Lab 06_01: Deploy a JDBC Driver

Performance Checklist

Lab Overview:

In this exercise, you will deploy the JDBC driver for the PostgreSQL database.

Lab Resources/Configuration:

Lab Files Location:	n/a
Application URL:	`http://192.168.0.xx:9990/console/App.html#domain-deployments`

Success Criteria: After completing this exercise, the PostgreSQL driver will appear in the list of deployments in your Domain.

Outcome: After the driver is installed, you will be able to define a datasource.

Lab Outline:

1. Prepare the JDBC JAR for Deployment
2. Define the **`java.sql.Driver`** File
3. Update the JAR
4. Add the JDBC JAR to the Content Repository
5. Deploy the Driver

☐ 1. Prepare the JDBC JAR for Deployment

☐ 1.1. Open a terminal window (or command prompt on Windows) and **cd** into your ***LABS*/`installs`** folder. You should see a Postgres JDBC JAR file in the **`installs`** folder with the name **`postgresql-9.0-801.jar`**.

☐ 1.2. Enter the following command to view the contents of the JDBC JAR file (without extracting the file):

```
jar tf postgresql-9.0-801.jar
```

Notice this JAR file already has a folder named **`META-INF`**, but there is no **`services`** subfolder of **`META-INF`** with a **`java.sql.Driver`** file. You will add this in the next step.

What is the name of the driver class file that is in this JAR?

_____

☐ 1.3. You will now create the necessary folders and file to make this JDBC JAR file deployable. From the **`installs`** folder, enter the following command to make a new folder named **`META-INF`**:

```
mkdir META-INF
```

```
mkdir META-INF/services
```

☐ 2. Define the java.sql.Driver File

☐ 2.1. Open a text editor and create a new, blank text file.

☐ 2.2. Add the following single line of text to the file:

```
org.postgresql.Driver
```

☐ 2.3. Save the text file in your ***LAB*/installs/META-INF/services** folder with the filename **java.sql.Driver**.

☐ 3. Update the JAR

☐ 3.1. Enter the following command from your **installs** folder to add your **META-INF/services/java.sql.Driver** file to the JDBC JAR:

```
jar uf postgresql-9.0-801.jar META-INF/services/java.sql.Driver
```

☐ 3.2. Verify this worked by entering the following command:

```
jar tf postgresql-9.0-801.jar
```

Make sure your **java.sql.Driver** file is where it is supposed to be in the JDBC JAR file.

☐ 4. Add the JDBC JAR to the Content Repository

☐ 4.1. On the **Runtime** page of the **Management Console**, select **Manage Deployments**.

☐ 4.2. Click the **Add Content** button. On Step 1 of the wizard, browse to and select the **postgresql-9.0-801.jar** file that you just updated in the previous step.

☐ 4.3. Click on the **Next** button. Step 2 of the wizard does not require any changes, so click the **Save** button.

☐ 4.4. Verify that your Postgres JDBC JAR file appears in the list of available deployments under **Content Repository**.

- ☐ 5. Deploy the Driver
  - ☐ 5.1. Using the Management Console, deploy and enable the Postgres JDBC JAR onto both the **dev-group** and the **production-group**.
  - ☐ 5.2. To verify the driver is deployed on your Servers, check the output of your host2 and host3 terminal windows (or view the server log files). You should see an output similar to:

```
Starting deployment of "postgresql-9.0-801.jdbc4.jar"
Deploying non-JDBC-compliant driver class org.postgresql.Driver (version
 9.0)
Register module: Module "deployment.postgresql-9.0-801.jar:main" from
 Service Module Loader
Deployed "postgresql-9.0-801.jdbc4.jar"
```

  - ☐ 5.3. Stop here! Your driver is deployed and ready for use. In the next lab, you will define a datasource that uses the driver you just deployed.

Lab 06_02: Define a Datasource

Performance Checklist

Lab Overview:

In this exercise, you will define a datasource that will be used in an upcoming Unit for an application named JBTravel.

Lab Resources/Configuration:

Lab Files Location:	*LABS*/Lab06_02
Application URL:	http://192.168.0.xx:9990/console/App.html#datasources

Success Criteria: After completing this exercise, you have a datasource connection pool that you can verify using the **dstest.war** application.

Outcome: A deployed datasource whose JNDI name is **java:jboss/JBTravelDatasource**.

Lab Outline:

1. Configure the Datasource
2. Verify the Datasource Configuration
3. Test the Datasource
4. Modify the Datasource

☐ 1. Configure the Datasource

- ☐ 1.1. Go to the **Profiles** page of the Management Console.
- ☐ 1.2. Change the profile to **ha**.
- ☐ 1.3. Select the **Datasources** link under the **Connector** subsystem.
- ☐ 1.4. Click the **Add** button.
- ☐ 1.5. Enter **"JBTravel"** for the **Name**.
- ☐ 1.6. Enter **"java:jboss/JBTravelDatasource"** for the JNDI Name.
- ☐ 1.7. Click the **Next** button.
- ☐ 1.8. On Step 2 of the wizard, select the **postgresql-9.0-801.jar** driver deployed on the **production-group** Server Group.
- ☐ 1.9. Click the **Next** button.
- ☐ 1.10. The Connection URL is **jdbc:postgresql://localhost:5432/postgres**.
- ☐ 1.11. Enter **postgres** for both the Username and Password, then click the **Done** button.

- ☐ 1.12. Select **JBTravel** in the list of **Available Datasources**, then click the **Enable** button to enable the datasource.

☐ 2. Verify the Datasource Configuration

- ☐ 2.1. In the terminal windows of host2 and host3, look for the following log event:

```
[Server:production-server-A] 09:44:10,760
 INFO [org.jboss.as.connector.subsystems.datasources] (MSC service thread
 1-6) JBAS010400: Bound data source [java:jboss/JBTravelDatasource]
```

- ☐ 2.2. From the CLI, enter the following command:

```
/profile=ha/subsystem=datasources/data-source=JBTravel:read-resource
```

- ☐ 2.3. You can also open the **domain.xml** file for the Domain Controller on machine1. You should see a new **<datasource>** entry with **jndi-name** equal to **java:jboss/JBTravelDatasource**.

☐ 3. Test the Datasource

- ☐ 3.1. Deploy **dstest.war** to the **production-group** Server Group. The **dstest.war** file is found in your ***LABS/Lab06_02*** folder.
- ☐ 3.2. Point your browser to `http://192.168.0.xx:38080/dstest/` which is the **dstest** application on **production-server-A**.
- ☐ 3.3. Enter **java:jboss/JBTravelDatasource** for the JNDI name.
- ☐ 3.4. Enter **jbtravel.airport** for the table name.
- ☐ 3.5. Click the **Submit** button to test the datasource.
- ☐ 3.6. Read the results page and verify that the datasource lookup was successful. If the database connection worked, you will be looking at all the airports available in the **jbtravel** database.
- ☐ 3.7. Click the **Back** button in your browser and run the test again on the **test.person** table.

☐ 4. Modify the Datasource

In this step, you will configure some of the available settings of the connection pool using the CLI.

- ☐ 4.1. Enter the following commands to view the current settings of the JBTravel datasource:

```
cd profile=ha/subsystem=datasources/data-source=JBTravel
:read-resource(recursive=true)
```

Notice several of the attributes of the JBTravel datasource are undefined.

- ☐ 4.2. Enter the following command, which sets the minimum pool size of the JBTravel datasource to 5:

```
:write-attribute(name=min-pool-size,value=5)
```

- ☐ 4.3. Verify the change was made:

```
:read-resource(recursive=true)
```

- ☐ 4.4. Go to the **Profiles** page of the Management Console, and change the **Profile** to **ha**.
- ☐ 4.5. Select the **JBTravel** datasource in the list of **Available Datasources**, then disable it by clicking the **Disable** button.
- ☐ 4.6. Click on the **Pool** tab below the list of datasources. You should see that **Min Pool Size** is 5.
- ☐ 4.7. Click the **Edit** button and set the **Max Pool Size** to 20.
- ☐ 4.8. Select the checkbox labeled **Prefill enabled**.
- ☐ 4.9. Click the **Save** button to save your changes, then click the **Enable** button to enable the JBTravel datasource again.
- ☐ 4.10. Open the file **`domain.xml`** of your Domain Controller on machine1 using a text editor. Verify that the changes you made using the CLI and the Management Console appear in the **`domain.xml`** configuration file.

Lab 06_03: Define an XA Datasource

Performance Checklist

Lab Overview:

In this exercise, you will define an XA datasource for the PostrgeSQL database.

Lab Resources/Configuration:

Lab Files Location:	n/a
Application URL:	`http://192.168.0.xx:0000/console/App.html#datasources`

Success Criteria: After completing this exercise, you will have an XA datasource defined.

Outcome: An XA datasource with a JNDI named **`java:jboss/JBTravelXA`**.

Lab Outline:

1. Define the XA Datasource
2. Enable the Datasource
3. Verify the XA Datasource

☐ 1. Define the XA Datasource

☐ 1.1. Go to the **Datasource** page of the **ha** profile on your **Profiles** page of the Management Console.

☐ 1.2. Click on the **XA Datasources** tab to view the list of JDBC XA Datasources. The list will be empty.

☐ 1.3. Click the **Add** button to start the **Create XA Datasource** wizard. Enter **JBTravel_XA** for the **Name** and **java:jboss/JBTravelXA** for the JNDI name, then click **Next**.

☐ 1.4. Select the **postgresql-9.0-801.jar** driver.

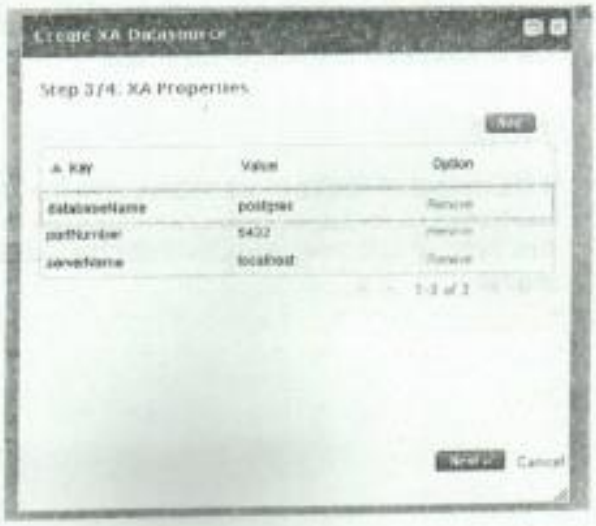
☐ 1.5. Enter **`org.postgresql.xa.PGXADataSource`** in the field labeled **XA Data Source Class**. Click **Next**.

☐ 1.6. In Step 3 of the wizard, define three XA properties (using the **Add** button):

```
serverName = localhost
```

```
portNumber = 5432
databaseName = postgres
```

The settings should look like:



Important

The settings you are entering in this lab are unique to the PostgreSQL driver class . If you are using a different driver, then check the Java documentation for that driver's XA datasource class to determine what the connection properties are.

- ☐ 1.7. Click the **Next** button. Enter postgres for both the **Username** and the **Password**.
- ☐ 1.8. Click the **Done** button to complete the wizard. You should see **JBTravel_XA** now in the list of **Available XA Datasources**.

☐ 2. Enable the Datasource

- ☐ 2.1. Select **JBTravel_XA** in the list of **Available XA Datasources** and click the **Enable** button.
- ☐ 2.2. Check the terminal window for both **host2** and **host3**. Verify that **java:jboss/JBTravelXA** has been bound on both **production-server-A** and **production-server-B**.

☐ 3. Verify the XA Datasource

- ☐ 3.1. Point your web browser to the **dstest** application: `http://192.168.0.xx:38080/dstest/`.
- ☐ 3.2. Enter **java:jboss/JBTravelXA** for the JNDI name.
- ☐ 3.3. Enter **jbtravel.airport** for the table name.
- ☐ 3.4. Click the **Submit** verify to test your XA datasource. If you see the following text on the web page:

```
Successfully looked up DataSource named java:jboss/JBTravelXA
Successfully connected to database.
Attempting query "SELECT * FROM ibtravel.airport"
```

followed by a list of airports, then your XA datasource is configured properly and is working! If you are getting an error message, check the output in the terminal window of **host2** for any helpful log errors.

Note

If the output of **host2** is not helpful, view the file **server.log** in your **machine2/domain/servers/production-server-A/log** folder. Any detailed error messages involving the datasource will appear in this log file.