Installing JD Edwards EnterpriseOne on Amazon RDS for Oracle

December 2016

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Foreword and Acknowledgement

I would like to take this opportunity to gratefully acknowledge the significant contribution from Marc Teichtahl, AWS Solutions Architect, and Shannon Moir, Lead Engineer at Myriad IT to this whitepaper.

Marc and Shannon were instrumental in researching and testing the installation of the JDE application with the connection to Amazon RDS for Oracle. They were able to document each step of the process. This document is a result of their hard work and meticulous attention to detail.

I cannot thank Marc and Shannon enough for their effort and hope for fruitful cooperation in the future.

- Tsachi Cohen

Manager, Software Development, DBS Relational

Abstract

Amazon Relational Database Service (Amazon RDS) is a flexible, cost-effective, easy-to-use service that makes it easy to run relational databases in the cloud.

In this whitepaper, we help you understand how to deploy Oracle's JD Edwards EnterpriseOne (version 9.2) using Amazon RDS for Oracle. Because this whitepaper focuses on the database components of the installation process, items such as JD Edwards EnterpriseOne application servers and application server node scaling will not be covered.

Introduction

There are two ways to deploy the Oracle database backend for a JD Edwards EnterpriseOne installation on Amazon Web Services (AWS): by using a database managed by the Amazon Relational Database Service (Amazon RDS),[1] or by deploying and managing a database on Amazon Elastic Compute Cloud (Amazon EC2)[2] infrastructure. This whitepaper focuses on the deployment of JD Edwards EnterpriseOne in an AWS environment using Amazon RDS for Oracle.

Why JD Edwards EnterpriseOne on Amazon RDS

Simplicity, scalability and stability are all important reasons to install the JD Edwards EnterpriseOne applications suite on Amazon RDS. Integrated high availability features provide seamless recoverability between AWS Availability Zones (AZs) without the complications of log shipping and Oracle Data Guard. Using RDS, you can quickly back up and restore your database to a point in time and also change the size of the server or speed of the disks— all within the AWS Management Console. Management advantages are also at your fingertips with the AWS mobile application.

All this, coupled with intelligent monitoring and management tools, provides a complete solution for implementing Oracle Database in Amazon RDS for use with JD Edwards EnterpriseOne. When designing your JD Edwards EnterpriseOne footprint, you can consider the entire lifecycle of JD Edwards EnterpriseOne on AWS, which includes complete disaster recovery. Instead of disaster recovery being an afterthought, it's encapsulated in the design fundamentals of "design for failure".

When your installation is complete, you can take backups, refresh subsidiary environments, and manage and monitor all critical aspects of your environment from the AWS Management Console. You can enable monitoring to ensure that everything is sized correctly and performing well.

Amazon RDS for Oracle is a great fit for JD Edwards EnterpriseOne. JD Edwards EnterpriseOne also provides for heterogeneous database support, which means that there is a loose coupling between Enterprise Resource Planning (ERP) and the database, allowing installation of Microsoft SQL Server, for example, as an alternative to Oracle.

Licensing

Purchase of JD Edwards EnterpriseOne includes the Oracle Technology Foundation component. Oracle Technology Foundation for JD Edwards EnterpriseOne provides all the software components you need to run Oracle's JD Edwards EnterpriseOne applications. Designed to help reduce integration and support costs, it's a complete package of the following integrated open standards software products that enable you to easily implement and maintain your JD Edwards EnterpriseOne applications:

- Oracle Database 12c Standard Edition 2
- Oracle Fusion Middleware
- JD Edwards EnterpriseOne Tools

If you have these licenses, you can take advantage of the Amazon RDS for Oracle "Bring-Your-Own-License" ("BYOL") option. Please refer to the Oracle Cloud Licensing Policy for more detail.[3]

> **Note:** With the "BYOL" option you may need to acquire additional licenses for standby database instances when running Multi-AZ deployments. Refer to the JD Edwards EnterpriseOne Licensing Information User Manual for a detailed description of the restricted use licenses provided in Oracle Technology Foundation for the JD Edwards EnterpriseOne product.[4]

Some historical JD Edwards EnterpriseOne licensing agreements do not include Oracle Technology Foundation. If that is the case for you, you can simply choose the Amazon RDS "License Included" option, which includes licensing costs in the hourly price of the service. If you have questions about any of your licensing obligations, please contact your JD Edwards EnterpriseOne licensing representative.

For additional details on licensing Oracle Database on AWS, please refer to the Oracle Cloud Licensing Policy.[5]

Performance Management

Instance Sizing

Increasing the performance of a database (DB) instance requires an understanding of which server resource is the performance constraint. If database performance is limited by CPU, memory, or network throughput,

you can scale up by choosing a larger instance type. In an Amazon RDS environment, this type of scaling is very easy.

Amazon RDS supports several DB instance types. At the time of this writing, instance types that support the Standard Edition 2 (SE2) socket requirements range from

- the small "micro" (db.t2.micro) to
- db.m4.10xlarge, which features 40 virtual cores, 160 GB of memory, 10 Gbps of network performance, and is Provisioned IOPS-optimized; and
- the memory-optimized db.r3.8xlarge with 32 virtual cores, 244 GB of memory, and 10 Gbps of network performance.

For current details and options, see the Amazon RDS for Oracle home page.[6]

The first time you start your Amazon RDS DB instance, choose the instance type that seems most relevant in terms of the number of cores and amount of memory. Using that as the starting point, you can then monitor the performance to determine whether it is a good fit or whether you need to pick a larger or smaller instance type.

You can modify the instance class for your Amazon RDS DB instance by using the AWS Management Console or the AWS command line interface (AWS CLI), or by making application programming interface (API) calls in applications written with the AWS Software Development Kit (SDK). Modifying the instance class will cause a restart of your DB instance, which you can set to occur right away or during the next weekly maintenance window that you specify when creating the instance (this setting can also be changed).

Increasing Instance Storage Size

Amazon RDS also makes it easy to scale up your storage without restarting the instance or interrupting active processes. The main reason to increase the Amazon RDS storage size is to accommodate database growth, but you can also do this to improve I/O. For an existing DB instance, you might observe some I/O capacity improvement if you scale up your storage.

Disk I/O Management—Provisioned IOPS

Provisioned IOPS (I/O operations per second) is a storage option that gives you control over your database storage performance by enabling you to

specify your IOPS rate. Provisioned IOPS is designed to deliver fast, predictable, and consistent I/O performance. At the time of this writing, you can provision from 1,000 IOPS to 30,000 IOPS, with corresponding storage ranging from 100 GB to 6 TB. You can start small and scale up in increments of 1,000 IOPS.

Here are some important things to keep in mind about Provisioned IOPS in Amazon RDS:

- The ratio between the amount of storage and Provisioned IOPS should be between 3 and 10. For example, a 100 GB database could have Provisioned IOPS set between 300 and 1,000.
- If you are using Provisioned IOPS storage, we recommend that you use DB instance types that are optimized for Provisioned IOPS. You can also convert a DB instance that uses standard storage to use Provisioned IOPS storage.
- The actual amount of your I/O throughput can vary, depending on your workload.

High Availability

The Oracle database provides a variety of features to enhance the availability of your databases. You can use the following Oracle Flashback technology features in both Amazon RDS and in Amazon EC2, which support multiple types of data recovery:

- **Flashback Table** recovers tables to a specific point in time. This can be helpful when a logical corruption is limited to one table or a set of tables instead of to the entire database.
- **Flashback Transaction Query** enables you to see all the changes made by a specific transaction.
- **Flashback Query** enables you to query any data at some point in time in the past.

In addition to these features, you should design a database architecture that protects you against hardware failures, data center problems, and disasters. You can do this by using replication technologies and the high availability features of Amazon RDS described in the following section.

High Availability Features of Amazon RDS

Amazon RDS makes it easy to create a high availability architecture. First, in the event of a hardware failure, Amazon RDS automatically replaces the compute instance powering your deployment. Second, Amazon RDS supports Multi-AZ deployments, where a secondary (or standby) Oracle DB instance is provisioned in a different Availability Zone (location) within the same region. This architecture allows the database to survive a failure of the primary DB instance, network, and storage, or even of the Availability Zone. The replication between the two Oracle DB instances is synchronous, helping to ensure that all data written to disk on the primary instance is replicated to the standby instance. This feature is available for all editions of Oracle, including the ones that do not include Oracle Data Guard, providing you with out-of-the-box high availability at a very competitive cost. **Error! Reference source not found.** shows an example of a high availability architecture in Amazon RDS.

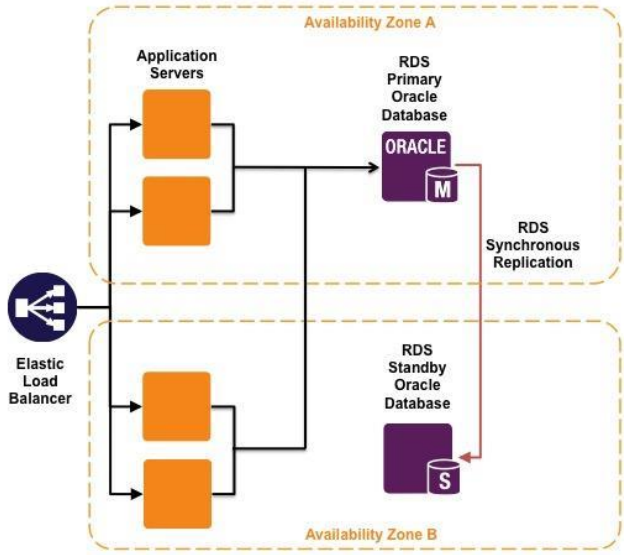


Figure 1: High availability architecture in Amazon RDS

You should also deploy the rest of the application stack, including application and web servers, in at least two Availability Zones to ensure that your applications continue to operate in the event of an Availability Zone failure. In the design of your high availability implementation, you can also take advantage of Elastic Load Balancing, which automatically distributes the load across application servers in multiple Availability Zones.[7]

A failover to the standby DB instance typically takes between one and three minutes and will occur in the event of any of the following:

- Loss of availability in the primary Availability Zone
- Loss of network connectivity to the primary DB instance
- Compute unit failure on the primary DB instance
- Storage failure on the primary DB instance
- Scaling of the compute class of your DB instance, either up or down
- Software patching

Running Amazon RDS in multiple Availability Zones has additional benefits:

- The Amazon RDS daily backups are taken from the standby DB instance, which means that there is usually no I/O impact to your primary DB instance.
- When you need to patch the operating system or replace the compute instance, updates are applied to the standby DB instance first. When complete, the standby DB instance is promoted as the new primary DB instance. The availability impact is then limited to the failover time, resulting in a shorter maintenance window.

Oracle Security in Amazon RDS

Amazon RDS allows you to control network access to your DB instances using security groups. By default, network access is limited to other hosts in the Amazon Virtual Private Cloud (Amazon VPC) where your instance is deployed.[8]

Using AWS Identity and Access Management (IAM), you can manage access to your Amazon RDS DB instances.[9] For example, you can authorize (or deny) administrative users under your AWS account to create, describe, modify, or delete an Amazon RDS DB instance. You can also enforce multi-factor authentication (MFA). For more information about using IAM to manage administrative access to Amazon RDS, see Authentication and Access Control for Amazon RDS.[10]

Amazon RDS offers optional storage encryption that uses AES-256 encryption and automatically encrypts any snapshots and snapshot restores. You can control who can decrypt your data by using AWS Key Management Service (AWS KMS).[11]

In addition, Amazon RDS supports several Oracle Database security features:

- Transparent Data Encryption (TDE) protects data at rest for customers who have purchased the Oracle Advanced Security Option. TDE provides transparent encryption of stored data to support your privacy and compliance efforts. Applications do not have to be modified and will continue to work as before. Data is automatically encrypted before it is written to disk and automatically decrypted when reading from storage. Key management is built in, which eliminates the task of creating, managing, and securing encryption keys. You can choose to encrypt tablespaces or specific table columns using industry-standard encryption algorithms, including Advanced Encryption Standard (AES) and Data Encryption Standard (Triple DES).
- Amazon RDS can protect data in motion using Secure Sockets Layer (SSL), or Native Network Encryption that protects data in motion using the Oracle Net Services. You can choose between AES, Triple DES, and RC4 encryption.
- Oracle Virtual Private Database (VPD) enables you to create security policies to control database access at the row and column level. Essentially, Oracle VPD adds a dynamic **WHERE** clause to an SQL statement that is issued against the table, view, or synonym to which an Oracle VPD security policy was applied. Oracle VPD enforces security to a fine level of granularity directly on database tables, views, or synonyms. Because you attach security policies directly to these database objects and the policies are automatically applied whenever a user accesses data, there is no way to bypass security.
- Fine Grained Auditing (FGA) can be understood as policy-based auditing. It lets you specify the conditions necessary to generate an audit record. FGA policies are programmatically bound to a table or view. They allow you to audit an event only when conditions that you define are true, for example, only if a specific column has been selected or updated. Because every access to a table is not always recorded, this creates more meaningful audit trails. This can be critical given the often commercially sensitive nature of the data retained in the JD Edwards EnterpriseOne backend databases.

Installing JD Edwards EnterpriseOne on an Amazon RDS for Oracle DB Instance

Installing JD Edwards EnterpriseOne is often seen as a complex task that involves setting up a server manager and the JD Edwards EnterpriseOne deployment server, followed by installing the platform pack.

In this section, we show you an alternative process for installing the platform pack, which is tailored to ensure a successful installation of JD Edwards EnterpriseOne on an Amazon RDS for Oracle database (DB) instance (referred to hereafter as an Oracle DB instance).

Assumptions

To install JD Edwards EnterpriseOne on Amazon RDS for Oracle, we assume the following:

- You are familiar with the JD Edwards EnterpriseOne installation process and have an understanding of the fundamentals of AWS architecture.
- You have a functional AWS account with appropriate IAM permissions.
- You have created an Amazon VPC with associated Subnet-Groups and Security-Groups and it is ready for use by the Amazon RDS for Oracle service.
- All configuration is executed from the "us-east-1" AWS Region.
- You have a local database on your deployment server that you can connect to with Oracle SQL Developer.

> **Note:** The deployment server will have two separate sets of Oracle binaries: a 32-bit client and a 64-bit server engine (named **e1local**).

Preparation

The process described in this whitepaper is based on the standard JD Edwards EnterpriseOne installation processes, which are described in the JD Edwards EnterpriseOne Applications Installation Guide.[12]

Prior to continuing, follow the standard JD Edwards EnterpriseOne Applications Installation Guide until section 6.8, "Understanding the Oracle Installation".

Once complete, follow the rest of the instructions in this whitepaper to successfully install JD Edwards EnterpriseOne on an Oracle DB instance.

Key Installation Tasks

The key elements of installing JD Edwards EnterpriseOne on an Oracle DB instance include:

- Creating the instance
- Configuring the SQL *Plus Instant Client
- Installing the platform pack
- Modifying the original installation scripts that are provided

Creating Your Oracle DB Instance

Using the AWS Management Console, follow these steps.

1. Select **Services** from the top menu bar.
2. Navigate to **RDS**.

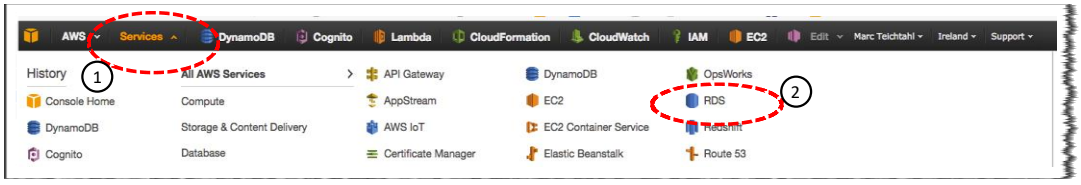


Figure 2: Navigating to the RDS Dashboard

This opens the Amazon RDS dashboard (see Figure 3) where you will create your Oracle DB instance.

3. Select **Launch a DB Instance** to continue.

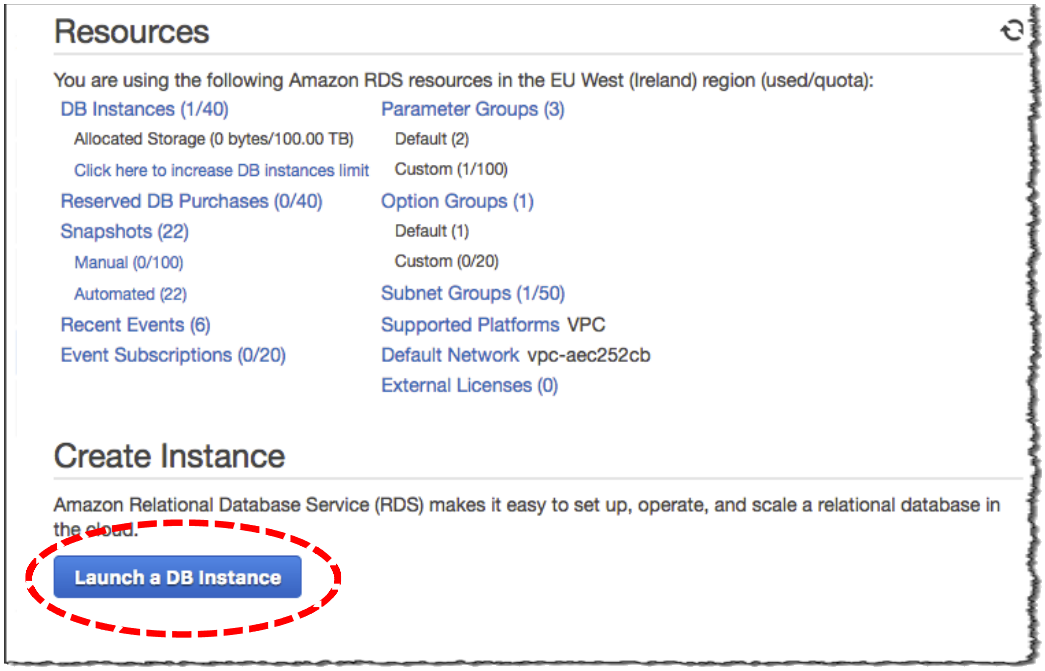


Figure 3: Launching a DB instance

4. To create an Oracle SE2 environment from the Amazon RDS Engine Selection screen, do the following:

   a. Choose **Oracle**.

   b. Choose **Select** next to **Oracle SE Two**.

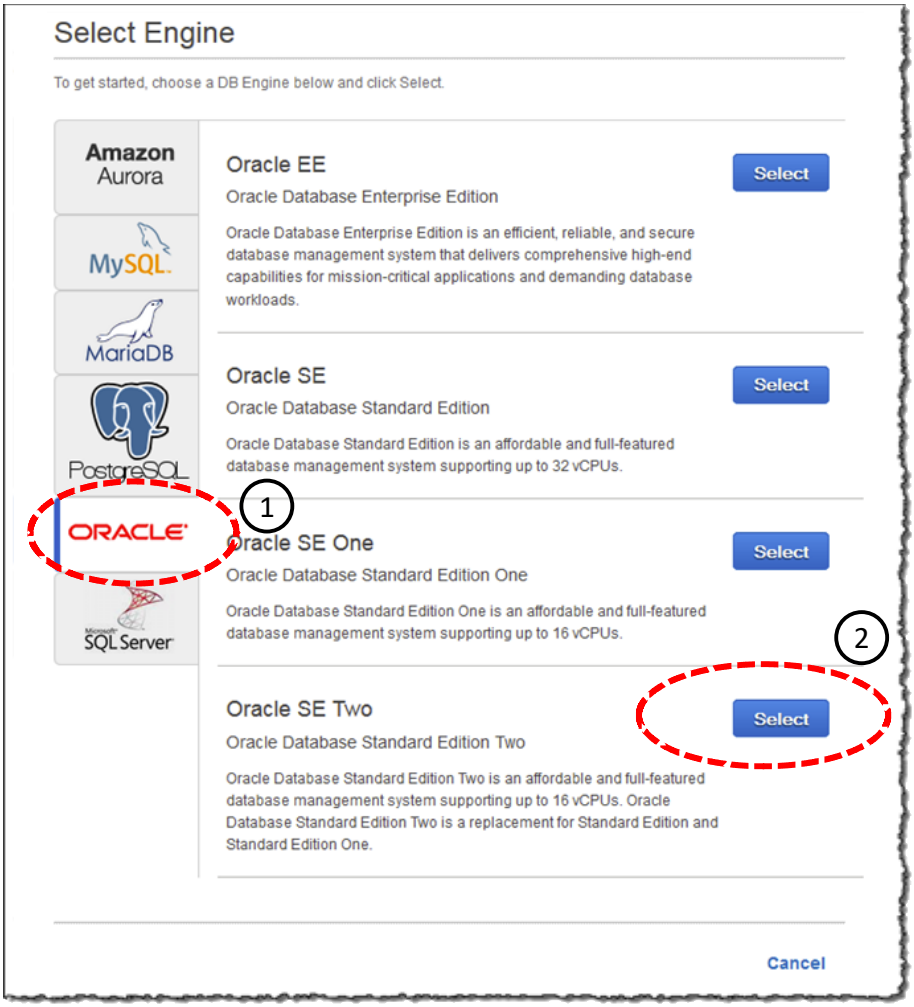


Figure 4: Choosing Oracle SE Two for the engine

5. The AWS Management Console recommends the default values for a production-ready environment or a development environment. For the purposes of this whitepaper, you will use a production environment. Select **Oracle SE Two** under **Production**, and then choose **Next Step**.

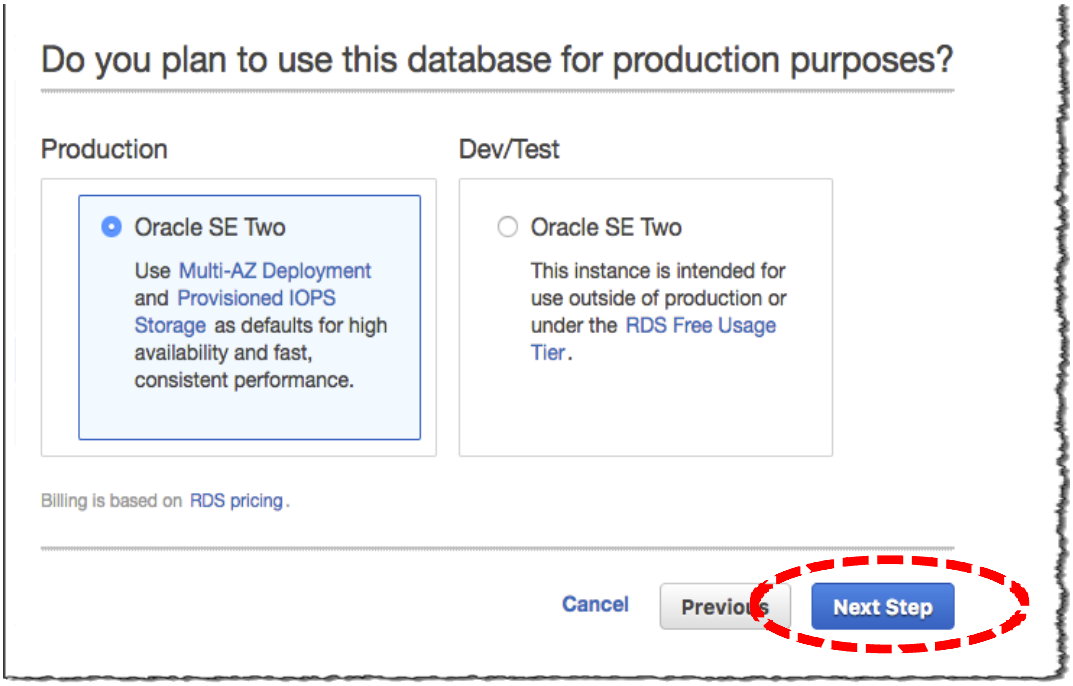


Figure 5: Using a production environment

6. On the **Specify DB Details** page, configure the Oracle DB instance in **Instance Specifications**.

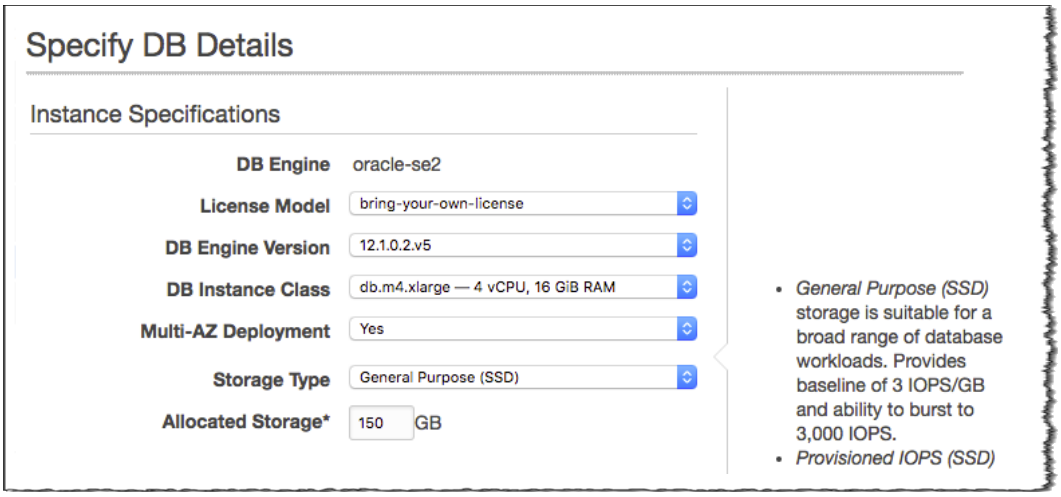


Figure 6: Configuring the Oracle DB instance

For the purposes of this whitepaper, use the settings in Table 1. These settings can be tailored to meet your specific requirements. We encourage you to consult with a JD Edwards EnterpriseOne supplier to ensure these settings are appropriate for your specific use case.

Table 1: Instance Specifications settings to use

DB Engine	oracle-se2 [a]
License Model	bring-your-own-license
DB Engine Version	12.1.0.2v5
DB Instance Class	db.m4.xlarge – 4 vCPU, 16 GiB RAM
Multi-AZ Deployment	Yes
Storage Type	General Purpose SSD
Allocated Storage	150 Gb
a) oracle-se2 must be used in compliance with the latest Oracle licensing. Please contact Oracle should further information be required.	

7. Proceed to update the **Instance Specifications** accordingly.
8. In **Settings**, configure the Oracle DB instance identifier and the database master username and password. For the purposes of this whitepaper, use the settings in Table 2. Then choose **Next Step**.

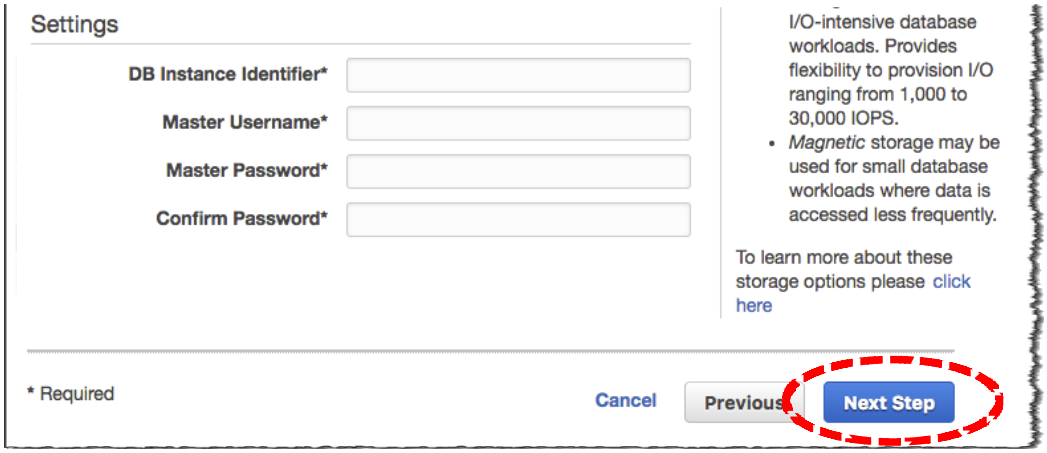


Figure 7: Configuring Oracle DB instance identifier and database master username and password

Table 2: Settings for DB instance identifier, username, and password

DB Instance Identifier	jde92poc
Master Username	jde92pocMaster
Master Password	jde92pocMasterPassword

Note: The rest of this procedure assumes that you have already created a VPC to accommodate the Amazon RDS for Oracle installation, and that the VPC name used is **JDE92**. If you need help, you can find an AWS CloudFormation template at http://jde92poc.s3-website-ap-southeast-2.amazonaws.com/jde_cfn.zip.[13]

On the next screen, you can use the **Configure Advanced Setting** section to configure the appropriate network and security, database, backup, monitoring, and maintenance as follows.

9. In **Network & Security**, use the preconfigured VPC (**JDE92**) and the settings shown in Figure 8. If you have appropriately configured Subnet Groups and VPN Security Groups, you can use them here.

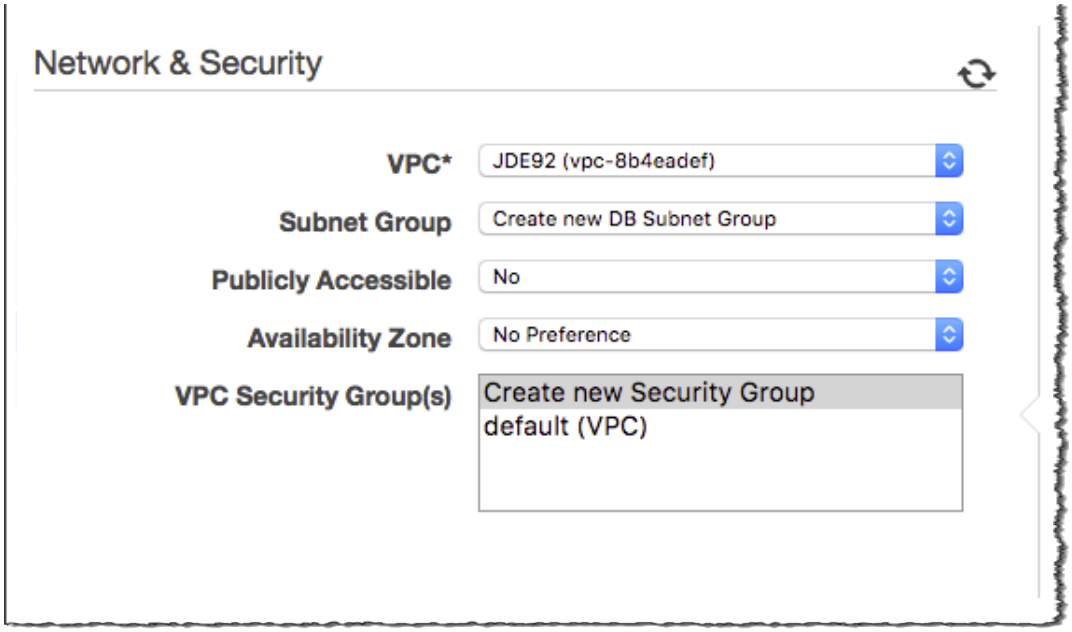


Figure 8: Configure network and security settings

10. In **Database Options**, use the following settings.

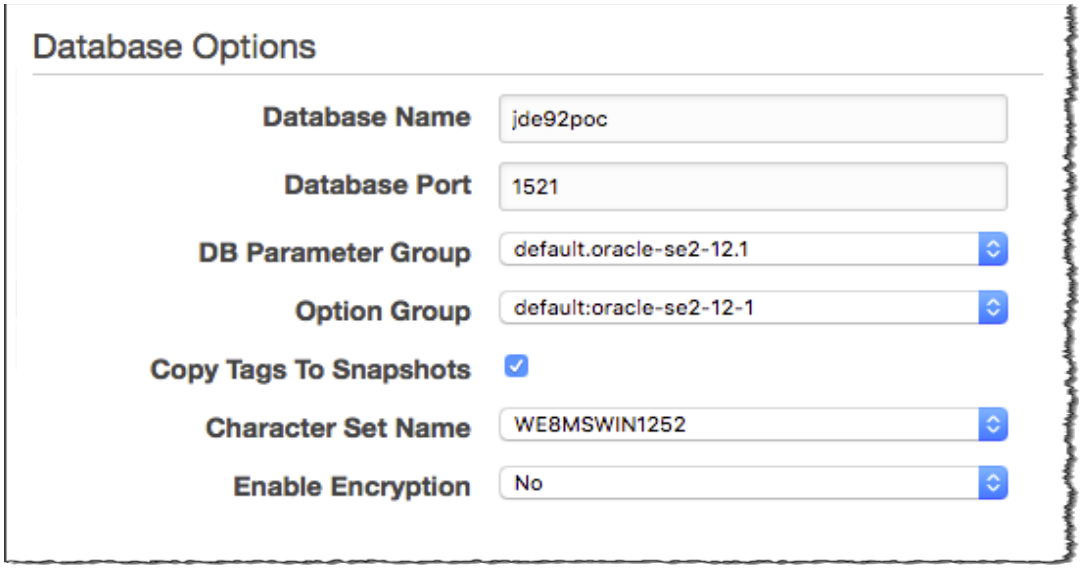


Figure 9: Configuring database options

11. In the **Backup**, **Monitoring**, and **Maintenance** sections you can use the default settings. However, because they do not impact the ability to install JD Edwards EnterpriseOne, we encourage you to experiment with and test these settings. Choose **Launch DB Instance** to launch the Oracle DB instance.

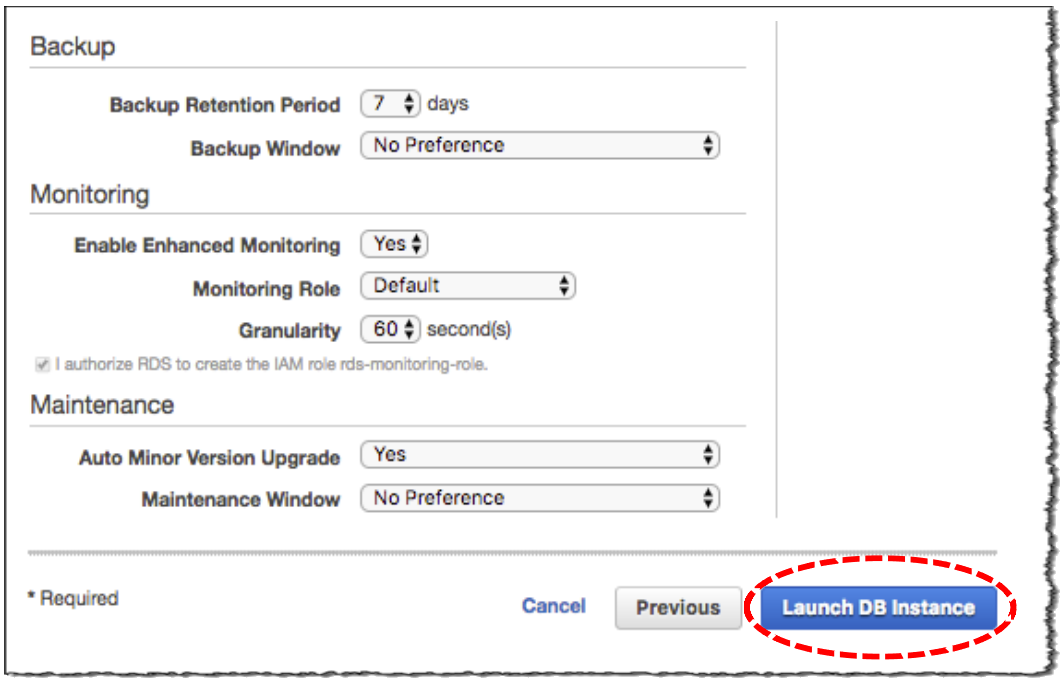


Figure 10: Selecting optional preferences

12. Creation of the Oracle DB instance begins. This can take some time to complete. Choose **View Your DB Instances** to view the progress.

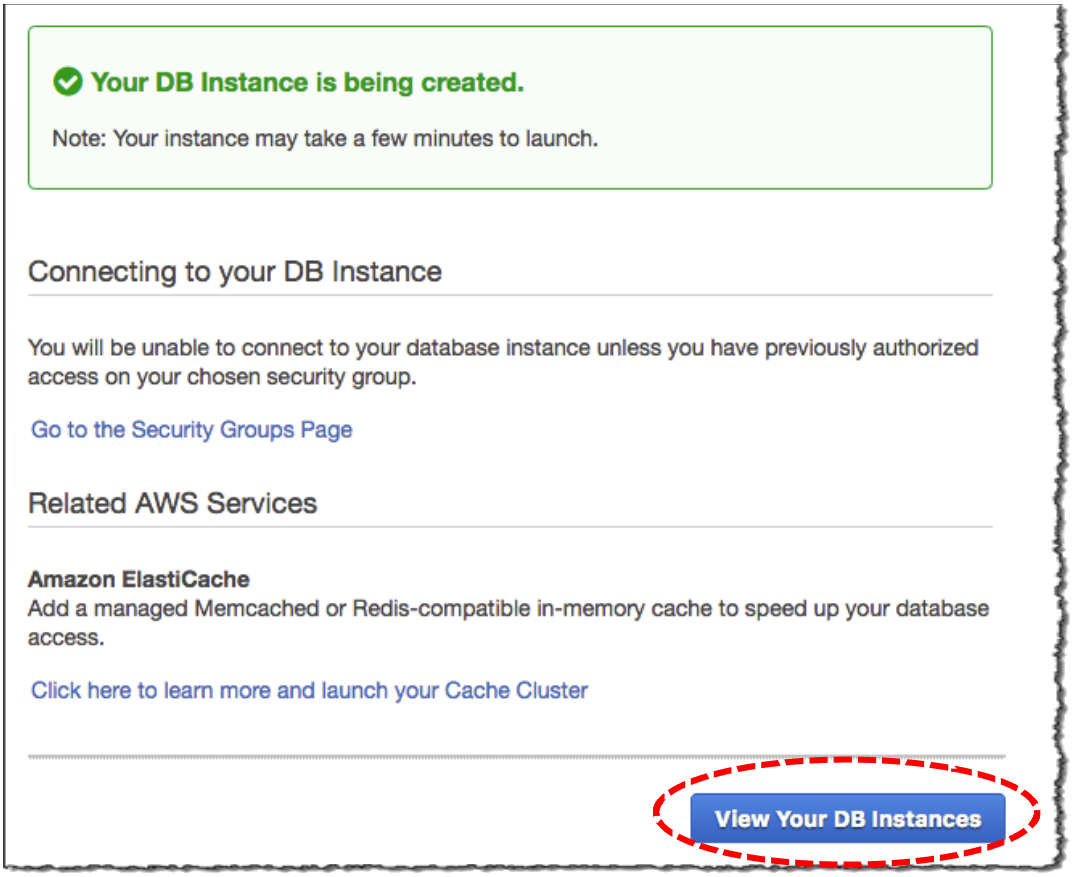


Figure 11: Viewing creation progress

13. Choose the **refresh** icon to watch the progress of the Oracle DB instance creation.

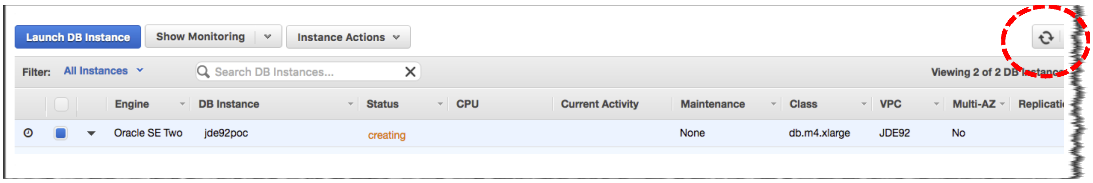


Figure 12: Refreshing the progress view

When the Oracle DB instance is available for use, the **Status** changes to **available**.

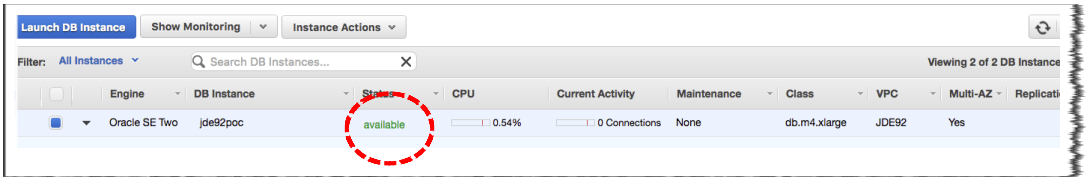


Figure 13: Oracle DB instance is available for use

Connecting to Your Oracle DB Intsance

When Amazon RDS creates the Oracle DB instance, it also creates an endpoint. Using this endpoint, you can construct the connection string required to connect directly with your Oracle DB instance.

To allow network requests to your running Oracle DB instance, you will need to authorize access.[14] For a detailed explanation of how to construct your connection string and get started, see Amazon RDS User Guide.[15]

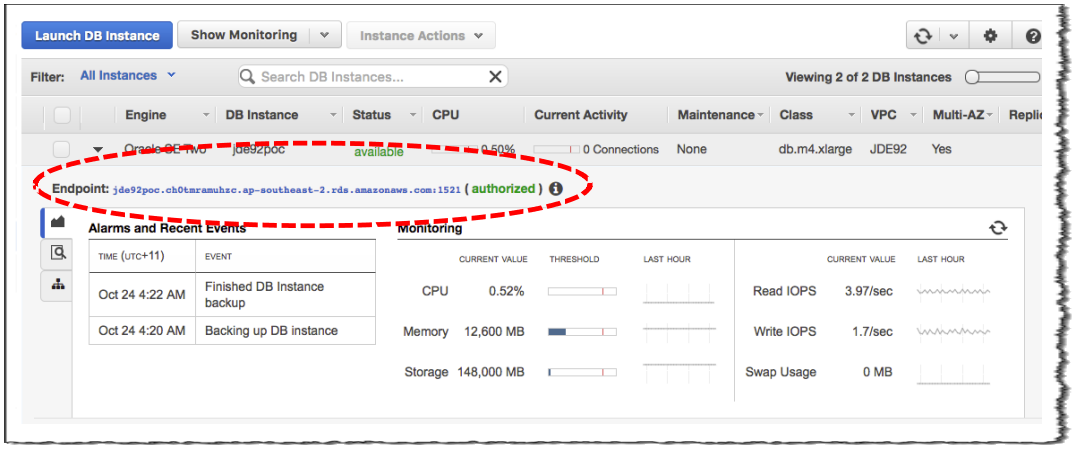


Figure 14: Endpoint for the Oracle DB instance

The endpoint is allocated a Domain Name System (DNS) entry, which you can use for connecting. However, to facilitate a better installation experience of JD Edwards EnterpriseOne, a CNAME record is created so that the endpoint can be more human-readable.

The CNAME should be created in the Amazon Route 53 local internal zone and should point to the new Oracle DB instance.

> **Note:** Creating an Amazon Route 53 record set is beyond the scope of this document. For more assistance, see the Amazon Route53 User Guide.[16]

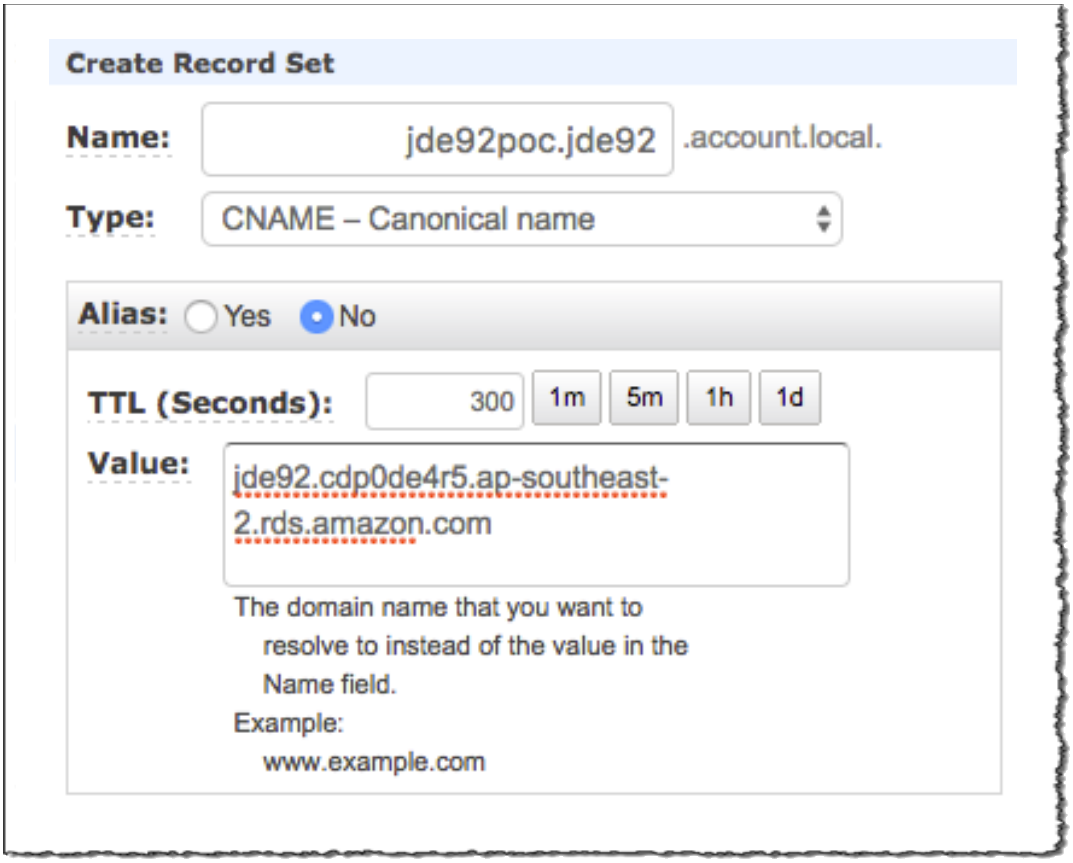


Figure 15: CNAME record set

To ensure that connectivity is permitted from the internal subnets in both Availability Zones, you will need to edit the security group for the Oracle DB instance.

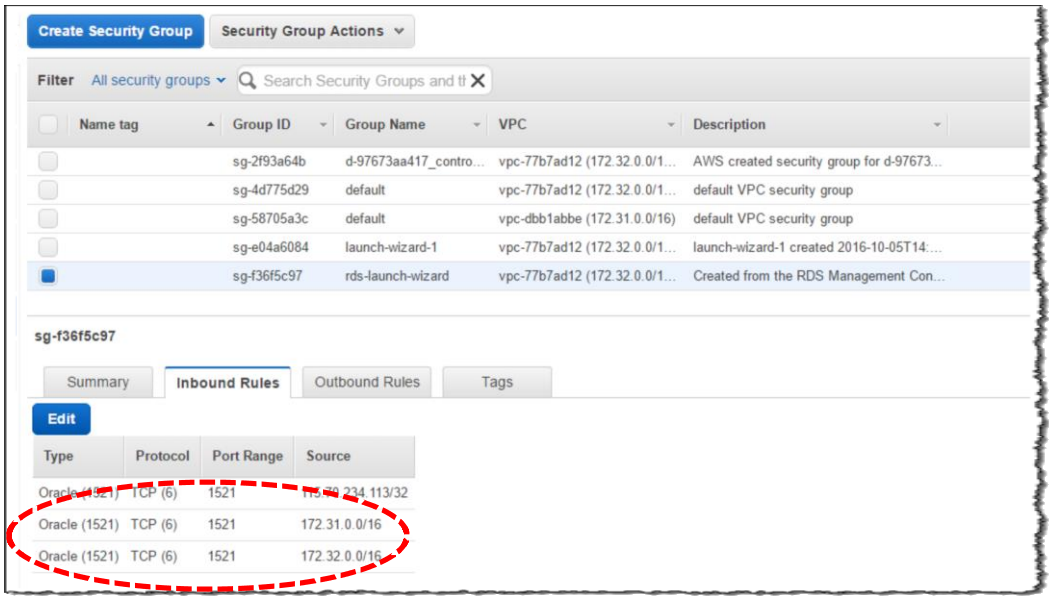


Figure 16: Updating the security group

Configure SQL Developer

Oracle SQL Developer is used to validate that the appropriate connectivity and permissions are in place and that the Oracle DB instance is accessible. SQL Developer is installed by default with your Oracle client; however, you can download a standalone version of SQL Developer here.[17]

The configuration information used to create the Oracle DB instance will be used as the SQL Developer configuration parameters (shown below) that are required to connect to the Oracle DB instance.

1. In the **New/Select Database Connection** dialog box, choose **Test** to perform a test connection to the Oracle DB instance.

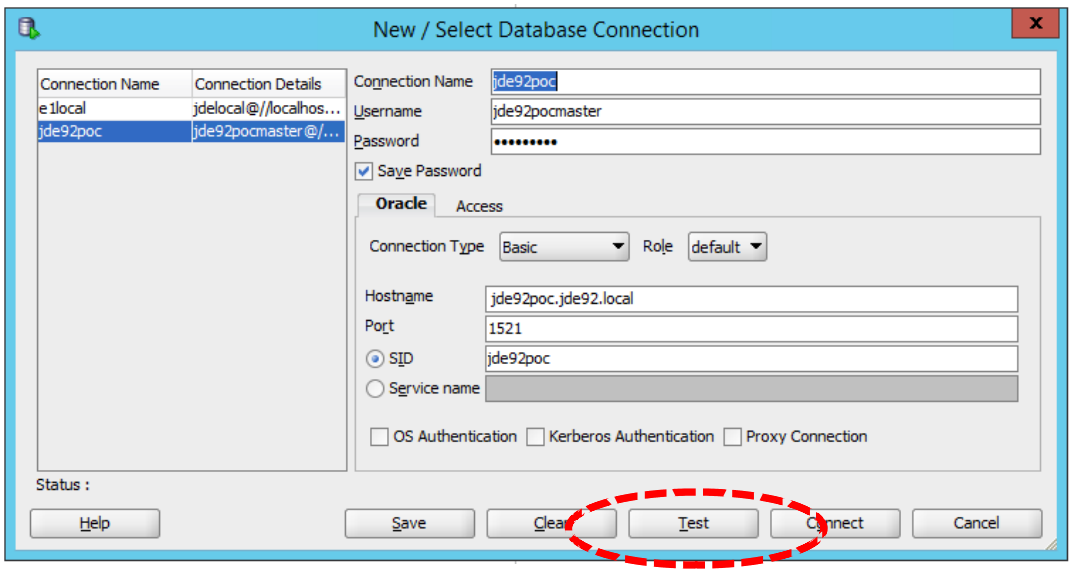


Figure 17: Testing the connection to the Oracle DB instance

A status of **Success** indicates that the test connection has executed and successfully connected to the Oracle DB instance.

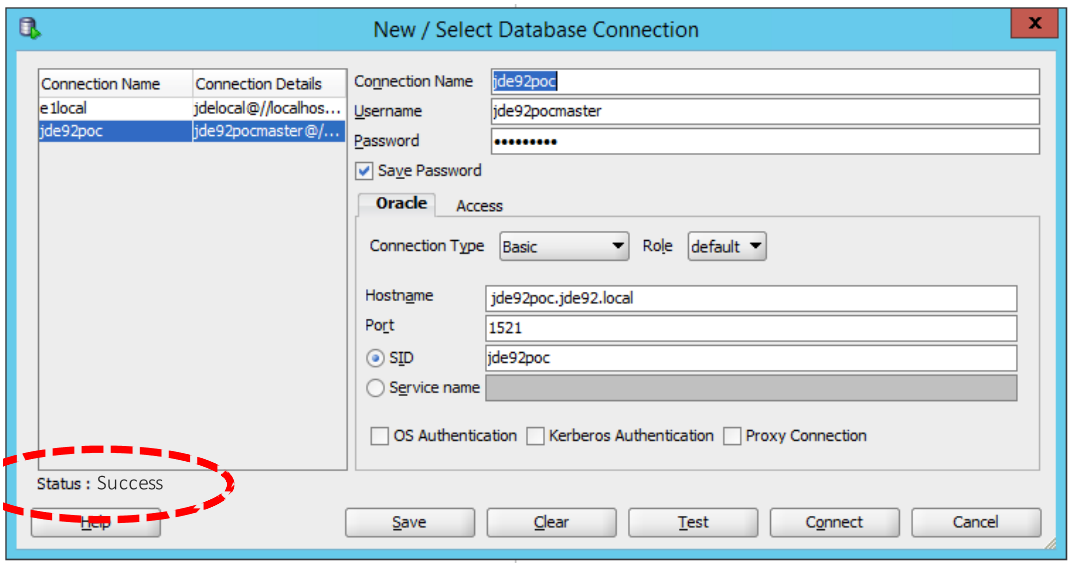


Figure 18: Test connection success

At this point, connectivity to both **e1local** and **jde92poc** has been proven using the default 64-bit drivers supplied with SQL Developer.

> **Note:** The 64-bit driver is selected by default due to the order of the client drivers in the **Servers** environment variable.

2. To check the deployment server path variables, in File Explorer (assuming Microsoft Windows 10), right-click **This PC** and choose **Properties**.

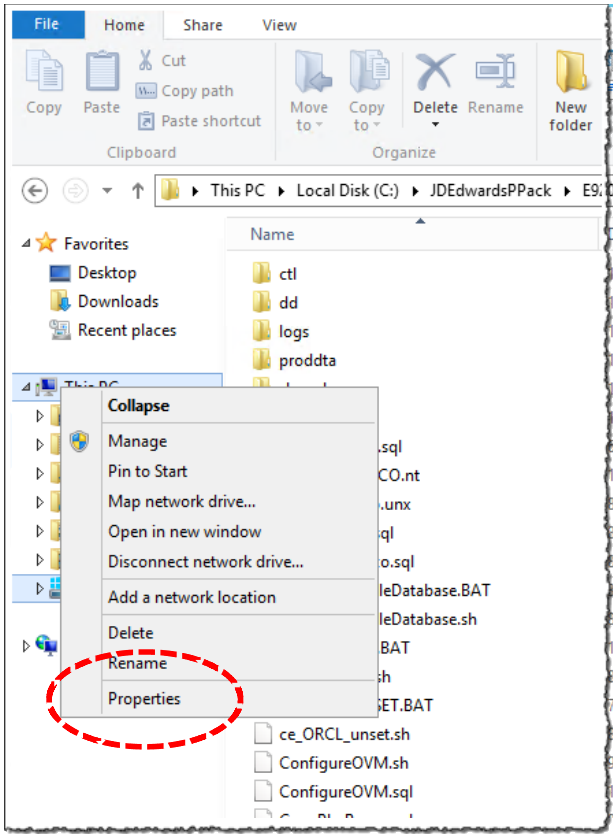


Figure 19: Open Properties to check the deployment server path variables

3. On the **Advanced** tab, choose **Environment Variables**.

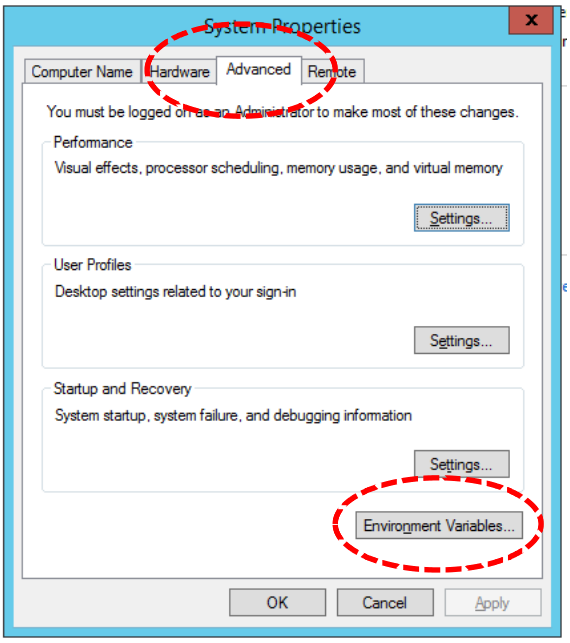


Figure 20: Accessing environment variables

4. Locate the **Path** environment system variable in the list.

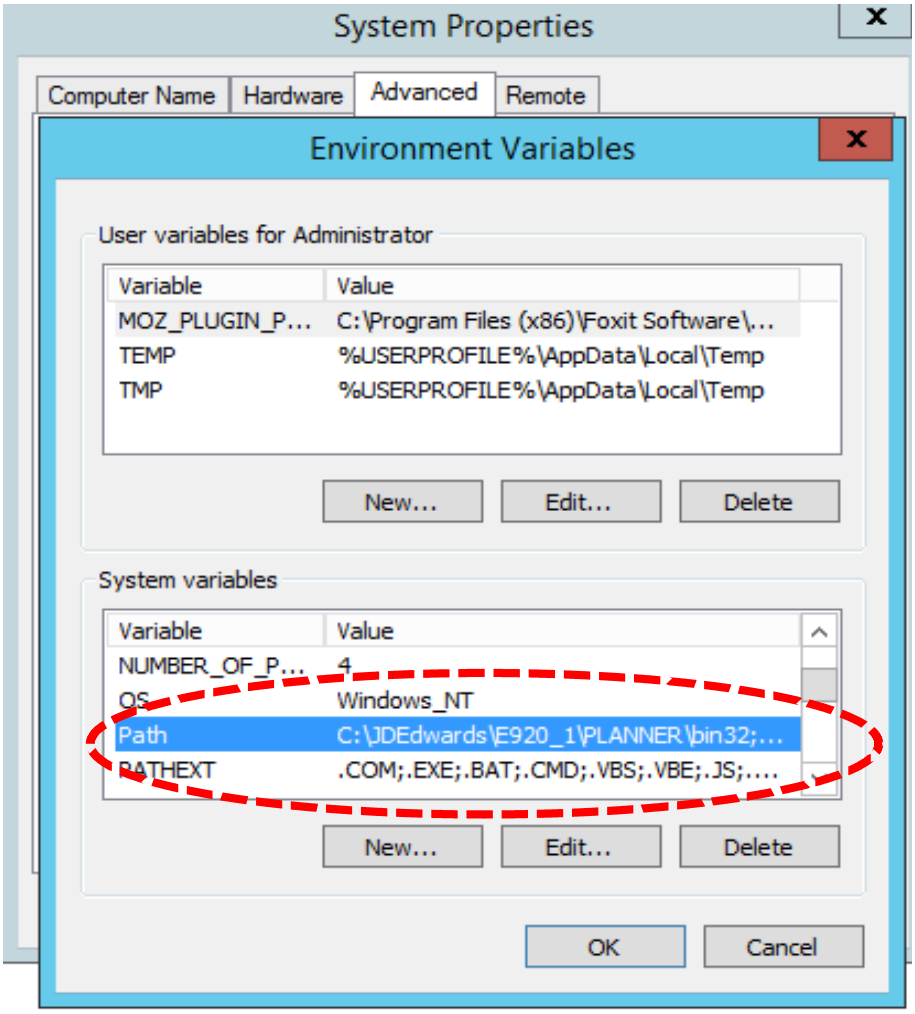


Figure 21: Path system variable

This allows the observation of the **Path** environment system variable. The following example shows the 64-bit binaries listed before the 32-bit binaries for Oracle.

```
"C:\JDEdwards\E920_1\PLANNER\bin32;C:\JDEdwards\E920_1\system\bin32;
C:\Oracle64db\E1Local\bin;C:\app\e1dbuser\product\12.1.0\client_1\bi
n;C:\ProgramData\Oracle\Java\javapath;%SystemRoot%\system32;%SystemR
oot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerSh
ell\v1.0\;C:\Program  Files\Amazon\cfn-bootstrap\;C:\Program
Files\Amazon\AWSCLI\"
```

5. To ensure that the remainder of the installation process works, it is critical that SQL*Plus works correctly—specifically, name resolution with **tnsnames.ora**. From the deployment server EC2 instance, open a command window and enter the following command.

```
tnsping e1local
```

As shown here, the file used for **tnsping** is located in the **C:\Oracle64db\E1Local\network\admin** folder.

Figure 22: tnsping file location

It is from within this directory that you'll make changes to the **tnsnames.**ora file, specifically, configuration of the **e1local** database (64-bit installation).

6. This step relates to the 64-bit libraries, not to the libraries that the JD Edwards EnterpriseOne deployment server code uses.

   The JD Edwards EnterpriseOne deployment server code uses 32-bit executables and the **tnsnames.ora** file on the client side to connect to databases (which are 64-bit). These are located in **C:\app\e1dbuser\product\12.1.0\client_1\network\admin** for this reference installation.

   Ensure that the Oracle DB instance is in the **tnsnames.ora** file in both locations (32-bit and 64-bit).

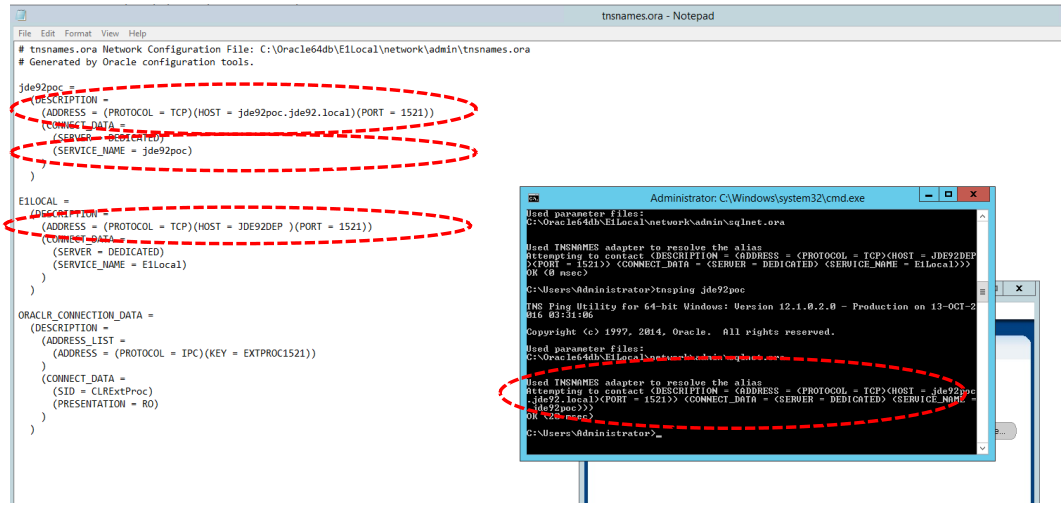


Figure 23: Checking tnsnames.ora file locations

In order to proceed, you must be able to SQL*Plus to the Oracle DB instance using **tnsnames.ora**.

Installing the Platform Pack

The platform pack will be run from the deployment server, connecting to a remote database.

To proceed, you need the Oracle Platform Pack for Windows. You can obtain it from https://edelivery.oracle.com with the appropriate MOS (My Oracle Support) login.

The following screenshots are taken from the Oracle Platform Pack installation wizard. In the this section, the installation directory is **C:\software\windowsPlatformPack\install**.

1. To run the Java-based installation program for the Oracle Platform Pack for Windows, execute **setup.exe** from within the installation directory. The following initial screen is displayed.

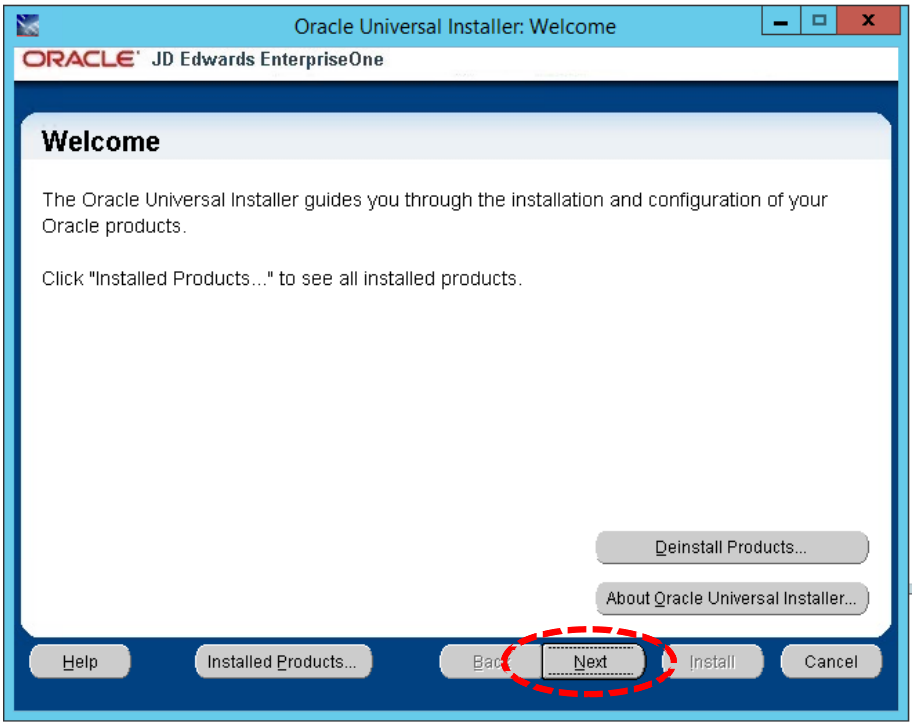


Figure 24: Oracle Universal Installer

2. Choose **Next**.
3. Only a database installation is required, so choose **Database**, and then choose **Next**.

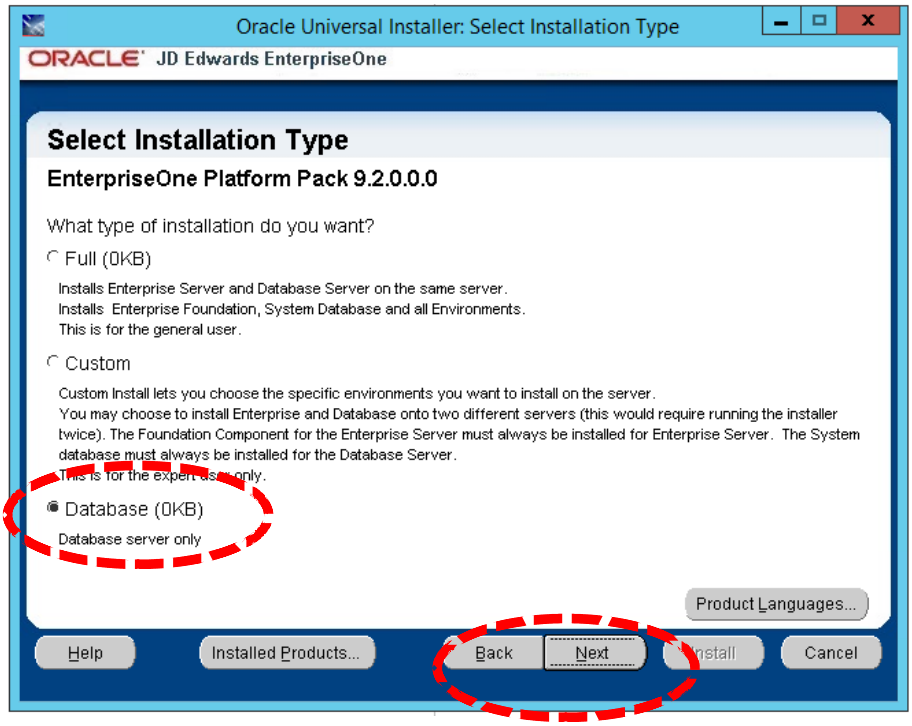


Figure 25: Choose database installation only

4. On the next screen, the **Name** field can be left as the suggested default. However, you should choose where to locate the installer files, based upon the installation preferences. This is a temporary location, and you can remove these files once the database is populated. Choose **Next**.

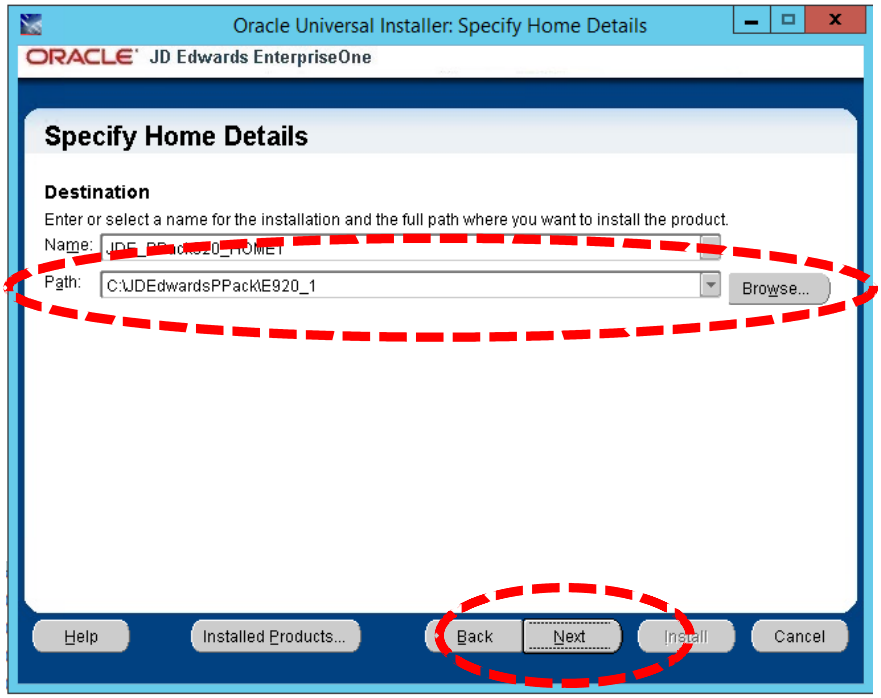


Figure 26: Temporary location for installer files

5. Because this is a new installation, choose **Install**, and then choose **Next**.

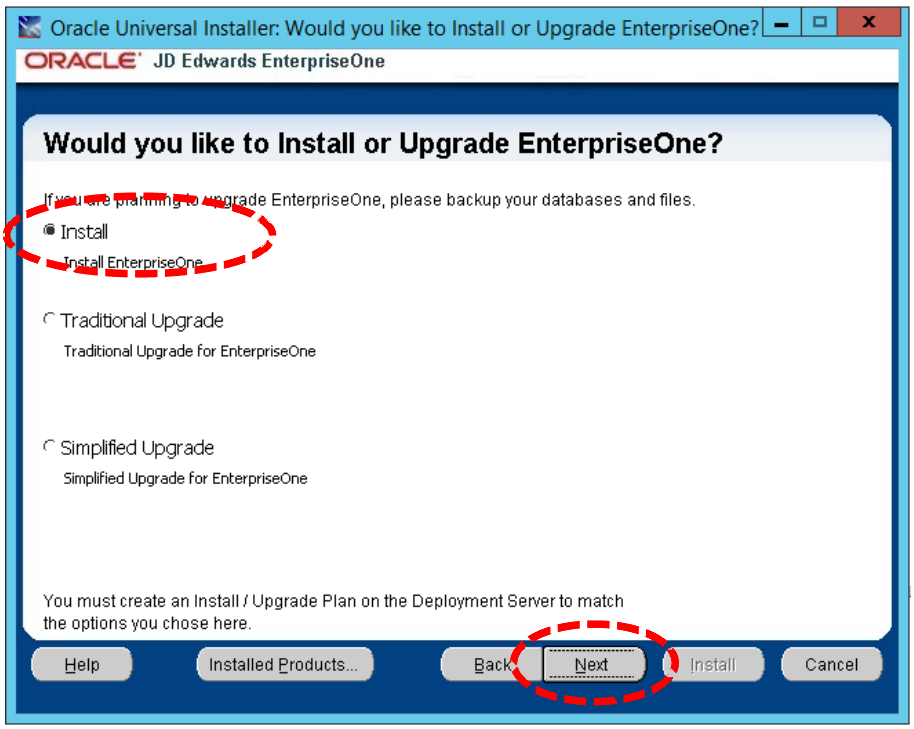


Figure 27: Starting a new installation

6. You need to provide the details of the database server; however, the database server name is not important, and the name of the deployment server can be used. Choose **Next**.

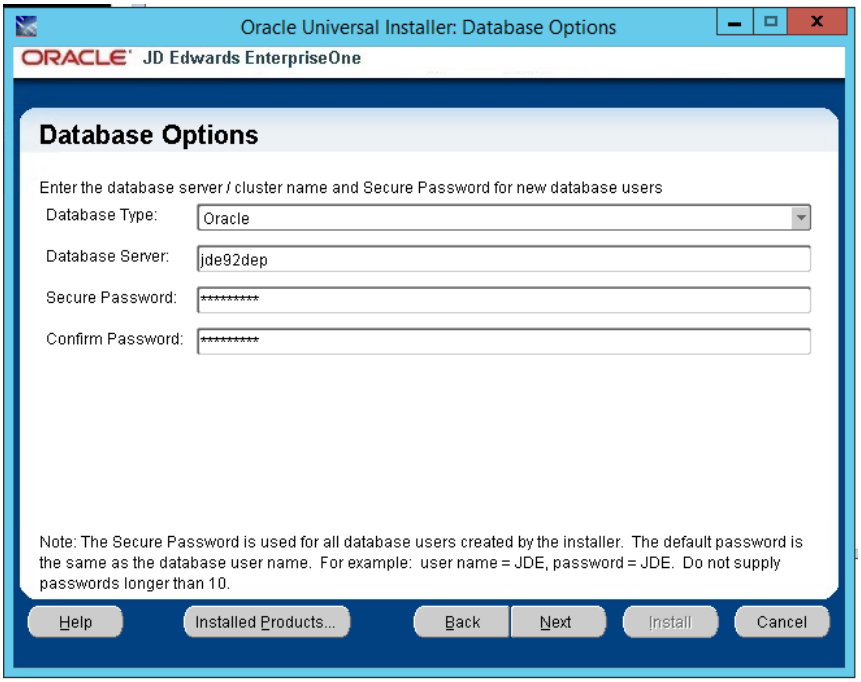


Figure 28: Database server details

7. When providing the **Administration and End User Roles**, use the defaults and choose **Next**.

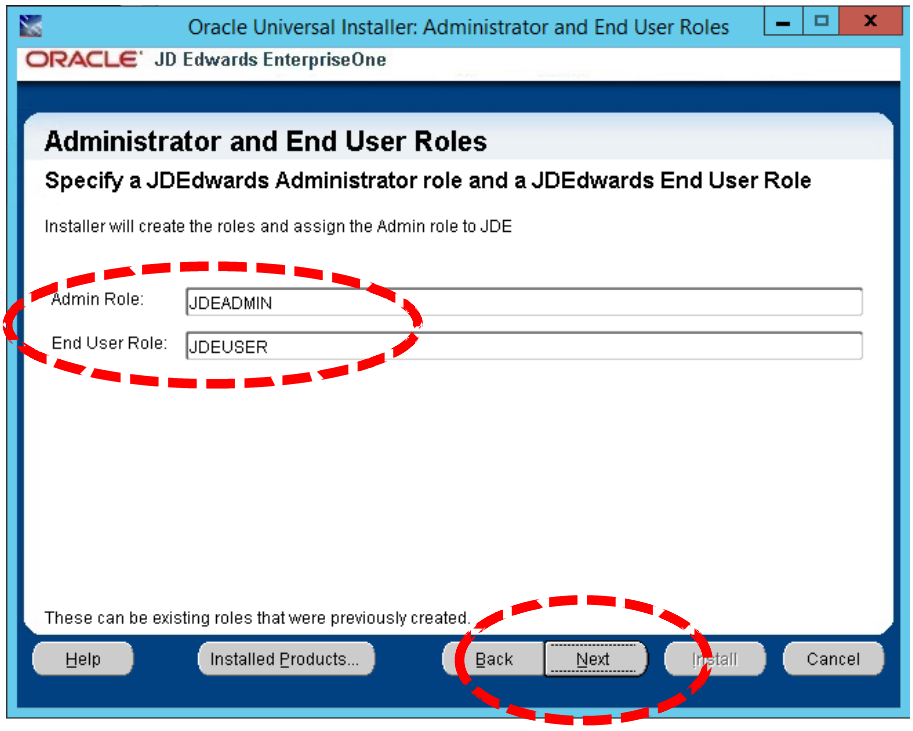


Figure 29: Default administrator and end user roles

The following warning is appropriate for our installation.

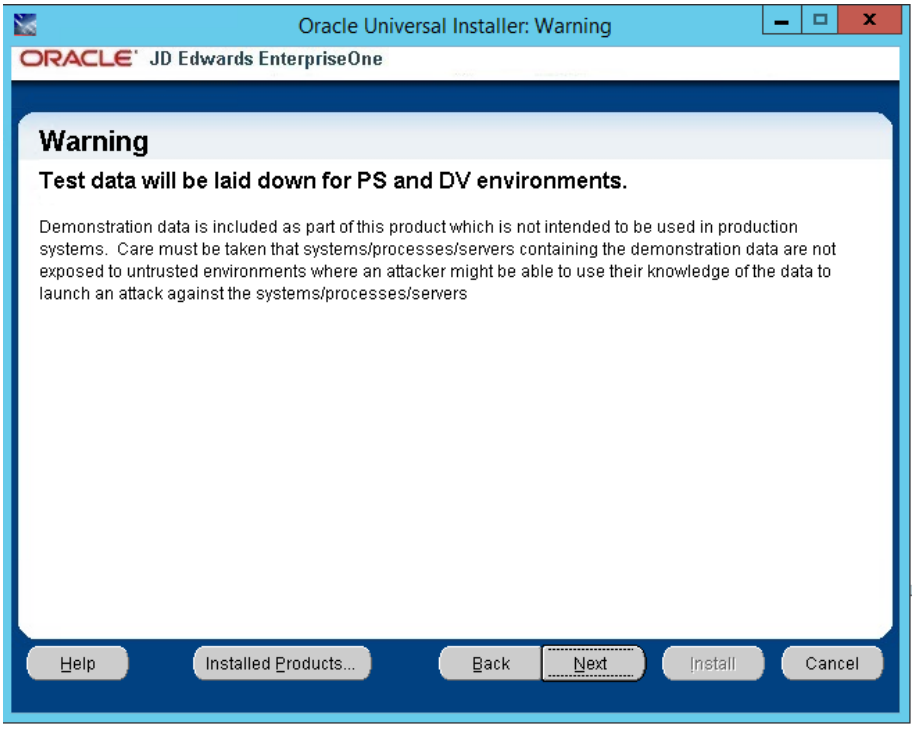


Figure 30: Warning for test data

8. Choose **Next**.

   At this stage, you can ignore the Database Server name warning.

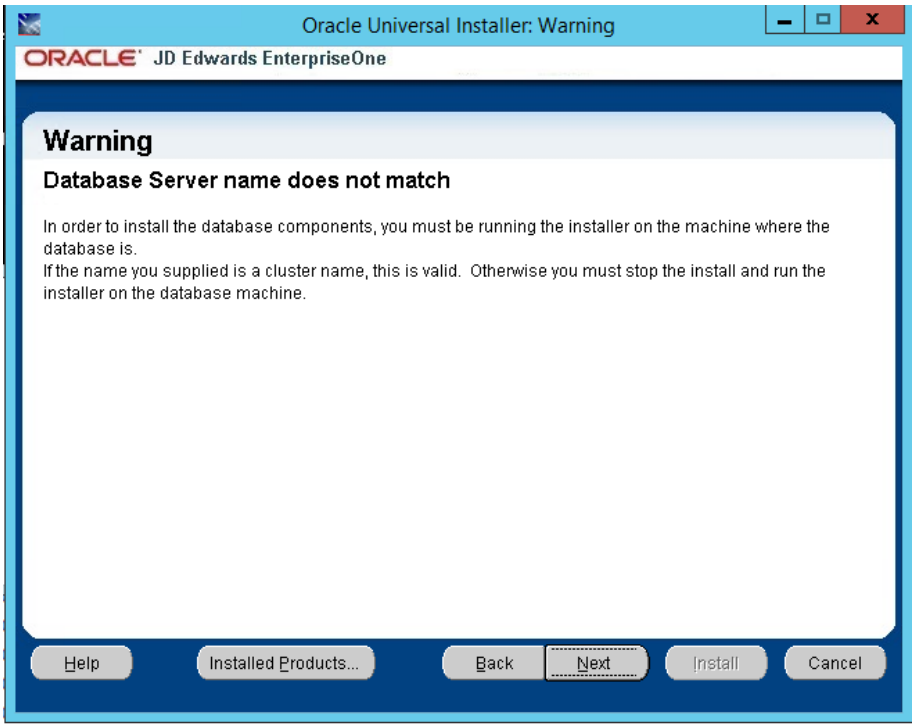


Figure 31: Database server name warning

   The following steps address this warning. Configuration for the Oracle DB instance and a username and password are supplied on the form. Unique string identifiers are provided for the tablespace directory (**c:\tablespace001**) and the Index tablespace directory (**c:\indexspace001**). These will be replaced at a later stage of the installation process.

9. Choosing **Run Scripts Manually** defers the execution of the installation scripts. Should the installation scripts execute at this stage, the installation will fail. Choose **Next**.

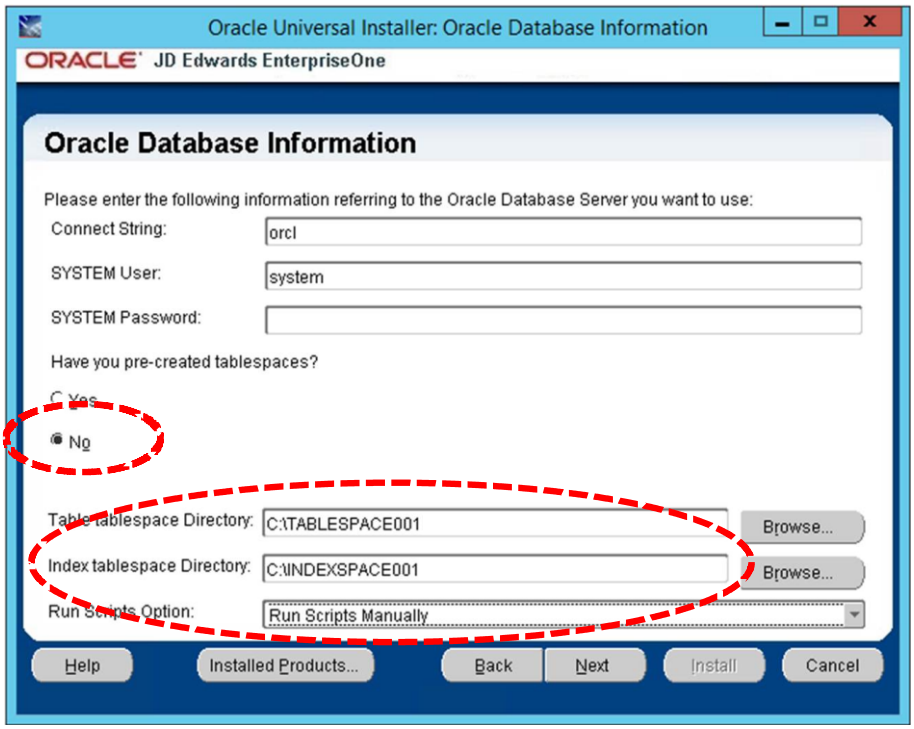


Figure 32: Choosing to run scripts manually

The installation process will attempt to connect to **jde92poc** using the information you provided. This connection must succeed in order for the installation to proceed. The following indicates that the installation process was able to connect to the Oracle DB instance specified (**jde92poc**).

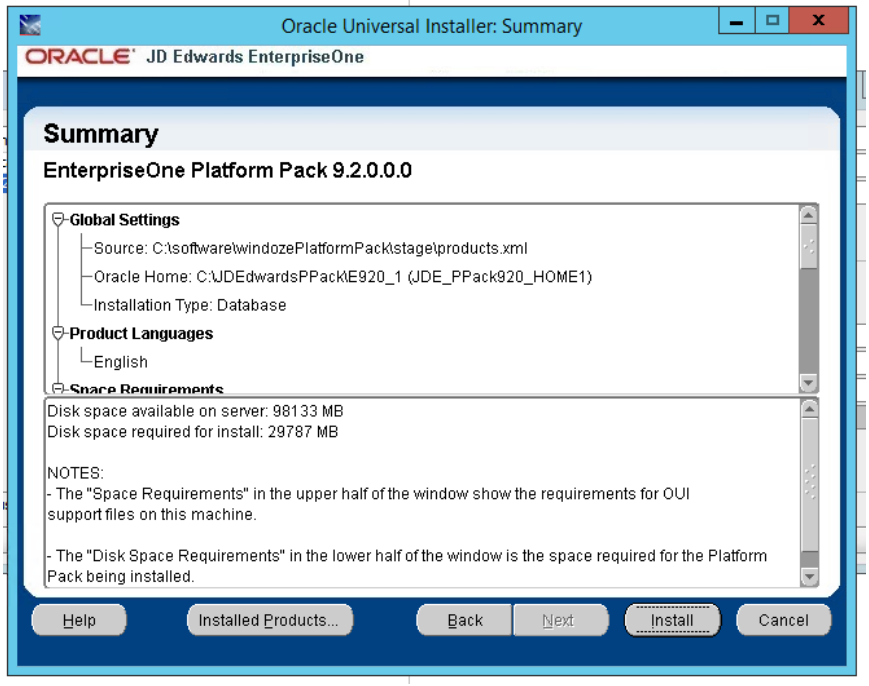


Figure 33: Installation process connected to the Oracle DB instance

10. Choose **Install**.

    The installation process starts, as shown in the following screenshot, and creates a set of specific database installation scripts for the options selected throughout the platform pack installation wizard.

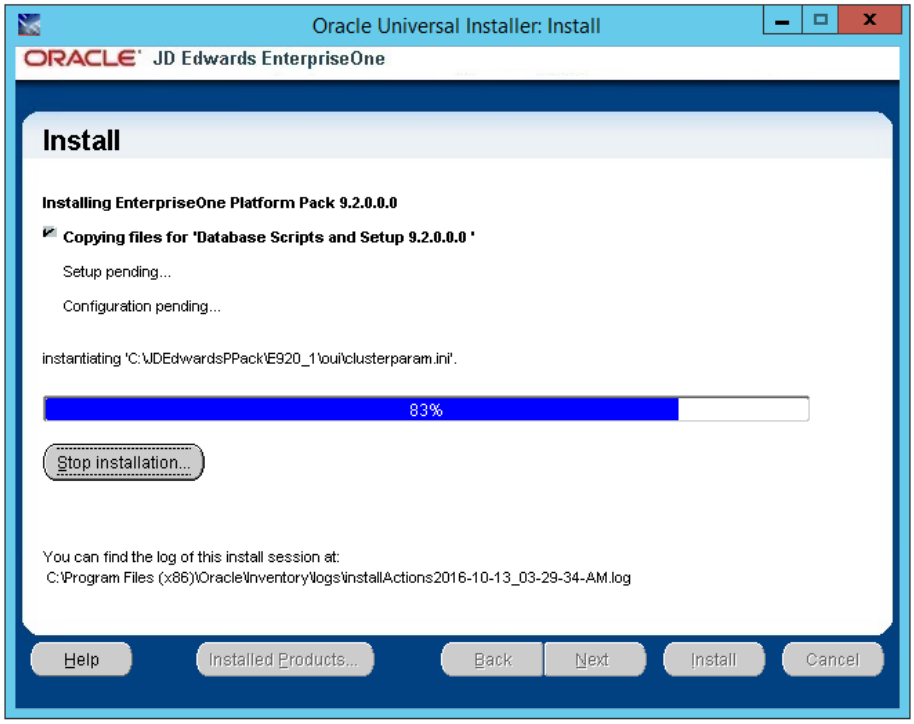


Figure 34: Installation begins

Once complete, instead of the vanilla scripts, all of custom values you provided are configured. Remember that the option **Run Scripts Manually** was selected, so the database is not loaded but scripts are created specifically for the current input parameters.

As the installation process proceeds, you can view logging at **C:\JDEdwardsPPack\E920_1**.

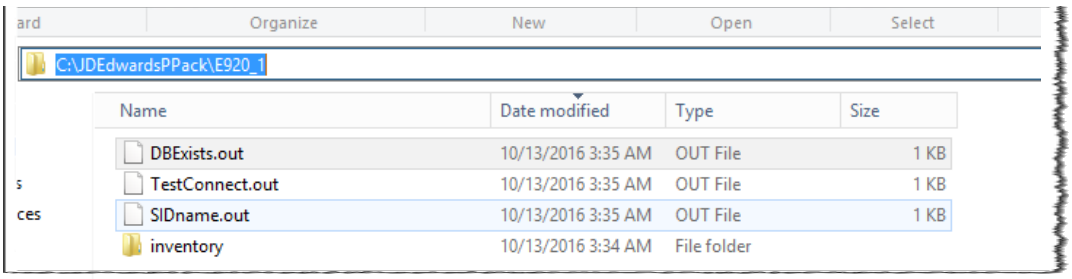


Figure 35: Logging information

Modifying the Default Scripts

After modifying the default scripts, all of "post installation wizard" installation scripts are created; however, it is assumed that they are going to run on the database server itself. As a result, you need to modify these scripts to ensure a seamless installation on the Oracle DB instance.

When you view the specified installation directory (**C:\JDEdwardsPPack\E920_1**), notice that a folder structure such as the following one shown was created. You will make the required modifications within this directory.

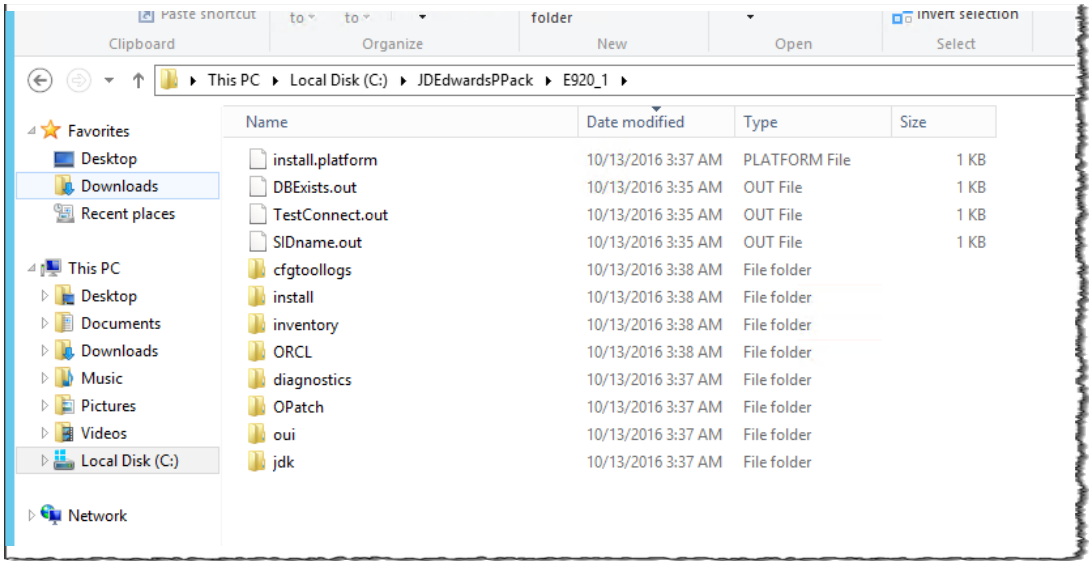


Figure 36: Folder structure for the installation directory

We can summarize the modifications required to achieve a seamless installation as follows.

Create the JDE Installers standard data pump directories in Amazon RDS instance. Amazon RDS for Oracle provides functionality allowing the creation of datapump directories using rdsadmin.rdsadmin_util.create_directory. This functionality will be used to create the three database directories that the installation process needs.

Change the syntax of the CREATE TABLESPACE statements. Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files, and control files. When creating data files and log files, you cannot specify physical file names.

Rename the pristine data dump. Change the name of the pristine data dump file and also the import data script for the TEST environment and pristine environment (the standard scripts change the import DIR; we are going to change the filename).

Change the database grants. Change the database grants to remove "create any directory", as this is not a grant that works on RDS.

- **Change the "dpump_dir1" entry in all scripts to DATA_PUMP_DIR.** The Data Pump files need to be moved from the various directories on the deployment server install media to the DATA_PUMP_DIR directory on the RDS DB instance using DBMS_FILE_TRANSFER.PUT_FILE. See Changing dpump_dir1 for additional details.
- **Change the syntax of the CREATE TABLESPACE statements.** Amazon RDS supports Oracle Managed Files (OMF) only for data files, log files, and control files. When creating data files and log files you cannot specify physical file names. See Changing the Syntax of the CREATE TABLESPACE Statements for additional details.
- **Rename the pristine data dump file and the import data script.** Change the name of the pristine data dump file and also the import data script for the TEST environment and pristine environment (the standard scripts change the import DIR, and we are going to change the filename). See Renaming the "pristine data dump" file and the Import Data Script for additional details.
- **Change the database grants.** Change the database grants to remove "create any directory", as this is not a grant that works on Amazon RDS. See Changing the Database Grants for additional details.

To make the modification of the default installation scripts easier, you can download a script that contains all of the previous modifications: http://jde92poc.s3-website-ap-southeast-2.amazonaws.com/ORCL_set.zip.[18] Using this script will reduce the amount of manual work you need to do in the installation process.

Throughout this process, the updated scripts are located in the **ORCL** directory. You can run these scripts at any time by executing the following command. However, this is the master script for the database installation and you should **NOT** run it at this stage.

```
cmd> InstallOracleDatabase.BAT
```

If, throughout this process, you make any mistakes or encounter failures, execute the following command. This command completely unloads and drops any database components that were created by the installation script.

```
cmd> drop_db.bat
```

We recommend that you back up **ALL** the scripts in the **ORCL** directory. If required, you can run the installer again to generate a set of new, pristine scripts.

Create the JDE Installers Standard Data Pump Directories

From SQL Developer connected to the Amazon RDS for Oracle database instance, perform the following steps. The Windows global search and replace commands were completed using notepad++.[19] However, you can use any text editor.

Changing dpump_dir1

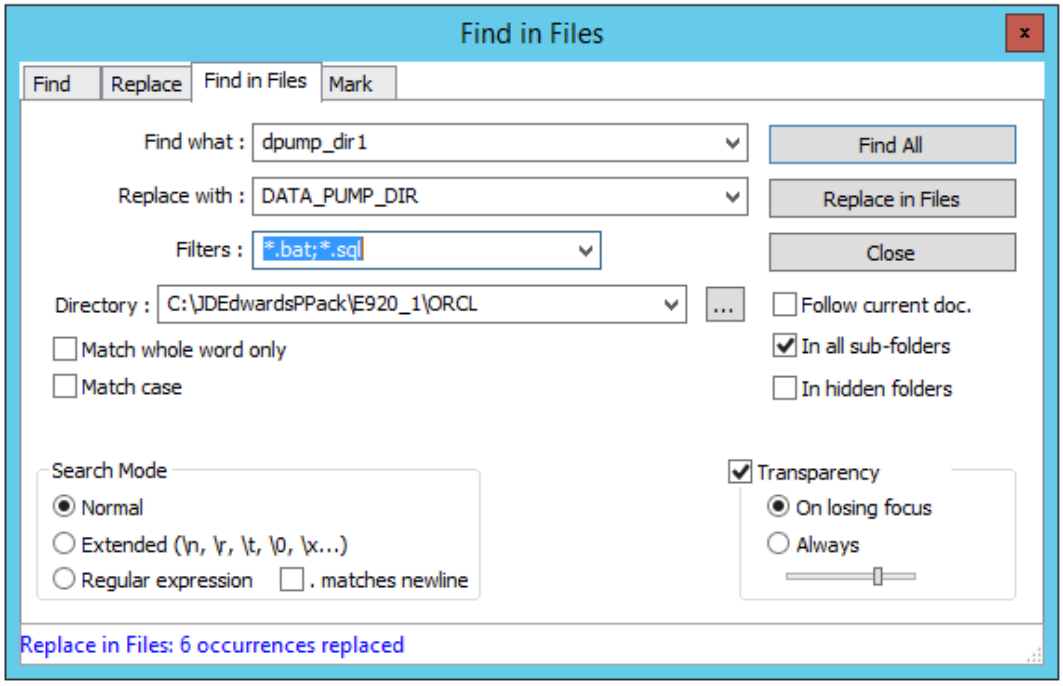


Figure 37: Find and replace the DATA_PUMP_DIR

```
Sqldeveloper> exec
rdsadmin.rdsadmin_util.create_directory('log_dir1');
```

```
Sqldeveloper> exec
rdsadmin.rdsadmin_util.create_directory('dpump_dir1');
```

Confirmation messages like the following will be displayed; you can safely ignore them.

```
anonymous block completed

anonymous block completed
```

You can confirm that the directory was created by running the following SQL statement.

```
SELECT directory_name, directory_path FROM dba_directories;
```

At the bottom of the list in the figure, in the lower right of the screen, you can see the two new database directories that were created.

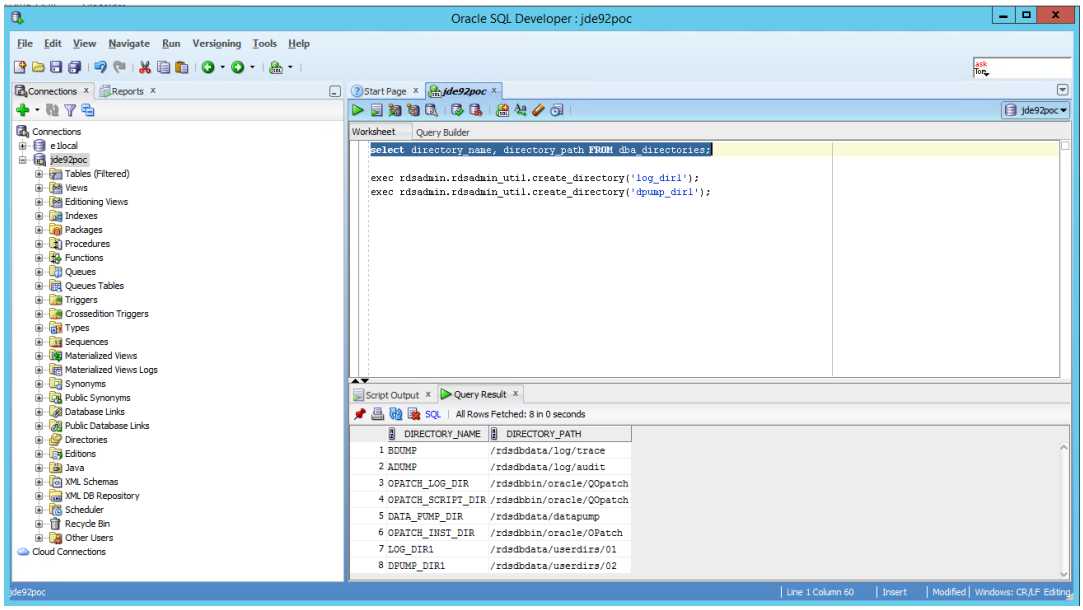


Figure 38: Viewing the selected directories

Above is the global search and replace for *.sql and *.bat files in the **c:\JDEdwardsPPack\E920_1\ORCL** directory. Replace the following.

From Value	To Value
dpump_dir_1	DATA_PUMP_DIR
log_dir1	DATA_PUMP_DIR

For example, the following…

```
impdp %SYSADMIN_USER%/%SYSADMIN_PSSWD%@%CONNECT_STRING%
DIRECTORY=dpump_dir1
DUMPFILE=RDBSPEC01.DMP,RDBSPEC02.DMP,RDBSPEC03.DMP,RDBSPEC04.DMP
LOGFILE=log_dir1:Import_%USER%.log TABLE_EXISTS_ACTION=TRUNCATE
EXCLUDE=USER
```

…becomes this.

```
impdp %SYSADMIN_USER%/%SYSADMIN_PSSWD%@%CONNECT_STRING%
DIRECTORY=DATA_PUMP_DIR
DUMPFILE=RDBSPEC01.DMP,RDBSPEC02.DMP,RDBSPEC03.DMP,RDBSPEC04.DMP
LOGFILE=DATA_PUMP_DIR:Import_%USER%.log TABLE_EXISTS_ACTION=TRUNCATE
EXCLUDE=USER
```

Changing the Syntax of the CREATE TABLESPACE Statements

By default, pristine **CREATE TABLESPACE** statements found in the files, such as **crtabsp_co.nt, crtabsp_sh.nt** and **crtabsp_env.nt**, look like the following example.

```
CREATE TABLESPACE &&PATH.&&RELEASE.t
logging
datafile '&&TABLE_PATH\&&PATH.&&RELEASE.t01.dbf' size 1500M,
         '&&TABLE_PATH\&&PATH.&&RELEASE.t02.dbf' size 1500M
autoextend on next 60M maxsize 5000M
extent management local autoallocate
segment space management auto
online;
```

These statements need to be modified to reflect the example below.

```
CREATE bigfile TABLESPACE &&PATH.&&RELEASE.t
logging
datafile SIZE 1500M AUTOEXTEND ON MAXSIZE 5G;
```

> **Note:** Generally, this next step of applying updates is either a manual or a scripted task due to differences in many of the tablespaces.

The following updates need to be applied.

crtabsp_co.nt

```
create bigfile tablespace &&PATH.&&RELEASE.t
loggingdatafile size 1500M AUTOEXTEND ON MAXSIZE 5G ;

create bigfile tablespace &&PATH.&&RELEASE.i
logging
datafile size 1500M AUTOEXTEND ON MAXSIZE 5G ;
```

crtabsp_sh.nt

```
create bigfile tablespace sy&&RELEASE.t
logging
datafile  size 250M  AUTOEXTEND ON MAXSIZE 750M;

create bigfile tablespace sy&&RELEASE.i
logging
datafile  size 100M AUTOEXTEND ON MAXSIZE 750M;

create bigfile tablespace svm&&RELEASE.t
logging
datafile  size 10M AUTOEXTEND ON MAXSIZE 150M;

create bigfile tablespace svm&&RELEASE.i
logging
datafile size 10M AUTOEXTEND ON MAXSIZE 150M;

create bigfile tablespace ol&&RELEASE.t
logging
datafile size 250M AUTOEXTEND ON MAXSIZE 350M;

create bigfile tablespace ol&&RELEASE.i
logging
datafile  size 100M AUTOEXTEND ON MAXSIZE 150M;

create bigfile tablespace dd&&RELEASE.t
logging
datafile  size 350M AUTOEXTEND ON MAXSIZE 450M;

create bigfile tablespace dd&&RELEASE.i
logging
datafile  size 125M AUTOEXTEND ON MAXSIZE 750M;
```

crtabsp_env.nt

```
create bigfile tablespace &&ENV_OWNER.ctli
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 1500M ;

create bigfile tablespace &&ENV_OWNER.ctlt
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 1500M ;

create bigfile tablespace &&ENV_OWNER.dtai
logging
datafile  size 1000M AUTOEXTEND ON MAXSIZE 4500M ;

create bigfile tablespace &&ENV_OWNER.dtat
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 4500M ;
```

Renaming the Pristine Data Dump File and the Import Data Script

These changes are made to **ORCL\InstallOracleDatabase.BAT**.

As explained earlier, you are changing DTA to DDTA to load the DEMO data as opposed to the empty tables.

```
--approx line 363 - PRISTINE
@rem ------------------------------------------------------------
@set USER=%PS_DTA_USER%
@set PSSWD=%PS_DTA_PSWD%
@set FROMUSER=%PS_DTA_FROMUSER%
@set LOAD_TYPE=DDTA
@set JDE_DTA=%DATABASE_INSTALL_PATH%\demodta
@echo ************************************************************
@echo      Create and load %USER% Business Data tables
@echo
@echo "Calling Load for %PS_DTA_USER% load type DTA">>logs\OracleStatus.txt
@echo "InstallOracleDatabase: #6 call load %PS_DTA_USER% DTA TESTDTA
@call Load.bat
@if ERRORLEVEL 4 (
@goto abend
```

```
--approx line 554 - TESTDTA
@rem ------------------------------------------------------------
@if "%RUN_MODE%"=="INSTALL" (
@set USER=%DV_DTA_USER%
@set PSSWD=%DV_DTA_PSWD%
@set FROMUSER=%PS_DTA_FROMUSER%
@set LOAD_TYPE=DDTA
@set JDE_DTA=%DATABASE_INSTALL_PATH%\demodta
@echo ************************************************************
@echo      Create and load %DV_DTA_USER% Business Data tables
@echo
@echo "Calling Load for %DV_DTA_USER% load type DTA">>logs\OracleStatus.txt
@call Load.bat
@if ERRORLEVEL 4 (
@goto abend
```

Changing the Database Grants

Create_dir.sql has the following statement that you need to change. Amazon RDS for Oracle does not support creating directories on the RDS instance, so you must remove this statement.

Before

```
grant create session, create table, create view, create any directory,
select any dictionary to jde_role;
```

After

```
grant create session, create table, create view, select any dictionary to
jde_role;
```

Advanced Configuration

Start an SQL Developer session to the RDS DB instance and log in as the administrative user (**jde92pocmaster**).

Running the following SQL command…

```
SELECT directory_name, directory_path FROM dba_directories;
```

…results in this.

```
DIRECTORY_NAME          DIRECTORY_PATH
------------------      --------------------------
BDUMP                   /rdsdbdata/log/trace
ADUMP                   /rdsdbdata/log/audit
OPATCH_LOG_DIR          /rdsdbbin/oracle/QOpatch
OPATCH_SCRIPT_DIR       /rdsdbbin/oracle/QOpatch
DATA_PUMP_DIR           /rdsdbdata/datapump
OPATCH_INST_DIR         /rdsdbbin/oracle/OPatch
LOG_DIR1                /rdsdbdata/userdirs/01
DPUMP_DIR1              /rdsdbdata/userdirs/02
```

To see files in DATA_PUMP_DIR1 directory, run the following.

```
SELECT * FROM TABLE
(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR1')) ORDER BY mtime;
```

```
SELECT * FROM TABLE (RDSADMIN.RDS_FILE_UTIL.LISTDIR('LOG_DIR1'))
ORDER BY mtime;
```

The following command deletes a single file named **Import_TESTCTL_CTL.log** from the LOG_DIR1 directory stored on the Oracle DB instance.

```
exec utl_file.remove('LOG_DIR1','Import_TESTCTL_CTL.log');
```

```
exec utl_file.fremove('DATA_PUMP_DIR','Import_TESTCTL_CTL.log');
```

The DATA_PUMP_DIR is used in the following SQL command to generate deletes for all log files in LOG_DIR1**DATA_PUMP_DIR**.

```
SELECT 'exec utl_file.fremove(''DATA_PUMP_DIR'',''' || filename ||
''');' FROM TABLE (RDSMDMIN.RDS_FILE_UTIL.LISTDIR('LOG_DIR1')) WHERE
filename LIKE '%log' ORDER BY mtime;
```

Moving DMP Files

When connected to **e1local** on the deployment server using SQL Developer, run the following commands.

```
DROP DATABASE LINK jde92poc;

CREATE DATABASE LINK jde92poc CONNECT TO jde92pocmaster IDENTIFIED
BY "aws_Poc_Password"
USING'(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=jde92poc.jde92.local
)(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=jde92poc)
))';

SELECT directory_name, directory_path FROM dba_directories;

'C:\Oracle64db\admin\e1local\dpdump';
```

These commands create the following:

- A new database directory to read the dump files from the deployment server
- A database link to the Amazon RDS for Oracle DB instance to be a conduit to move the dump files from the deployment server to the Oracle DB instance

Copying DMP Files from an ORCL Directory to a Specified DATA_PUMP Directory

Locate *.dmp files in the **ORCL** directory and copy them to **C:\Oracle64db\admin\e1local\dpdump**, as defined in the previous **e1local** database directory (**DATA_PUMP_SRM**).

You'll see that there are two **DUMP_DTA.DMP** files in the find results. The one in **demodta** must be renamed **DUMP_DDTA.DMP**. It's important to name it exactly as specified, because there are associated changes in the import scripts. **DUMP_DTA.DMP** comes from **ORCL\proddta**.

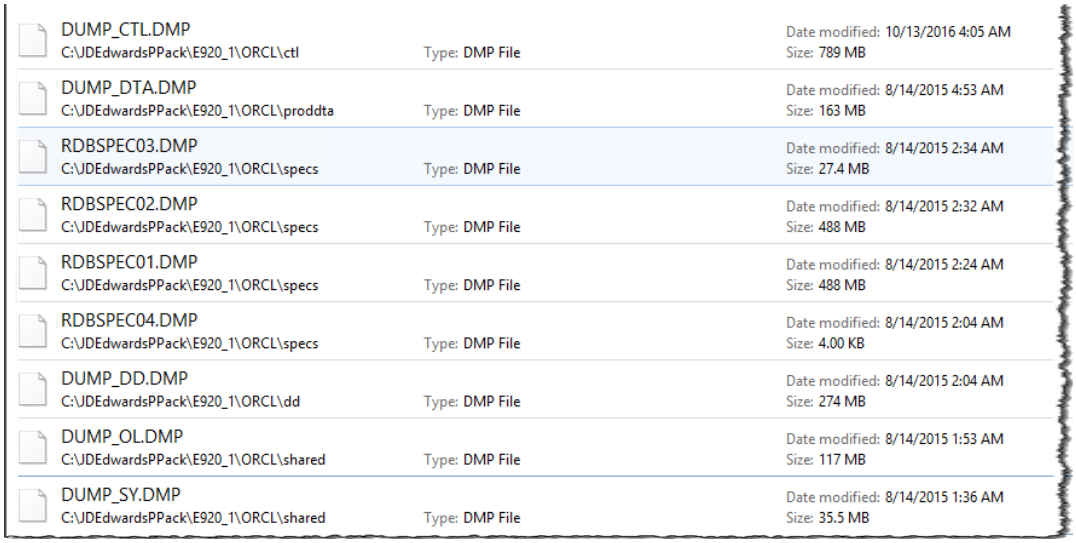


Figure 39: List of DATA_PUMP_SRM directory

The reason for this renaming is that one of the dump files (the larger one) is for **DEMO** data, which is imported into **TESTDTA** and **PRISTINE**, while the smaller file (**DUMP_DTA.DMP**) does not contain any data – just table and index structures.

Now, all of the *.dmp files that require copying into the Oracle DB instance are in an **e1local** directory named **DATA_PUMP_SRM**. It's time to move these files to the RDS DB instance directory named **DPUMP_DIR1** that we recently created.

The following is how this directory looks on the deployment server.

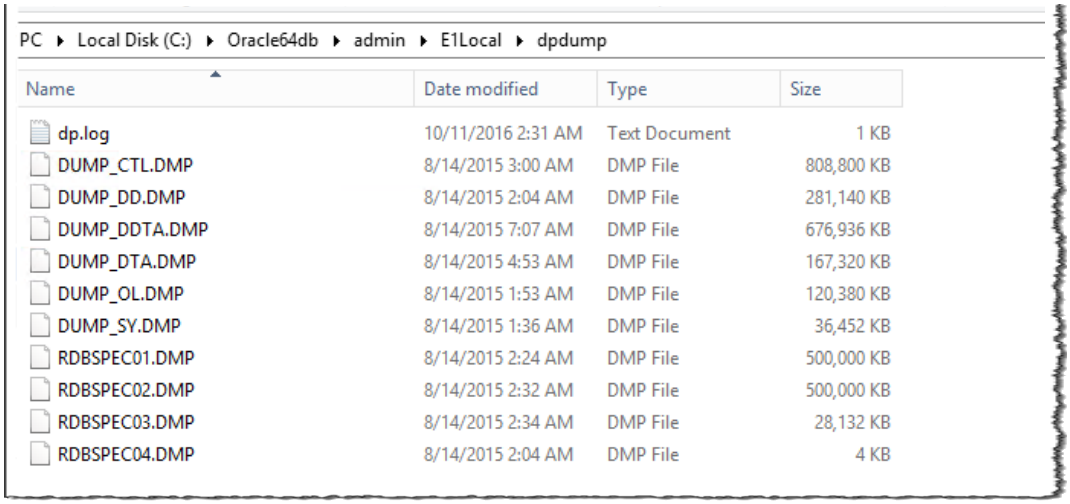


Figure 40: DPUMP_DIR1 directory on deployment server

In the Appendix we provide a script you can use to copy the dump files from the deployment server to the RDS DB instance via a database link. Execute this script from SQL Developer connected to the **e1local** database.

When these commands finish successfully, you can execute the following command against the Oracle DB instance (**jde92poc**) to ensure that the files have arrived.

```
SELECT substr(filename,1,30),type, filesize, MTIME
FROM TABLE(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DPUMP_DIR1')) ORDER BY
mtime;
```

The following output indicates that the files were transferred correctly.

```
SUBSTR(FILENAME,1,30)          TYPE          FILESIZE  MTIME
------------------------------ ---------- ---------- ---------
DUMP_CTL.DMP                   file         828211200  14-OCT-16
RDBSPEC01.DMP                  file         512000000  14-OCT-16
RDBSPEC02.DMP                  file         512000000  14-OCT-16
RDBSPEC03.DMP                  file          28807168  14-OCT-16
RDBSPEC04.DMP                  file              4096  14-OCT-16
DUMP_DTA.DMP                   file         171335680  14-OCT-16
DUMP_DD.DMP                    file         287887360  14-OCT-16
DUMP_OL.DMP                    file         123269120  14-OCT-16
DUMP_SY.DMP                    file          37326848  14-OCT-16
DUMP_DDTA.DMP                  file         693182464  14-OCT-16
```

```
create bigfile tablespace &&ENV_OWNER.ctli
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 1500M ;

create bigfile tablespace &&ENV_OWNER.ctlt
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 1500M ;

create bigfile tablespace &&ENV_OWNER.dtai
logging
datafile  size 1000M AUTOEXTEND ON MAXSIZE 4500M ;

create bigfile tablespace &&ENV_OWNER.dtat
logging
datafile size 1000M AUTOEXTEND ON MAXSIZE 4500M ;
```

Figure 41: Confirming files are transferred

Change the Database Grants to Not Include Create Any Directory

Because Amazon RDS Oracle does not support creating directories on the RDS instance, the creation of directories in the installation scripts must be done manually. You do this by using the AWS custom function **rdsadmin.rdsadmin_util.create_directory**.

Grants before

```
grant create session, create table, create view, create any directory,
select any dictionary to jde_role;
```

Grants after

```
grant create session, create table, create view, select any dictionary to
jde_role;
```

Running the Installer

At this point, you have made all the modifications that are required to facilitate the smooth installation of JD Edwards EnterpriseOne.

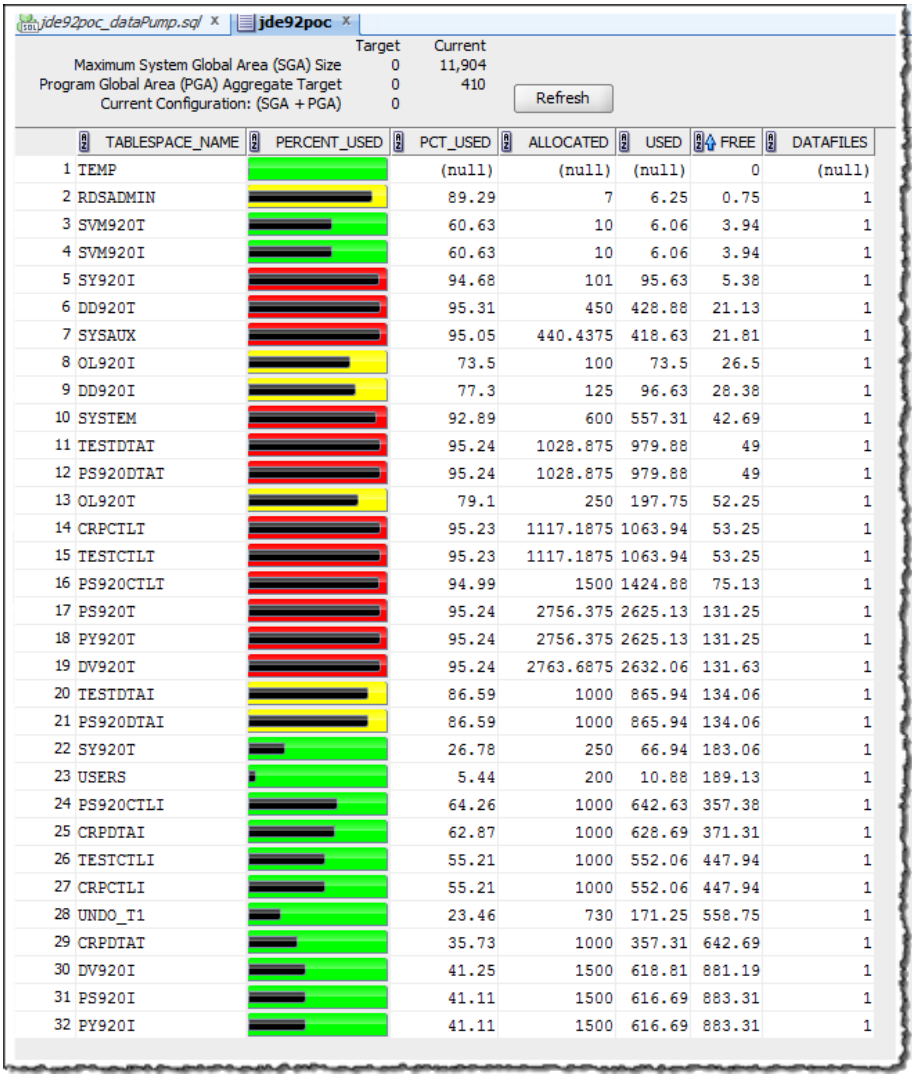
If you encounter any issues, be sure that anything you defined in the installation wizard is also defined in **ORCL\ORCL_SET.BAT**. If you forget items such as passwords or settings, you can retrieve them from this file. However, be sure to delete this file when the installation is complete.

Open a command window on the deployment server and run **InstallOracleDatabase.bat** from the **C:\JDEdwardsPPack\E920_1\ORCL** directory.

You can use **C:\JDEdwardsPPack\E920_1\ORCL\logs** to track progress and view the script output.

You cannot view the output of the data pump operations, because they are not multiples of the block size of the database.

When the installation is complete, you should see that the database is populated. The following screenshot is from Oracle SQL Developer and shows you the properties of the target database. All JD Edwards EnterpriseOne tablespaces now have space allocated and tables created.



jde92poc_dataPump.sql | jde92poc

	Target	Current
Maximum System Global Area (SGA) Size	0	11,904
Program Global Area (PGA) Aggregate Target	0	410
Current Configuration: (SGA + PGA)	0	

Refresh

	TABLESPACE_NAME	PERCENT_USED	PCT_USED	ALLOCATED	USED	FREE	DATAFILES
1	TEMP		(null)	(null)	(null)	0	(null)
2	RDSADMIN		89.29	7	6.25	0.75	1
3	SVM920T		60.63	10	6.06	3.94	1
4	SVM920I		60.63	10	6.06	3.94	1
5	SY920I		94.68	101	95.63	5.38	1
6	DD920T		95.31	450	428.88	21.13	1
7	SYSAUX		95.05	440.4375	418.63	21.81	1
8	OL920I		73.5	100	73.5	26.5	1
9	DD920I		77.3	125	96.63	28.38	1
10	SYSTEM		92.89	600	557.31	42.69	1
11	TESTDTAT		95.24	1028.875	979.88	49	1
12	PS920DTAT		95.24	1028.875	979.88	49	1
13	OL920T		79.1	250	197.75	52.25	1
14	CRPCTLT		95.23	1117.1875	1063.94	53.25	1
15	TESTCTLT		95.23	1117.1875	1063.94	53.25	1
16	PS920CTLT		94.99	1500	1424.88	75.13	1
17	PS920T		95.24	2756.375	2625.13	131.25	1
18	PY920T		95.24	2756.375	2625.13	131.25	1
19	DV920T		95.24	2763.6875	2632.06	131.63	1
20	TESTDTAI		86.59	1000	865.94	134.06	1
21	PS920DTAI		86.59	1000	865.94	134.06	1
22	SY920T		26.78	250	66.94	183.06	1
23	USERS		5.44	200	10.88	189.13	1
24	PS920CTLI		64.26	1000	642.63	357.38	1
25	CRPDTAI		62.87	1000	628.69	371.31	1
26	TESTCTLI		55.21	1000	552.06	447.94	1
27	CRPCTLI		55.21	1000	552.06	447.94	1
28	UNDO_T1		23.46	730	171.25	558.75	1
29	CRPDTAT		35.73	1000	357.31	642.69	1
30	DV920I		41.25	1500	618.81	881.19	1
31	PS920I		41.11	1500	616.69	883.31	1
32	PY920I		41.11	1500	616.69	883.31	1

Figure 42: Properties of the target database

You've now completed all the tasks for installing JD Edwards EnterpriseOne on the Amazon RDS Oracle DB instance. The following steps enable you to verify that you can connect to the populated instance.

Logging into JD Edwards EnterpriseOne on the Deployment Server

Click the application launch icon, shown below, to start JD Edwards EnterpriseOne.

Figure 43: Launch JD Edwards EnterpriseOne

The JD Edwards EnterpriseOne login screen is displayed.

Figure 44: JD Edwards EnterpriseOne login

Logging in to the **JDEPLAN** environment, as shown, does not connect to **jde92poc** at all—it only connects to **e1local**.

To test the database installation, you must log in to **DV920**.

Figure 45: Logging in to DV920 for testing

If your login succeeds, the next step is to log out and then log back in to the **jdeplan** environment and continue with the standard installation. Because there are no further deviations from a standard installation beyond this point, you can proceed to create an installation plan and run the installation workbench. Follow the instructions in section 5 of the JD Edwards EnterpriseOne installation process, "Working with Installation Planner for an Install".[20]

Validation and Testing

The successful completion of the installation workbench will give you confidence that the Amazon RDS Oracle database installation is working. Proceeding to install web servers and enterprise servers and connecting them to the Amazon RDS for Oracle DB instance are some of the remaining installation steps.

Please remember to delete the dump files on the Amazon RDS instance to ensure that they do not contribute to the amount of storage you are using on the Amazon RDS instance. Any files stored in database directories contribute to the space you are using in the Amazon RDS instance.

Use the following statement to build the commands you need to run to delete the DMP files. Run this statement only when you know that your installation succeeded.

```
SELECT 'exec utl_file.fremove(''DPUMP_DIR1'',''' || filename || ''');'
FROM table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DPUMP_DIR1'))
WHERE filename LIKE '%DMP' ORDER BY mtime;
```

Conclusion

This whitepaper described many of the capabilities and advantages of using AWS and Amazon RDS as the foundation for installing the JD Edwards EnterpriseOne application. Specifically, this whitepaper focused on a way of configuring Amazon RDS for Oracle as the underlying database for the JD Edwards EnterpriseOne application.

The whitepaper articulated all the steps for installing the JD Edwards EnterpriseOne application, and the steps required to set up an Amazon RDS Oracle DB instance.

Having JD Edwards EnterpriseOne and Amazon RDS for Oracle running in the AWS Cloud enables you to enjoy the advantages of simple deployment, high availability, security, scalability, and many additional services supported by Amazon RDS and AWS.

Appendix A: Dumping Deployment Service to RDS

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_CTL.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_CTL.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'RDBSPEC01.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'RDBSPEC01.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'RDBSPEC02.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'RDBSPEC02.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'RDBSPEC03.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'RDBSPEC03.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'RDBSPEC04.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'RDBSPEC04.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_DTA.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_DTA.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_DD.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_DD.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_OL.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_OL.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_SY.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_SY.DMP',
    destination_database          => 'jde92poc'
  );
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_SRM',
    source_file_name              => 'DUMP_DDTA.DMP',
    destination_directory_object  => 'DPUMP_DIR1',
    destination_file_name         => 'DUMP_DDTA.DMP',
    destination_database          => 'jde92poc'
  );

END;
```

Notes

[1] http://aws.amazon.com/rds

[2] http://aws.amazon.com/ec2

[3] http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf

[4] http://docs.oracle.com/cd/E53430_01/EOTLI/toc.htm

[5] http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf

[6] http://aws.amazon.com/rds/oracle/

[7] http://aws.amazon.com/elasticloadbalancing

[8] http://aws.amazon.com/vpc

[9] http://aws.amazon.com/iam

[10] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAM.html

[11] http://aws.amazon.com/kms

[12] https://docs.oracle.com/cd/E61420_01/EOIUO/toc.htm

[13] http://jde92poc.s3-website-ap-southeast-2.amazonaws.com/jde_cfn.zip

[14] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html

[15] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[16] http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-creating.html

[17] http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index-098778.html

[18] http://jde92poc.s3-website-ap-southeast-2.amazonaws.com/ORCL_set.zip

[19] https://notepad-plus-plus.org/

[20] https://docs.oracle.com/cd/E24902_01/doc.91/e23311/install_planner_install.htm#EOIWO00005