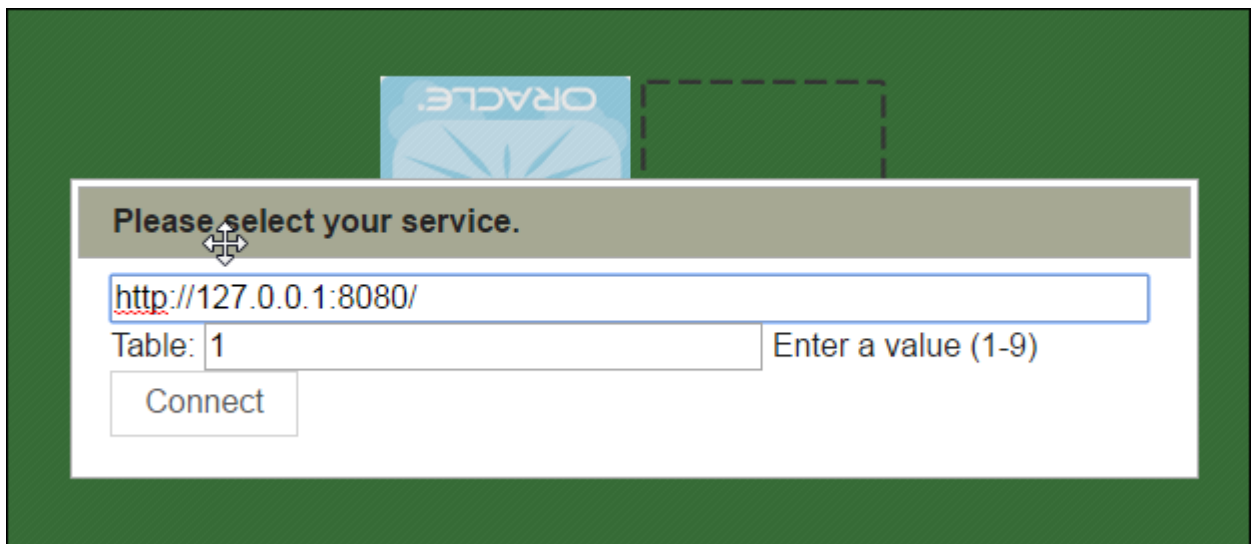
[Download Link](#)

Testing the Locally Deployed BlackJack Application

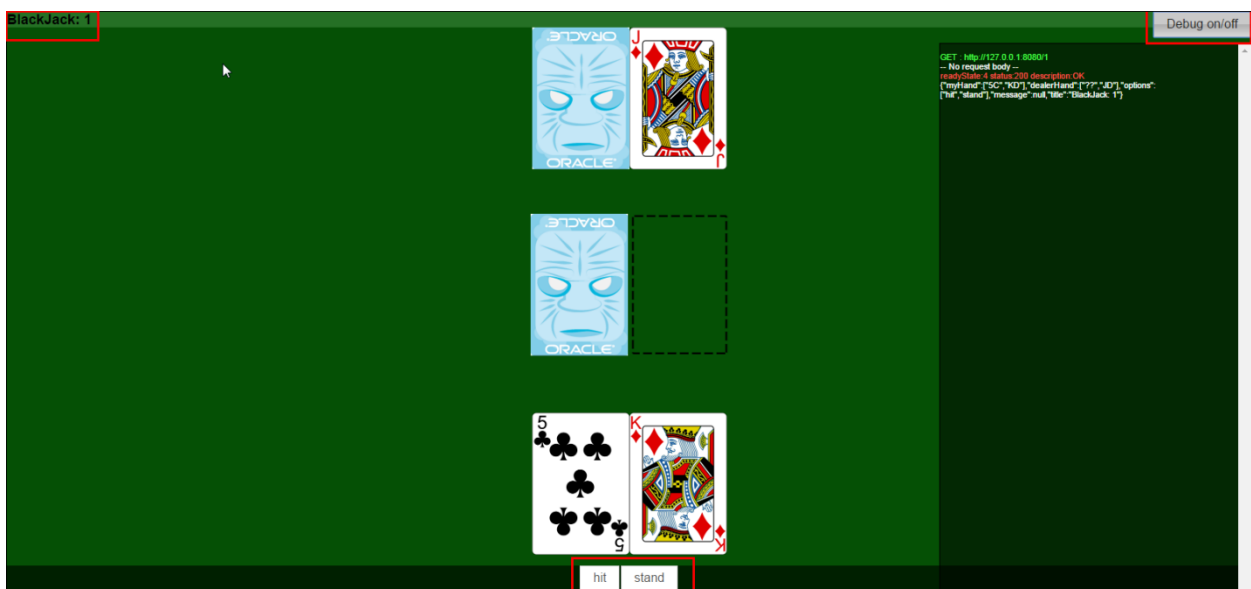
An HTML-5 client application has been developed and supplied with the BlackJack application to test its functionality once deployed on a local/remote server.

Use the following instructions to test the BlackJack application.

1. Open Windows Explorer and navigate to the **cloud > BlackJack > html5-client** directory.
2. Open the index.html file with a browser.
3. Make sure that the first field, **Service**, is populated with <http://127.0.0.1:8080/> value, enter a number between 1 and 9 in the second field, and then click Connect.



4. Once you connect to the gaming console, click the **Debug on/off** button to view the Debug console.



You can use the **Hit** and **Stand** buttons available on the UI to play the game.

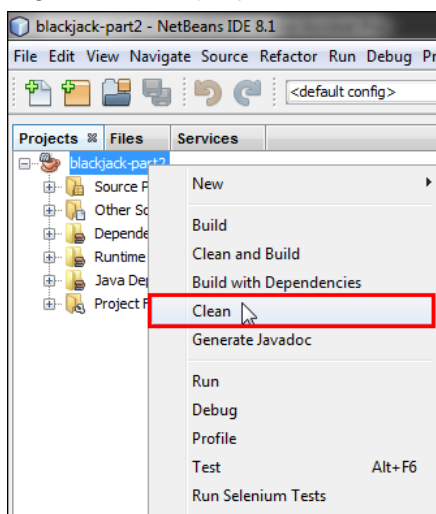
This HTML5 Client application interacts with the BlackJack gaming application deployed on Tomcat Server running locally on your computer. Close this HTML5 Client application once you are done.

Generating Application Archive Files for the BlackJack Application

Oracle Application Container Cloud can deploy and run Java Platform, Standard Edition (Java SE), and Node.js applications. First, to deploy our application, we compress the application in a **ZIP or Gzipped Tar (TGZ)** archive file, which includes the required configuration information. Then, we use the Oracle Application Container Cloud graphical user interface (GUI) to deploy our application. With the application deployed, we can test and run our application and manage the application's size.

Use the following steps to deploy the application to OACCS(Oracle Application Container Cloud Service).

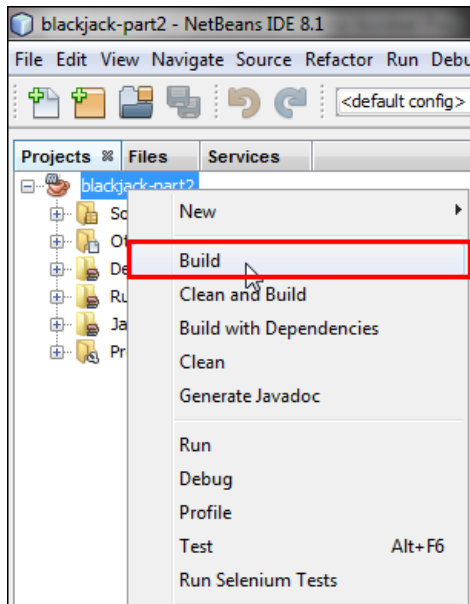
1. Open the blackjack-part2 application in Netbeans if it is not opened already.
2. Right-click the project and click **Clean**.



3. Open Windows Explorer, navigate to **cloud > BlackJack > blackjack-part2**, and make a note of the directory structure and its contents.

Name	Date modified	Type	Size
src	1/30/2017 6:39 PM	File folder	
bin.xml	4/26/2016 10:44 AM	XML Document	1 KB
manifest.json	4/26/2016 10:49 AM	JSON File	1 KB
pom.xml	4/26/2016 10:44 AM	XML Document	3 KB

4. Switch to Netbeans, right-click the project, and click **Build**.



5. Switch to the **cloud > BlackJack > blackjack-part2** directory and notice that a new directory named **target** is created.
6. Examine the **target** directory. You will notice that **.zip and .tar.gz** distribution files have been generated. These are application archive files that we can use to deploy to OACCS.

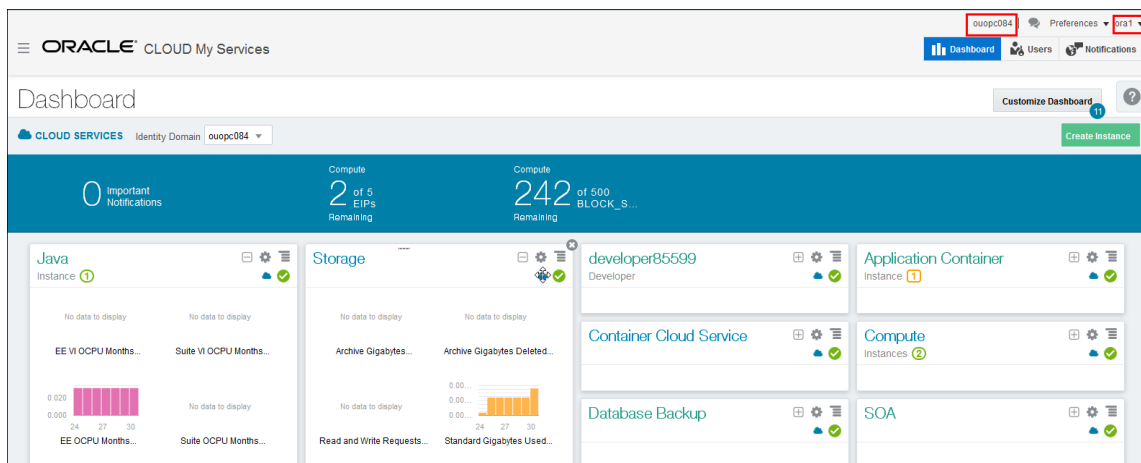
archive-tmp	1/30/2017 7:36 PM	File folder	
classes	1/30/2017 7:36 PM	File folder	
generated-sources	1/30/2017 7:36 PM	File folder	
maven-archiver	1/30/2017 7:36 PM	File folder	
maven-status	1/30/2017 7:36 PM	File folder	
test-classes	1/30/2017 7:36 PM	File folder	
blackjack-part2-1.0.jar	1/30/2017 7:36 PM	Executable Jar File	13,091 KB
blackjack-part2-1.0.jar.original	1/30/2017 7:36 PM	ORIGINAL File	11 KB
blackjack-part2-1.0-dist.tar.gz	1/30/2017 7:36 PM	gz Archive	11,691 KB
blackjack-part2-1.0-dist.zip	1/30/2017 7:36 PM	zip Archive	13,091 KB

Activating Oracle Application Container Cloud Service (OACCS)

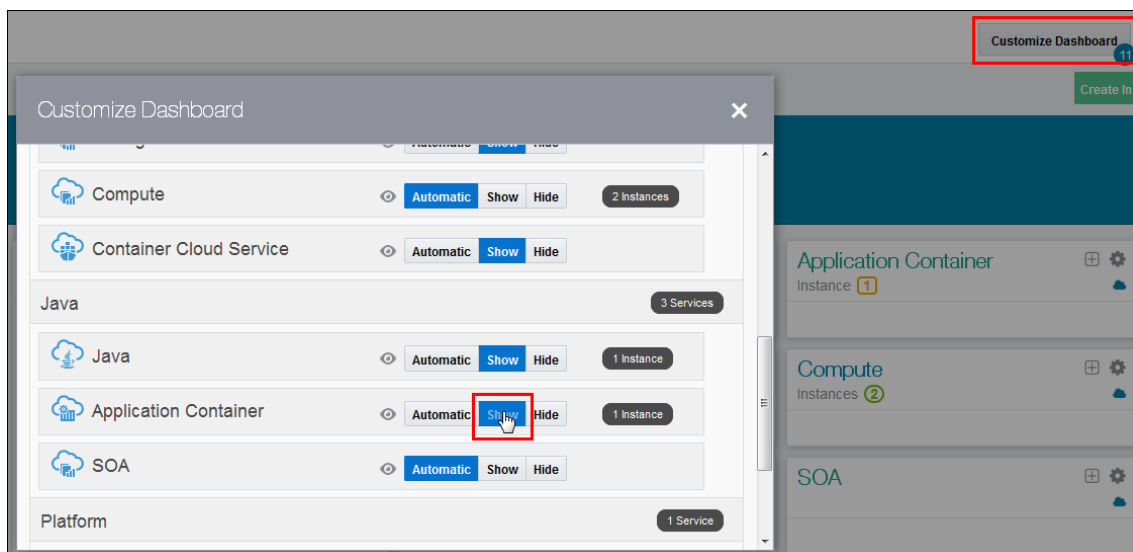
Important Note: The cloud login credentials and link are required to perform this part of the lab activity. Gather this information from the email you have received from Oracle and keep it handy.

For the purpose of creating this document, a cloud instance from the EMEA region Data Center was used. You will get a cloud instance from the NAMER region Data Center; select the Data Center accordingly.

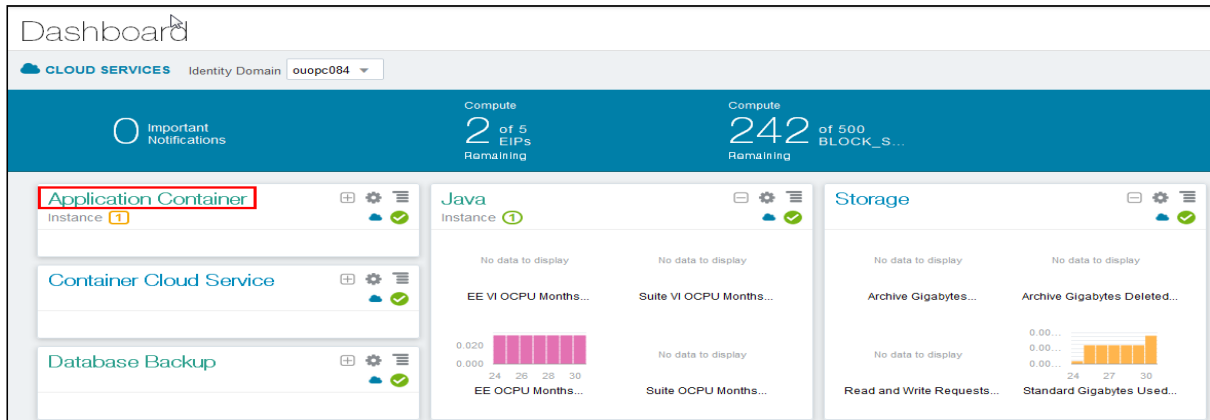
1. Sign In to Oracle Cloud account (Refer to **Activating Developer Cloud Service** section for detailed instruction on how to Sign In)
2. On a successful Sign In, we can see the **Identity Domain Name** and the **Username** on the Welcome page.



3. Services that are assigned to your account will be visible on the Dashboard. If the **Application Container** service is not visible, click the **Customize Dashboard** button and **Show** button for **Application Container** to make it visible on the Dashboard.



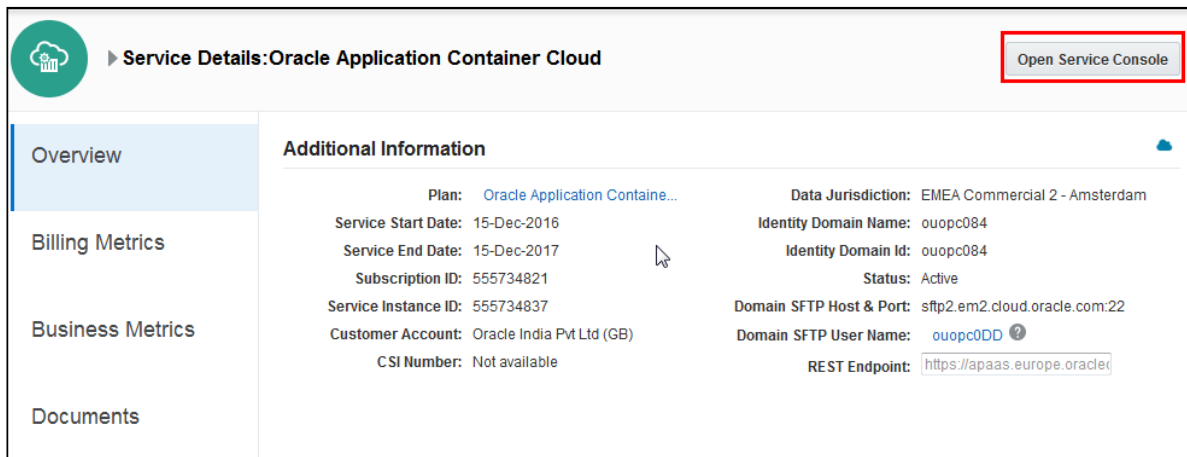
- Click the **Application Container** on the Dashboard to go to the **Service Details: Oracle Application Container Cloud** page.



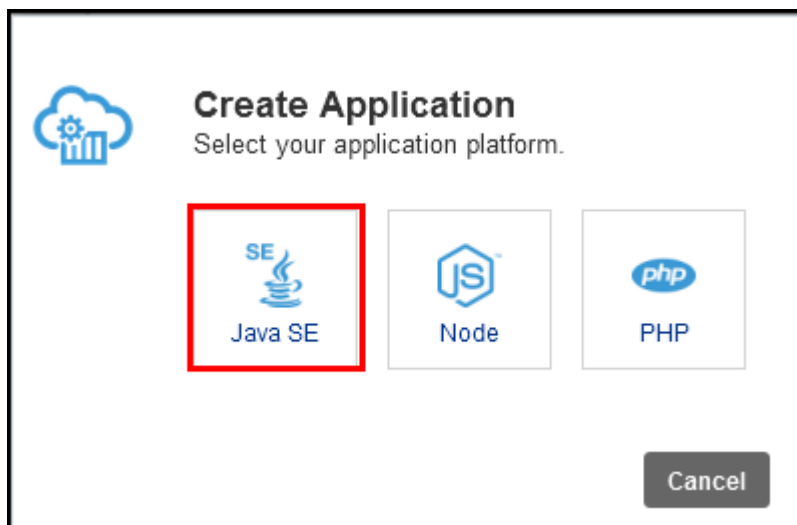
Deploying the BlackJack Application on OACCS

Use the following instructions to deploy the BlackJack application archive that you generated for OACCS as part of the previous exercise.

1. Click the **Open Service Console** button.



2. Click the **Create Application** and **Java SE** buttons.



3. In the **Create Application** dialog box, enter **BlackJack-part2** for the application name, select **Monthly** for the subscription type, and enter **Deploying BlackJack Application** in the Notes field. For the Application Archive field, select **Upload Archive**.

Create Application

Application

* Name: BlackJack-part2

Subscription: ☒ Monthly ☐ Hourly

* Application Archive: ☒ Deploy a Sample Application ☐ Get Archive from Storage Path ☐ Upload Archive

Notes: Deploying BlackJack Application

Instances

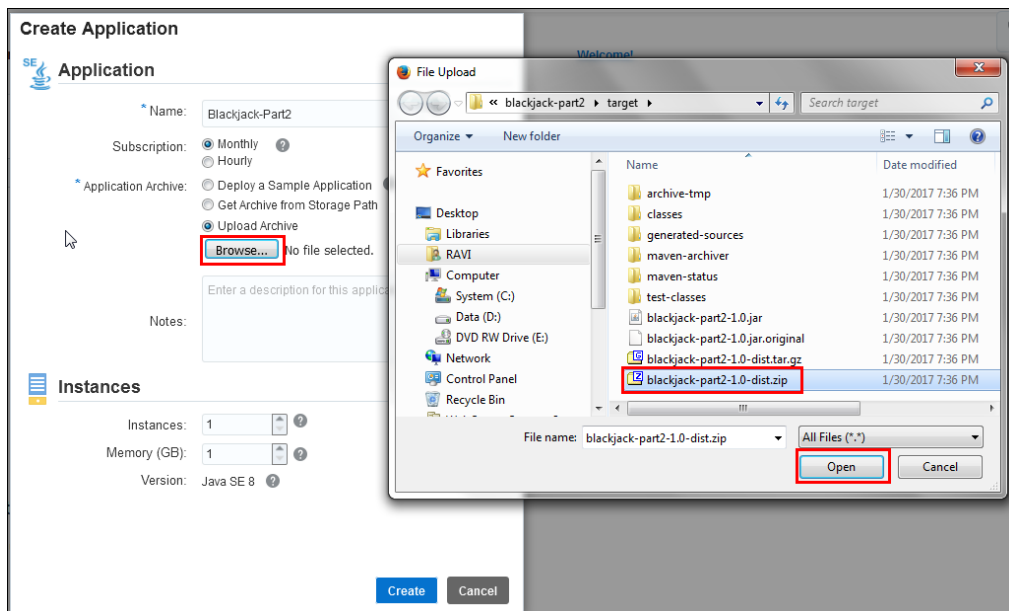
Instances: 1

Memory (GB): 1

Version: Java SE 8

Create Cancel

4. Browse and select the **blackjack-part2-1.0-dist.zip** file from the **target** directory.



5. The **Create Application** dialog box now shows the selected file. Under **Instance**, review the number of instances and the memory size, and make any necessary adjustments. Click **Create** to deploy your application to Oracle Application Container Cloud.

Create Application

Application

* Name: Blackjack-Part2 ?

Subscription: ☒ Monthly ?
☐ Hourly

* Application Archive: ☐ Deploy a Sample Application ?
☐ Get Archive from Storage Path
☒ Upload Archive

Browse... blackjack-part2-1.0-dist.zip

Enter a description for this application.

Notes:

Instances

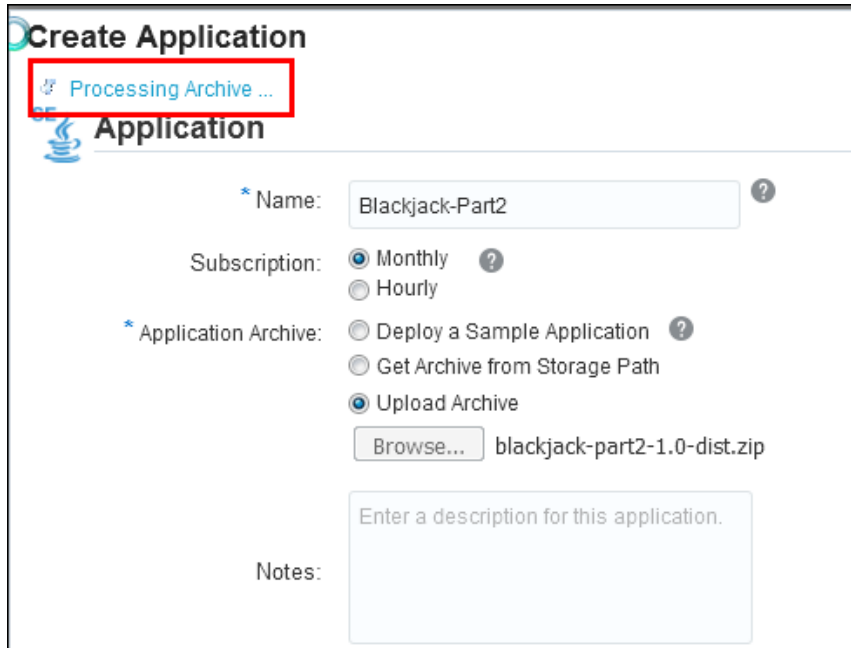
Instances: 1 ?

Memory (GB): 1 ?

Version: Java SE 8 ?

Create Cancel

6. A status message appears indicating that it is **Processing Archive**.



Create Application

Processing Archive ...

Application

* Name:

Subscription: ☒ Monthly ☐ Hourly

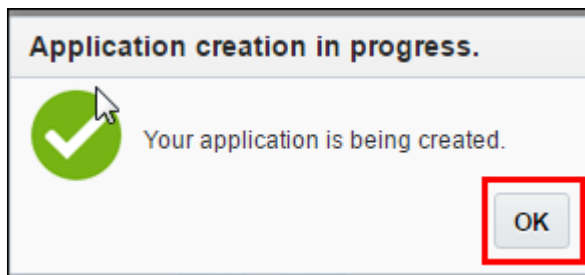
* Application Archive: ☐ Deploy a Sample Application ☐ Get Archive from Storage Path ☒ Upload Archive

blackjack-part2-1.0-dist.zip

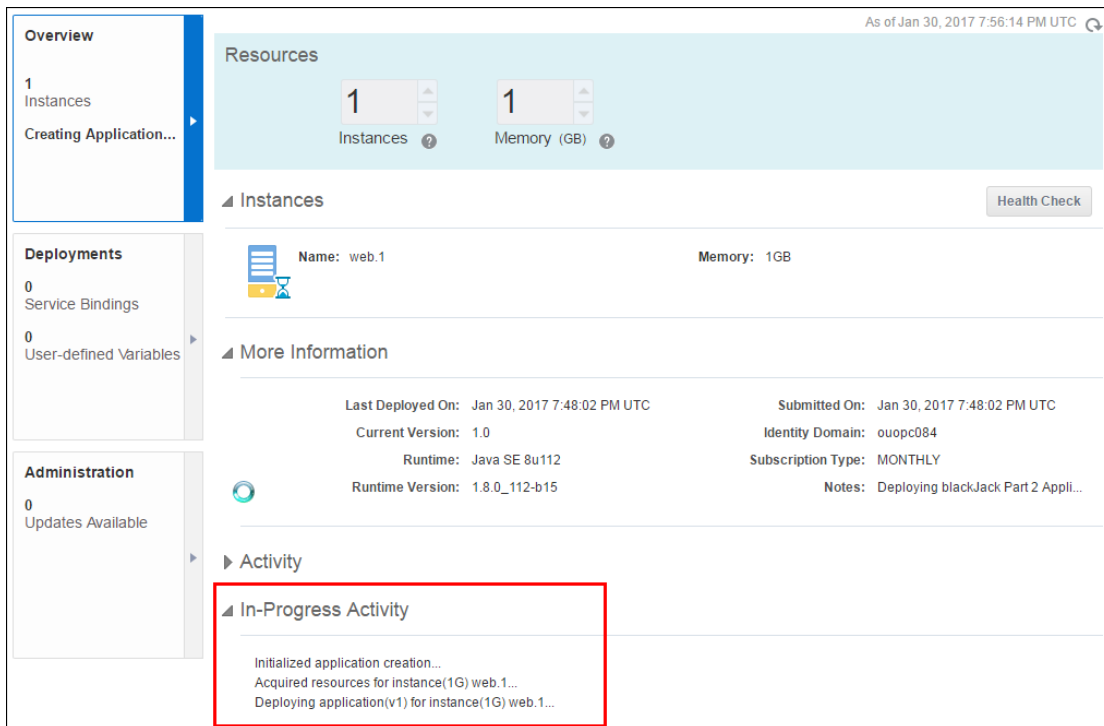
Enter a description for this application.

Notes:

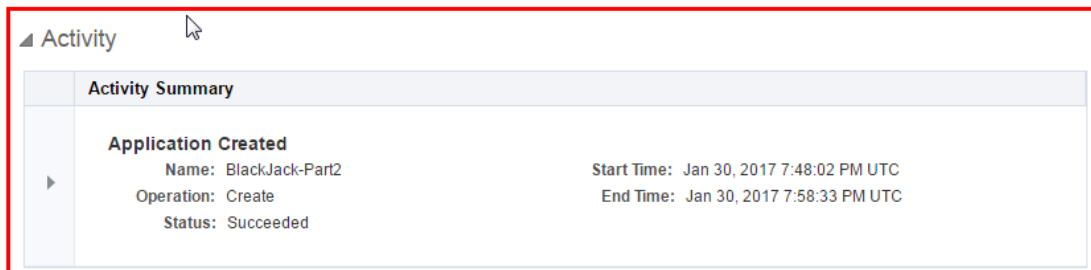
7. After the archived application is uploaded, the service determines whether the archive is properly configured. If it is, the following dialog box appears. Click **OK**.



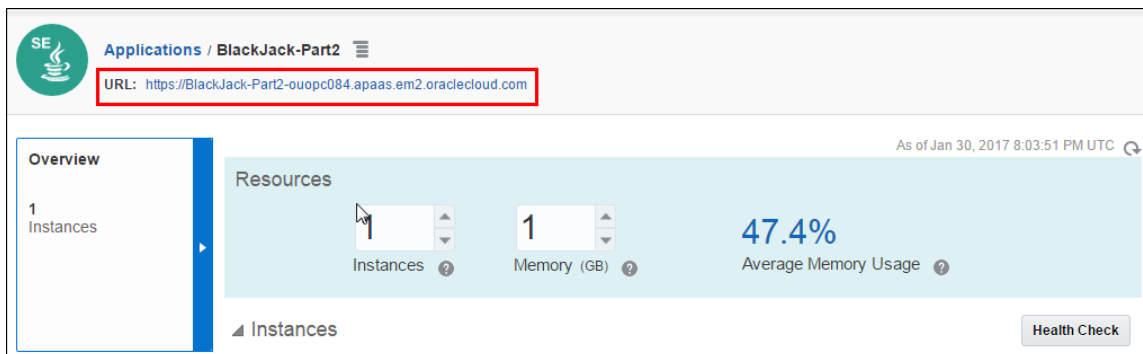
8. It will take several minutes to deploy the application. The deployment status can be viewed under the In-Progress Activity section.



9. You should see a **Status: Succeeded** message in the Activity section once the application has been deployed successfully.



10. Copy the application URL and paste it in a notepad. We will need this URL for testing purposes.

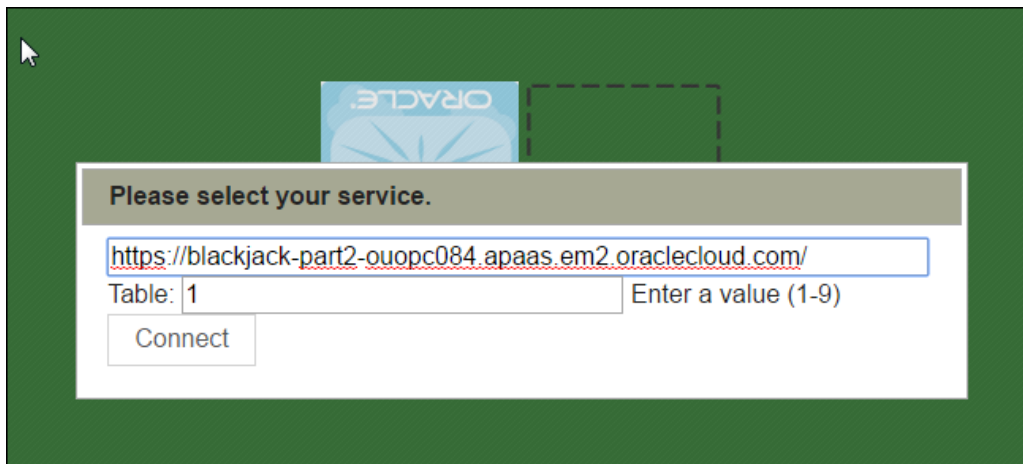


Testing the BlackJack Application Deployed on OACCS

An HTML-5 client application has been developed and supplied with the BlackJack application to test its functionality once deployed on a local/remote server.

Use the following instructions to test the BlackJack application.

1. Open Windows Explorer and navigate to the **cloud > BlackJack > html5-client** directory.
2. Open the index.html file with a browser.
3. Make sure that the first field, **Service**, is populated with the URL you copied in the previous exercise and add a /(forward slash) at the end, <https://blackjack-part2-ouopc084.apaas.em2.oraclecloud.com/> . Enter a number between 1 and 9 in second field, and then click Connect.



4. Once you connect to the gaming console, click the **Debug on/off** button to view the Debug console.



Note: You can use the **Hit** and **Stand** buttons available on the UI to play the game.

This HTML5 Client application interacts with the BlackJack gaming application deployed on OACCS on cloud.

Troubleshooting Tips

These are some of the issues that you may encounter while performing these exercises and below are the troubleshooting tips and procedures that you can follow to resolve them.

Proxy Issue

You may encounter proxy issues with Maven and Netbeans if you are part of a secured network and behind a fire wall. This is primarily because when you are creating a new project in Maven or running an existing Maven project, it tends to download several configuration files and the download will fail if the proxy settings are not done.

Note: Ask your event manager or network administrator for the proxy address

Resolving proxy issues in Maven:

1. Open the C:\Maven\apache-maven-3.3.9\conf\settings.xml file with a text editor like Notepad++.
2. Add the following lines under the <proxies> tag:

```
<proxy>

    <id>Oracle</id>

    <active>true</active>

    <protocol>http</protocol>

    <host>ENTER YOUR PROXY ADDRESS</host>

    <port>80</port>

    <nonProxyHosts>localhost|oracle.com</nonProxyHosts>

</proxy>
```

3. Replace **ENTER YOUR PROXY ADDRESS** within the <host> tag with your proxy and save the file.

Resolving proxy issues in Netbeans:

1. Open the C:\Program Files\NetBeans 8.1\java\maven\conf\settings.xml file with a text editor like Notepad++.
2. Add the following lines under the <proxies> tag:

```
<proxy>

    <id>Oracle</id>

    <active>true</active>

    <protocol>http</protocol>
```

```
<host>ENTER YOUR PROXY ADDRESS</host>

<port>80</port>

<nonProxyHosts>localhost|oracle.com</nonProxyHosts>

</proxy>
```

3. Replace **ENTER YOUR PROXY ADDRESS** within the <host> tag with your proxy and save the file.

Port Conflict Issue

You may encounter a problem in running this project due to a port conflict issue. This application will be deployed to Apache Tomcat Server and it requires **8080** local port number to listen to the client request. Make sure you stop the services running on **8080** local port number.

TCPView tool can be used to identify and terminate the process using this port number.

[Download Link](#)