



# Oracle Autonomous Transaction Processing Hands-on Lab Guide



# Oracle Autonomous Transaction Processing Hands on Lab Guide

---

Lab Introduction .....	4
<i>Lab software prerequisites</i> .....	4
<b>Section 1 - Introduction to Autonomous Transaction Processing .....</b>	<b>9</b>
Lab 1. Connecting to the Oracle Cloud .....	10
<i>Connecting to your Lab Virtual Machine</i> .....	11
<i>Signing in to the OCI Console</i> .....	12
<i>Locating your ATP Instance</i> .....	13
Lab 2. Securely Connecting to Autonomous Transaction Processing .....	18
<i>Managing Priorities on Autonomous Transaction Processing</i> .....	19
<i>Downloading the credentials wallet</i> .....	20
<i>Connecting to the database using SQL Developer</i> .....	23
<b>Section 2 - Focus on DBAs .....</b>	<b>26</b>
Lab 3. Scaling an Automatic Transaction Processing Instance .....	27
<i>Scaling your instance</i> .....	28
Lab 4. Managing and Monitoring Autonomous Transaction Processing .....	34
<i>About Backup and Recovery on Autonomous Transaction Processing</i> .....	34
<i>Exploring Service Console</i> .....	35
<i>Stopping your ATP instance</i> .....	40
<i>Starting your ATP instance</i> .....	41
Lab 5. Data Loading into Autonomous Transaction Processing .....	44
<i>Loading a file from local storage</i> .....	44
<i>Loading a file from Object Storage</i> .....	49
<i>Importing a Data Pump Export</i> .....	54
Lab 6. DBA Exploration of ATP with SQL Developer .....	60
Lab 7: Workload testing and analysis using Swingbench with Autonomous Transaction Processing (optional) .....	71
<i>Preparing to use Swingbench</i> .....	71
<i>Running Swingbench</i> .....	73
<i>Want to learn more about Swingbench?</i> .....	77
<b>Section 3- Focus on Developers .....</b>	<b>78</b>
Lab 8: Using Node.js with Autonomous Transaction Processing .....	80
<i>Creating your first Node.js application</i> .....	80
<i>Configuring Node.js for use with ATP</i> .....	81
<i>Connect to an ATP Database with Node.js</i> .....	83
<i>Using Node.js script to select from an ATP database</i> .....	85
<i>Using Node.js script to select from an ATP database with a bind variable</i> .....	86
<i>Using JSON data in an ATP database</i> .....	87
Lab 9: Using Python with Autonomous Transaction Processing .....	93
<i>Verify Your Lab Environment</i> .....	94
<i>Configuring Python to run with ATP and sample scripts</i> .....	95
Lab 10: Using Docker Containers with Autonomous Transaction Processing.....	110
<i>Populating the ATP Database for the Docker/Node.js application</i> .....	111
<i>Creating your Docker image</i> .....	114
<i>Running the Node.js application in your Docker image</i> .....	117

## **Oracle** Autonomous Transaction Processing Hands-on Lab Guide

<i>Creating and running your own customized Docker Container Image .....</i>	122
<i>Optional – Clean your Docker environment. ....</i>	124
Want to Learn More? .....	125
<i>Recommended Reading .....</i>	125
<i>Additional Resources.....</i>	125
Appendix A – Provisioning Process Walkthrough .....	128
<i>Creating your Autonomous Transaction Processing Database.....</i>	128
Appendix B – Creating and preparing a user to access Object Storage .....	136
Appendix C – Connecting to the database using Oracle Machine Learning Notebook .....	139
<i>Creating an OML user .....</i>	139

# Lab Introduction

In this hands-on lab you will get first-hand experience with Oracle's new Autonomous Transaction Processing (ATP) service. Oracle ATP delivers a self-driving, self-securing, self-repairing database service that can instantly scale to meet demands of mission critical transaction processing and mixed workload applications.

The lab is structured in three sections

- Section 1 – Introduction to ATP
- Section 2 – Focus on DBA activities
- Section 3 – Focus on Developer features

The audience for this hands-on-lab is expected to be database administrators and application developers. The lab expects you to understand the basic concepts of an Oracle database, the UNIX command line and be comfortable using the vi editor or the built in gedit editor.

All attendees must complete the 'Introduction to ATP' section, but then can select labs from the other sections.

## Lab software prerequisites

This lab requires you to install the following desktop application so that you can complete this hands-on lab:

- An PDF File reader.
- A VNC viewer. There are many VNC viewer packages both commercial and open source. Suggested packages are TigerVNC viewer or TightVNC Viewer but if you already have a preferred VNC viewer you can use this. TigerVNC viewer has a simpler install process, as it is a standalone executable, but has fewer features.

Some pdf viewers may have trouble correctly copying and pasting the format required for some of the scripts within this guide. All long SQL statements and scripts can be found in the **/home/oracle/labScripts** directory of the supplied VM. Where appropriate the script or SQL filename will be provided by the output of its listing within the guide.

### Installing TigerVNC Viewer

Follow these steps to if you want to download and install TigerVNC Viewer.

*macOS*

A better VNC viewer for macOS is realVNC which can be obtained from  
<https://www.realvnc.com/en/connect/download/viewer/macos/>

#### *Windows 64-bit*

Download '[vncviewer64-1.9.0.exe](#)' from  
<https://bintray.com/tigervnc/stable/tigervnc/1.9.0#files> and save it to your desktop.  
It is a self-contained executable file, which requires no further installation.

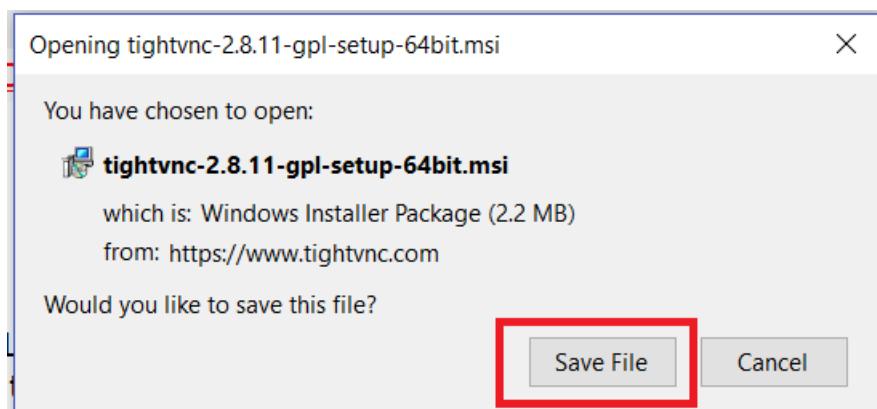
### **Installing TightVNC Viewer**

Follow these steps if you want to download and install TightVNC Viewer.

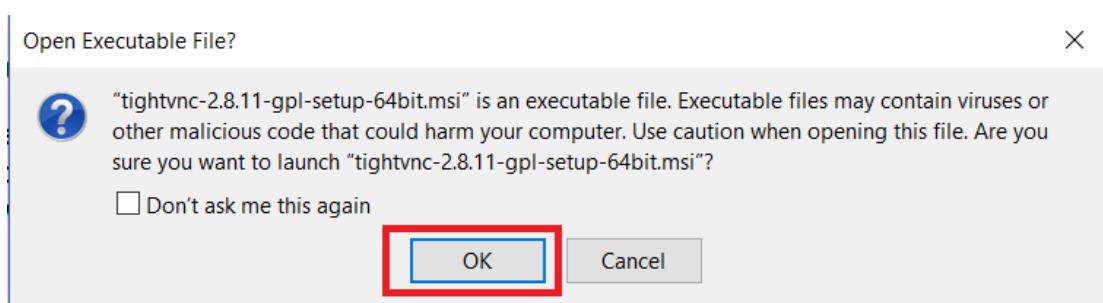
#### *Windows 64-bit*

Select the 'Installer for Windows (64-bit)' from  
<https://www.tightvnc.com/download.php>

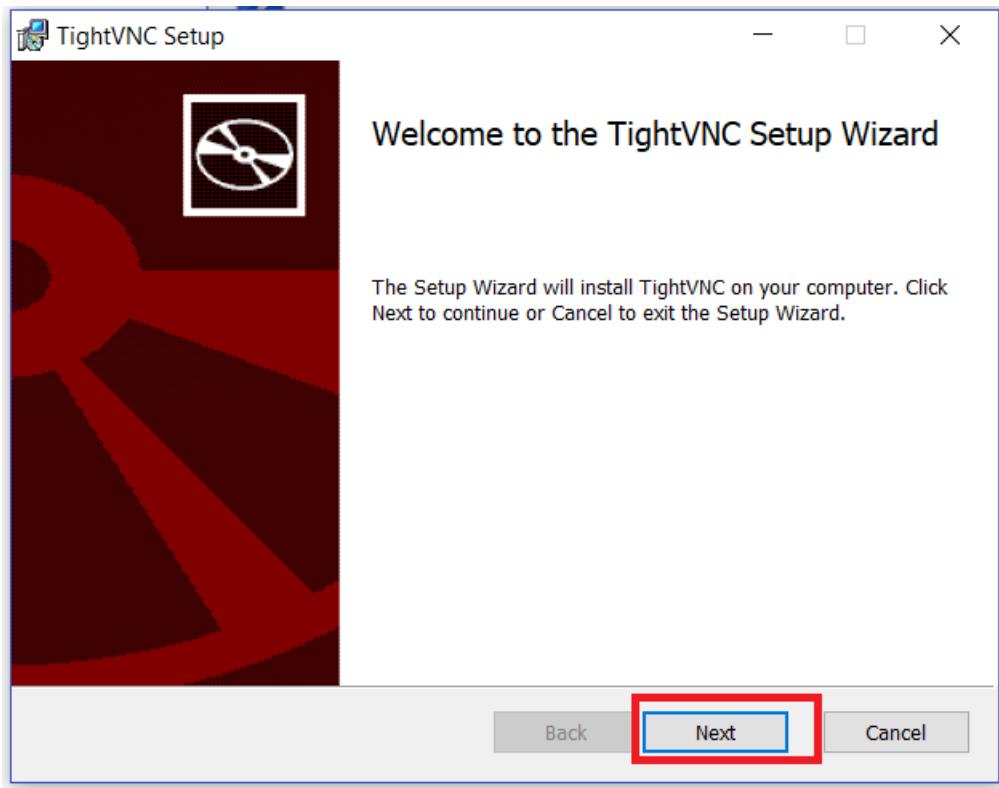
When prompted, select to save the file.



Locate the saved file on your machine and double click to execute. Windows will prompt you to check if you want to open the executable file. Select ok

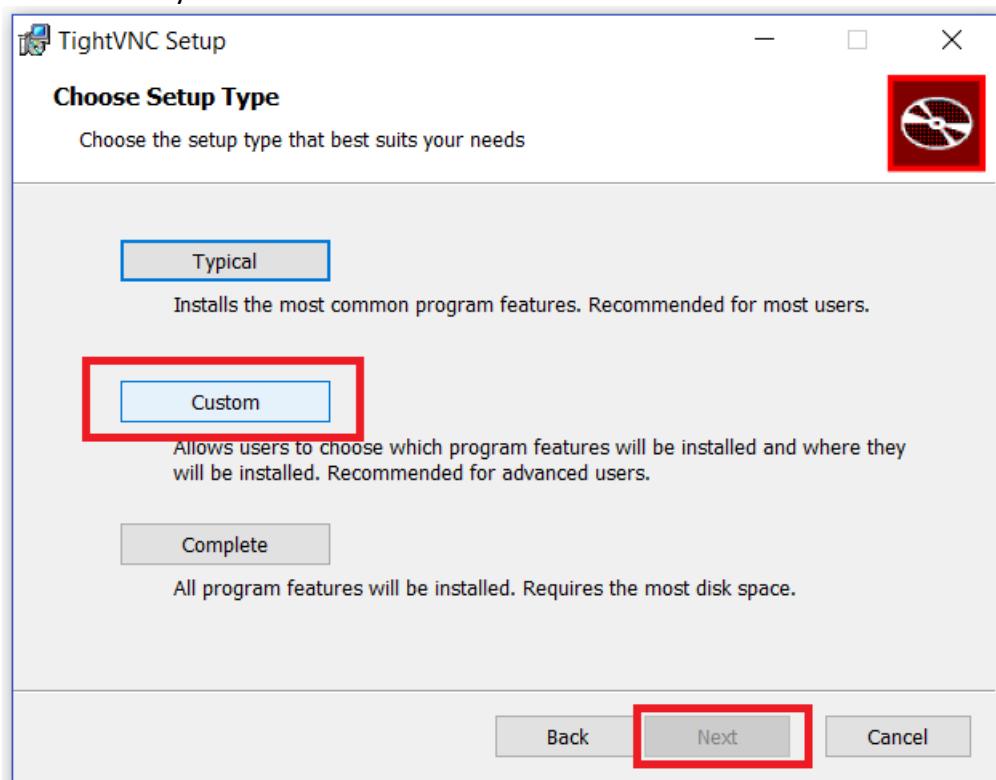


This will start the TightVNC install wizard. Select 'Next' to continue the install.

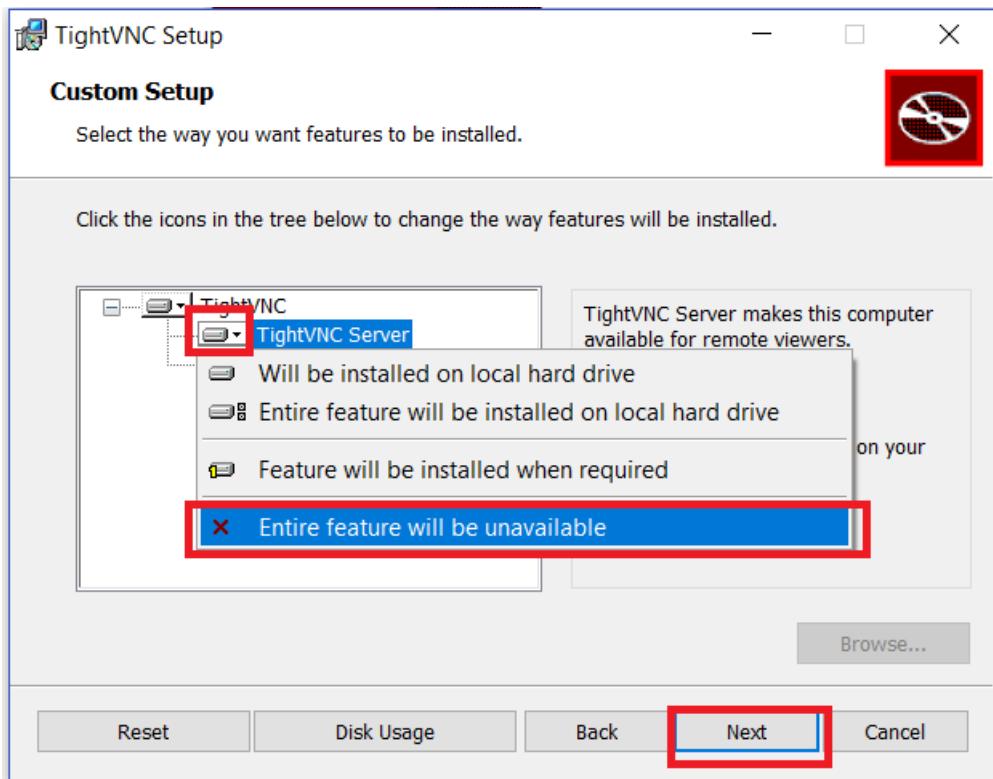


Review and accept the license terms if you agree. Select **Next** to continue.

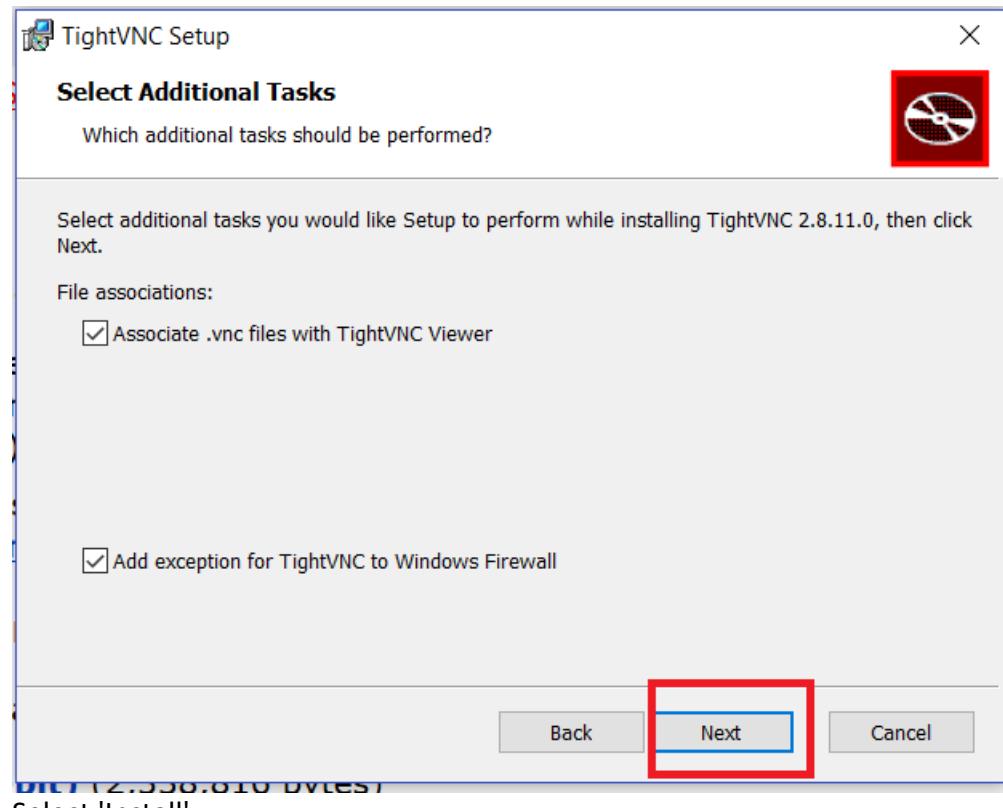
For this lab we only require the TightVNC Viewer, rather than the entire package. Select to carry out a **Custom** installation and select **Next**.



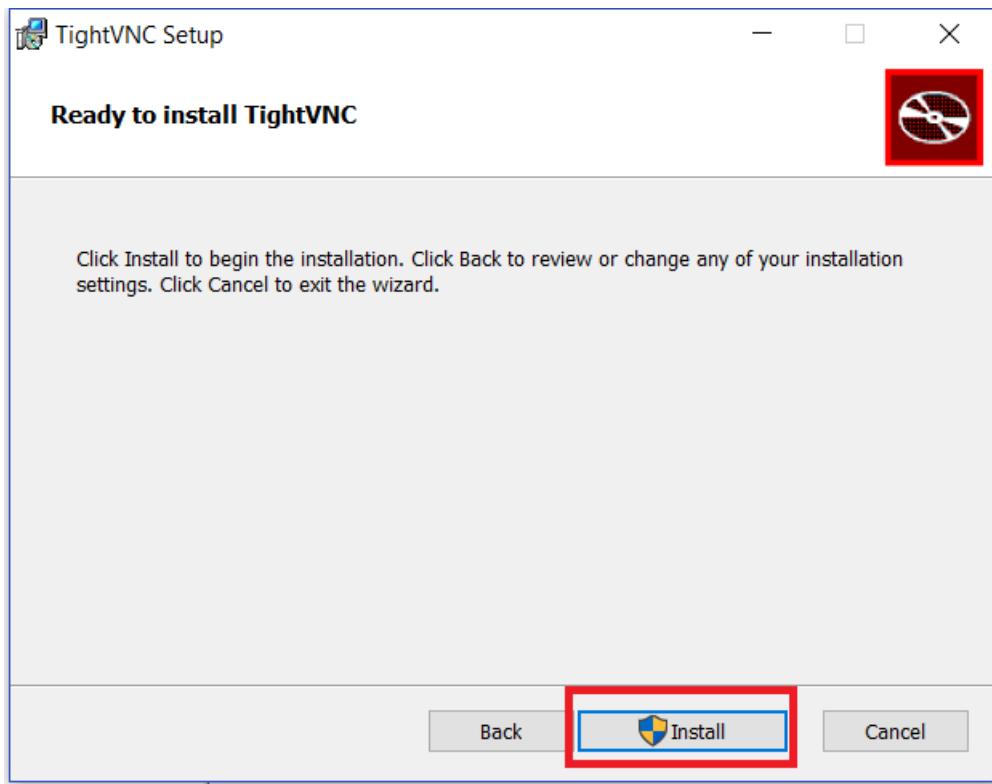
If you do not wish to install the server component, select the drop down next to **TightVNC Server** and select 'X Entire Feature will be unavailable'.



On the 'Select Additional Tasks' screen, choose if you want to change the file associations and the Firewall rules and select 'Next'



Select 'Install'



# Section 1 - Introduction to Autonomous Transaction Processing



# Lab 1 – Connecting to the Oracle Cloud

# Lab 1. Connecting to the Oracle Cloud

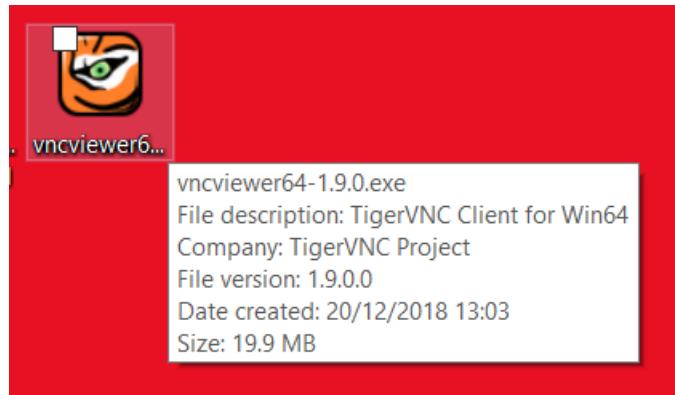
## Objectives:

- Learn how to login to the Oracle Cloud Console
- Learn how to provision a new Autonomous Transaction Processing (ATP) Instance

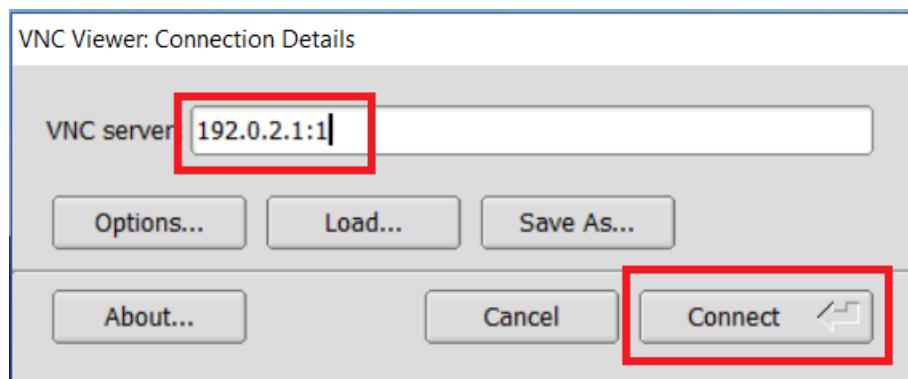
## Connecting to your Lab Virtual Machine

To reduce the amount of software that needs to be installed on your local machine we have prepared a Linux virtual machine to act as your desktop.

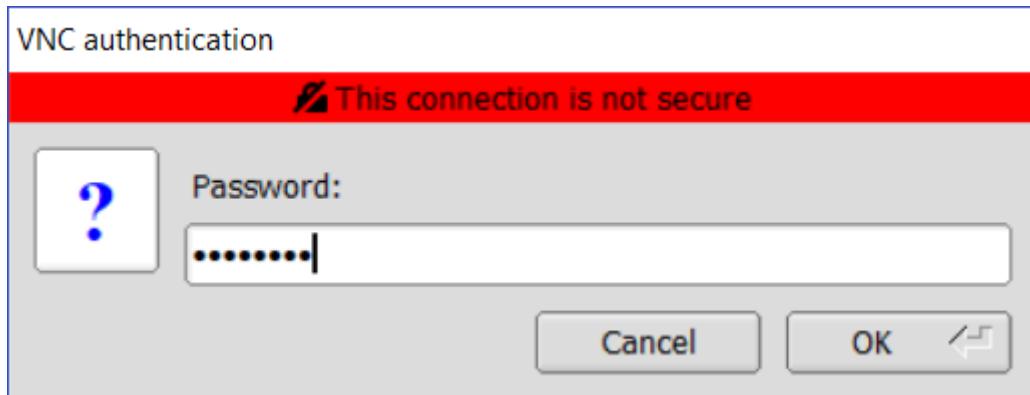
Start TigerVNC viewer (or your choice of VNC viewer) by double clicking on the icon on your local desktop.



Enter connection information provided by your lab leader and select **Connect**. You will use a different IP address than the example in the screenshot below.



Enter the supplied password and hit OK



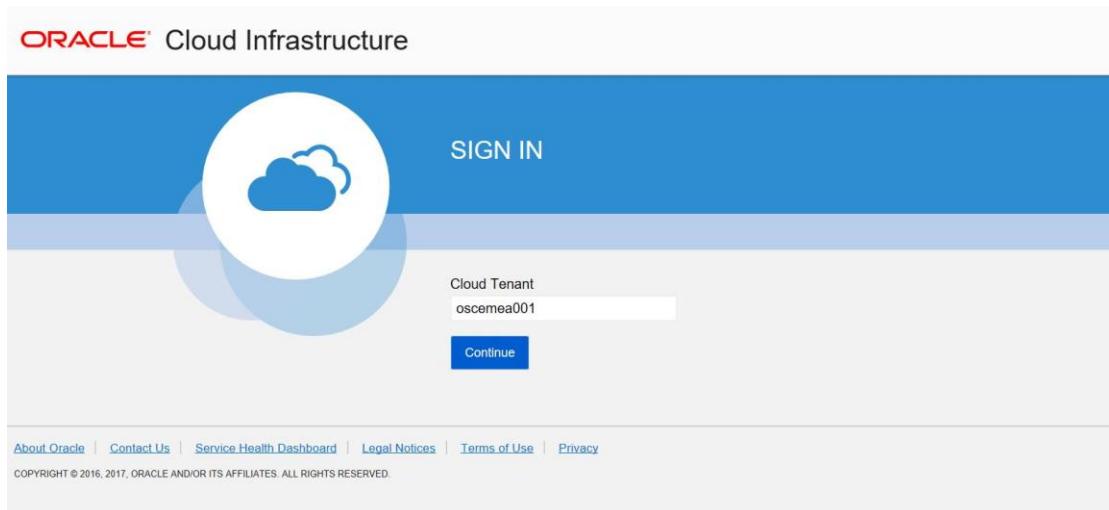
The VNC session will open, and an Oracle Linux desktop session will be displayed.



## Signing in to the OCI Console

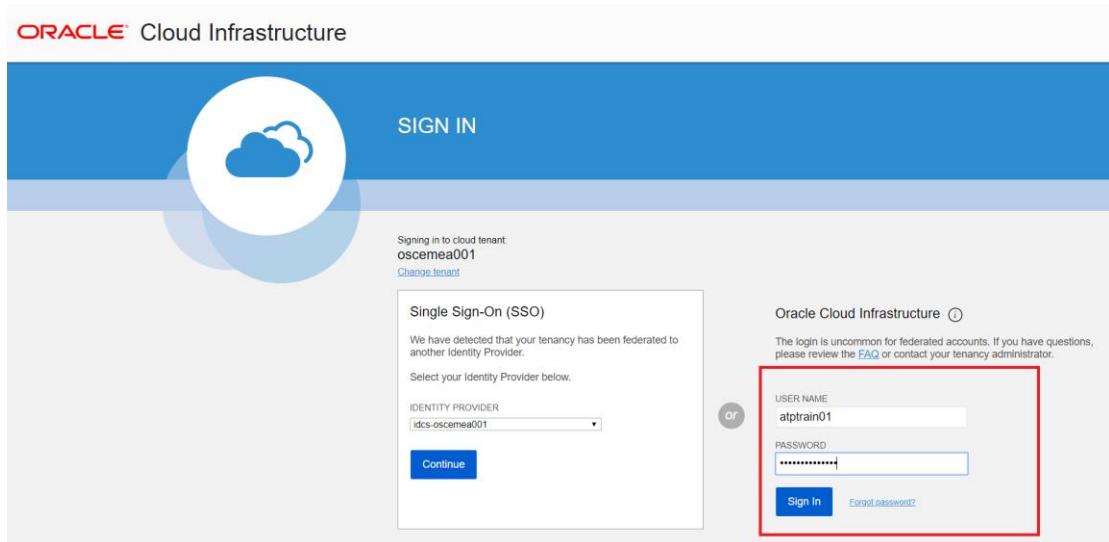
Start Firefox on your Lab VM by double clicking on the Firefox Logo.

Go to the [OCI console](https://console.eu-frankfurt-1.oraclecloud.com). <https://console.eu-frankfurt-1.oraclecloud.com> For this lab we are using the Frankfurt region for our ATP instances, even though other regions may be geographically closer as this is where our tenancy has capacity to support the workshop.



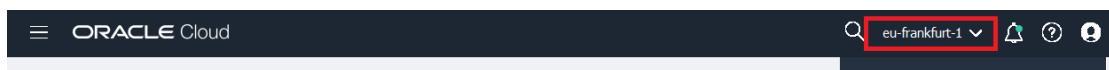
Enter a cloud tenancy of “**oscemea001**” and click continue.

On the right-hand section of the page under the title “Oracle Cloud Infrastructure” enter your username and password. Click “**Sign In**”.



This will bring you to the console home page.

Verify that your region is set to eu-frankfurt-1 on the top right-hand side of your screen, if not, use the dropdown list to select ‘eu-frankfurt-1’

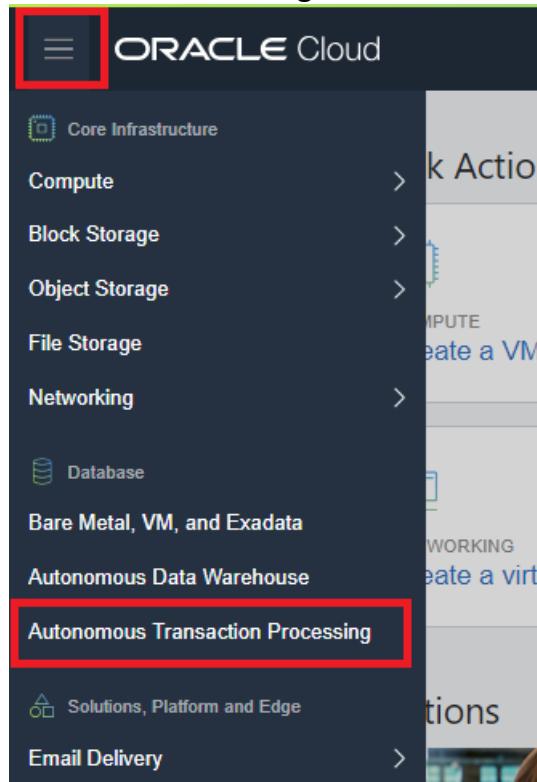


## Locating your ATP Instance

As part of the preparation for this lab, ATP instances have been pre-created. Your lab leader will have demonstrated a provisioning process live and the process is documented using screenshots in Appendix A.

Click on the **MENU** link at the top left of the page.

This will produce a drop-down menu, where you should select “**Autonomous Transaction Processing**”



This will take you to the management console page for ATP in the root compartment of the tenancy. The red warning icon "Forbidden" indicates that the Policies within the tenancy does not allow your user to create ATP Instances in the root compartment.

The screenshot shows the 'Autonomous Database' management console. On the left, there are filters for 'List Scope' (set to 'oscemea001 (root)'), 'State' (set to 'Any state'), and 'Workload Type' (set to 'ADW'). On the right, a table titled 'Autonomous Databases in oscemea001 (root) Compartment' is displayed. The table has columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. A single row is present, showing a status of 'Forbidden' with a red exclamation mark icon. At the bottom of the table, it says 'No Autonomous Databases < Page'.

To locate your ATP instance in this tenancy you need to select a compartment.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Click on the pulldown menu marked **Compartment**.

In your list of compartments expand **ATP\_Workshop** by clicking the ‘+’.

Your instructor will tell you which sub compartment of **ATP\_Workshop** to use.

### Autonomous Database

#### List Scope

##### COMPARTMENT

oscemea001 (root) 

Search compartments

-  oscemea001 (root)
  -  ATP\_Workshop
  - ATP\_Delegate
  - ManagedCompartmentsForPaaS

Any state 

*Note – Your list of compartments may be different to the one shown above. In these lab notes we will use the compartment called **ATP\_Delegate** in all the screenshots/examples. Your compartment may be different.*

The main page will now change to show the list of ATP instances within your compartment, as shown below:

*Note: this lab uses the same tenancy and compartment for all lab attendees. You will see ATP instances listed on this page which have already been created in preparation for your hands-on lab.*

Autonomous Database

Autonomous Databases in ATP\_Delegate Compartment

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
dd25ATP	Available	dd25ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:13:58 GMT
dd24ATP	Available	dd24ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:09:57 GMT
dd23ATP	Available	dd23ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:05:56 GMT
dd22ATP	Available	dd22ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:01:55 GMT
dd21ATP	Available	dd21ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:57:54 GMT
dd20ATP	Available	dd20ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:53:53 GMT
dd19ATP	Available	dd19ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:49:52 GMT
dd18ATP	Available	dd18ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:45:51 GMT
dd17ATP	Available	dd17ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:41:50 GMT
dd16ATP	Available	dd16ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:37:49 GMT
dd15ATP	Available	dd15ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:33:48 GMT

*Note: the list of instances is sorted by provisioning date, not by name. Refer to your lab information sheet to check which ATP instance has been assigned to you for this lab.*

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

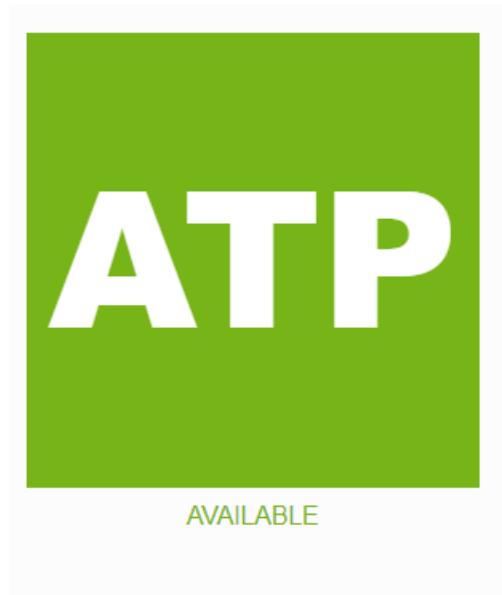
Click on the name of your ATP instance from the list of instances. You may have to scroll down to locate it in the list.

The screenshot shows the Oracle Autonomous Database interface. On the left, there's a sidebar with 'Autonomous Database' and 'List Scope' sections. Under 'List Scope', 'COMPARTMENT' is set to 'ATP\_Delegate'. Below that is a note: 'Don't see what you're looking for?'. Under 'Filters', 'STATE' is set to 'Any state' and 'WORKLOAD TYPE' is set to 'All'. In the center, there's a table titled 'Autonomous Databases in ATP\_Delegate Compartment'. The table has columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. The table lists several instances, with 'MELATPTRAIN01' highlighted by a red box. The data for 'MELATPTRAIN01' is as follows:

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
MELATPTRAIN01	Available	MELATPTRAIN01	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT
dd1ATP	Available	dd1ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:13:58 GMT
dd2ATP	Available	dd2ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:05:57 GMT
dd22ATP	Available	dd22ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:01:55 GMT
dd21ATP	Available	dd21ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:57:54 GMT
dd20ATP	Available	dd20ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:53:53 GMT
dd19ATP	Available	dd19ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:49:52 GMT
dd18ATP	Available	dd18ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:45:51 GMT

This will display more information about your instance and you should notice the various menu buttons that help you manage your new instance

A summary of your instance lifecycle status is shown in the large box on the left.



Additional summary information about your instance is displayed, including the Workload Type . You can also see the Lifecycle Status reported in this region.

The screenshot shows the 'Autonomous Database Information' page for the 'MELATPTRAIN01' instance. At the top, there are tabs for 'Autonomous Database Information' and 'Tags'. The main content area displays the following details:

Workload Type: Transaction Processing	Created: Fri, 08 Mar 2019 08:38:38 GMT
Display Name: MELATPTRAIN01	Compartment: oracledemabautodb (root)/ATP_Workshop/ATP_Delegate
Database Name: MELATPTRAIN01	OCID: l-au7frq Show Copy
CPU Core Count: 1	License Type: Bring Your Own License
Storage (TB): 1	Lifecycle State: Available

Return to the main page which list all your ATP instances by clicking on the Autonomous Database link at the top of the page:

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Autonomous Database Details

**MELATPTRAIN01**

DB Connection Service Console Scale Up/Down

Autonomous Database Information Tags

**Workload Type:** Transaction Processing  
**Display Name:** MELATPTRAIN01  
**Database Name:** MELATPTRAIN01  
**CPU Core Count:** 1  
**Storage (TB):** 1



AVAILABLE

Autonomous Transaction Processing Database Details

**MELATPTRAIN01**

DB Connection Service Console Scale Up/Down

Autonomous Transaction Processing Database Information

**Display Name:** MELATPTRAIN01  
**Database Name:** MELATPTRAIN01  
**Database Version:** 18.4.0.0  
**CPU Core Count:** 1  
**Storage (TB):** 1



AVAILABLE

You are now ready to start Lab 2.



# Lab 2 - Securely Connecting to Autonomous Transaction Processing

# Lab 2. Securely Connecting to Autonomous Transaction Processing

## Objectives:

- Learn about the different Consumer Groups in Autonomous Transaction Processing (ATP)
- Learn how to download the credential wallet for your ATP instance
- Learn how to securely connect desktop tools to ATP

Applications and tools connect to ATP databases by using Oracle Net Services (also known as SQL\*Net). SQL\*Net supports a variety of connection types to ATP databases, including Oracle Call Interface (OCI), ODBC drivers, JDBC OC, and JDBC Thin Driver. Unlike other cloud services you do not get a UNIX command line interface on the system hosting your ATP instance, this reduces the complexity and need of UNIX skills required to administer it.

The sample SQL scripts for this lab are available in your VM under the directory `/home/oracle/labScripts/lab2`.

## Managing Priorities on Autonomous Transaction Processing

The priority of user requests in ATP is determined by the database service the user is connected to. Users are required to select a service when connecting to the database. The service names are in the format:

- *database\_name\_tpurgent*
- *database\_name\_tp*
- *database\_name\_low*
- *database\_name\_medium*
- *database\_name\_high*

These services map to the LOW, MEDIUM, HIGH, TP and TPURGENT consumer groups. For example, a user connecting to *database\_name\_low* service uses the consumer group LOW.

The basic characteristics of these consumer groups are:

- **tpurgent:** The highest priority application connection service for time critical transaction processing operations. This connection service supports manual parallelism.
- **tp:** A typical application connection service for transaction processing operations. Queries run serially.

- **high:** A high priority application connection service for reporting and batch operations with low concurrency requirements. All operations run in parallel (if you have multiple OCPU assigned to your instance) and are subject to queuing.
- **medium:** A typical application connection service for reporting and batch operations. All operations run in parallel and are subject to queuing.
- **low:** A lowest priority application connection service for high concurrency reporting or batch processing operations. Queries run serially.

As a database administrator and an application developer you need to select the database service based on your performance, concurrency and parallelism requirements.

## Downloading the credentials wallet

As ATP only accepts secure connections to the database, you need to download the wallet file containing your credentials first.

The wallet is downloaded from the ATP service console, or from the "**DB Connection**" button on the instance details page. To access the ATP Service console, find your database on the table listing ATP instances and click on the three vertical dots on the right-hand side.

In the pop-up menu select **Service Console**.

Autonomous Databases in ATP_Delegate Compartment						
Create Autonomous Database						
Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
dd18ATP	Available	dd18ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:49:52 GMT
dd19ATP	Available	dd19ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:45:51 GMT
dd20ATP	Available	dd20ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:53:53 GMT
dd21ATP	Available	dd21ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:01:55 GMT
dd22ATP	Available	dd22ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:05:56 GMT
dd23ATP	Available	dd23ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:09:57 GMT
dd24ATP	Available	dd24ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:13:58 GMT
dd25ATP	Available	dd25ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:17:59 GMT
MELATTRAIN01	Available	MELATTRAIN01	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT

This will open a new browser tab for the Service Console.

If you are prompted to do so, sign in to the service console with the following information.

**Username:** admin

**Password:** The administrator password you specified during provisioning

Oracle Autonomous Transaction Processing Hands-on Lab Guide

Database name:  
**MELATPTRAIN01**

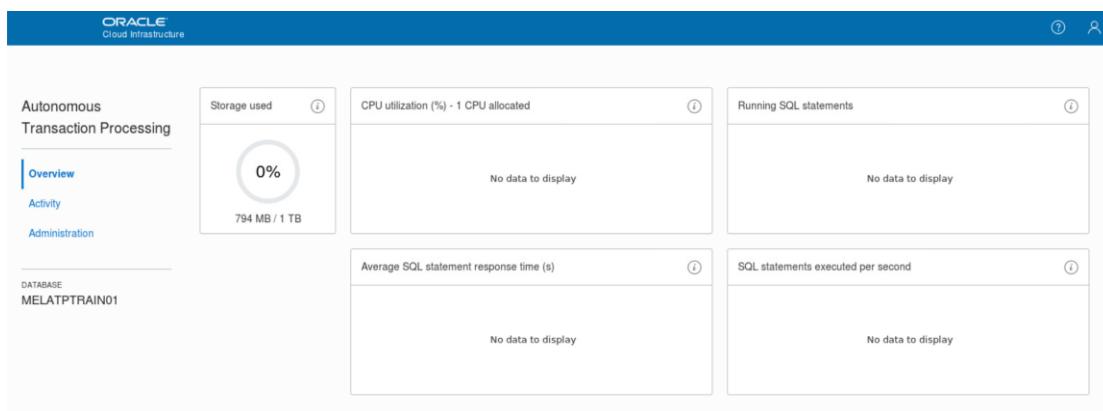
Sign in with your Oracle Autonomous Transaction Processing database credentials

USERNAME  
admin

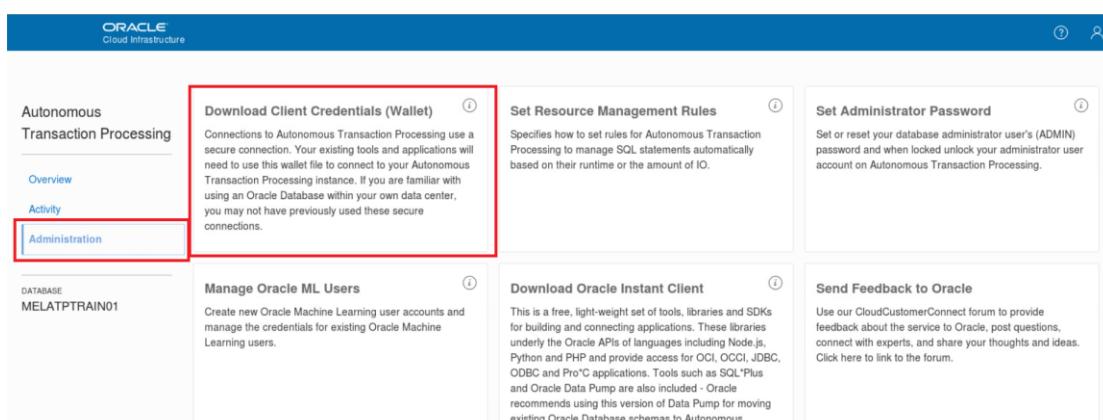
PASSWORD  
•••••••••••••••

**Sign in**

You will now see the main **dashboard** page for your instance. As we have not generated any load into the instance, it may have ‘No data to display’ in the information panes. Later lessons will explore the **Overview** and **Activity** tabs in more detail.



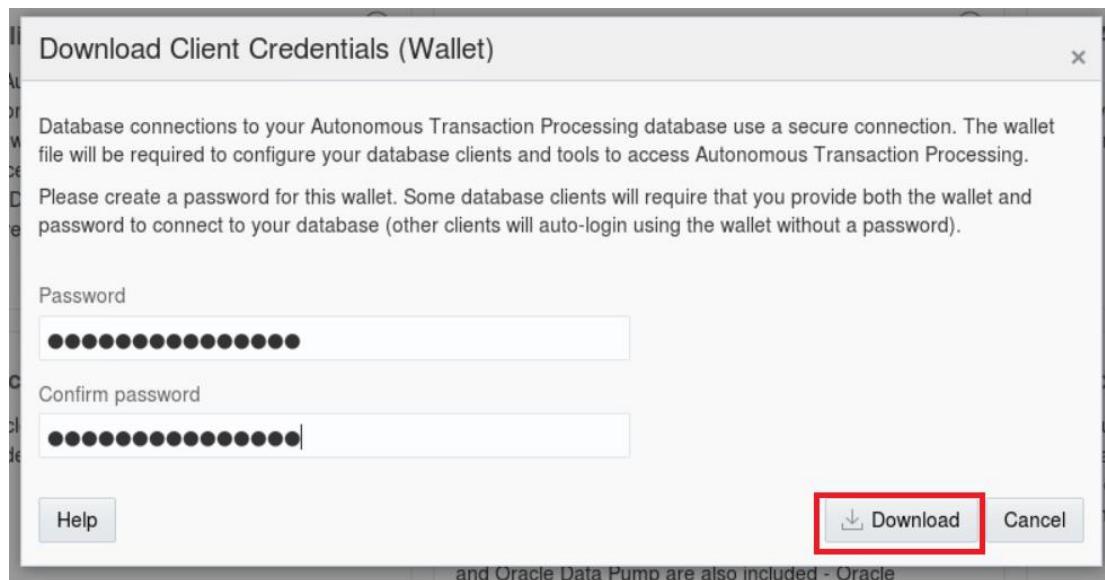
Click the “**Administration**” link in the left-hand side menu and click “**Download Client Credentials**” to download the wallet.



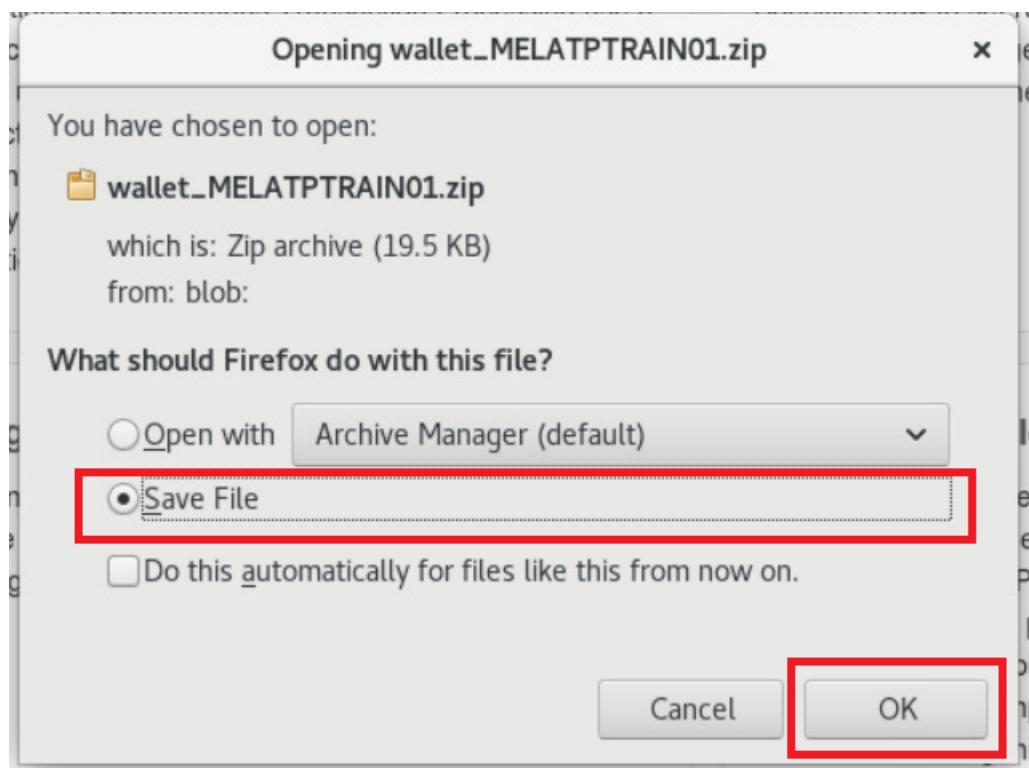
Specify a password for the wallet. Some applications require this password when connecting to the database, for example some JDBC thin applications will require this password to use as the keystore password. Note that this password is separate

from the admin password and can be set to a different value. For this lab, you could use the value ATPwelcome-1234 or another memorable password of your choice.

Click Download to download the wallet file to your lab virtual machine.



Select to Save the file, and then click ok. This will save the file in the default downloads location \$HOME/Downloads

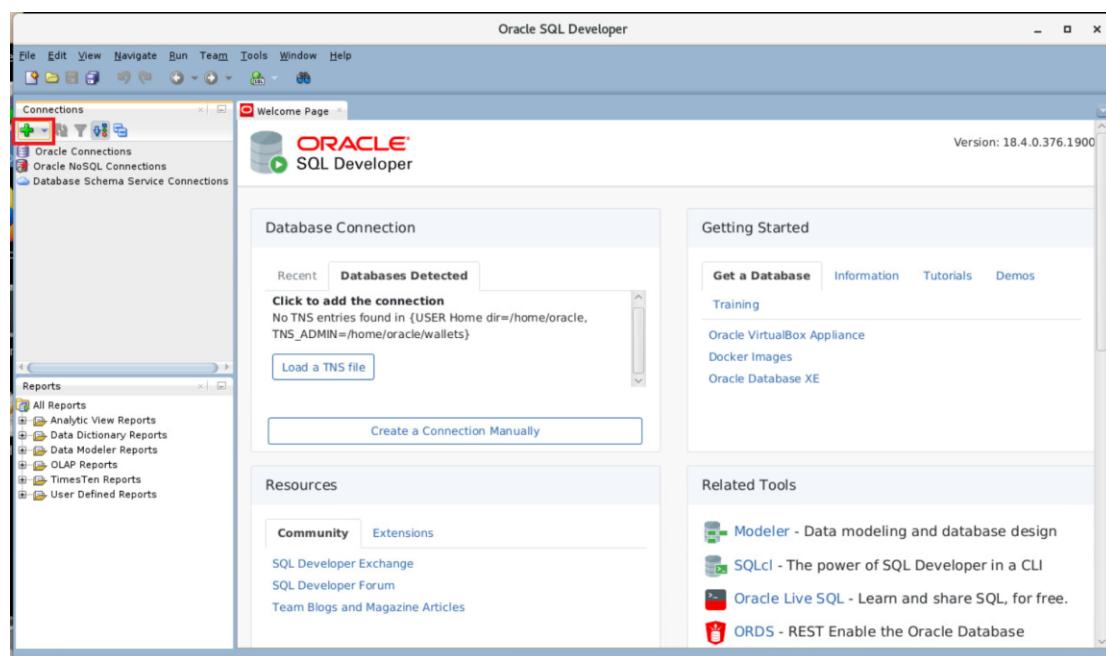


## Connecting to the database using SQL Developer

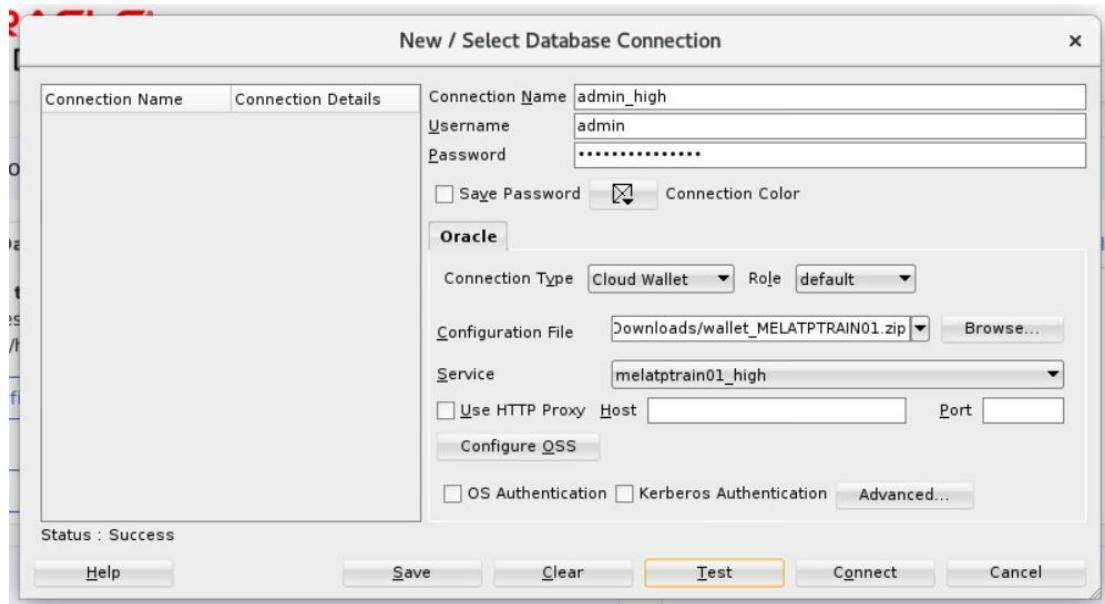
Minimise your Firefox window, and on your Lab VM desktop, start SQL Developer by double clicking on the icon.



Click the Create Connection icon in the Connections toolbox on the top left of the SQL Developer homepage.



This will open the **New>Select Database Connection Screen**. See below the screenshot for information on how to complete this form.



**Connection Name:** admin\_high

**Username:** admin

**Password:** The admin password that was specified provisioning process. This will have been provided by your lab leaded.

**Connection Type:** Cloud Wallet

**Configuration File:** Enter the full path for the wallet file you downloaded earlier in the lab or click the Browse button to point to the locate the file (by default it will be under your Downloads directory).

**Service:** You must select the correct service for your ATP Instance. The Wallet will contain the service names for all the ATP databases in the tenancy so this list could be long. Please make sure that you are selecting your database. As discussed previously there are 5 pre-configured database services for each database. Pick <your databasename>\_high for this lab. For example, if you created a database named **melatptrain01** select **melatptrain01\_high** as the service.

Test your connection by clicking the Test button, if it succeeds save your connection information by clicking Save, then connect to your database by clicking the Connect button.

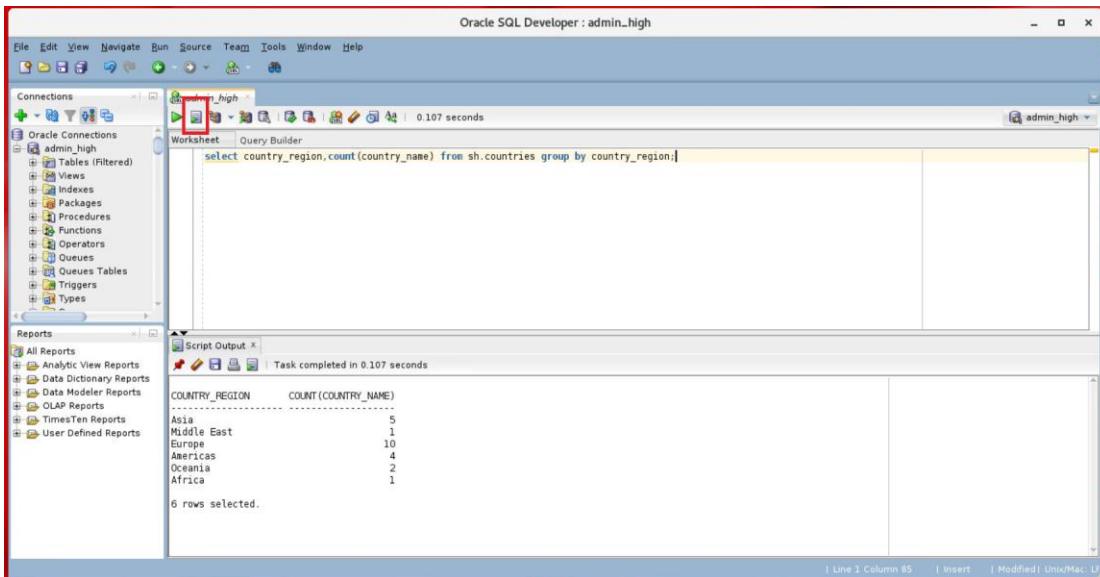
You can now run a test query using the sample data in the SH schema.

In the SQL Worksheet enter the following SQL:

```
select country_region, count(country_name) from sh.countries group by country_region;
```

and press the 'Run Script' button or press F5.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



*Note – You do not have to use GUI tools to access an ATP instance. Other Oracle Client utilities such as SQL Plus can connect to the ATP instance using a wallet.*

# Section 2 -

# Focus on DBAs



# Lab 3 - Scaling an ATP Instance

# Lab 3. Scaling an Automatic Transaction Processing Instance

## Objectives:

- Learn how to scale your Autonomous Transaction Processing (ATP) instance.

Scaling in the context of an ATP database means increasing or decreasing the amount of CPU or storage resources allocated to the service. Scaling an ATP instance is easy, flexible and can be done without any downtime so your applications continue to run unaffected while the scaling operation is in progress.

The sample SQL scripts for this lab are available in your VM under the directory **/home/oracle/labScripts/lab3**.

## Scaling your instance

To demonstrate that scaling an instance does not cause downtime we will run a long running query in SQL Developer and alter the number of CPUs available to the ATP instance while the SQL is executing.

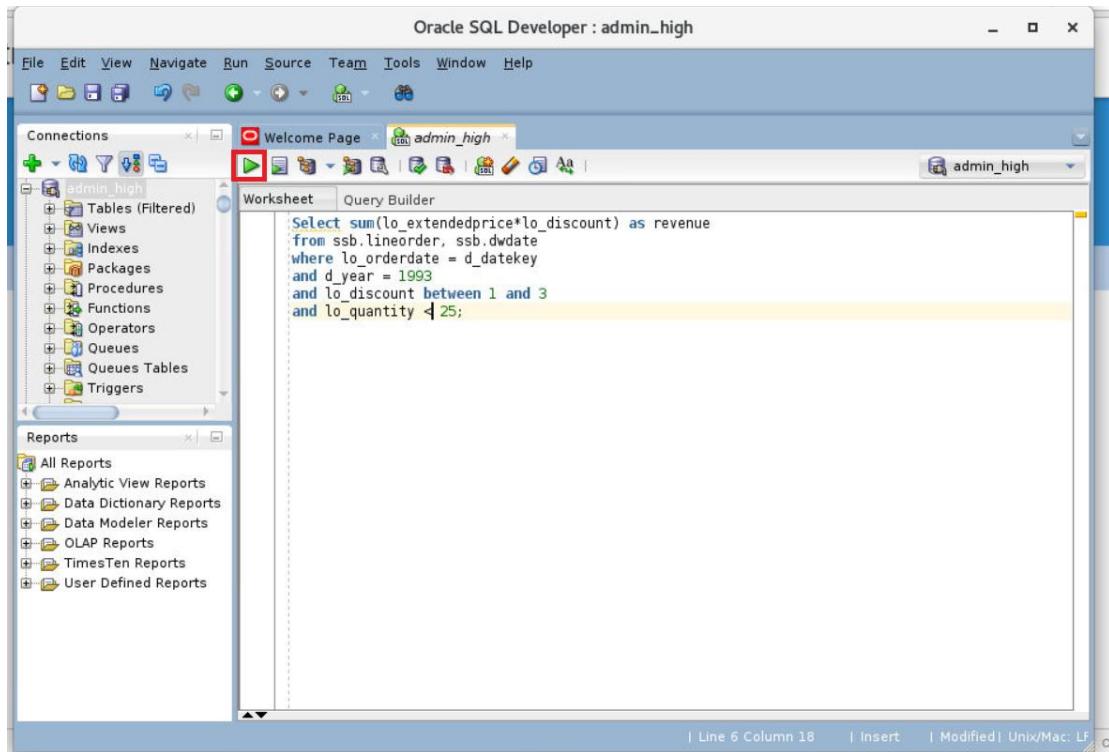
Start SQL Developer and open your **admin\_high** connection.

Paste the following long running SQL into the SQL Worksheet and select **Run Statement**. It will run for at least 10 minutes, allowing time to carry out the scaling operations while it is still executing. The sql can be found in the file **/home/oracle/labScripts/lab3/lab3\_long\_running.sql**

```
select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;
```

Leave this window open and the statement executing.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Return to your Firefox window that you used to during the provisioning exercise.

Navigate back to the **Autonomous Transaction Processing** page in the **ATP\_Delegate** compartment.

Click on the name of your database to open the instance details screen.

The screenshot shows the "Autonomous Databases in ATP\_Delegate Compartment" list. On the left, there are filters for COMPARTMENT (ATP\_Delegate), STATE (Any state), and WORKLOAD TYPE (All). The main table lists databases with columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. One row, "MELATPTTRAIN01", is highlighted with a red box.

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
MELATPTTRAIN01	Available	MELATPTTRAIN01	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT
dd25ATP	Available	dd25ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:13:58 GMT
dd24ATP	Available	dd24ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:09:57 GMT
dd23ATP	Available	dd23ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:05:56 GMT
dd22ATP	Available	dd22ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:01:55 GMT
dd21ATP	Available	dd21ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:57:54 GMT
dd20ATP	Available	dd20ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:53:53 GMT
dd19ATP	Available	dd19ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:49:52 GMT
dd18ATP	Available	dd18ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:45:51 GMT

Select the **Scale Up/Down** button.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

MELATPTRAIN01

DB Connection Service Console Scale Up/Down Stop Actions

Autonomous Database Information Tags

Workload Type: Transaction Processing  
Display Name: MELATPTRAIN01  
Database Name: MELATPTRAIN01  
CPU Core Count: 1  
Storage (TB): 1

Created: Fri, 08 Mar 2019 08:38:38 GMT  
Compartment: oraclemelatptrain01 (root)/ATP\_Workshop/ATP\_Delegate  
OCID: ...autfraq Show Copy  
License Type: Bring Your Own License  
Lifecycle State: Available

Resources Backups

Backups are automatically created daily.

Create Manual Backup

Name	State	Type	Started	Ended
No items found.				

Showing 0 item(s) < Page 1 >

Fill in the form with the following information.

Change the **CPU core count** from 1 to 2 (i.e. you are going to add 1 OCPU)  
Leave **the storage capacity** set at 1

Scale Up/Down

help cancel

CPU CORE COUNT

2

The number of CPU cores to enable. Available cores are subject to your tenancy's service limits.

STORAGE (TB)

1

The amount of storage to allocate.

Update

Click “**Update**” after setting the values in the form as shown above.

This will return you to the instance page and the status will now show “**Scaling in progress**” and the “**Scale Up/Down**” button is grayed out.

MELATPTRAIN01

DB Connection Service Console Scale Up/Down Start Actions

Autonomous Database Information Tags

Workload Type: Transaction Processing  
Display Name: MELATPTRAIN01  
Database Name: MELATPTRAIN01  
CPU Core Count: 1  
Storage (TB): 1

Created: Fri, 08 Mar 2019 08:38:38 GMT  
Compartment: oraclemelatptrain01 (root)/ATP\_Workshop/ATP\_Delegate  
OCID: ...autfraq Show Copy  
License Type: Bring Your Own License  
Lifecycle State: Scaling In Progress...

Resources Backups

Backups are automatically created daily.

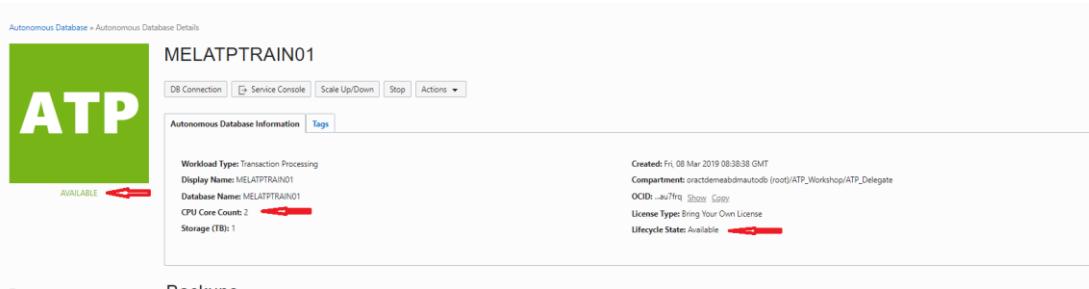
Create Manual Backup

Name	State	Type	Started	Ended
No items found.				

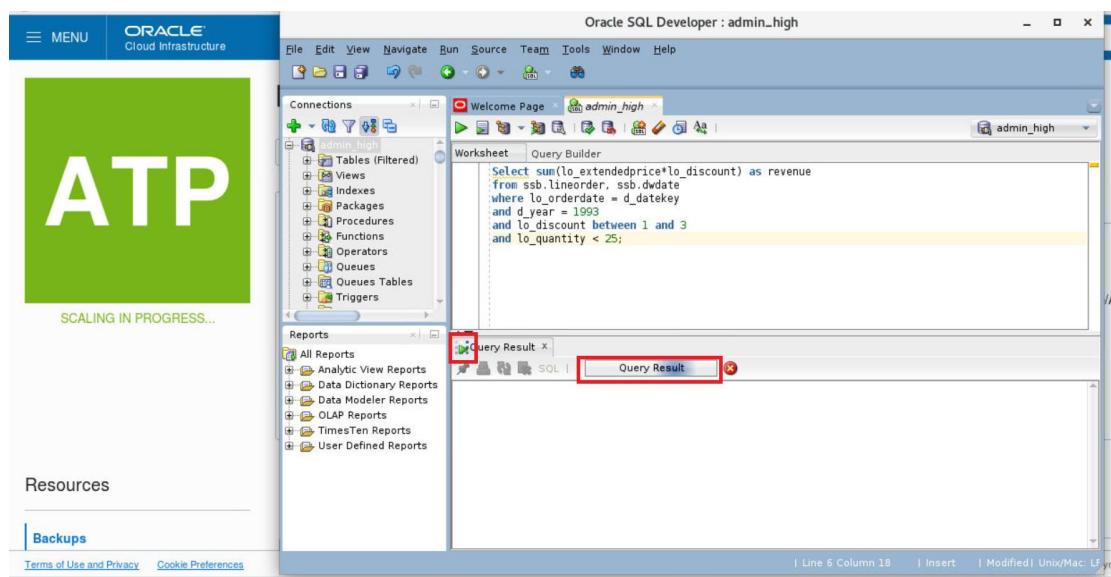
The page will automatically refresh once the scale operation has completed. The Lifecycle state will return to “**Available**”

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Note that the “ATP” square remains green during the scaling process. This means that applications can continue running during the scale operation without downtime.



Bring your SQL Developer window back to the foreground. Verify that the query is still executing and has not been interrupted. You can see this by looking at the Query Result Tab, and the Query Result bar will be moving right to left and there will be rotating symbols around the icon. Once you have verified the query has not been interrupted, proceed to scaling your instance back to 1 OCPU.



Now scale your instance back to 1 OCPU. Return to your Firefox Window.

On the Instance Details Screen select **Scale Up/Down** button.  
Fill in the form with the following information.

Change the **CPU core count** from 2 to **1** (i.e. you are going to remove 1 OCPU)  
Leave the **storage capacity** set at 1

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows a 'Scale Up/Down' dialog box. It has two input fields: 'CPU CORE COUNT' set to 1, and 'STORAGE (TB)' set to 1. Both fields have up and down arrows for adjustment. Below each field is a descriptive message: 'The number of CPU cores to enable. Available cores are subject to your tenancy's service limits.' and 'The amount of storage to allocate.'. At the bottom left is a blue 'Update' button.

Click “**Update**” after setting the values in the form as shown above.

This will return you the instance page and the status will now show “**Scaling in progress**” and the “**Scale Up/Down**” button is grayed out.

The screenshot shows the 'Autonomous Database Details' page for database 'MELATPTRAIN01'. On the left is a green ATP logo with 'SCALING IN PROGRESS...' text. The main area displays database information: Workload Type: Transaction Processing, Display Name: MELATPTRAIN01, Database Name: MELATPTRAIN01, CPU Core Count: 2, Storage (TB): 1. To the right, detailed information is shown: Created: Fri, 08 Mar 2019 08:38:38 GMT, Compartment: orclcdemeabdmautodb (root)/ATP\_Workshop/ATP\_Delegate, OCID: oc1.vcn1.azoty.20190308t083838z.1, License Type: Bring Your Own License, Lifecycle State: Scaling In Progress.. Below the main area are tabs for 'Resources' and 'Backups'.

Bring your SQL Developer window back to the foreground. Verify that the query is still executing and has not been interrupted. You can see this by looking at the Query Result Tab, and the Query Result bar will be moving right to left and there will be rotating symbols around the icon.

Once the Lifecycle state of your instance has returned to “**Available**” you are now ready to proceed with the next exercise.

Leave this SQL statement executing in the background, it can be used as part of the monitoring lab.



# Lab 4 - Managing and monitoring ATP

# Lab 4. Managing and Monitoring Autonomous Transaction Processing

## **Objectives:**

- Learn how to use the Autonomous Transaction Processing (ATP) Cloud Service Console
- Learn how to monitor the performance of your ATP instance
- Learn how to monitor individual SQL statements
- Learn how to stop and start your ATP instance
- Learn about the ATP backup and restore processes

In this section you will use the Cloud Console and the service console to manage and monitor your ATP instance. You will learn how you can stop and start your instance, and how you could recover your database if needed.

You will use the service console to monitor the performance of your ATP instance.

## **About Backup and Recovery on Autonomous Transaction Processing**

ATP automatically backs up your database for you. The retention period for automatic and manual backups is 60 days. You can restore and recover your database to any point-in-time in this retention period.

### **Manual Backups**

You do not have to do any manual backups for your database as ATP backs up your database automatically. You can do manual backups using the cloud console; for example, if you want to take a backup before a major change to make restore and recovery faster. The manual backups are put in a Cloud Object Storage bucket in your tenancy, which you must configure before you carry out manual backups. When you initiate a point-in-time recovery ATP decides which backup to use for faster recovery.

### **Recovery**

You can initiate recovery for your ATP database using the cloud console. ATP automatically restores and recovers your database to the point-in-time you specify.

You can see what backups have been created for your instance on the ATP Database Details page. As your instances were only provisioned today, there will be no backups. A sample output is below.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows the Oracle Cloud Service Console for the MELATPTRAIN01 instance. At the top, there's a green banner with the word 'ATP' and the text 'SCALING IN PROGRESS...'. Below the banner, there are tabs for 'Autonomous Database Information' and 'Tags'. Under 'Autonomous Database Information', it lists the Workload Type as 'Transaction Processing', Display Name as 'MELATPTRAIN01', Database Name as 'MELATPTRAIN01', CPU Core Count as 2, and Storage (TB) as 1. It also shows the creation date as Fri, 08 Mar 2019 08:38:38 GMT, Compartment as 'oractelemelatptrain01 (root)/ATP\_Workshop/ATP\_Delegate', OCID as 'ocid1.instance.oc1.ashzq...', License Type as 'Bring Your Own License', and Lifecycle State as 'Scaling In Progress...'. Below this, there's a 'Backups' section with a table showing a single backup entry: 'Mar 08, 2019 08:48:21 AM UTC' (Active, Incremental, initiated by Auto Backup), 'Started' at 'Fri, 08 Mar 2019 08:48:09 GMT', and 'Ended' at 'Fri, 08 Mar 2019 08:48:21 GMT'. The entire 'Backups' table row is highlighted with a red box.

## Exploring Service Console

There are two routes to the Service Console:-

1. Connect to the Service Console for your instance by selecting it from the menu next to the name of your ATP instance on the list of ATP Instances in your compartment.

The screenshot shows the list of Autonomous Databases in the 'ATP\_Delegate' compartment. On the left, there's a sidebar with 'List Scope' and 'Filters' (set to 'Any state'). The main area displays a table of databases with columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. The 'MELATPTRAIN01' database is listed as 'Available' with 'dd21ATP' as its database name, 1 CPU core, 1 TB storage, 'Transaction Processing' workload type, and created on 'Fri, 08 Mar 2019 08:38:38 GMT'. To the right of the table, there are several buttons: 'View Details', 'Service Console' (which is highlighted with a red box), 'Copy OCID', 'Apply Tag(s)', and 'Terminate'. The 'Service Console' button is the one being focused on.

2. Or you can click the name of your instance to open the instance page and select the 'Service Console' button.

The screenshot shows the instance details for 'MELATPTRAIN01'. At the top, there's a green banner with the word 'ATP' and the text 'AVAILABLE'. Below the banner, there are tabs for 'Autonomous Database Information' and 'Tags'. Under 'Autonomous Database Information', it lists the same details as the previous screenshot. The 'Service Console' button is highlighted with a red box at the top of the page, just below the banner.

## Console Overview

The Overview page shows real-time and historical information about the utilization of the service. Your page may have 'No Data to Display' in the monitoring panes as your instance is newly provisioned.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



The components on this page are:

- **Storage:** This chart shows the total and used storage capacity of the service. It indicates what percentage of the space is currently in-use.
- **CPU utilization (%):** This chart shows the historical CPU utilization of the service.
- **Running SQL statements:** This chart shows the average number of running SQL statements historically.
- **Average SQL statement response time (s):** This chart shows the average response time of SQL statements historically.

The default retention period for performance data is eight days. So, the CPU utilization, running statements, and average SQL response time charts show data for up to the last eight days by default.

## Activity Page

Select the **Activity** link on the left-hand side.



The **Activity** page defaults to a near real time view, which shows performance data for the last hour. Selecting the '**Time Period**' tab allows you to change the time slider to view historic performance data if required.

## Monitored SQL

Select the '**Monitored SQL**' Tab. If you have the long running SQL statement from the previous lab shown in this table, use it to explore the interface, otherwise, just review the provided screen shots.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

STATUS	SQL TEXT	DURATION	START TIME	END TIME
✓ DONE		37.68 s		
✓ DONE (ALL ROWS)	Select sum(lt_extendedprice*lt_discount) as revenue from ssb.lineorder, ssb.dwddate where lt_orderdate = d_datekey and d_year = 1993 and lt_discount between 1 and 3 and lt_quantity < 25	15.71 min	Mon, 07 Jan 2019 14:54:45 GMT	Mon, 07 Jan 2019 15:00:15 GMT
✗ DONE (ERROR)	Select /*+no_result_cache*/ c_city, c_region, count(*) From ssb.customer Group by c_cit	1.75 min	Mon, 07 Jan 2019 13:18:12 GMT	Mon, 07 Jan 2019 13:18:15 GMT
✓ DONE	CREATE INDEX "SOEDEMO"."ITEM_PRODUCT_IX" ON "SOEDEMO"."ORDER_ITEMS"	6 s	Mon, 07 Jan 2019 12:27:56 GMT	Mon, 07 Jan 2019 12:27:56 GMT
✓ DONE	CREATE INDEX "SOEDEMO"."ITEM_ORDER_IX" ON "SOEDEMO"."ORDER_ITEMS"	5 s	Mon, 07 Jan 2019 12:27:56 GMT	Mon, 07 Jan 2019 12:27:56 GMT
✓ DONE	CREATE INDEX "SOEDEMO"."INV_WAREHOUSE_IX" ON "SOEDEMO"."INVENTORIES"	3 s	Mon, 07 Jan 2019 12:27:55 GMT	Mon, 07 Jan 2019 12:27:55 GMT
✓ DONE	CREATE INDEX "SOEDEMO"."INV_PRODUCT_IX" ON "SOEDEMO"."INVENTORIES"	6 s	Mon, 07 Jan 2019 12:27:53 GMT	Mon, 07 Jan 2019 12:27:53 GMT
✓ DONE	CREATE UNIQUE INDEX "SOEDEMO"."ORDER_ITEMS_PK" ON "SOEDEMO"."ORDI	9 s	Mon, 07 Jan 2019 12:27:52 GMT	Mon, 07 Jan 2019 12:27:52 GMT
✓ DONE	BEGIN dbms_stats.postprocess_stats(owner, tabname, lobj, lobjn, flags, rawstats, se	5 s	Mon, 07 Jan 2019 12:25:58 GMT	Mon, 07 Jan 2019 12:25:58 GMT
✓ DONE	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("ADDRESSES")1*/ INTO	11 s	Mon, 07 Jan 2019 12:25:52 GMT	Mon, 07 Jan 2019 12:25:52 GMT
✓ DONE	BEGIN SYS.KUPW\$WORKER.MAIN(SYS_IMPORT_FULL,'01','ADMIN'); END;	6.03 min	Mon, 07 Jan 2019 12:22:14 GMT	Mon, 07 Jan 2019 12:22:14 GMT

The **Monitored SQL** tab shows information about current and past monitored SQL statements. By default, Real-Time SQL Monitoring automatically starts when a SQL statement runs in parallel, or when it has consumed at least 5 seconds of CPU or I/O time in a single execution.

To see the detailed SQL monitor report for a statement, select the statement and right click to bring up the menu and click **Show Details**. This page can take a little while to populate with the data.

The **Overview** tab shows general information for the statement.

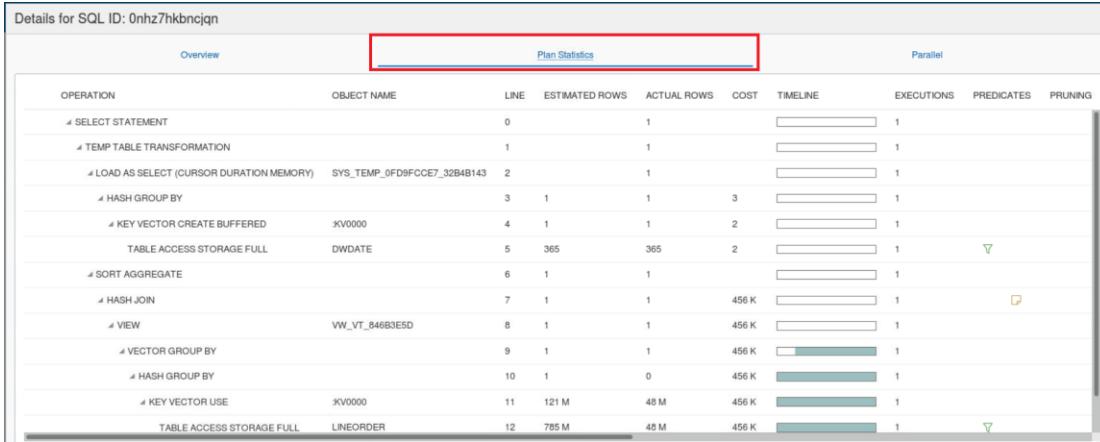
General	
Status	DONE (ALL ROWS)
Execution started	01/07/2019 14:54:45
Last refresh time	01/07/2019 15:10:28
Execution ID	63866050
User	ADMIN@BUMK28FBV/PYBRGK_MELATPTRAIN01
Consumer group	HIGH

Time & Wait Statistics	
Duration	15.71 min
Database Time	15.8 min
Activity %	100

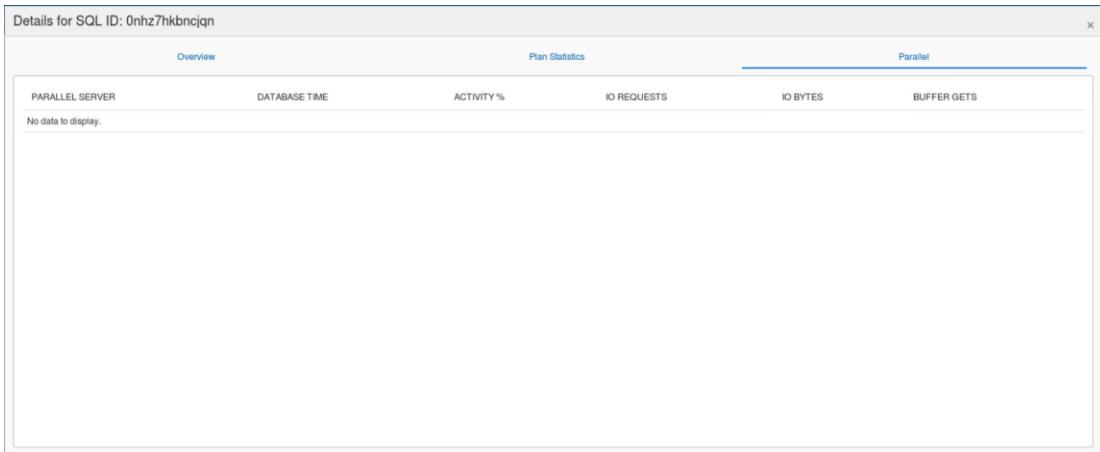
I/O Statistics	
Buffer Gets	21.16 M
I/O Requests	165.5 K
I/O Bytes	161.45 GB

The **Plan Statistics** tab shows the runtime execution plan of the statement.

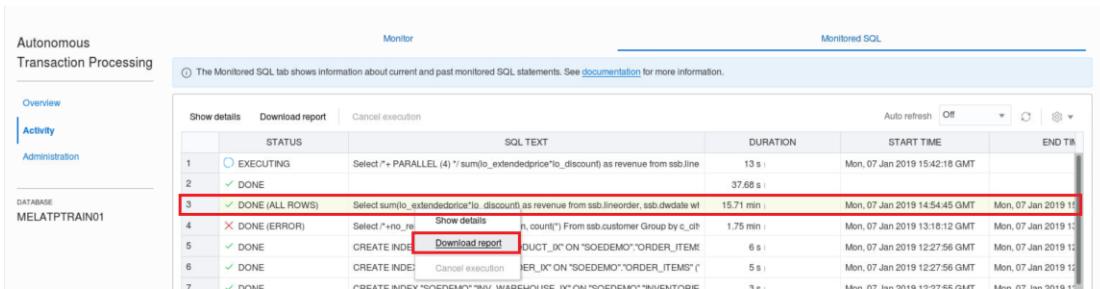
## Oracle Autonomous Transaction Processing Hands-on Lab Guide



If your statement uses parallel processing this will be documented on the **Parallel** tab

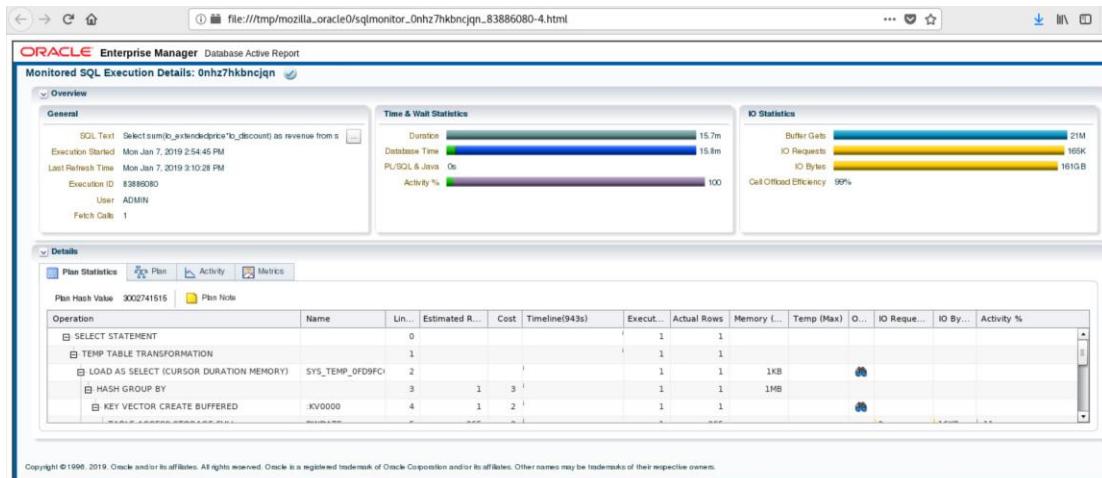


You can download the SQL Monitor report in HTML format by selecting the SQL statement, right clicking to bring up the menu and clicking 'Download report'



This will download a html document giving details about the SQL statement (note – to view the report you need Adobe Flash plugin in your browser). If you want you can download the report and open it in your Firefox browser.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



### Cancelling Execution

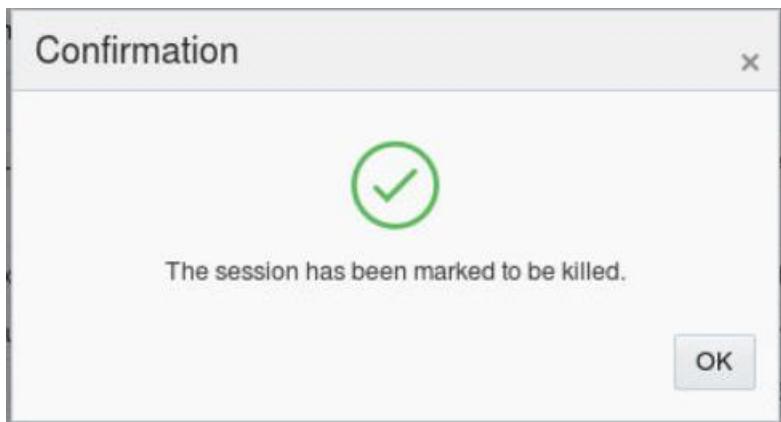
There is a ‘Cancel execution’ button. This can be used to cancel the execution of currently running SQL statements. This process is shown in screen shot only.  
Select the running SQL you want to cancel, and select **Cancel Execution**

STATUS	SQL TEXT	DURATION	START TIME	END TIME
EXECUTING	Select /*+ PARALLEL (4) */ sum(ss_extendedprice)*ss_discount as revenue from ssb.lineorder	13 s	Mon, 07 Jan 2019 15:42:18 GMT	
DONE		37.68 s		
DONE (ALL ROWS)	Select sum(ss_extendedprice)*ss_discount as revenue from ssb.lineorder, ssb.datedeal w/	15.71 min	Mon, 07 Jan 2019 14:54:45 GMT	Mon, 07 Jan 2019 11:
DONE (ERROR)	Select /*+ no_result_cache */ c_city, c_region, count(*) From ssb.customer Group by c_cit	1.75 min	Mon, 07 Jan 2019 13:16:12 GMT	Mon, 07 Jan 2019 1:
DONE	CREATE INDEX "SOEDEMO"."ITEM_PRODUCT_IK" ON "SOEDEMO"."ORDER_ITEMS"	6 s	Mon, 07 Jan 2019 12:27:56 GMT	Mon, 07 Jan 2019 1:
DONE	CREATE INDEX "SOEDEMO"."ITEM_ORDER_IK" ON "SOEDEMO"."ORDER_ITEMS"	5 s	Mon, 07 Jan 2019 12:27:56 GMT	Mon, 07 Jan 2019 1:
DONE	CREATE INDEX "SOEDEMO"."INV_WAREHOUSE_IK" ON "SOEDEMO"."INVENTORIES"	3 s	Mon, 07 Jan 2019 12:27:55 GMT	Mon, 07 Jan 2019 1:
DONE	CREATE INDEX "SOEDEMO"."INV_PRODUCT_IK" ON "SOEDEMO"."INVENTORIES"	6 s	Mon, 07 Jan 2019 12:27:53 GMT	Mon, 07 Jan 2019 1:
DONE	CREATE UNIQUE INDEX "SOEDEMO"."ORDER_ITEMS_PK" ON "SOEDEMO"."ORDERS"	9 s	Mon, 07 Jan 2019 12:27:52 GMT	Mon, 07 Jan 2019 1:
DONE	BEGIN obms_stats.postprocess_stats(owner, tabname, lobj, lobjn, flags, rawstats, se	5 s	Mon, 07 Jan 2019 12:25:58 GMT	Mon, 07 Jan 2019 1:
DONE	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("ADDRESSES",1) */ INTO	11 s	Mon, 07 Jan 2019 12:25:52 GMT	Mon, 07 Jan 2019 1:

There will be a Kill session pop up displayed.



And then a confirmation window that the session has been marked to be killed.

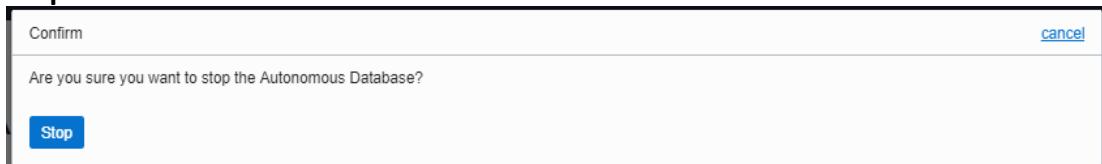


## Stopping your ATP instance

Go to the page listing ATP instances in your compartment, which you used during the provisioning exercise and open the name of your ATP instance to open to instance page. Click the **Stop** button.

A screenshot of the Oracle Cloud Infrastructure (OCI) Autonomous Database Details page for an instance named "MELATPTRAIN01". The instance status is "AVAILABLE". In the top navigation bar, there are buttons for "DB Connection", "Service Console", "Scale Up/Down", "Stop" (which is highlighted with a red box), and "Actions". Below the instance name, there are tabs for "Autonomous Database Information" (selected) and "Tags". Under "Autonomous Database Information", there are details like Workload Type (Transaction Processing), Display Name (MELATPTRAIN01), Database Name (MELATPTRAIN01), CPU Core Count (1), Storage (TB) (1), and Lifecycle State (Available). On the left, there are "Resources" and "Backups" sections. On the right, there are creation details (Created: Fri, 08 Mar 2019 08:38:38 GMT, Compartment: orclatempelautodb (root)/ATP\_Workshop/ATP\_Delegate, OCID: oc1t-azf7fr&gt;Show Copy), License Type (Bring Your Own License), and Lifecycle State (Available).

In the pop-up dialog confirm that you want to stop your instance by clicking on the **Stop** button.



The instance page will now show that your instance is stopping

Autonomous Database > Autonomous Database Details

MELATPTRAIN01

DB Connection Service Console Scale Up/Down Start Actions ▾

Autonomous Database Information Tags

Workload Type: Transaction Processing  
Display Name: MELATPTRAIN01  
Database Name: MELATPTRAIN01  
CPU Core Count: 1  
Storage (TB): 1

Resources Backups

Backups are automatically created daily.

Once the instance has stopped the page will automatically refresh to show the new state



Note that stopping your instance stops metering and charging for your compute resources, based on a full-hour cycle of usage. You can come back later and start your instance instantly anytime.

## Starting your ATP instance

Go to the Cloud Console you used during the previous exercise and open the Instances screen. Click the **Start** button.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

MELATPTRAIN01

DB Connection Service Console Scale Up/Down **Start** Actions ▾

Autonomous Database Information Tags

Workload Type: Transaction Processing  
Display Name: MELATPTRAIN01  
Database Name: MELATPTRAIN01  
CPU Core Count: 1  
Storage (TB): 1

Created: Fri, 08 Mar 2019 08:3  
Compartment: oractdemaebc  
OCID: ...au7frq [Show](#) [Copy](#)  
License Type: Bring Your Own  
Lifecycle State: Stopped

Resources Backups

Backups are automatically created daily.

Create Manual Backup

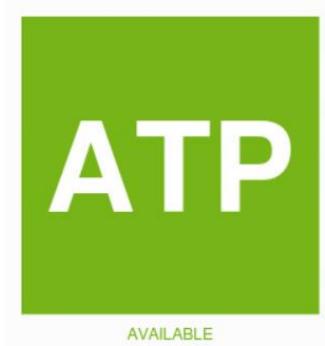
In the pop-up dialog confirm that you want to start your instance by clicking on the **Start** button.



The instance page will now show that your instance is starting



Starting the instance will take a few minutes. The Lifecycle state will change to Available when the instance has started.





# Lab 5 - Data Loading into ATP



# Lab 5. Data Loading into Autonomous Transaction Processing

## Objectives:

- You will learn how to load a local data file into your Autonomous Transaction Processing (ATP) instance.
- You will learn how to use the DBMS\_CLOUD package to load data into your ATP Instance from Object Storage.
- You will use Oracle Data Pump to load data from an export dump file.

You can load data into ATP Database using Oracle Database tools, and Oracle and 3<sup>rd</sup> party data integration tools.

You can load data:

- From files local to your client computer
- From files stored in a cloud-based object store

For the fastest data loading experience Oracle recommends uploading the source files to a cloud-based object store, such as Oracle Cloud Infrastructure Object Storage, before loading the data into your ATP Database.

To load data from files in the cloud into your ATP database, use the new PL/SQL DBMS\_CLOUD package. The DBMS\_CLOUD package supports loading data files from the following Cloud sources: Oracle Cloud Infrastructure Object Storage, Oracle Cloud Infrastructure Object Storage Classic, Azure Blob Storage and Amazon AWS S3.

This lab shows how to load data from Oracle Cloud Infrastructure Object Storage using two of the procedures in the DBMS\_CLOUD package:

- **Create\_credential:** Stores the object store credentials in your ATP database schema. You will use this procedure to create object store credentials in your ATP admin schema.
- **Copy\_data:** You will use this procedure to load tables in your admin schema with data from data files staged in the Oracle Cloud Infrastructure.

The sample SQL and shell scripts for this lab are available in your VM under the directory **/home/oracle/labScripts/lab5**.

## Loading a file from local storage

In this section you will load a local file from your machine into ATP:

To use as the source file, download this file to your Lab VM using Firefox:

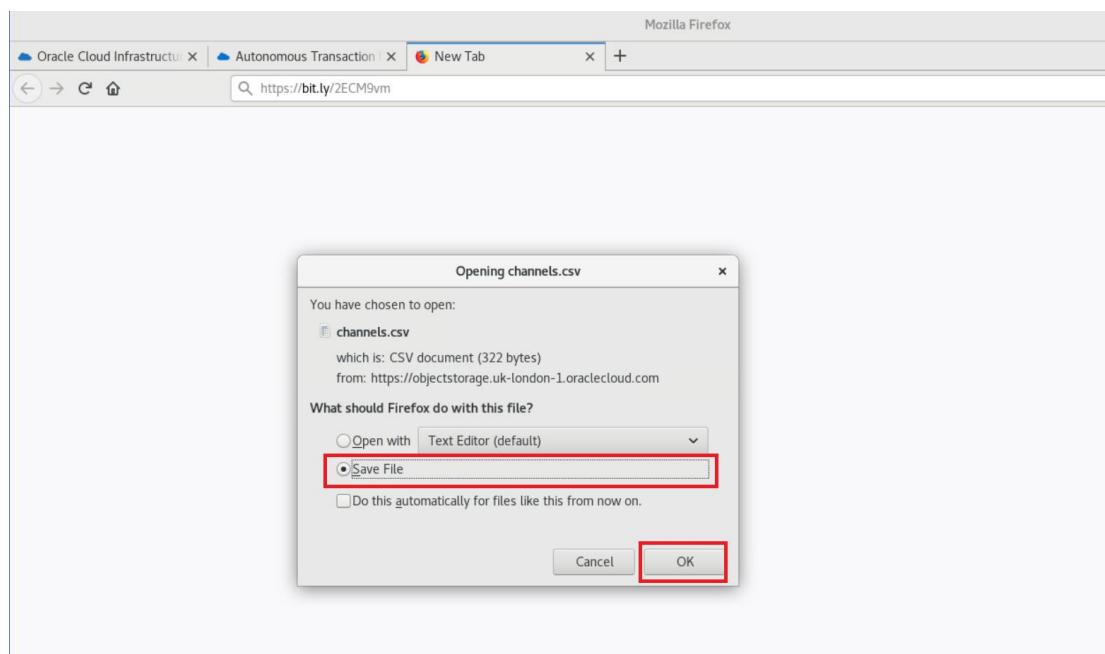
## Oracle Autonomous Transaction Processing Hands-on Lab Guide

[https://objectstorage.uk-london-1.oraclecloud.com/p/Xs-twgmNVlyOrpzCsnuTQlhMVXuiJ\\_XZ0jJ2VkJTeKnw/n/oscemea001/b/DEMO\\_DATA/o/channels.csv](https://objectstorage.uk-london-1.oraclecloud.com/p/Xs-twgmNVlyOrpzCsnuTQlhMVXuiJ_XZ0jJ2VkJTeKnw/n/oscemea001/b/DEMO_DATA/o/channels.csv)

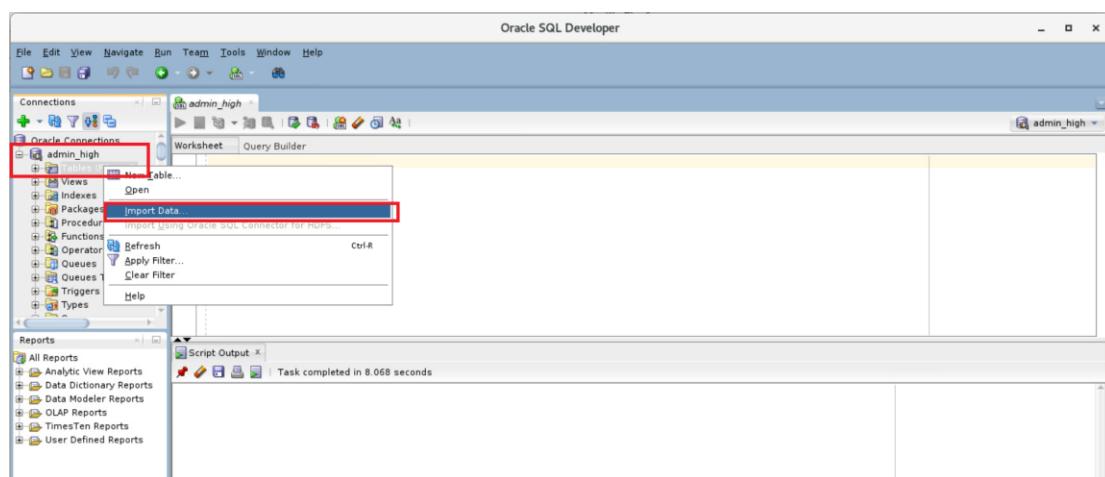
or use this short link

<https://bit.ly/2uu7zot>

Select **Save File**. This will save the file to default downloads location for the Firefox browser, **\$HOME/Downloads**.

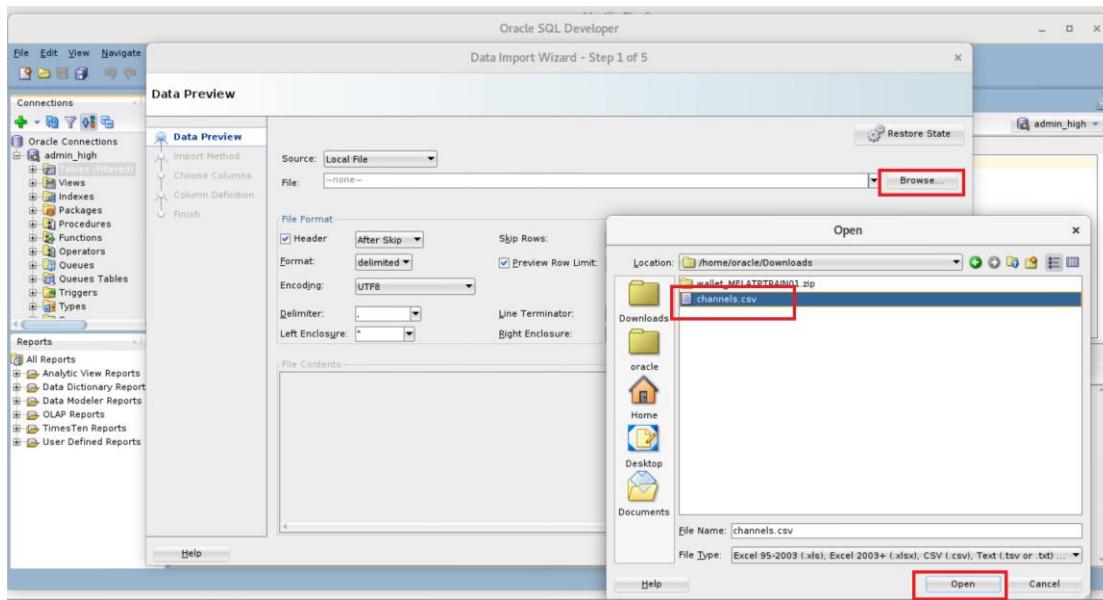


Go to SQL Developer, expand your **admin\_high** connection and right click Tables, then click Import Data.

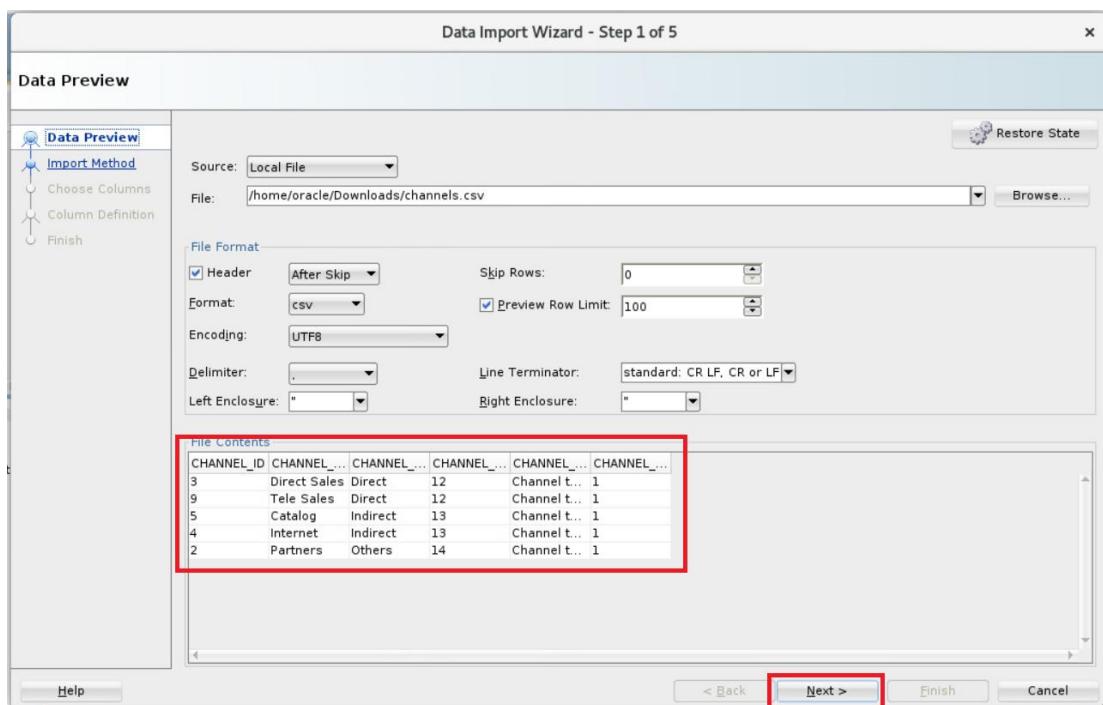


This will open the data import wizard. Click Browse and locate the channels.csv file you downloaded. (It will usually be in the Downloads folder under your home directory).

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

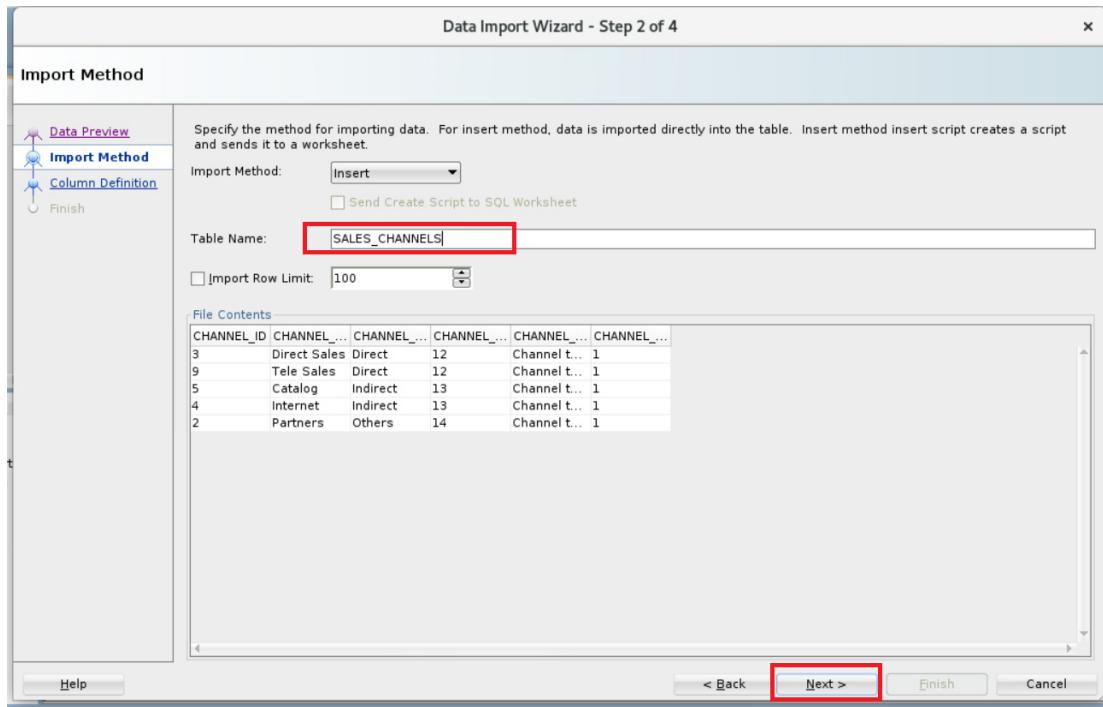


When you select the file you will see the file contents previewed in the import wizard.

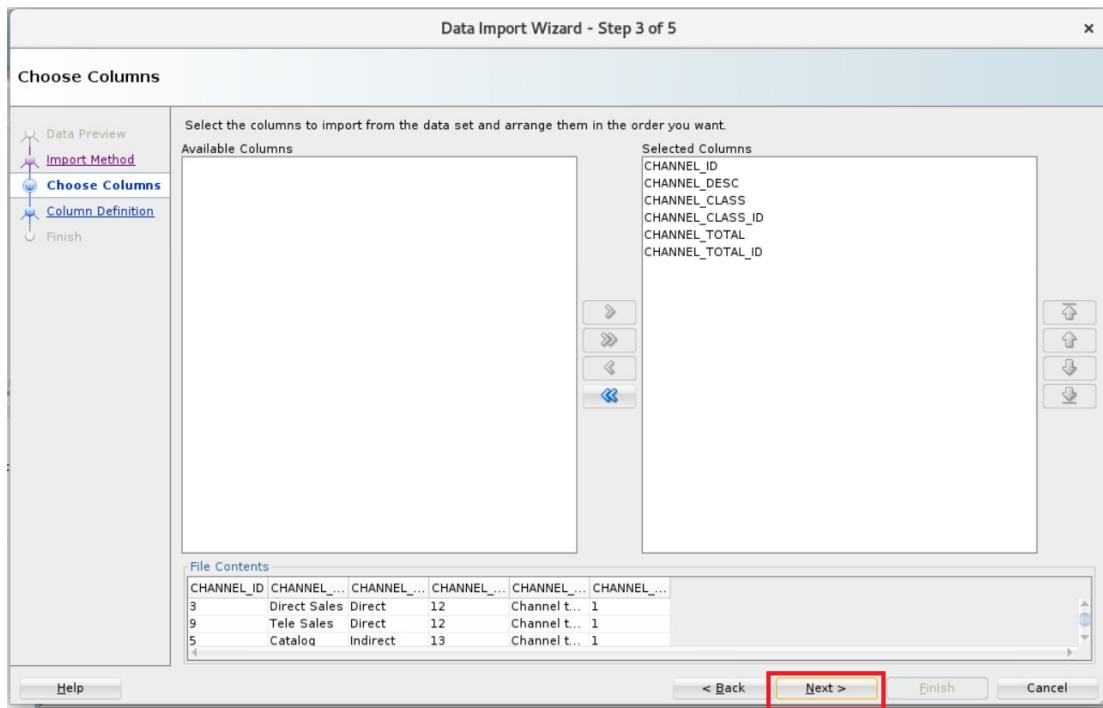


Click Next. In the next screen enter SALES\_CHANNELS as the table name you will create and load into.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

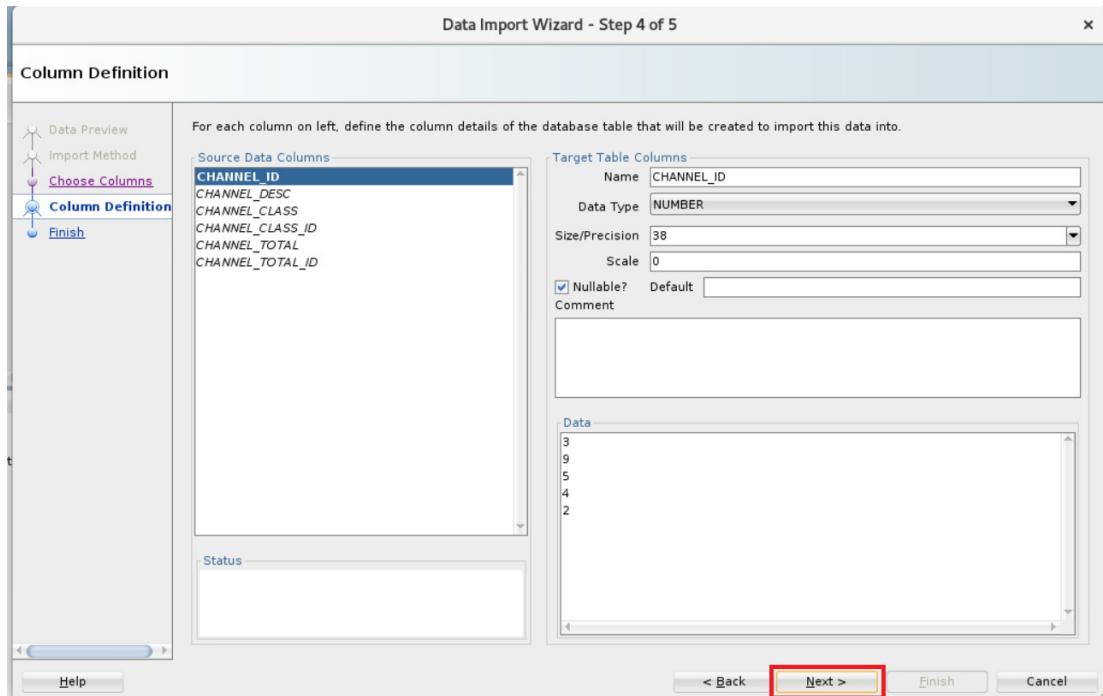


Click Next. The next screen allows you to select the columns you want for this table. For this exercise leave the columns as-is which means the table will have all columns available.

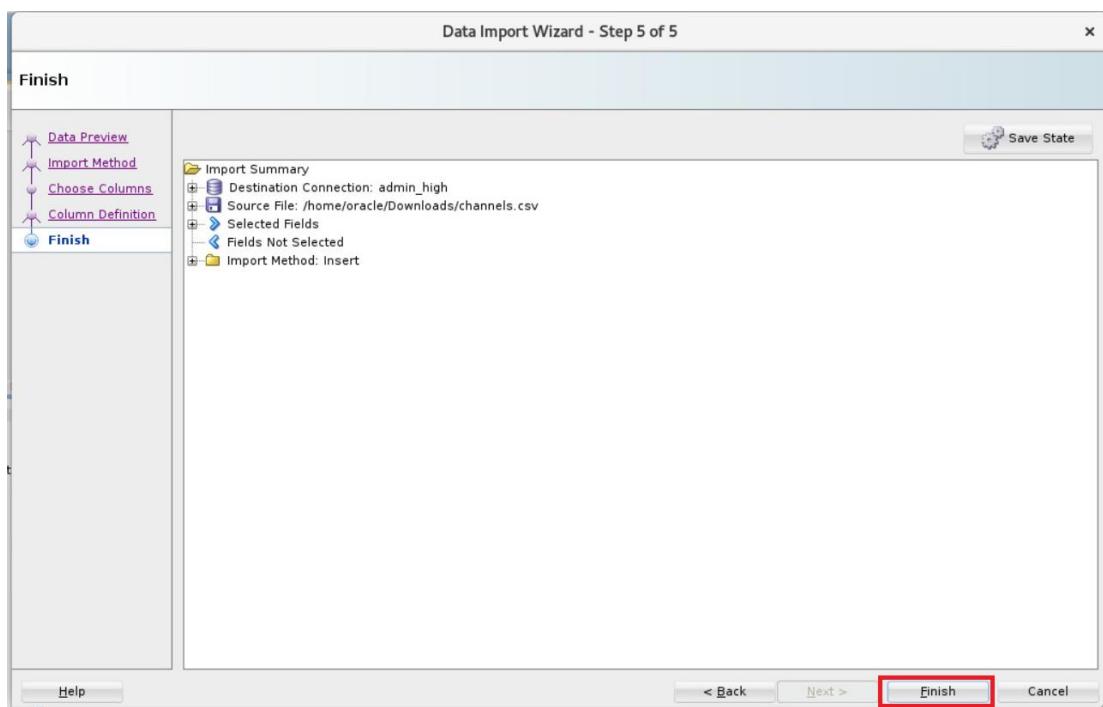


Click Next. The next screen allows you to look at the data types for each column. You can change the data types if you need to. For this exercise leave the data types as default.

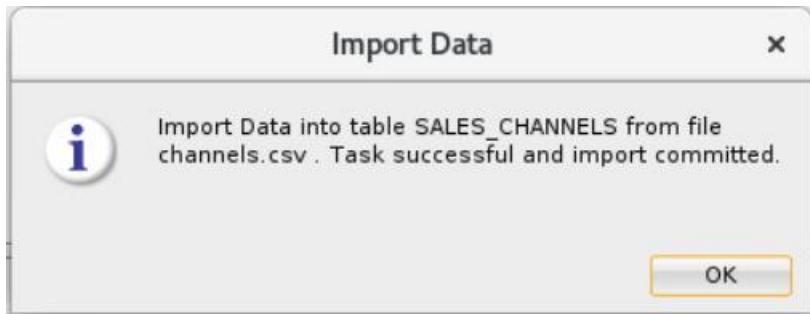
## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Click Next. The next page will display a summary for the import operation.



Click Finish to complete the import wizard and start the data load. When the data load finishes you will see a message saying the import was completed.



Your source file is now loaded into ATP. You can run a query on the table in SQL Worksheet to see your data.

```
select * from sales_channels;
```

A screenshot of the Oracle Database SQL Worksheet interface. The top navigation bar shows 'admin\_high'. The 'Worksheet' tab is selected. In the main area, a query is entered: 'select \* from sales\_channels;'. Below the query, the results are displayed in a table titled 'Script Output' with a 'Query Result' tab selected. The results show five rows of data from the SALES\_CHANNELS table. The table has columns: CHANNEL\_ID, CHANNEL\_DESC, CHANNEL\_CLASS, CHANNEL\_CLASS\_ID, CHANNEL\_TOTAL, and CHANNEL\_TOTAL\_ID. The data is as follows:

CHANNEL_ID	CHANNEL_DESC	CHANNEL_CLASS	CHANNEL_CLASS_ID	CHANNEL_TOTAL	CHANNEL_TOTAL_ID
1	3 Direct Sales	Direct		12 Channel total	1
2	9 Tele Sales	Direct		12 Channel total	1
3	5 Catalog	Indirect		13 Channel total	1
4	4 Internet	Indirect		13 Channel total	1
5	2 Partners	Others		14 Channel total	1

## Loading a file from Object Storage

### Information about how the environment was prepared

To load data from the Oracle Cloud Infrastructure Object Storage you will need a Cloud user with the appropriate privileges to read data from the Object Store. A user 'atp\_oss\_access' with the right setup and authentication token has been pre-created for you to use in the next step. If you are interested in seeing an outline of how it was setup, this is documented in Appendix B.

### Storing your object store authentication token credentials in the database

To access data in the Object Store you must enable your database user to authenticate itself with the Object Store using your object store account and authentication token.

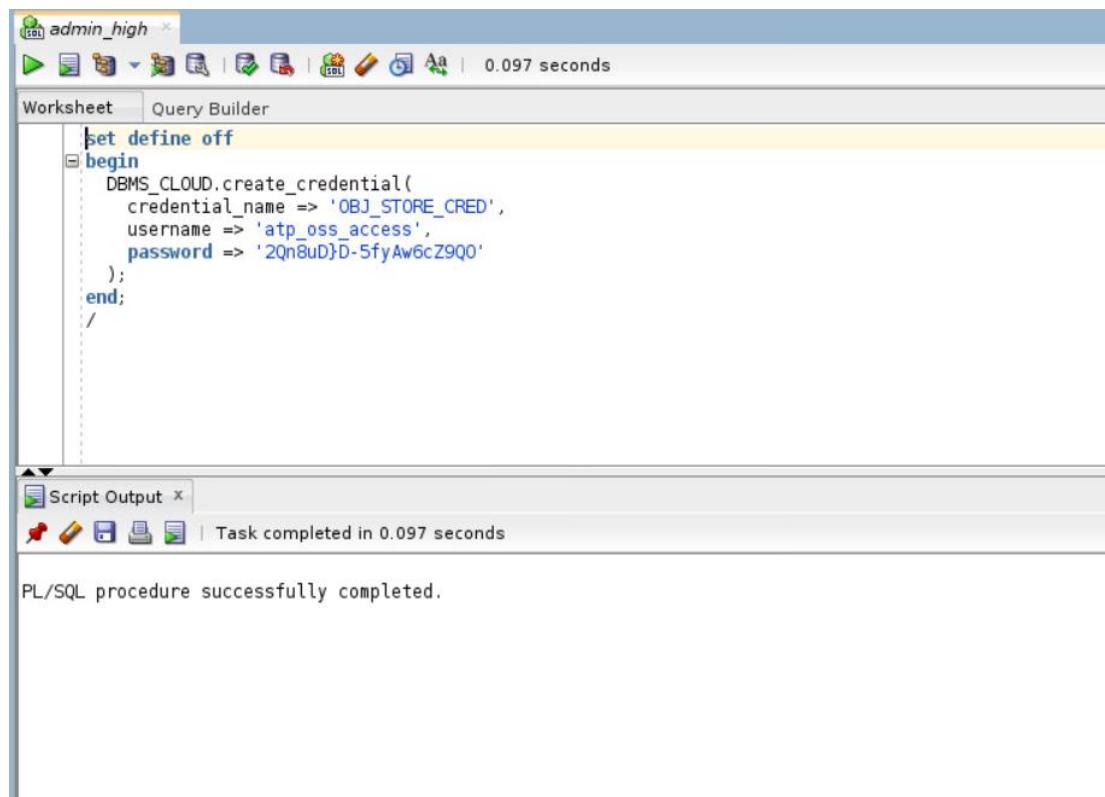
You do this by creating a private CREDENTIAL object for your user that stores this information encrypted in your ATP instance. This encrypted connection information is only usable by your user schema.

Within the SQL worksheet of SQL Developer in your **admin\_high** connection, execute the following code to store the object store credential in the database:

The below sql can be found in the file

**/home/oracle/labScripts/lab5/lab5\_create\_credential.sql**

```
set define off
begin
  DBMS_CLOUD.create_credential(
    credential_name => 'OBJ_STORE_CRED',
    username => 'atp_oss_access',
    password => '2Qn8uD}D-5fyAw6cZ9QO'
  );
end;
/
```



The screenshot shows the Oracle SQL Developer interface. The title bar says "admin\_high". The main window is titled "Worksheet" and contains the PL/SQL code for creating a credential. The code is as follows:

```
set define off
begin
  DBMS_CLOUD.create_credential(
    credential_name => 'OBJ_STORE_CRED',
    username => 'atp_oss_access',
    password => '2Qn8uD}D-5fyAw6cZ9QO'
  );
end;
/
```

The "Script Output" tab at the bottom shows the result:

```
PL/SQL procedure successfully completed.
```

Now you are ready to load data from Object Store

## Loading Data from Object Store

You can use the PL/SQL package DBMS\_CLOUD directly to load the data from object store. It can be invoked directly or implicitly by using tools within SQL Developer.

Connect as your admin user in SQL Developer using your **admin\_high** connection.

Prepare your destination table by running the create table statement

The sql can be found in **/home/oracle/labScripts/lab5/lab5\_create\_countries.sql**

```
CREATE TABLE COUNTRIES (
    COUNTRY_ID          NUMBER,
    COUNTRY_ISO_CODE    CHAR(2),
    COUNTRY_NAME        VARCHAR2(40),
    COUNTRY_SUBREGION   VARCHAR2(30),
    COUNTRY_SUBREGION_ID NUMBER,
    COUNTRY_REGION      VARCHAR2(20),
    COUNTRY_REGION_ID   NUMBER,
    COUNTRY_TOTAL        VARCHAR2(30),
    COUNTRY_TOTAL_ID    NUMBER,
    COUNTRY_NAME_HIST   VARCHAR2(40)
);
```

The screenshot shows the Oracle SQL Developer interface. The top menu bar has tabs for 'Welcome Page', 'admin\_high', and 'COUNTRIES'. Below the menu is a toolbar with various icons. The main area is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the following SQL code:

```

CREATE TABLE COUNTRIES (
    COUNTRY_ID          NUMBER,
    COUNTRY_ISO_CODE    CHAR(2),
    COUNTRY_NAME        VARCHAR2(40),
    COUNTRY_SUBREGION   VARCHAR2(30),
    COUNTRY_SUBREGION_ID NUMBER,
    COUNTRY_REGION      VARCHAR2(20),
    COUNTRY_REGION_ID   NUMBER,
    COUNTRY_TOTAL       VARCHAR2(30),
    COUNTRY_TOTAL_ID    NUMBER,
    COUNTRY_NAME_HIST   VARCHAR2(40)
);

```

Below the code, the 'Script Output' tab shows the message: 'Table COUNTRIES created.' and 'Task completed in 0.056 seconds'. The bottom status bar indicates '1 Line 24'.

Next, paste the following code to copy data from object storage using the DBMS\_CLOUD package. We use the copy\_data procedure of the DBMS\_CLOUD package to copy the data (countries.csv) staged in your object store.

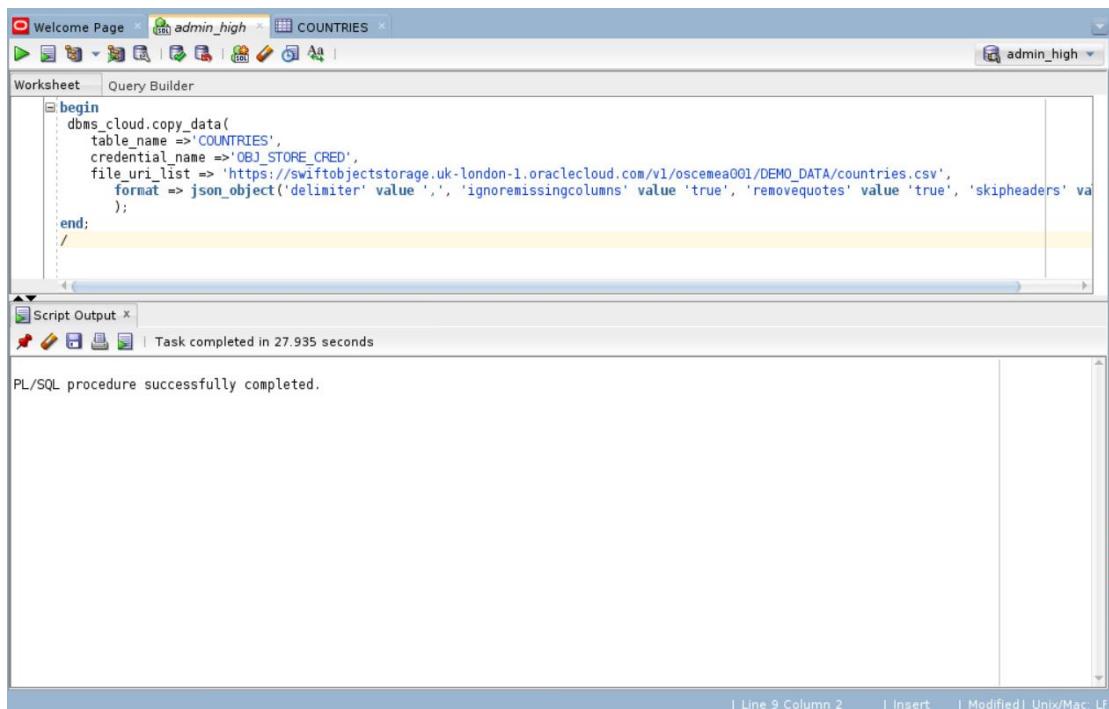
The below sql can be found under  
[/home/oracle/labScripts/lab5/lab5\\_dbms\\_cloud\\_sample.sql](/home/oracle/labScripts/lab5/lab5_dbms_cloud_sample.sql) in your vm.

```

begin
    dbms_cloud.copy_data(
        table_name =>'COUNTRIES',
        credential_name =>'OBJ_STORE_CRED',
        file_uri_list => 'https://swiftobjectstorage.uk-london-
1.oraclecloud.com/v1/oscemea001/DEMO_DATA/countries.csv',
        format => json_object('delimiter' value ',',
        'ignoremissingcolumns' value 'true', 'removequotes' value 'true',
        'skipheaders' value '1')
    );
end;
/

```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



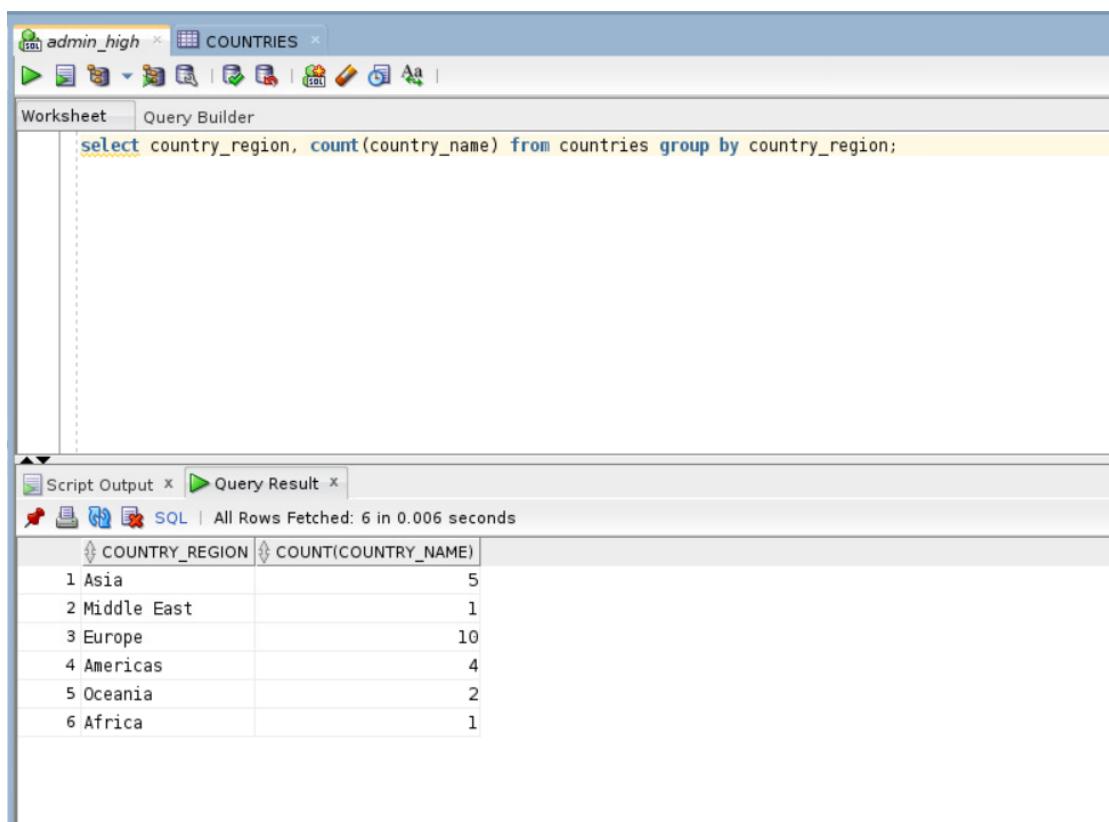
The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'admin\_high', and 'COUNTRIES'. The main area is titled 'Worksheet' and contains a 'Query Builder' pane. The code in the query builder is:

```
begin
  dbms_cloud.copy_data(
    table_name =>'COUNTRIES',
    credential_name =>'OBJ_STORE_CRED',
    file_uri_list => 'https://swiftobjectstorage.uk-london-1.oraclecloud.com/v1/oscemea001/DEMO_DATA/countries.csv',
    format => json_object('delimiter' value ',' , 'ignoremissingcolumns' value 'true', 'removequotes' value 'true', 'skipheaders' value '1');
  end;
/
```

Below the worksheet, a 'Script Output' tab is visible with the message: 'Task completed in 27.935 seconds'. The status bar at the bottom indicates 'PL/SQL procedure successfully completed.'

You can verify that the data has loaded by running a simple query on your COUNTRIES table.

```
select country_region, count(country_name) from countries group by country_region;
```



The screenshot shows the Oracle SQL Developer interface with the same session context as the previous one. The 'Worksheet' tab is active, displaying the query:

```
select country_region, count(country_name) from countries group by country_region;
```

Below the worksheet, a 'Query Result' tab is open, showing the output of the query:

COUNTRY_REGION	COUNT(COUNTRY_NAME)
1 Asia	5
2 Middle East	1
3 Europe	10
4 Americas	4
5 Oceania	2
6 Africa	1

## Importing a Data Pump Export

### How the Data Pump Export was Prepared

To simulate migrating from an on-premise platform to the cloud, this export was generated on an Oracle SPARC SuperCluster in an on-premise data center, following the documentation <https://docs.oracle.com/en/cloud/paas/atp-cloud/atpug/load-data.html#GUID-30DB1EEA-DB45-49EA-9E97-DF49A9968E24>

The export command used was:

```
$ expdp system/MyDemoPa55word exclude=cluster,db_link parallel=8
schemas=soe dumpfile=exp_soe_150_%u.dmp directory=mnet
```

This created an export dump consisting of 8 files. The files were uploaded to an Object Storage bucket within the tenancy used for this lab.

The source schema was a Swingbench SOE schema, scaled at 150MB.

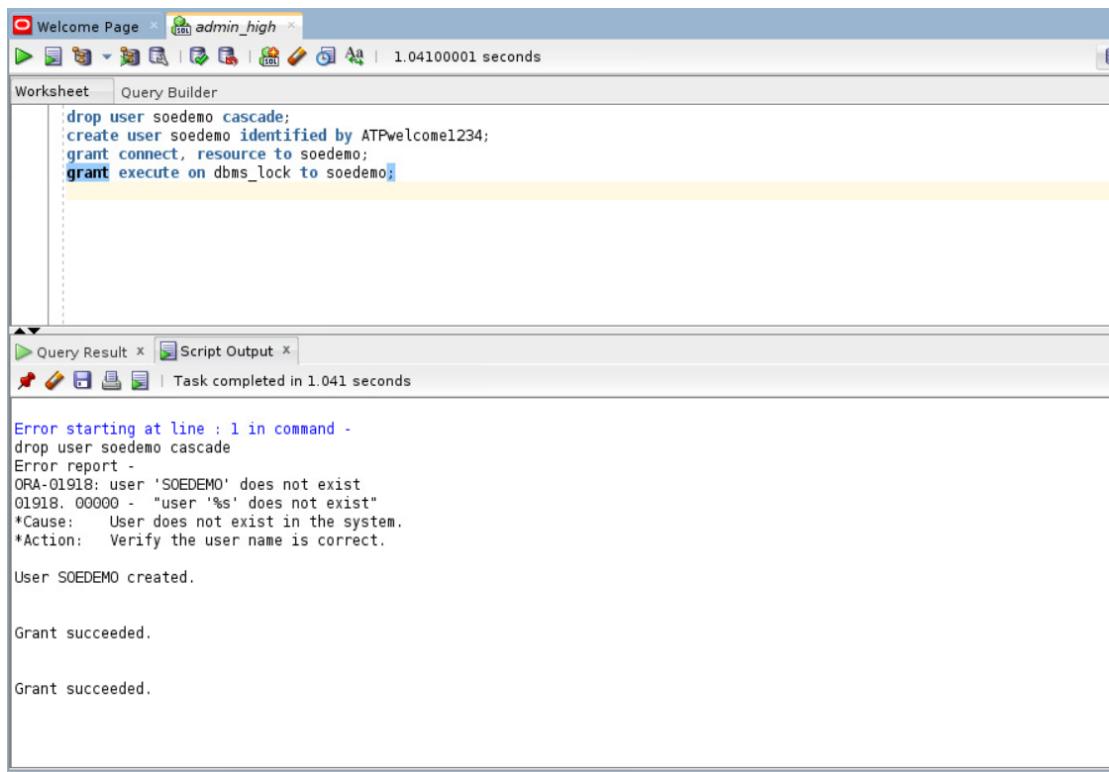
### Creating the Destination Schema

We are going to re-map the schema from the one in the dumpfiles to a new schema. Within the SQL worksheet of SQL Developer in your **admin\_high** connection, execute the following code to drop any existing schema with the same name, and create your new schema. This sql can be found in the file

**/home/oracle/labScripts/lab5/lab5\_create\_user.sql**

```
drop user soedemo cascade;
create user soedemo identified by ATPwelcome1234;
grant connect, resource to soedemo;
grant execute on dbms_lock to soedemo;
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



```
drop user soedemo cascade;
create user soedemo identified by ATPwelcome1234;
grant connect, resource to soedemo;
grant execute on dbms_lock to soedemo;
```

```
Error starting at line : 1 in command -
drop user soedemo cascade
Error report -
ORA-01918: user 'SOEDEMO' does not exist
01918. 00000 - "user '%s' does not exist"
*Cause: User does not exist in the system.
*Action: Verify the user name is correct.

User SOEDEMO created.

Grant succeeded.

Grant succeeded.
```

## Running Data Pump

Now go to a terminal window in your TigerVNC session connected to the LAB VM.

The location of the wallet file directory is not mandated by the cloud software. For this lab we will use the directory **/home/oracle/wallets** to store our wallet file. The wallet zip file should still be available on your system under your **\$HOME/Downloads** directory from the previous labs. If not, then you will have to download it again via the admin console.

```
cd $HOME
cp $HOME/Downloads/wallet_*.zip wallets
cd $HOME/wallets
unzip wallet_*.zip

[oracle@demo-long-v4 ]$ cd $HOME
[oracle@demo-long-v4 ]$ cp $HOME/Downloads/wallet_*.zip wallets
[oracle@demo-long-v4 ]$ cd $HOME/wallets
[oracle@demo-long-v4 ]$ unzip wallet_*.zip
Archive: wallet_MELATPTRAIN01.zip
  inflating: cwallet.sso
  inflating: tnsnames.ora
  inflating: truststore.jks
  inflating: ojdbc.properties
  inflating: sqlnet.ora
  inflating: ewallet.p12
  inflating: keystore.jks
[oracle@demo-long-v4 ]$
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Next you edit the sqlnet.ora to set the wallet location to your /home/oracle/wallets directory

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="/home/oracle/wallets")))
SSL_SERVER_DN_MATCH=yes
```

Verify that LD\_LIBRARY\_PATH includes the instantclient\_18\_3 directory

```
echo $LD_LIBRARY_PATH

[oracle@demo-long-v4 wallets]$ echo $LD_LIBRARY_PATH
/home/oracle/instantclient_18_3
```

If not set it using the command

```
export LD_LIBRARY_PATH=/home/oracle/instantclient_18_3
```

Verify the TNS\_ADMIN environment variable is set to /home/oracle/wallets

```
echo $TNS_ADMIN

[oracle@demo-long-v4 ~]echo $TNS_ADMIN
/home/oracle/wallets
[oracle@demo-long-v4 ~]
```

If not set it using the command

```
export TNS_ADMIN=/home/oracle/wallets

[oracle@demo-long-v4 wallets]$ export TNS_ADMIN=/home/oracle/wallets
```

Go to the Oracle Instant Client directory

```
cd $HOME/instantclient_18_3
```

This installation of Oracle Instant Client includes both the Basic package and the Tools package to allow us to run impdp.

The impdp command that for your import is based on the following information.  
The %U in the dump file expands to a 2-digit incrementing integer starting with 01.

**Connection string:** admin/<your admin password>@<your database>\_high

**Directory:** data\_pump\_dir

**Credential:** OBJ\_STORE\_CRED (the same credential we used to load a file from object\_storage)

**REMAP\_SCHEMA:** SOE:SOEDEMO

**Dumpfile:** [https://swiftobjectstorage.uk-london-1.oraclecloud.com/v1/oscemea001/DEMO\\_DATA/exp\\_atp\\_soe\\_150\\_%U.dmp](https://swiftobjectstorage.uk-london-1.oraclecloud.com/v1/oscemea001/DEMO_DATA/exp_atp_soe_150_%U.dmp)

**p**

**Parallel:** 8

To this lab simpler, there is a wrapper shell script which collects your password and connect string and calls the 'impdp' executable. In the UNIX terminal in your VM change directory to /home/oracle/labScripts/lab5. Then execute the script lab5\_import\_wrapper.sh, giving the admin password and connect string on the command line.

```
cd /home/oracle/labScripts/lab5  
./lab5_import_wrapper.sh <admin password> <connect string>
```

The script will list the generated impdp command. Review this and enter 'Y' if you are happy to proceed.

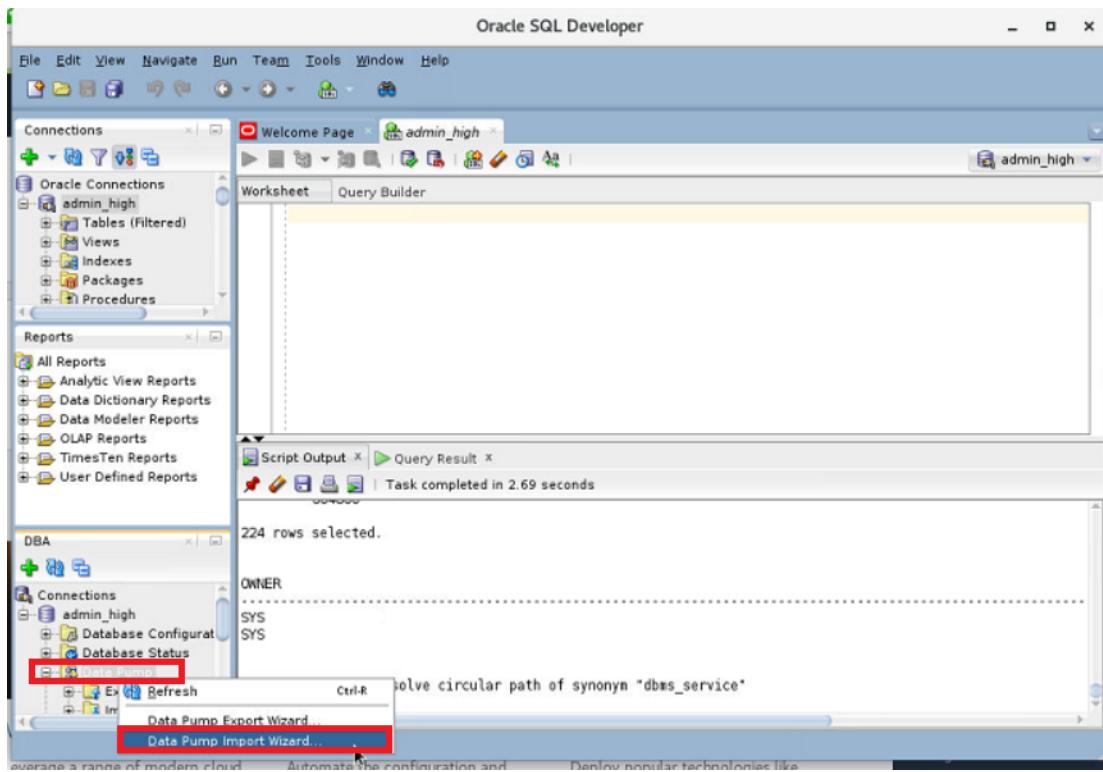
```
[oracle@demo-long-v4 ]$ cd /home/oracle/labScripts/lab5  
[oracle@demo-long-v4 ]$ ./lab5_import_wrapper.sh ATPwelcome-1234 MELATPTRAIN01_high  
This is the import command that has been generated  
impdp admin/ATPwelcome-1234@MELATPTRAIN01_high directory=data_pump_dir credential=OBJ_STORE_CRED REMAP_SCHEMA=SOE:SOEDEMO dumpfile=https://swiftobjectstorage.uk-london-1.oraclecloud.com/v1/oscemea001/DEMO_DATA/exp_atp_soe_150_%U.dmp PARALLEL=8  
Do you want to run it [Y/N]  
Y  
  
Import: Release 18.0.0.0.0 - Production on Fri Feb 1 13:11:39 2019  
Version 18.3.0.0.0  
  
Copyright (c) 1982, 2018, Oracle and/or its affiliates. All rights reserved.  
  
Connected to: Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production  
-
```

This import should take under 10 minutes to complete.

You will observe some errors during the import schema creation phase. One is because the schema already exists. The other is the schema originally had the 'manage scheduler' privilege, which cannot be granted by the admin user in your ATP instance. There may also be a compilation error on the Package 'ORDERENTRY'.

**Note:** You could also have carried out the import using the Oracle SQL Developer 'Data Pump Import Wizard'

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



You have successfully tested three methods of loading data into your ATP instance.

For a list of all the possible methods, see the product documentation

<https://docs.oracle.com/en/cloud/paas/atp-cloud/atpug/load-data.html>



# Lab 6 - DBA

## Exploration of ATP with SQL Developer

# Lab 6. DBA Exploration of ATP with SQL Developer

## Objectives:

- Explore the DBA view for Autonomous Transaction Processing (ATP) in SQL Developer

In this lab you will get an overview of the DBA view in SQL Developer and how to explore configuration and settings in your ATP environment. As part of the autonomous nature of ATP, some of these settings cannot be adjusted by the DBA.

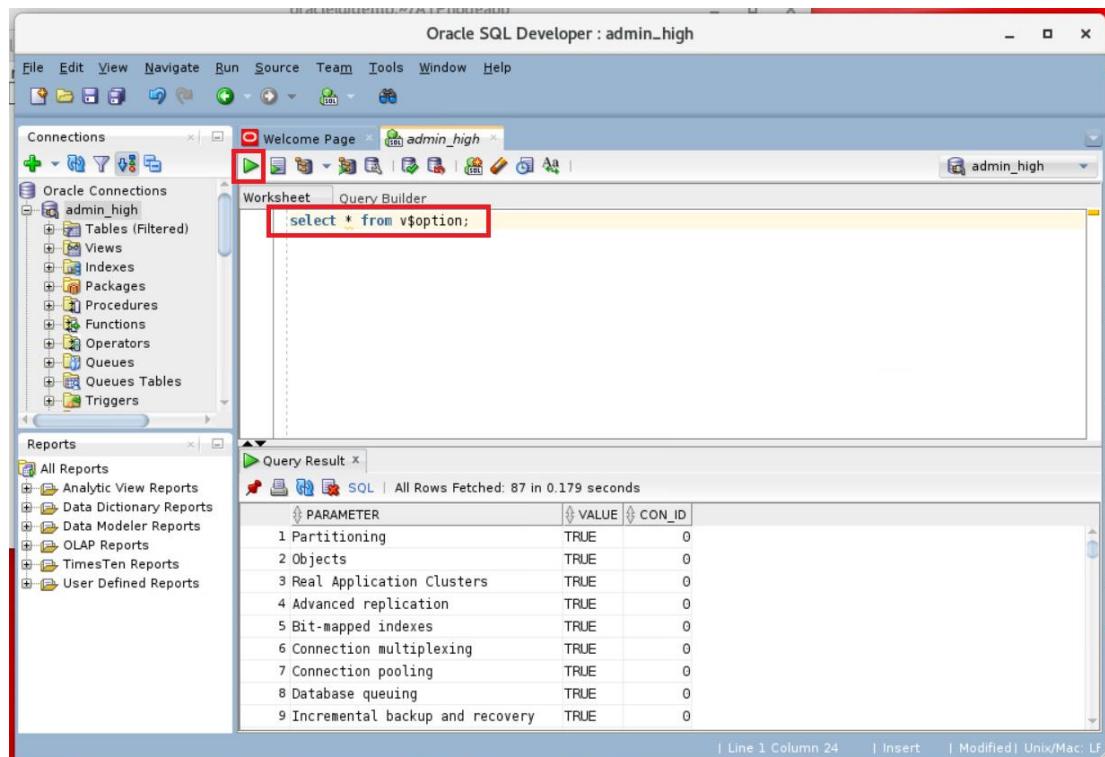
For this lab make sure you use your “ADMIN” user connection. Regular user accounts will not be able to view any system configuration information.

ATP includes many of the database options that are widely used in your databases and applications.

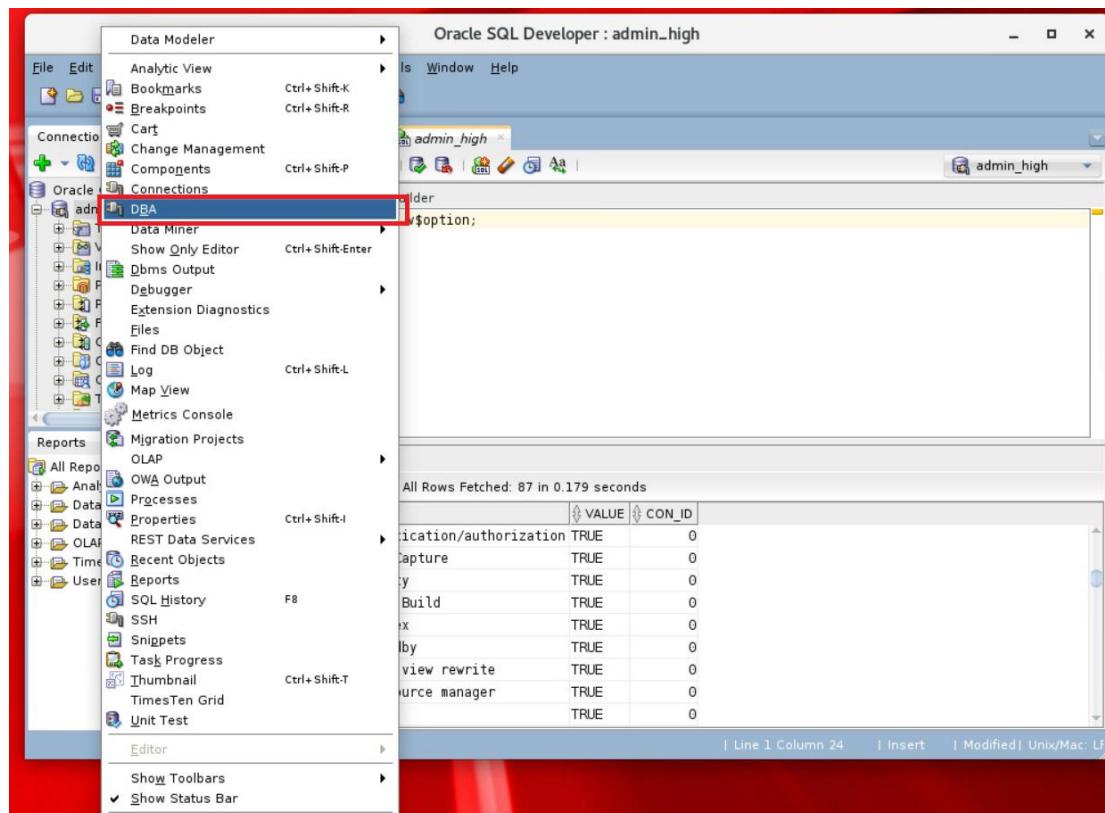
Start SQL Developer. Connect to your **admin\_high** connection.

In your SQL worksheet run the following command to get a list of options in your ATP instance.

```
select * from v$option;
```

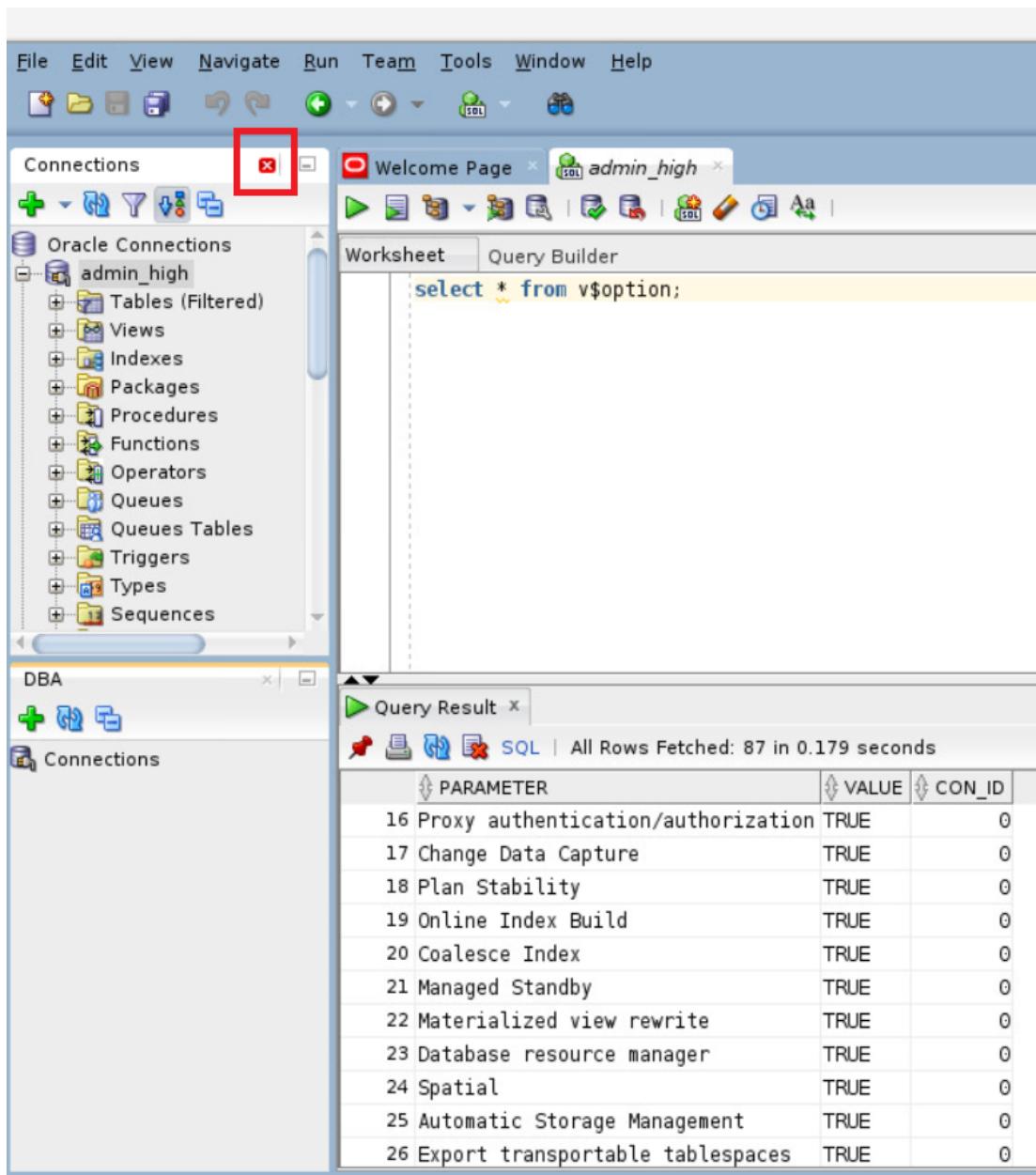


Next you will examine the DBA view in SQL Developer. In the main screen select **View->DBA**. If you cannot see the DBA option under View, go to “Window” and select “Reset Window to Factory Setting”, this should let you see the DBA option under View.



The DBA view appears in the bottom left pane. If you need more space to examine the DBA view you can close the Connections window or any other un-needed windows by clicking the ‘X’ on their menu bar.

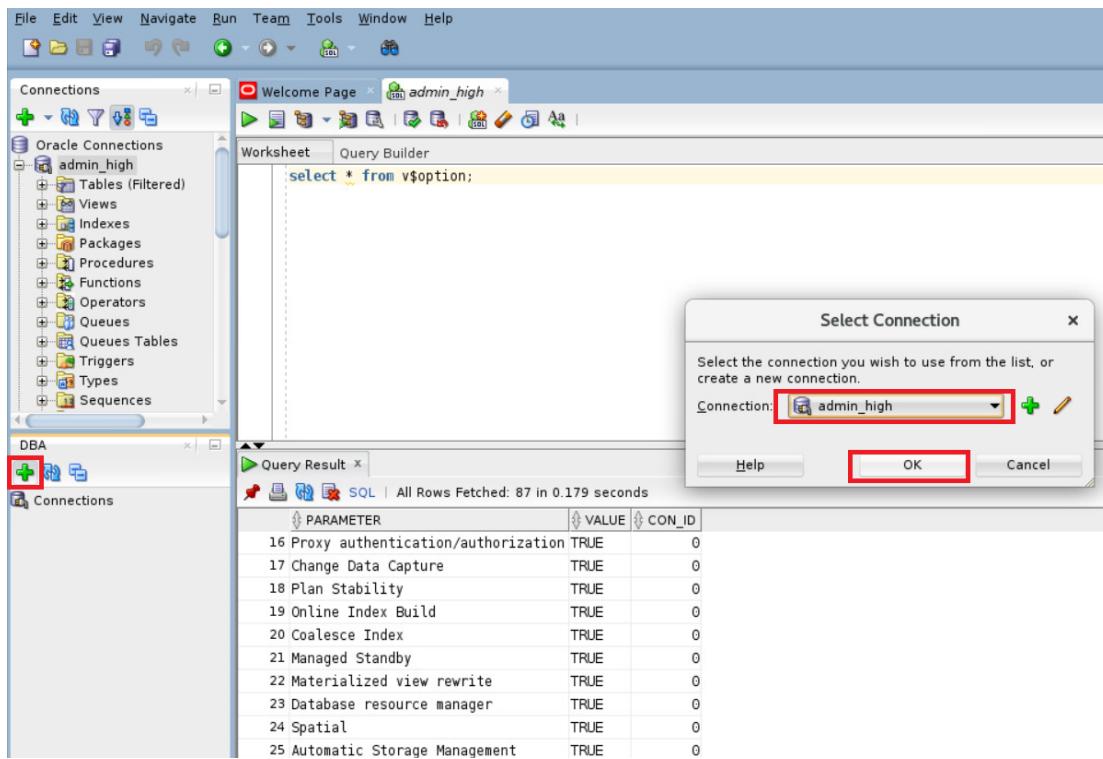
## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Click the connections green plus sign in the DBA view menu.

On the connection menu pop up, select your **admin\_high** connection and select **OK**.

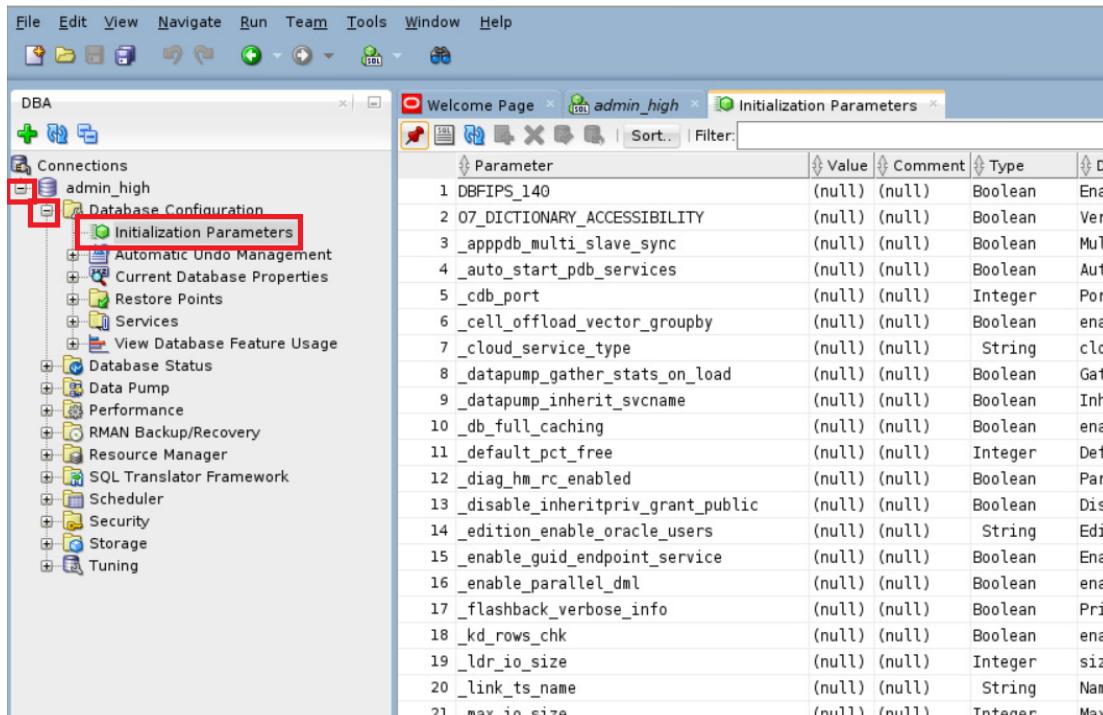
## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Expand your **admin\_high** connection by clicking on the ‘+’ symbol.

Expand the **Database Configuration** by clicking on the ‘+’ symbol.

View the Initialization parameters by clicking on **Initialization Parameters**.



Expand **Services** by clicking on the ‘+’

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

You can see the services that are configured for your ATP instance.

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : SERVICE ADMIN.BUMK28FBVPYBRGK\_MELATPTRAIN01\_high.atp.oraclecloud.com@admin\_high". The left sidebar under "DBA" shows a tree view with "Connections" expanded, showing "admin\_high" and its sub-nodes: "Database Configuration", "Initialization Parameters", "Automatic Undo Management", "Current Database Properties", "Restore Points", and "Services". The "Services" node is highlighted with a red box. Under "Services", there is a single entry: "BUMK28FBVPYBRGK\_MELATPTRAIN01\_high.atp.oraclecloud.com", which is also highlighted with a red box. To the right of the tree view is a "Welcome Page" tab with the URL "admin\_high" and the page content showing a table of service properties:

Name	Value
1 NAME	BUMK28FBVPYBRGK_MELATPTRAIN01_high.atp.oraclecloud.com
2 NETWORK_NAME	BUMK28FBVPYBRGK_MELATPTRAIN01_high.atp.oraclecloud.com
3 INSTANCES	2
4 SERVICE_ID	1
5 CREATION_DATE	26-NOV-18
6 GOAL	NONE
7 DTP	N
8 AQ_HA_NOTIFICATIONS	NO
9 CLB_GOAL	LONG
10 FAILOVER_METHOD	(null)
11 FAILOVER_TYPE	(null)
12 FAILOVER_RETRIES	(null)
13 FAILOVER_DELAY	(null)
14 MIN_CARD	(null)
15 MAX_CARD	(null)

In **View Database Features Usage** you can see the features that have been marked as in use. This view is not real-time and so you may not have any features marked as in use in your ATP instance.

The screenshot shows the Oracle SQL Developer interface with the title bar "Wed 17:02 Oracle SQL Developer". The left sidebar under "DBA" shows a tree view with "Connections" expanded, showing "admin\_high" and its sub-nodes: "Database Configuration", "Initialization Parameters", "Automatic Undo Management", "Current Database Properties", "Restore Points", and "View Database Feature Usage". The "View Database Feature Usage" node is highlighted with a red box. To the right of the tree view is a "Welcome Page" tab with the URL "admin\_high" and the page content showing a table titled "Usage | High Water Marks":

NAME	CURRENTLY_USED	DETECTED_USAGES	TOTAL_SAMPLES	FIRST_USAGE_DATE	LAST_USAGE_DATE
1 ACFS	TRUE	1	0 (null)	30-JAN-19	30-JAN-19
2 ACFS	FALSE	0	0 (null)	(null)	(null)
3 ACFS Encryption	FALSE	0	0 (null)	(null)	(null)
4 ACFS Encryption	FALSE	0	1 (null)	(null)	(null)
5 ACFS Snapshot	FALSE	0	0 (null)	(null)	(null)
6 ACFS Snapshot	FALSE	0	1 (null)	(null)	(null)
7 ADDM	FALSE	0	0 (null)	(null)	(null)
8 ADDM	FALSE	0	1 (null)	(null)	(null)
9 ASM Filter Driver	FALSE	0	1 (null)	(null)	(null)
10 ASM Filter Driver	FALSE	0	0 (null)	(null)	(null)
11 ASO native encryption and checksumming	FALSE	0	1 (null)	(null)	(null)
12 ASO native encryption and checksumming	FALSE	0	0 (null)	(null)	(null)
13 AWR Baseline	FALSE	0	1 (null)	(null)	(null)
14 AWR Baseline	FALSE	0	0 (null)	(null)	(null)
15 AWR Baseline Template	FALSE	0	0 (null)	(null)	(null)
16 AWR Baseline Template	FALSE	0	1 (null)	(null)	(null)
17 AWR Report	FALSE	0	1 (null)	(null)	(null)
18 AWR Report	FALSE	0	0 (null)	(null)	(null)
19 Active Data Guard - Real-Time Query on Physical Standby	FALSE	0	1 (null)	(null)	(null)
20 Active Data Guard - Real-Time Query on Physical Standby	FALSE	0	0 (null)	(null)	(null)
21 Adaptive Plans	TRUE	1	130-JAN-19	30-JAN-19	30-JAN-19
22 Adaptive Plans	FALSE	0	0 (null)	(null)	(null)
23 Advanced Index Compression	FALSE	0	1 (null)	(null)	(null)
24 Advanced Index Compression	FALSE	0	0 (null)	(null)	(null)
25 Advanced Replication	FALSE	0	0 (null)	(null)	(null)
26 Advanced Replication	FALSE	0	1 (null)	(null)	(null)
27 Application Continuity	FALSE	0	0 (null)	(null)	(null)
28 Application Continuity	FALSE	0	1 (null)	(null)	(null)
29 Application Express	FALSE	0	1 (null)	(null)	(null)
30 Application Express	FALSE	0	0 (null)	(null)	(null)

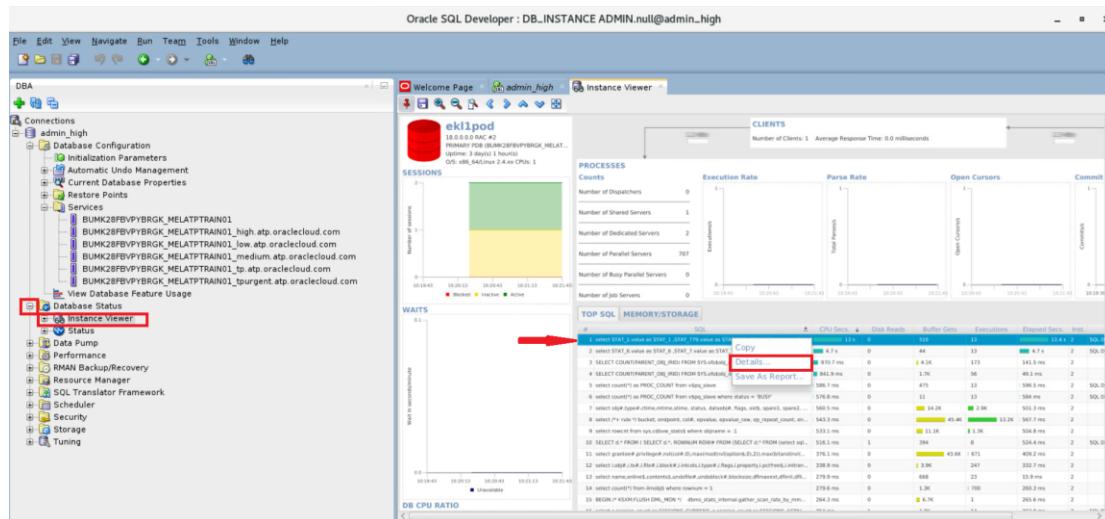
Expand **Database Status** by clicking on '+' and double click on **Instance Viewer**.

This displays a lot of information about the instance such as usage, configuration as well as the TOP SQL running on the instance.

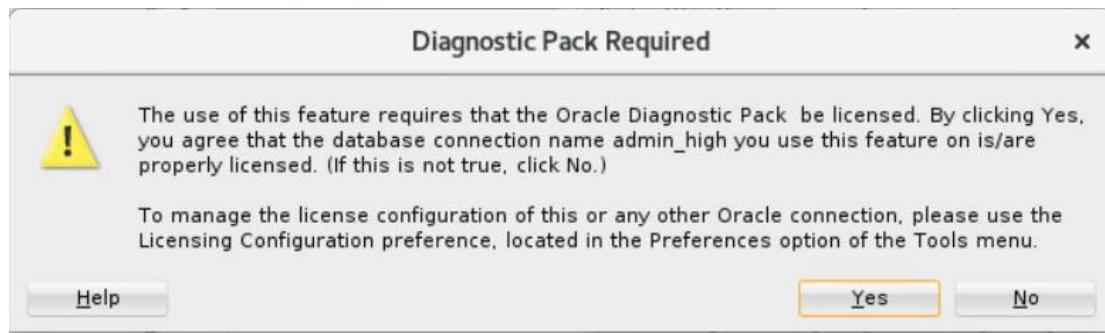
You can select a **SELECT STATEMENT** from **TOP SQL**, right click and select **Details**.

Oracle Autonomous Transaction Processing Hands-on Lab Guide

Since your instance was started it has not run many user SQL statements. This means that you may not have any statements that have interesting Explain Plan or parallelism.



You may receive a warning about Diagnostic Pack and/or Tuning Pack Click **Yes** on these popups to continue the lab.



This will create a new tab in the main window called SQL Detail. You can see examine the explain plan for your SQL here.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows the Oracle Database Control interface. On the left, the DBA navigation pane is open, showing various database management options like Connections, Database Configuration, and Services. The main window is titled "Welcome Page" and has tabs for "admin\_high", "Instance Viewer", and "SQL Details". A red arrow points to the "SQL Details" tab. Below it, the "Explain Plan" section is highlighted with a red border. It displays the execution plan for a query involving multiple SELECT statements and nested loops.

The **Resource Manager** menu contains information about different consumer groups and plans defined, and the current plan in effect. Explore the entries here, most of them will have contextual activities if you select and right click on them.

Verify which plan is active by selecting **Plans**.

This screenshot shows the Oracle Database Control interface focusing on the Resource Manager. In the left navigation pane, under "Resource Manager", the "Plans" node is selected and highlighted with a red box. The main pane displays a table of resource plans, with the "OLTP\_PLAN" row highlighted and labeled "ACTIVE".

This information is also available under the **Settings** link under **Resource Manager**.

This screenshot shows the Oracle Database Control interface focusing on the Resource Manager. In the left navigation pane, the "Settings" node under "Resource Manager" is selected and highlighted with a red box. The main pane displays the "Active Resource Plan" section, which lists "1 OLTP\_PLAN" and highlights it with a red box.

Select **Resource Manager -> Plans-> OLTP\_PLAN**

This will populate the main pane with an overview of the **OLTP\_PLAN**

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Name	Value
1 Plan	OLTP_PLAN
2 Status	ACTIVE
3 Is Subplan	NO
4 Management Method	EMPHASIS
5 Description	(null)

Select the **Resource Allocation** tab to see how the resources are shared between the groups.

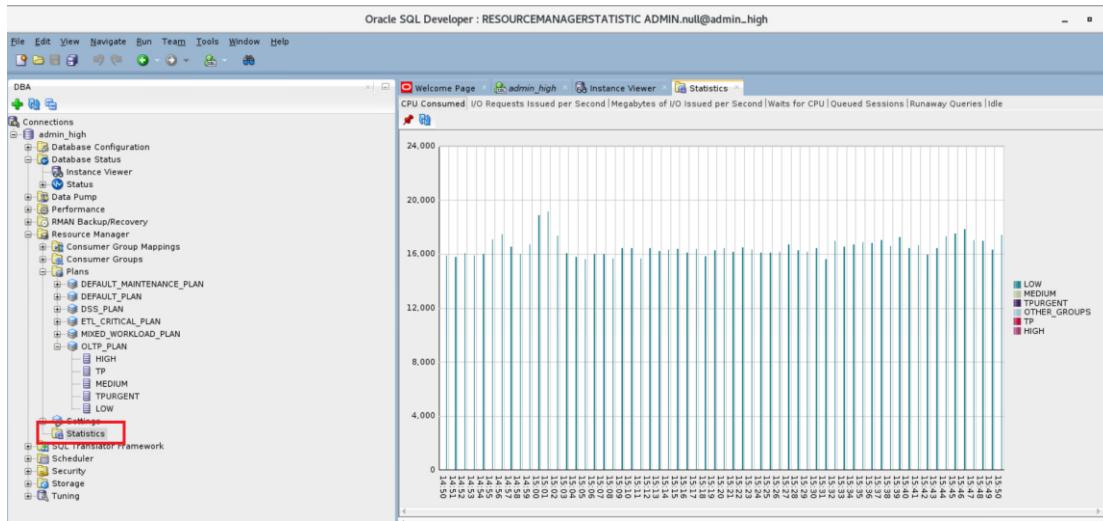
Group/Subplan	Shares
1 TPURGENT	12
2 HIGH	4
3 MEDIUM	2
4 LOW	1
5 TP	8

Select the **Directive Values** tab to see the Maximum degree of parallelism associated with each group.

Group	Max Degree of Parallelism
1 TPURGENT	1
2 HIGH	4
3 MEDIUM	1
4 LOW	1
5 TP	1

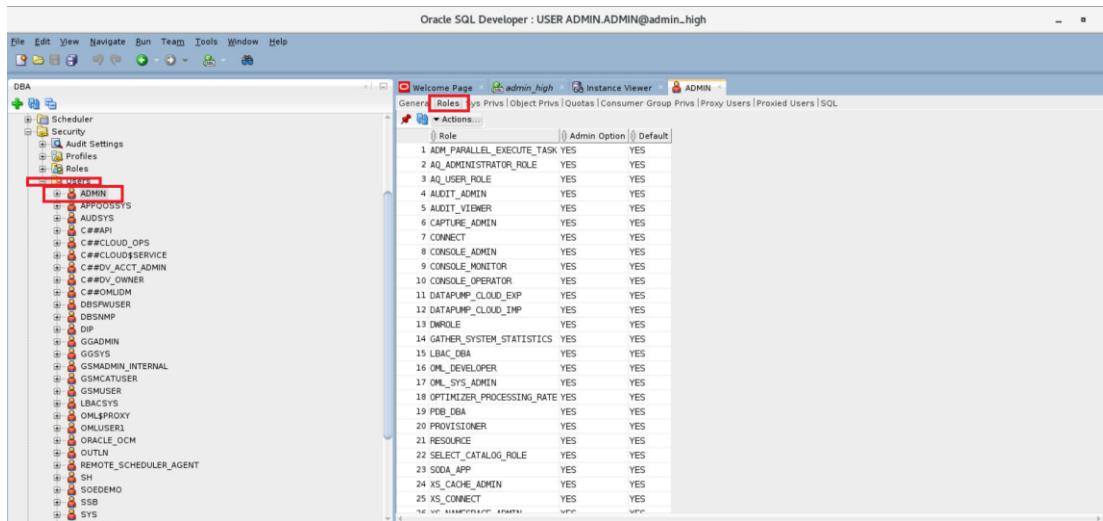
**Resource Manager->Statistics** provides graphical views into system usage by plan group. Double click on Statistics and select the tabs to explore the screens.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

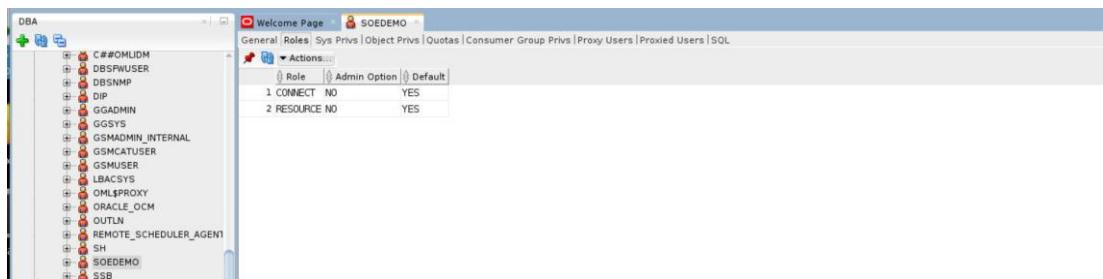


In the Security Section you can examine the users in the database and their roles and privileges.

Select **Security -> Users -> Admin** and list the roles associated with this user.



Compare this list with that of the SOEDEMO user if you created it in a previous lab.



You could also view the System privileges assigned to these users.

The **Storage** section can be used to display all the storage characteristics of the ATP database. Most of these characteristics cannot be changed, as the Autonomous Database automatically manages and optimizes storage for the service. You can still examine how the tablespace is configured.

### Select Storage -> Tablespaces -> Data

This view may take a long time to refresh. Any data in a database will be stored in the DATA Tablespace. Below is a screenshot of the parameters of the DATA Tablespace . Examine the different entries in this area.

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : TABLESPACE ADMIN.DATA@admin\_high". The left sidebar is titled "DBA" and contains a tree view of database components. The "Storage" node under "Connections" is expanded, and its "Tablespaces" node is also expanded, with the "DATA" tablespace selected. A red box highlights the "Actions..." button next to the tablespace name. The main pane displays a table titled "Actions..." with 26 rows of parameters and their values. The first few rows are:

Name	Value
1 TABLESPACE_NAME	DATA
2 BLOCK_SIZE	8192
3 INITIAL_EXTENT	65536
4 NEXT_EXTENT	(null)
5 MIN_EXTENTS	1
6 MAX_EXTENTS	2147483645
7 MAX_SIZE	2147483645
8 PCT_INCREASE	(null)
9 MIN_EXTLEN	65536
10 STATUS	ONLINE
11 CONTENTS	PERMANENT
12 LOGGING	LOGGING
13 FORCE_LOGGING	NO
14 EXTENT_MANAGEMENT	LOCAL
15 ALLOCATION_TYPE	SYSTEM
16 PLUGGED_IN	NO
17 SEGMENT_SPACE_MANAGEMENT	AUTO
18 DEF_TAB_COMPRESSION	DISABLED
19 RETENTION	NOT APPLY
20 BIGFILE	YES
21 PREDICATE_EVALUATION	STORAGE
22 ENCRYPTED	YES
23 COMPRESS_FOR	(null)
24 DEF_INMEMORY	DISABLED
25 DEF_INMEMORY_PRIORITY	(null)
26 DEF_INMEMORY_DISTRIBUTE	(null)

This concludes the SQL Developer Lab. SQL Developer is a very comprehensive powerful tool and you should continue to explore the different benefits it offers.



# Lab 7- Workload testing and analysis using Swingbench with ATP

# Lab 7: Workload testing and analysis using Swingbench with Autonomous Transaction Processing (optional)

## Objectives:

- Using Swingbench to demo and analyse workloads with Autonomous Transaction Processing (ATP)

You can optionally run some small tests to familiarize yourself with the tool and connecting it to ATP.

Swingbench is a suite of utilities developed by Dominic Giles that creates Oracle Database objects and data, creating workloads that can then be monitored or used for testing in OLTP and Datawarehouse environments. It is written in Java and so can be run on many platforms. It is a free tool and provided without licenses and support. Among the many uses for Swingbench is the ability to provide a comprehensive demo of the database and its associated tools and functionality. At the end of this lab you can find links to much more information on Swingbench and its use.

The sample shell scripts for this lab are available in your VM under the directory `/home/oracle/labScripts/lab7`.

## Preparing to use Swingbench

In your TigerVNC connection to your Lab VM, open a terminal window.

Verify that you have java 1.8 available in your environment. The precise build is not important.

```
java -version
```

```
$ java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

We have already downloaded and unzipped Swingbench to your Lab VM.

Normally the first step of using Swingbench is to create a schema for your benchmark using one of the provided benchmark generators, for example, `oewizard` for the Order Entry benchmark. As part of Lab 5, you imported a schema 'SOEDEMO' which was a benchmark schema generated using the Swingbench utility '`oewizard`', so we can skip this step.

## Verify your schema is usable

Verify your schema imported correctly by running the Swingbench utility script with the validate parameter. There is a wrapper script available

**/home/oracle/labScripts/lab7/lab7\_charbench\_validate.sh**. Edit this script, making sure to select your wallet file and correct connect string, changing the values in black :

```
#!/bin/bash
export PATH=$PATH:$HOME/swingbench/bin
sbutil -soe \
-cf $HOME/wallets/wallet_MELATPTRAIN01.zip \
-cs MELATPTRAIN01_medium \
-u soedemo \
-p ATPwelcome1234 \
-val
```

Execute the script.

```
cd /home/oracle/labScripts/lab7
./lab7_charbench_validate.sh
```

```
$ cd /home/oracle/labScripts/lab7
$ ./lab7_charbench_validate.sh
There appears to be an issue with the current Order Entry Schema. Please see below.
-----
|Object Type      |    Valid|   Invalid|   Missing|
-----
|Table           |    10|       0|       0|
|Index           |    26|       0|       0|
|Sequence         |     5|       0|       0|
|View             |     2|       0|       0|
|Code             |     0|       1|       0|
-----
List of missing or invalid objects.
Invalid Code : ORDERENTRY,
```

If there is an invalid package error then you should recompile it in SQL Developer

```
alter package soedemo.orderentry compile body;
alter package soedemo.orderentry compile;
```

To see how many rows were inserted on each table use the script

**/home/oracle/labScripts/lab7/lab7\_charbench\_tables.sh**. Edit this script, making sure to select your wallet file and correct connect string, changing the values in black :

```
#!/bin/bash
export PATH=$PATH:$HOME/swingbench/bin
sbutil -soe \
-cf $HOME/wallets/wallet_MELATPTRAIN01.zip \
-cs MELATPTRAIN01_medium \
-u soedemo \
-p ATPwelcome1234 \
```

**-tables**

And then run it.

```
cd /home/oracle/labScripts/lab7
./lab7_charbench_tables.sh
```

The results from the script should be similar to below. You may have to expand your terminal window to get the formatting to fit the width:

Table Name	Rows	Blocks	Size	Compressed?	Partitioned?
CUSTOMERS	150,000	2,747	256.0MB		Yes
LOGON	357,448	1,619	256.0MB		Yes
ORDER_ITEMS	644,281	5,184	256.0MB		Yes
ORDERS	214,469	3,165	256.0MB		Yes
ADDRESSES	225,000	2,795	256.0MB		Yes
CARD_DETAILS	225,000	1,880	256.0MB		Yes
INVENTORIES	894,824	2,153	17.0MB	Disabled	No
PRODUCT_DESCRIPTIONS	1,000	34	320KB	Disabled	No
PRODUCT_INFORMATION	1,000	28	256KB	Disabled	No
ORDERENTRY_METADATA	4	4	64KB	Disabled	No
WAREHOUSES	1,000	7	64KB	Disabled	No
Total Space				1.5GB	

If you will be conducting performance testing with Swingbench it is recommended that you collect statistics. Edit the script

**/home/oracle/labScripts/lab7/lab7\_charbench\_stats.sh** making sure to select your wallet file and correct connect string, changing the values in black :

```
#!/bin/bash
export PATH=$PATH:$HOME/swingbench/bin
sbutil -soe \
-cf $HOME/wallets/wallet_MELATPTRAIN01.zip \
-cs MELATPTRAIN01_medium \
-u soedemo \
-p ATPwelcome1234 \
-stats
```

And then run it.

```
./lab7_charbench_stats.sh
```

```
$ ./lab7_charbench_stats.sh
Collecting statistics for the schema
Collected statistics in : 0:08:07.933
```

## Running Swingbench

You are ready to run Swingbench workloads on ATP. Workloads are simulated by users submitting transactions to the database. This is configured by an xml file. You can fine tune the benchmark (e.g. transaction mix) or set standard parameters such as username and password by editing this file. You can also override some of these values using the command line.

## Run the workload against the \_medium service

We are overriding the following load related parameters

- rt specifies the total run time of the benchmark in hours and minutes.
- uc specifies the number of users who will connect to the system.
- r specifies the output file for the summary results.

**Please note:** charbench will place a high load on the database, so don't run for long periods of time. The command below will terminate the run automatically after three minutes. You can stop running charbench at any time with Ctrl C.

Generate load on your database by running the charbench utility. Do this by running the script `/home/oracle/labScripts/lab7/lab7_charbench_medium.sh`. Edit the script to contain your wallet and connect string information, highlighted below in black, before running it. Ensure that you are connecting to the **\_medium** service for your database.

```
#!/bin/bash
export PATH=$PATH:$HOME/swingbench/bin
charbench -c $HOME/swingbench/configs/SOE_Server_Side_V2.xml \
-cf $HOME/wallets/wallet_MELATPTRAIN01.zip \
-cs MELATPTRAIN01_medium \
-u soedemo \
-p ATPwelcome1234 \
-rt 0:03 \
-uc 32 \
-r /home/oracle/labScripts/lab7/medium32.xml
```

The benchmark will run for 3 minutes before completing.

```
$ ./lab7_charbench_medium.sh
Author : Dominic Giles
Version : 2.6.0.1082

Results will be written to /home/oracle/labScripts/lab7/medium32.xml.
Hit Return to Terminate Run...

Time          Users    TPM      TPS
2:20:38 PM     0       8483    44
Completed Run.
```

Examine the performance of your benchmark by reviewing the summary information at the start of the output xml file using the command

```
head -12 /home/oracle/labScripts/lab7/medium32.xml
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
(base) [oracle@demo-long bin]$ head -12 medium32.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Results xmlns="http://www.dominicgiles.com/swingbench">
  <Overview>
    <BenchmarkName>Order Entry (PLSQL) V2</BenchmarkName>
    <Comment>Version 2 of the SOE Benchmark running in the database using PL
/SQL</Comment>
    <TimeOfRun>Jan 14, 2019 5:08:47 PM</TimeOfRun>
    <TotalRunTime>0:03:00</TotalRunTime>
    <TotalLogonTime>0:00:07</TotalLogonTime>
    <TotalCompletedTransactions>1066</TotalCompletedTransactions>
    <TotalFailedTransactions>0</TotalFailedTransactions>
    <AverageTransactionsPerSecond>5.92</AverageTransactionsPerSecond>
    <MaximumTransactionRate>409</MaximumTransactionRate>
```

Note the following summary results from your file

TotalCompletedTransactions	
AverageTransactionsPerSecond	
MaximumTransactionRate	

### Run the workload against the \_high service

Now we will run the same test again using the '\_high' service, and specifying a different output file high32.xml

Do this by running the script

**/home/oracle/labScripts/lab7/lab7\_charbench\_high.sh**. Edit the script to contain your wallet and connect string information, highlighted below in black, before running it. Ensure that you are connecting to the **\_high** service for your database.

```
#!/bin/bash
export PATH=$PATH:$HOME/swingbench/bin
charbench -c $HOME/swingbench/configs/SOE_Server_Side_V2.xml \
-cf $HOME/wallets/wallet_MELATPTRAIN01.zip \
-cs MELATPTRAIN01_high \
-u soedemo \
-p ATPwelcome1234 \
-rt 0:03 \
-uc 32 \
-r /home/oracle/labScripts/lab7/high32.xml
```

```
$ ./lab7_charbench_high.sh
Author : Dominic Giles
Version : 2.6.0.1082

Results will be written to /home/oracle/labScripts/lab7/high32.xml.
Hit Return to Terminate Run...

Time          Users    TPM      TPS
5:21:06 PM      0     9080     145
Completed Run.
```

Review the summary information from the output high32.xml file using the command **head -12**

```
head -12 /home/oracle/labScripts/lab7/high32.xml
```

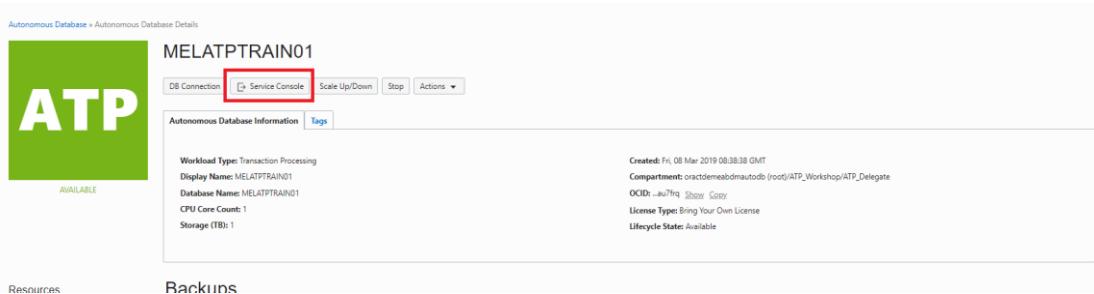
## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
(base) [oracle@demo-long bin]$ head -12 high32.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Results xmlns="http://www.dominicgiles.com/swingbench">
    <Overview>
        <BenchmarkName>Order Entry (PLSQL) V2</BenchmarkName>
        <Comment>Version 2 of the SOE Benchmark running in the database using PL
/SQL</Comment>
        <TimeOfRun>Jan 14, 2019 5:18:05 PM</TimeOfRun>
        <TotalRunTime>0:03:00</TotalRunTime>
        <TotalLogonTime>0:00:07</TotalLogonTime>
        <TotalCompletedTransactions>1003</TotalCompletedTransactions>
        <TotalFailedTransactions>0</TotalFailedTransactions>
        <AverageTransactionsPerSecond>5.57</AverageTransactionsPerSecond>
        <MaximumTransactionRate>395</MaximumTransactionRate>
```

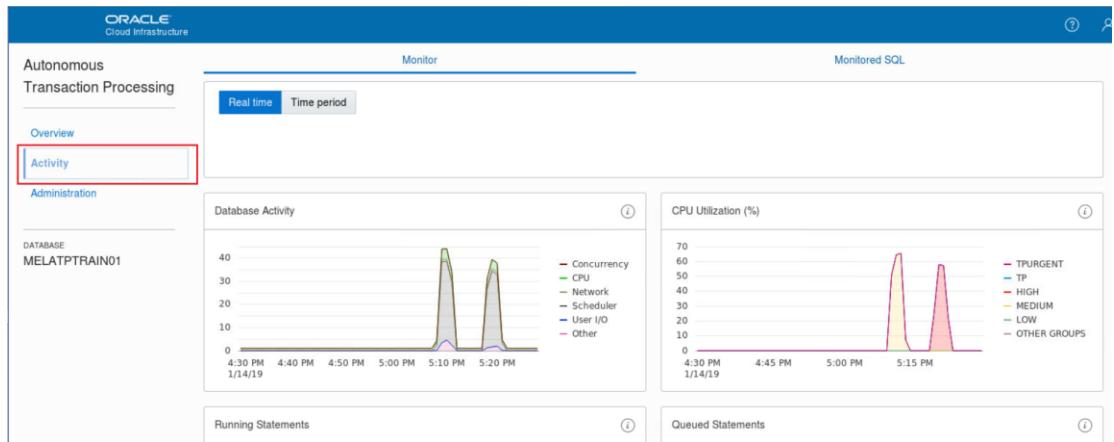
TotalCompletedTransactions	
AverageTransactionsPerSecond	
MaximumTransactionRate	

It is not expected that your benchmarks will have identical results, as there will be slight variances due to the short run times we are using for this demonstration. Unless there were transactions from resource consumer groups that have higher resource shares running at the same time, both the benchmark runs using \_high and the \_medium services will have had access to all the resources in the instance, and are both services that can support high concurrency. Unless your instance was under CPU pressure they should have similar results.

You can review the load on your system during the benchmark using the Service Console.



## Oracle Autonomous Transaction Processing Hands-on Lab Guide



## Want to learn more about Swingbench?

Visit <http://www.dominicgiles.com/swingbench.html> to see more usage instructions. See his blog at <http://www.dominicgiles.com/blog/blog.html> for more tips, tricks and innovations.

# Section 3- Focus on Developers



# Lab 8 - Using Node.js with ATP



# Lab 8: Using Node.js with Autonomous Transaction Processing

## Objectives:

- Learn how to build Node.js applications and connect them to an Oracle Autonomous Transaction Processing (ATP) database service

The following lab will use the following pre-installed software and libraries on your VM:

- Python 3.7
- Node.js
- Git
- libaio
- node
- npm

## Creating your first Node.js application

Open a terminal and navigate to the `/home/oracle/labScripts/lab8` directory. Run the following commands to show the versions of node and npm installed:

```
node -v  
npm -v
```

---

```
[oracle@LABVM61 ~]$ cd /home/oracle/labScripts/lab8  
[oracle@LABVM61 lab8]$ node -v  
v6.14.3  
[oracle@LABVM61 lab8]$ npm -v  
3.10.10  
[oracle@LABVM61 lab8]$ █
```

In the lab8 directory you will find all the node scripts that we are going to use for this lab. In a terminal window use cat to display the first script we are going to run, which is test.js

```
cat test.js
```

The script contents are shown below

```
const http = require('http');  
  
const hostname = '127.0.0.1';  
const port = 3000;
```

```
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Now run your first Node.js application. This application will simply print a message on a web browser.

In your terminal window from the **/home/oracle/labScripts/lab8** directory run:

```
node test.js
```

```
[oracle@LABVM61 lab8]$ node test.js
Server running at http://127.0.0.1:3000/
```

**Open up a new browser or browser tab and in the address bar enter:  
<http://127.0.0.1:3000/>**

You will see your first Node.js application running.



To terminate your Node.js program, **hit Ctrl-C** in your terminal window where you were running 'node test.js'

## Configuring Node.js for use with ATP

For the next step of the lab we need to make use of several node.js libraries that will be used to run the remaining lab examples. The libraries have already been installed on the VM using the npm utility. They are listed below for your information only:

- oracledb (Oracle database libraries)
- async (Libraries for working with asynchronous JavaScript)
- app (Libraries for web application development)
- express (Libraries for mobile applications)

We need to ensure the wallet file for your ATP instance is available and unzipped in the **/home/oracle/wallets** directory. It should still be available on your system under your \$HOME/Downloads directory from the previous labs. If not, then you will have to download it again via the admin console.

```

cd $HOME
cp $HOME/Downloads/wallet_*.zip wallets
cd $HOME/wallets
unzip wallet_*.zip

[oracle@demo-long wallets]$ pwd
/home/oracle/wallets
[oracle@demo-long wallets]$ ls
wallet_MELATPTRAIN01.zip
[oracle@demo-long wallets]$ unzip wallet_MELATPTRAIN01.zip
Archive: wallet_MELATPTRAIN01.zip
  inflating: cwallet.sso
  inflating: tnsnames.ora
  inflating: truststore.jks
  inflating: ojdbc.properties
  inflating: sqlnet.ora
  inflating: ewallet.p12
  inflating: keystore.jks
[oracle@demo-long wallets]$ ls -l
total 60
-rw-r--r-- 1 oracle dba 6661 Jan  9 17:10 cwallet.sso
-rw-r--r-- 1 oracle dba 6616 Jan  9 17:10 ewallet.p12
-rw-r--r-- 1 oracle dba 3240 Jan  9 17:10 keystore.jks
-rw-r--r-- 1 oracle dba 87 Jan  9 17:10 ojdbc.properties
-rw-r--r-- 1 oracle dba 114 Jan  9 17:10 sqlnet.ora
-rw-r--r-- 1 oracle dba 6359 Jan  9 17:10 tnsnames.ora
-rw-r--r-- 1 oracle dba 3335 Jan  9 17:10 truststore.jks
-rw-r--r-- 1 oracle dba 19916 Jan  9 17:10 wallet_MELATPTRAIN01.zip
[oracle@demo-long wallets]$ 

```

Several files need to be updated to reflect correct paths and names.

Edit the **ojdbc.properties** (open with vi or your favourite editor) to reflect the location of the wallet file. The line entry in the file should be as below, where **/home/oracle/wallets** should be the directory where your wallet files are located:

```
oracle.net.wallet_location=(SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY="/home/oracle/wallets")))
```

Edit the **sqlnet.ora** file to reflect the location of the wallet file. The line entry in the file should be as below, the directory where your wallet files are located:

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="/home/oracle/wallets")))
SSL_SERVER_DN_MATCH=yes
```

We need to ensure that the **TNS\_ADMIN** environment variable points to the wallet directory. Use the echo command to check the current value:

```
echo $TNS_ADMIN

[oracle@LABVM61 ~]$ echo $TNS_ADMIN
/home/oracle/wallets
[oracle@LABVM61 ~]$ 
```

In our case the variable is already set to point to **/home/oracle/wallets**. If you wanted to change the value of **TNS\_ADMIN** it can be done using the below command:

```
TNS_ADMIN=/home/oracle/wallets; export TNS_ADMIN
```

```
[oracle@demo-long nodejs]$ TNS_ADMIN=/home/oracle/wallets; export TNS_ADMIN
[oracle@demo-long nodejs]$ echo $TNS_ADMIN
/home/oracle/wallets
[oracle@demo-long nodejs]$ █
```

In Node.js scripts you can specify database credentials directly within the script, but it is easier to create one file that contains the credentials and then call this file in your Node.js scripts. That way you can independently change the credentials without having to update all your programs.

Look at the file in the **/home/oracle/labScripts/lab8** directory called **dbconfig.js**. The file is also listed below in your guide. This is a common file containing database credentials. It will be called from all Node.js scripts we run. Using vi (or other editor) edit the **dbconfig.js** file, replace the red text with your specific credential information, do not change anything else:

```
module.exports= {

  dbuser: 'admin',
  dbpassword: 'ATPwelcome-1234',
  connectString: 'melatptrain01_tp'
}
```

**dbuser** = admin (or another user account if you created another user)

**dbpassword** = admin's (or other users) password

**connectString** = your database \_tp service name. This must match an entry in the tnsnames.ora from your wallet.

## Connect to an ATP Database with Node.js

The next Node.js script we will use is **connectatp.js**. It will connect to your database and, if successful, display the username used to connect in a browser (address <http://localhost:4001/>). From the lab8 directory view the script in your terminal, using the cat command:

```
cat connectatp.js
```

The script is also listed below. The first few lines call Node libraries including the oracledb library. The third line calls the **dbconfig.js** that contains your database credentials. It is important to make sure the dbconfig.js file is in the same directory or the app will not find the database credentials.

```
var http = require('http');
var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');
let error;
let user;

oracledb.getConnection({
  user: dbConfig.dbuser,
  password: dbConfig.dbpassword,
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
        connectString: dbConfig.connectString
    },
    function(err, connection) {
        if (err) {
            error = err;
            return;
        }
        connection.execute('select user from dual', [], function(err,
result) {
            if (err) {
                error = err;
                return;
            }
            user = result.rows[0][0];
            console.log(`Check to see if your database connection
worked at http://localhost:4001/`);
            error = null;
            connection.close(function(err) {
                if (err) {
                    console.log(err);
                }
            });
        })
    }
);

http.createServer(function(request, response) {
    response.writeHead(200, {
        'Content-Type': 'text/plain'
    });
    if (error === null) {
        response.end('Connection test succeeded. You connected to ATP
as ' + user + '!');
    } else if (error instanceof Error) {
        response.write('Connection test failed. Check the settings
and redeploy app!\n');
        response.end(error.message);
    } else {
        response.end('Connection test pending. Refresh after a few
seconds...');

    }
}).listen(4001);
```

To execute the code, in a terminal window, go to the `/home/oracle/labScripts/lab8` directory and run the following command `node connectatp.js`. The script will wait on the command line and the application will listen on port 3050 for a http connection. Follow the instruction to check the connection status using Firefox while the script is running.

```
node connectatp.js
```

```
[oracle@demo-long-v4 ]$ cd /home/oracle/labScripts/lab8
[oracle@demo-long-v4 ]$ node connectatp.js
Check to see if your database connection worked at http://localhost:4001/
```

Go to your browser and the address 127.0.0.1:4001. You should see:



Exit your application by returning to your terminal window and entering <CTRL><C>.

## Using Node.js script to select from an ATP database

In the Node.js script **atpselect.js** you will run the same query we ran in Lab 2 on the SH schema, to test SQL through Node.js. The code below is the contents of **atpselect.js**. Notice again the library calls at the beginning to load the oracle library and the file with your connection information.

```

var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');

oracledb.getConnection({
    user: dbConfig.dbuser,
    password: dbConfig.dbpassword,
    connectString: dbConfig.connectString
},
function(err, connection) {
    if (err) {
        console.error(err.message);
        return;
    }
    connection.execute(
        // The statement to execute
        `Select count(*), co.country_name, co.country_region
         From sh.countries co, sh.customers cu
          where co.country_id=cu.country_id group by
co.country_region, co.country_name
          order by co.country_region`,
        // The callback function handles the SQL
execution results
        function(err, result) {
            if (err) {
                console.error(err.message);
                doRelease(connection);
                return;
            }
            console.log(result.rows);
            doRelease(connection);
        });
});

// Note: connections should always be released when not needed
function doRelease(connection) {
    connection.close(
        function(err) {
            if (err) {
                console.error(err.message);
            }
        });
}

```

To run the code, in a terminal, go to the **/home/oracle/labScripts/lab8** and view the code. Once you have view the code, run the following command to execute it:

```
node atpselect.js
```

You will see the following on your terminal:

```
[oracle@LABVM61 lab8]$ cd /home/oracle/labScripts/lab8
[oracle@LABVM61 lab8]$ node atpselect.js
[ [ 88, 'South Africa', 'Africa' ],
  [ 403, 'Argentina', 'Americas' ],
  [ 832, 'Brazil', 'Americas' ],
  [ 2010, 'Canada', 'Americas' ],
  [ 18520, 'United States of America', 'Americas' ],
  [ 712, 'China', 'Asia' ],
  [ 624, 'Japan', 'Asia' ],
  [ 597, 'Singapore', 'Asia' ],
  [ 383, 'Denmark', 'Europe' ],
  [ 3833, 'France', 'Europe' ],
  [ 8173, 'Germany', 'Europe' ],
  [ 7780, 'Italy', 'Europe' ],
  [ 708, 'Poland', 'Europe' ],
  [ 2039, 'Spain', 'Europe' ],
  [ 91, 'Turkey', 'Europe' ],
  [ 7557, 'United Kingdom', 'Europe' ],
  [ 75, 'Saudi Arabia', 'Middle East' ],
  [ 831, 'Australia', 'Oceania' ],
  [ 244, 'New Zealand', 'Oceania' ] ]
```

## Using Node.js script to select from an ATP database with a bind variable

Bind variables are commonly used in Node.js scripts and supported within the Oracle database. The following example shows how to bind a variable within Node.js code. The **atpselectwithbindvariable.js** script is listed below.

Note the section on binding the variable and also that we explicitly release connections at the end of the script, so resources are not wasted.

```
var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');

oracledb.getConnection({
  user: dbConfig.dbuser,
  password: dbConfig.dbpassword,
  connectString: dbConfig.connectString
},
function(err, connection) {
  if (err) {
    console.error(err.message);
    return;
  }
  connection.execute(
    // The statement to execute
    `SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME
      FROM sh.customers WHERE CUST_ID = :id`,
    { outFormat: oracledb.OBJECT } // query result
  )
  .then(function(result) {
    console.log(result.rows);
  })
  .catch(function(error) {
    console.error(error);
  })
  .finally(function() {
    connection.release();
  });
});
```

```

        },
        // The callback function handles the SQL
execution results
    function(err, result) {
        if (err) {
            console.error(err.message);
            doRelease(connection);
            return;
        }
        console.log(`We are specifically looking for customer ID
5992`);
        console.log(result.rows);
        doRelease(connection);
    });
});

// Note: connections should always be released when not needed
function doRelease(connection) {
    connection.close(
        function(err) {
            if (err) {
                console.error(err.message);
            }
        });
}

```

To run the code, in the terminal, go to the **/home/oracle/labScripts/lab8** directory and run the below script:

```
node atpselectwithbindvariable.js
```

Check the results on your terminal.

---

```
[oracle@LABVM61 lab8]$ cd /home/oracle/labScripts/lab8
[oracle@LABVM61 lab8]$ node atpselectwithbindvariable.js
We are specifically looking for customer ID 5992
[ { CUST_ID: 5992,
    CUST_FIRST_NAME: 'Brett',
    CUST_LAST_NAME: 'Hackett' } ]
[oracle@LABVM61 lab8]$
```

## Using JSON data in an ATP database

You can create tables in the Oracle database that have JSON columns. In this example you will drop and create the table **j\_purchaseorder**. You use the SQL condition **is\_json** as a check constraint to ensure that data inserted into a column is (well-formed) JSON data. Oracle recommends that you always use an **is\_json** check constraint when you create a column intended for JSON data. In this example you will use the constraint **ensure\_json** and the **is\_json** check. The script **atpcreatejsontable.js**, which is listed below, creates the table and constraints. Examine the syntax around table manipulation and JSON data.

```
var async = require('async');
var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');
```

```

var doconnect = function(cb) {
    oracledb.getConnection({
        user: dbConfig.dbuser,
        password: dbConfig.dbpassword,
        connectString: dbConfig.connectString
    },
    cb);
};

var dorelease = function(conn) {
    conn.close(function (err) {
        if (err)
            console.error(err.message);
    });
};

var dodrop = function (conn, cb) {
    conn.execute(
        `BEGIN
            EXECUTE IMMEDIATE 'DROP TABLE j_purchaseorder';
            EXCEPTION WHEN OTHERS THEN
                IF SQLCODE <> -942 THEN
                    RAISE;
                END IF;
            END;`,
        function(err) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log("Table dropped");
                return cb(null, conn);
            }
        });
};

var docreate = function (conn, cb) {
    conn.execute(
        "CREATE TABLE j_purchaseorder (po_document VARCHAR2(4000)
CONSTRAINT ensure_json CHECK (po_document IS JSON))",
        function(err) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log("Table created");
                return cb(null, conn);
            }
        });
};

async.waterfall(
    [
        doconnect,
        dodrop,
        docreate
    ],
    function (err, conn) {
        if (err) { console.error("In waterfall error cb: ==>", err,
"<=="); }
        if (conn)
            dorelease(conn);
    }
);

```

```
});
```

To run the code, from your terminal window, go to the directory **/home/oracle/labScripts/lab8** and run the following command:

```
node atpcREATEjsontable.js
```

The output will simply be two lines in the terminal, indicating the table was dropped and then created.

```
[oracle@vm61 lab8]$ cd /home/oracle/labScripts/lab8
[oracle@vm61 lab8]$ node atpcREATEjsontable.js
Table dropped
Table created
[oracle@vm61 lab8]$
```

In the following script you will extend the previous app and manipulate JSON data into Oracle objects for updating, analysis, inserts and deletes from the database.

Examine the following code from the file **atpSELECTjson.js**

```
var async = require('async');
var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');

var doconnect = function(cb) {
    oracledb.getConnection({
        user: dbConfig.dbuser,
        password: dbConfig.dbpassword,
        connectString: dbConfig.connectString
    },
    cb);
};

var dorelease = function(conn) {
    conn.close(function (err) {
        if (err)
            console.error(err.message);
    });
};

var dodrop = function (conn, cb) {
    conn.execute(
        `BEGIN
            EXECUTE IMMEDIATE 'DROP TABLE j_purchaseorder';
            EXCEPTION WHEN OTHERS THEN
                IF SQLCODE <> -942 THEN
                    RAISE;
                END IF;
            END;`,
        function(err) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log("Table dropped");
                return cb(null, conn);
            }
        }
    );
};
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
        }
    });

var docreate = function (conn, cb) {
    conn.execute(
        "CREATE TABLE j_purchaseorder (po_document VARCHAR2(4000)
CONSTRAINT ensure_json CHECK (po_document IS JSON))",
        function(err) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log("Table created");
                return cb(null, conn);
            }
        });
};

var checkver = function (conn, cb) {
    if (conn.oracleServerVersion < 1201000200) {
        return cb(new Error('This example only works with Oracle Database
12.1.0.2 or greater'), conn);
    } else {
        return cb(null, conn);
    }
};

var doinsert = function (conn, cb) {
    var data = { "userId": 1, "userName": "Chris", "location":
"Australia" };
    var s = JSON.stringify(data);
    conn.execute(
        "INSERT INTO j_purchaseorder (po_document) VALUES (:bv)",
        [s], // bind the JSON string for inserting into the JSON column.
        { autoCommit: true },
        function (err) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log("Data inserted successfully.");
                return cb(null, conn);
            }
        });
};

var dojsonquery = function (conn, cb) {
    console.log('1. Selecting JSON stored in a VARCHAR2 column');
    conn.execute(
        "SELECT po_document FROM j_purchaseorder WHERE JSON_EXISTS
(po_document, '$.location')",
        function(err, result) {
            if (err) {
                return cb(err, conn);
            } else {
                var js = JSON.parse(result.rows[0][0]); // just show first
record
                console.log('Query results: ', js);
                return cb(null, conn);
            }
        });
};
```

```

var dorelationalquerydot = function (conn, cb) {
    console.log('2. Using dot-notation to extract a value from a JSON
column');
    conn.execute(
        "SELECT po.po_document.location FROM j_purchaseorder po",
        function(err, result) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log('Query results: ', result.rows[0][0]); // just
show first record
                return cb(null, conn);
            }
        });
};

var dorelationalquery = function (conn, cb) {
    console.log('3. Using JSON_VALUE to extract a value from a JSON
column');
    conn.execute(
        "SELECT JSON_VALUE(po_document, '$.location') FROM
j_purchaseorder",
        function(err, result) {
            if (err) {
                return cb(err, conn);
            } else {
                console.log('Query results: ', result.rows[0][0]); // just
show first record
                return cb(null, conn);
            }
        });
};

async.waterfall(
    [
        doconnect,
        checkver,
        dodrop,
        docreate,
        doinsert,
        dojsonquery,
        dorelationalquerydot,
        dorelationalquery
    ],
    function (err, conn) {
        if (err) { console.error("In waterfall error cb: ==>", err,
"<=="); }
        if (conn)
            dorelease(conn);
    });
}

```

To run the code, from your terminal, go to **/home/oracle/labScripts/lab8** and run the code:

```
node atpselectjson.js
```

The terminal will show the output of the JSON manipulation functions that were defined in the code. Please review the code to see more detail on how the functions were used.

---

```
[oracle@vm61 lab8]$ cd /home/oracle/labScripts/lab8
[oracle@vm61 lab8]$ node atpselectjson.js
Table dropped
Table created
Data inserted successfully.
1. Selecting JSON stored in a VARCHAR2 column
Query results: { userId: 1, userName: 'Chris', location: 'Australia' }
2. Using dot-notation to extract a value from a JSON column
Query results: Australia
3. Using JSON_VALUE to extract a value from a JSON column
Query results: Australia
[oracle@vm61 lab8]$ █
```

This concludes the Node.js Lab. You have successfully worked with Node.js to create several simple applications demonstrating connectivity between Node.js and the ATP database.



# Lab 9 - Using Python with ATP

# Lab 9: Using Python with Autonomous Transaction Processing

## Objectives:

- Run basic Python code that interacts with an Oracle Autonomous Transaction Processing (ATP) Database

The objective of this lab is to run database operations against an Oracle ATP Database using Python. This lab will not teach you how to code in Python.

You will need to be comfortable editing files in a Linux environment.

The sample Python scripts which are used in this lab are available in your VM under the directory `/home/oracle/labScripts/lab9`.

## Verify Your Lab Environment

The Lab VM that you are using has the following pre-requisite software installed for you

- Anaconda
- Python 3.7.1
- Oracle Instant Client

Connect to your Lab VM using TigerVNC.

Verify that Python is available on your PATH and is in the `anaconda3` directory structure.

```
which python
```

```
$ which python  
~/anaconda3/bin/python
```

Verify the version of Python is 3.7.1

```
python --version
```

```
$ python -version  
Python 3.7.1
```

Verify the version of pip is 18.1

```
pip --version
```

```
$ pip -version  
pip 18.1 from /home/oracle/anaconda3/lib/python3.7/site-packages/pip (python 3.7)
```

We need to ensure the wallet file for your ATP instance is available and unzipped in the /home/oracle/wallets directory. It should still be available on your system under your \$HOME/Downloads directory from the previous labs. If not, then you will have to download it again via the admin console.

```
cd $HOME  
cp $HOME/Downloads/wallet_*.zip wallets  
cd $HOME/wallets  
unzip wallet_*.zip
```

Next you edit the sqlnet.ora to set the wallet file location

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =  
(DIRECTORY="/home/oracle/wallets")))  
SSL_SERVER_DN_MATCH=yes
```

## Configuring Python to run with ATP and sample scripts

For Python to work with ATP, the Python Oracle libraries must be installed. Verify that they are available using

```
pip list |grep -i oracle
```

```
$ pip list |grep -i oracle  
cx-Oracle 7.0.0
```

Set TNS\_ADMIN to the location of your wallet

```
export TNS_ADMIN=$HOME/wallets
```

Open a terminal and move to the python lab, where the scripts are staged.

```
cd $HOME/labScripts/lab9
```

### Executing SELECT statements

Execute a simple select (same select as on previous labs) through a Python script.

The python script **select.py** is the first script we are going to run. The script is listed in its entirety below. Using vi (or other editor) edit the script and change the connection credentials in the **con** line of the script (highlighted in black below) to reflect your account/password@connectstring information.

```
import cx_Oracle  
  
## Edit the below line to contain your password and connect string  
con = cx_Oracle.connect('admin/ATPwelcome-1234@MELATPTRAIN01_high')  
##  
  
sql = "select country_region, count(country_name) from sh.countries  
group by country_region"
```

```
cur = con.cursor()
cur.execute(sql)
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

Once you have edited your Python code execute it from the directory where the file is located:

```
python select.py
```

```
$ cd /home/oracle/labScripts/lab9
$ python select.py
[('Asia', 5), ('Middle East', 1), ('Europe', 10), ('Americas', 4), ('Oceania', 2),
 ('Africa', 1)]
```

A slight variation to the script will allow the results to be displayed in a loop one at a time on a new line until all results are displayed, unlike the previous code which prints all the results at once.

Edit file in the lab9 directory called **selectloop.py** by changing the connection credentials in the **con** line below to reflect your account/password@connectstring information. The complete selectloop.py program is listed below.

```
import cx_Oracle

## Edit the below line to contain your password and connect string
con = cx_Oracle.connect('admin/ATPwelcome-1234@MELATPTRAIN01_high')
##

cur = con.cursor()
sql = "select country_region, count(country_name) from sh.countries
group by country_region"
cur.execute(sql)
for result in cur:
    print (result)
cur.close()
con.close()
```

Execute the new script using the command python selectloop.py

```
python selectloop.py
```

You should receive the result.

```
$cd /home/oracle/labScripts/lab9
$python selectloop.py
('Asia', 5)
('Middle East', 1)
('Europe', 10)
('Americas', 4)
('Oceania', 2)
('Africa', 1)
```

To demonstrate some of the different ways to query and display results in Python, the **querytypes.py** script performs a query and then displays results in three different ways. The output of the scripts provides a brief description of the results display process.

Edit the file called **querytypes.py** by changing the connection credentials in the **con** line below to reflect your account/password@connectstring information. Examine the **querytypes.py** script below (or by viewing it from the terminal) before running it.

```
from __future__ import print_function
import cx_Oracle

## Edit the below line to contain your password and connect string
con = cx_Oracle.connect('admin/ATPwelcome-1234@MELATPTRAIN01_high')
##

sql = "select s_city,s_region,count(*) from ssb.supplier \
group by s_city,s_region order by count(*)"

print("Get all rows and then display all the results one row at a
time - this may take a few minutes")
cursor = con.cursor()
for result in cursor.execute(sql):
    print(result)
print()

print("Fetch one row at a time using fetchone - do this twice to show
2 results")
cursor.execute(sql)
row = cursor.fetchone()
print(row)
row = cursor.fetchone()
print(row)
print()

print("Fetch 3 rows using fetchmany - displayed all at once")
cursor.execute(sql)
res = cursor.fetchmany(numRows=3)
print(res)
con.close()
```

Execute your Python script. It will take a few minutes to completion and will display a long output

```
python querytypes.py
```

Sample output

```
$ cd /home/oracle/labScripts/lab9
$ python querytypes.py
Get all rows and then display all the results one row at a time - this may take a
few minutes
('UNITED ST9', 'AMERICA      ', 7792)
('GERMANY  1', 'EUROPE       ', 7812)
('ETHIOPIA 8', 'AFRICA        ', 7841)
('ETHIOPIA 0', 'AFRICA        ', 7842)
```

## Creating database objects

The goal of this section is to create a table using Python. The **createtable.py** will try to drop the table in case it already exists, and then create the new table.

Remember to edit the script to change the connection credentials in the **con** line below to reflect your account/password@connectstring information. The **createtable.py** is listed below. View it before running the script.

```
import cx_Oracle

## Edit the below line to contain your password and connect string
con = cx_Oracle.connect('admin/ATPwelcome-1234@MELATPTTRAIN01_high')
##

droptable = "DROP TABLE DEMO_COUNTRIES"

createtable = "CREATE TABLE DEMO_COUNTRIES ( \
    COUNTRY_ID          NUMBER, \
    COUNTRY_ISO_CODE    CHAR(2), \
    COUNTRY_NAME         VARCHAR2(40), \
    COUNTRY_SUBREGION   VARCHAR2(30), \
    COUNTRY_SUBREGION_ID NUMBER, \
    COUNTRY_REGION      VARCHAR2(20), \
    COUNTRY_REGION_ID   NUMBER, \
    COUNTRY_TOTAL        NUMBER(9), \
    COUNTRY_TOTAL_ID    NUMBER, \
    COUNTRY_NAME_HIST   VARCHAR2(40) \
)"

checkobject = "select object_name, object_type, created from
user_objects where object_name='DEMO_COUNTRIES'"
cur = con.cursor()
try:
    cur.execute(droptable)
except cx_Oracle.DatabaseError:
    print ("Table does not exist")

print ("Executed drop table statement")
cur.execute(createtable)
print ("Executed create table statement")
cur = con.cursor()
print ("Checking the table exists in user_objects")
cur.execute(checkobject)
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

Execute your Python script

```
python createtable.py
```

```
$ cd /home/oracle/labScripts/lab9
$ python createtable.py
Executed drop table statement
```

```
Executed create table statement
Checking the table exists in user_objects
[('DEMO_COUNTRIES', 'TABLE', datetime.datetime(2019, 1, 17, 16, 41, 37))]
```

## Using Python to Insert and Update data

The **insertupdate.py** script will:

- Drop the table if it already exists so this script can be run multiple times giving the same results.
- Create the table.
- Insert data into the table
- Update data in the table.

Remember to change the connection credentials in the **con** line (shown below) to reflect your account/password@connectstring information in the **insertupdate.py** script.

```
import cx_Oracle

## Edit the below line to contain your password and connect string
con = cx_Oracle.connect('admin/ATPwelcome-1234@MELATPTTRAIN01_high')
##

droptable = "DROP TABLE DEMO_COUNTRIES"

createtable = "CREATE TABLE DEMO_COUNTRIES ( \
    COUNTRY_ID          NUMBER , \
    COUNTRY_ISO_CODE    CHAR(2) , \
    COUNTRY_NAME        VARCHAR2(40) , \
    COUNTRY_SUBREGION   VARCHAR2(30) , \
    COUNTRY_SUBREGION_ID NUMBER, \
    COUNTRY_REGION      VARCHAR2(20) , \
    COUNTRY_REGION_ID   NUMBER , \
    COUNTRY_TOTAL        NUMBER(9), \
    COUNTRY_TOTAL_ID    NUMBER, \
    COUNTRY_NAME_HIST   VARCHAR2(40) \
)"

checktable="select * from DEMO_COUNTRIES"
# Try to drop the table ignore exception in case table does not exist.
cur = con.cursor()
try:
    cur.execute(droptable)
except cx_Oracle.DatabaseError:
    print ("Table does not exist")

print ("Executed drop table statement")
# Create our Table
cur.execute(createtable)
print ("Executed create table statement")
cur = con.cursor()
#here we insert data into the table, commit and query it
cur = con.cursor()
```

```
statement = 'insert into demo_countries(country_id, country_name,
country_region) values (:2, :4, :7)'
cur.execute(statement, (1, 'Paraguay', 'South America'))
con.commit()
print ("Inserted data about Paraguay")
cur.execute(checktable)
result = cur.fetchall()
print (result)

#here we update data in the table and query it
cur = con.cursor()
statement = 'update demo_countries set country_name = :1 where
country_name = :2'
cur.execute(statement, ("Republic of Paraguay", "Paraguay"))
con.commit()
print ("Updating contry name to Republic of Paraguay")
cur.execute(checktable)
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

Execute your script using **python insertupdate.py**

```
python insertupdate.py
```

```
$ cd /home/oracle/labScripts/lab9
$ python insertupdate.py
Executed drop table statement
Executed create table statement
Inserted data about Paraguay
[(1, None, 'Paraguay', None, None, 'South America', None, None, None, None)]
Updating contry name to Republic of Paraguay
[(1, None, 'Republic of Paraguay', None, None, 'South America', None, None, None, None)]
```

## Removing hard coded connect strings

In the previous scripts, the credential information for the ATP service was a hard coded **cx\_Oracle.connect** call. This is non-secure and hard to manage should something change with the credentials. With the Python os library it is possible to define an operating system environment variable with the credential information and then simply read that variable from inside the Python script. This provides a more secure mechanism to provide credentials and an easy way to reduce code changes due to database service changes.

To set the environment variable with your login credentials run the following in your command prompt window. The environment variable name is your choice, however as it will be referenced in the code later, for this Python lab you must use **atp\_connect**.

```
export atm_connect=<your user>/<your password>@<connect string>
```

For example:

```
$ export atm_connect=admin/ATPwelcome-1234@melatptrain01_tp
```

The next script will re-run the first select statement we ran in `select.py` but instead of hardcoding the connection credentials it will call the environment variable to get the connection information. To allow this, we will need to import the `os` library in Python. The script `selectconnectstring.py` is listed below.

```
import cx_Oracle
import os
connectstring = os.getenv('atm_connect')
print ("Got the following value for atm_connect")
print (connectstring)
con = cx_Oracle.connect(connectstring)
sql = "select country_region, count(country_name) from sh.countries
group by country_region"
cur = con.cursor()
cur.execute(sql)
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

Set your operating system environment variable `atm_connect` to the connect string for your database and execute the code using `python selectconnectstring.py`.

```
export atm_connect=admin/ATPwelcome-1234@melatptrain01_tp
python selectconnectstring.py
```

```
$ cd /home/oracle/labScripts/lab9
$ export atm_connect=admin/ATPwelcome-1234@melatptrain01_tp
$ python selectconnectstring.py
Got the following value for atm_connect
admin/ATPwelcome-1234@melatptrain01_tp
[('Asia', 5), ('Middle East', 1), ('Europe', 10), ('Americas', 4), ('Oceania', 2),
('Africa', 1)]
```

## Passing parameters

The `variables.py` script relies on the previous labs being successful. It will use the same Operating System variable for the connect string, and the `DEMO_COUNTRIES` table that you used previously. You will call a Python script and pass it three parameters. Passing parameters to a Python script requires using the `sys` library so the script includes an `import sys` statement. This is a very simple script with limited checking and error handling, so parameters must be passed exactly as shown or the script will return an error. The parameters passed will be inserted into the table, and the updated table contents will be displayed. The `variables.py` script is listed below:

```
import cx_Oracle
import os
import sys

connectstring = os.getenv('atm_connect')
con = cx_Oracle.connect(connectstring)
print ("Got the following value for atm_connect")
print (connectstring)
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
checktable='select country_id, country_name, country_region  from
demo_countries'
inserttable='insert into demo_countries(country_id, country_name,
country_region) values (:2, :4, :7)'

# Check that 3 arguments are passed in from the command line
if (len(sys.argv) < 4 or len(sys.argv) > 4):
    print (" This script requires exactly 3 parameters and received
", str(sys.argv))
    print (" A working example:  python variables.py 10 France
Europe")
    exit(1)

# Check that the first argument is a whole number
try:
    check = int(sys.argv[1])
except:
    print(" Error: Argument one needs to be a whole number ")
    print (" A working example:  python variables.py 10 France
Europe")
    exit(1)

print ("Arguments provided:", sys.argv[1], " ",sys.argv[2], " ",
sys.argv[3] )
id = sys.argv[1]
country = sys.argv[2]
region = sys.argv[3]

# Check that the target table exists
checkobject = "select count(table_name)  from user_tables where
table_name = upper('demo_countries')"
cur = con.cursor()
try:
    cur.execute(checkobject)
    result = cur.fetchall()
    if(int(result[0][0]) < 1):
        print ("Error: Table does not exist. Please run
createtable.py first")
        exit(1)
except cx_Oracle.DatabaseError:
    print ("Error: Table does not exist. Please run createtable.py
first")
    exit(1)

print ("Inserting your data")
cur = con.cursor()
cur.execute(inserttable, (id, country, region))
con.commit()
print ("Selecting from DEMO_COUNTRIES")
for result in cur.execute(checktable):
    print (result)
print()
cur.close()
con.close()
```

Ensure you have the connect string set as **atp\_connect**, and that when you execute the script you provide three parameters, the first of which must be numeric. E.g.  
**python variables.py 10 Finland Europe**

```
export atm_connect=admin/ATPwelcome-1234@melatptrain01_tp
python variables.py 10 Finland Europe
```

You should expect the following output

```
$ export atm_connect=admin/ATPwelcome-1234@melatptrain01_tp
$ python variables.py 10 Finland Europe
Got the following value for atm_connect
admin/ATPwelcome-1234@melatptrain01_tp
Arguments provided: 10    Finland    Europe
Inserting your data
Selecting from DEMO_COUNTRIES
(10, 'Finland', 'Europe')
(1, 'Republic of Paraguay', 'South America')
```

## Optional – Using Anaconda/Jupyter/Python with ATP

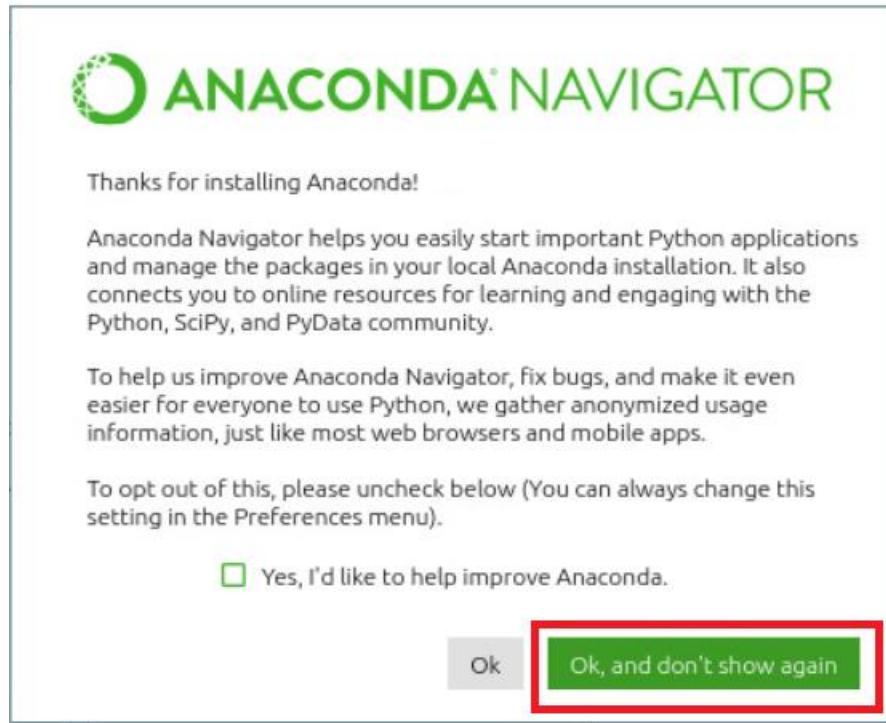
Integrated Development Environments (IDEs) are very popular with the development community. Among the most popular for Python are the combination of Anaconda/Jupyter or Spyder. Installing this environment is out of scope for this Hands on Lab, and so the required software is pre-installed in your Lab VM.

To start Anaconda, enter the following command in your terminal window.

```
$HOME/anaconda3/bin/anaconda-navigator
```

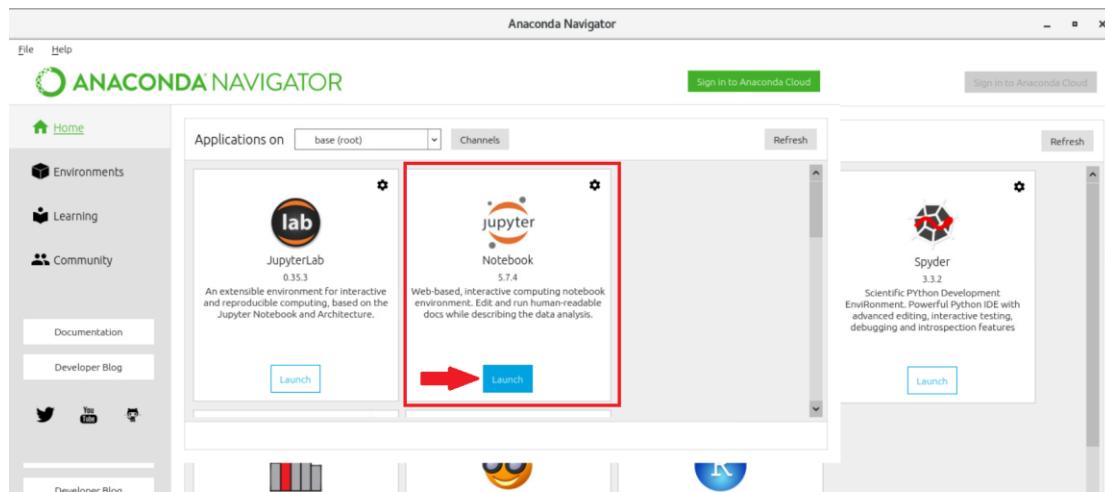
This will start the Anaconda Navigator GUI.

The first time you start Anaconda you will get a message box, select ‘OK, and don’t show again’

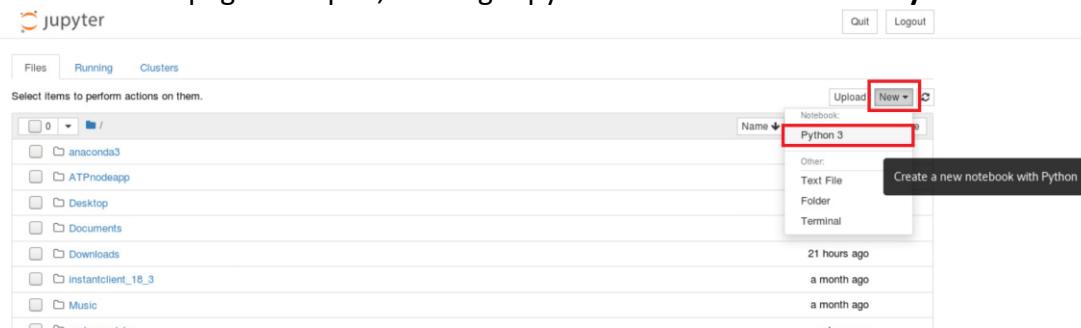


## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Locate **Jupyter Notebook** in Anaconda Navigator and select **Launch**.



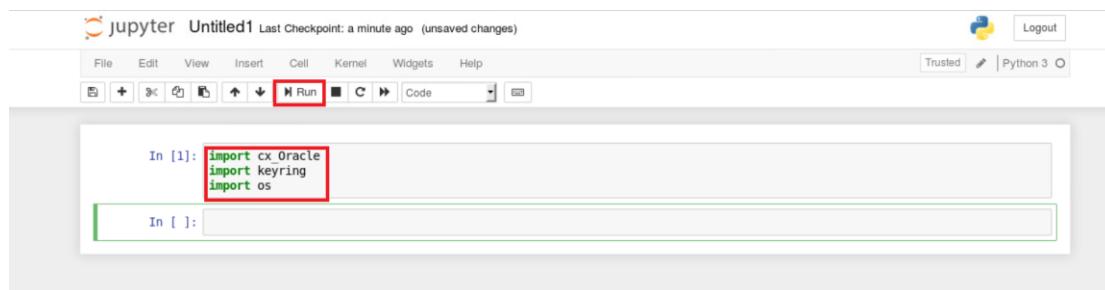
A new browser page will open, running Jupyter. Select **New** and then **Python 3**



This will open a new Python notebook. Python is an interpreted language, so we must load libraries to use every time an environment is started up. Libraries are loaded with the import command.

Copy the 3 import statements below and paste them directly in the box next to the **In[]:** prompt, then select **Run**.

```
import cx_Oracle  
import keyring  
import os
```



The import commands will not produce any output.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Run a simple command to display your PATH. Run the following command (copy and paste next to the **In[]** prompt and select **Run**):

```
print(os.environ["PATH"])
```

The screenshot shows a Jupyter Notebook window titled "Untitled1". The toolbar at the top includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A red box highlights the "Run" button in the toolbar. Below the toolbar, the code cell In [1] contains three lines of Python code: `import cx\_Oracle`, `import keyring`, and `import os`. The code cell In [2] contains the command `print(os.environ["PATH"])` and its output, which is the full path of the Oracle Instant Client and Anaconda bin directories. The status bar at the bottom indicates "Last Checkpoint: 4 minutes ago (unsaved changes)".

To connect to the database, it must be able to locate the wallet files. The TNS\_ADMIN variable must be set to the location of your unzipped wallet files. Below we set and then check the variable (the first command sets it, the second displays it back). Run the following command (copy and paste next to the **In[]** prompt and select **Run**):

```
os.environ['TNS_ADMIN'] = '/home/oracle/wallets'  
print(os.environ["TNS_ADMIN"])
```

The screenshot shows a Jupyter Notebook window titled "Untitled1". The toolbar at the top includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A red box highlights the "Run" button in the toolbar. Below the toolbar, the code cell In [1] contains three lines of Python code: `import cx\_Oracle`, `import keyring`, and `import os`. The code cell In [2] contains the command `print(os.environ["PATH"])` and its output, which is the full path of the Oracle Instant Client and Anaconda bin directories. The code cell In [3] contains two lines of Python code: `os.environ['TNS\_ADMIN'] = '/home/oracle/wallets'` and `print(os.environ["TNS\_ADMIN"])`. The output of this cell is the path "/home/oracle/wallets". The status bar at the bottom indicates "Last Checkpoint: 13 minutes ago (unsaved changes)".

To make SQL calls to ATP, we need to load an additional library to support calls to external sql databases. Run the command below (copy and paste next to the **In[]** prompt, make sure to include the % and select **Run**):

```
%load_ext sql
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows a Jupyter Notebook interface with the following steps:

- In [1]: `import cx_Oracle`  
`import keyring`  
`import os`
- In [2]: `print(os.environ["PATH"])`  
Output: /home/oracle/anaconda3/bin:/bin:/home/oracle/anaconda3/bin:/home/oracle/anaconda3/bin:/home/oracle/anaconda3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/oracle/instantclient\_18\_3:/home/oracle/.local/bin:/home/oracle/bin:/home/oracle/instantclient\_18\_3:/home/oracle/instantclient\_18\_3
- In [3]: `os.environ['TNS_ADMIN'] = '/home/oracle/wallets'`  
`print(os.environ['TNS_ADMIN'])`  
Output: /home/oracle/wallets
- In [4]: `%load_ext sql`
- In [ ]: (This cell is empty)

Next we will connect to the ATP database using a user name, password and service. Use your admin account and password created when the ATP database was created. The format of the command is (replace the black portion with your database and users information):

```
%sql oracle+cx_oracle://user:password@service
```

So your command would look like similar to this:

```
%sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp
```

Enter your command next to the In[] prompt, make sure to include the % and select Run.

```
In [5]: %sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp  
Out[5]: 'Connected: admin@None'
```

When the connection succeeds you will get the message 'Connected: admin@None'

To run a query, once connected use the **oracle+cx** library calls followed by the SQL statement (notice there is no ';' at the end of the statement). The SQL below is the same one we ran in previous labs.

The statement below can be found in **/home/oracle/labScripts/lab9/select.jupyter**. Copy this files content into the Jupyter box, replace the connection detail with your connection details, and click Run.

```
%%sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp  
SELECT channel_desc, TO_CHAR(SUM(amount_sold), '9,999,999,999')  
SALE$,  
      RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,  
      RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS  
custom_rank  
FROM sh.sales, sh.products, sh.customers, sh.times, sh.channels,  
sh.countries  
WHERE sales.prod_id=products.prod_id AND  
sales.cust_id=customers.cust_id
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
    AND customers.country_id = countries.country_id AND
sales.time_id=times.time_id
    AND sales.channel_id=channels.channel_id
    AND times.calendar_month_desc IN ('2000-09', '2000-10')
    AND country_iso_code='US'
GROUP BY channel_desc
```

```
Out[5]: 'Connected: admin@None'

In [10]: %%sql oracle+cx oracle://admin:ATPwelcome-1234@melatptrain01_tp
SELECT channel_desc, To_CHAR(SUM(amount_sold), '9,999,999,999') SALES$,
       RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,
       RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS custom_rank
FROM sh.sales, sh.products, sh.customers, sh.times, sh.channels, sh.countries
WHERE sales.prod_id=products.prod_id AND sales.cust_id=customers.cust_id
      AND customers.country_id = countries.country_id AND sales.time_id=times.time_id
      AND sales.channel_id=channels.channel_id
      AND times.calendar_month_desc IN ('2000-09', '2000-10')
      AND country_iso_code='US'
GROUP BY channel_desc
0 rows affected.

Out[10]:   channel_desc      sales$  default_rank  custom_rank
          Direct Sales    1,320,497           3            1
          Partners        800,871           2            2
          Internet        261,278           1            3
```

You can issue DDL commands via Jupyter. Next we will create a simple table. As in the last example copy the contents from the file **/home/oracle/labScripts/lab9/create-table.jupyter** below, replace the black text with your database and connection information, paste it in the box, and click Run.

```
%%sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp
CREATE TABLE J_COUNTRIES (
    COUNTRY_ID             NUMBER,
    COUNTRY_ISO_CODE       CHAR(2),
    COUNTRY_NAME            VARCHAR2(40),
    COUNTRY_SUBREGION       VARCHAR2(30),
    COUNTRY_SUBREGION_ID   NUMBER,
    COUNTRY_REGION          VARCHAR2(20),
    COUNTRY_REGION_ID      NUMBER,
    COUNTRY_TOTAL            NUMBER(9),
    COUNTRY_TOTAL_ID        NUMBER,
    COUNTRY_NAME_HIST       VARCHAR2(40)
)
```

```
In [14]: %%sql oracle+cx oracle://admin:ATPwelcome-1234@melatptrain01_tp
CREATE TABLE J_COUNTRIES (
    COUNTRY_ID             NUMBER,
    COUNTRY_ISO_CODE       CHAR(2),
    COUNTRY_NAME            VARCHAR2(40),
    COUNTRY_SUBREGION       VARCHAR2(30),
    COUNTRY_SUBREGION_ID   NUMBER,
    COUNTRY_REGION          VARCHAR2(20),
    COUNTRY_REGION_ID      NUMBER,
    COUNTRY_TOTAL            NUMBER(9),
    COUNTRY_TOTAL_ID        NUMBER,
    COUNTRY_NAME_HIST       VARCHAR2(40)
)
0 rows affected.

Out[14]: []
```

You will not get a message saying the table was created. You can check it created by querying the USER\_OBJECTS view. The below statement can be copied from the **/home/oracle/labScripts/lab9/check-table.jupyter** script, replace the black text with your database and connection information in the script and paste it in the box and click Run.

```
%%sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp
SELECT OBJECT_NAME, OBJECT_TYPE from USER_OBJECTS
where OBJECT_NAME='J_COUNTRIES'
```

```
In [17]: %%sql oracle+cx_oracle://admin:ATPwelcome-1234@melatptrain01_tp
SELECT OBJECT_NAME, OBJECT_TYPE from USER_OBJECTS
where OBJECT_NAME='J_COUNTRIES'
```

0 rows affected.

```
Out[17]:   object_name  object_type
              J_COUNTRIES      TABLE
```

This concludes the lab on using Python with ATP.



# Lab 10 - Using Docker Containers with ATP

# Lab 10: Using Docker Containers with Autonomous Transaction Processing

## Objectives:

- To build a Docker container running a Node.js application microservice.
- Connect this microservice to an Autonomous Transaction Processing (ATP) Database service running in the Oracle cloud.
- Package your Docker container image for deployment.

Microservices and Docker containers are very popular development and deployment environments. We will use a Node.js application, deploy it as a Docker container and show how ATP can be used as data management for these applications.

In a previous lab you ran Node.js scripts to learn the basics of Node.js and ATP. In this lab you will deploy a Node.js simple storefront application inside a Docker container which accesses information stored in an ATP database. You will create your Docker container from a build file available from Github.

Once your Docker image is built you will customize it to access your ATP database, and then package your customized image for deployment as a Docker image. This lab builds on an Oracle Learning Lab located [here](#). For any updates please refer to this site. As we are downloading a pre-created image, some values are hardcoded.

We have already downloaded the sample Node.js application from  
<https://github.com/cloudsolutionhubs/ATPnodeapp>  
and installed it on the VM under **/home/oracle/ATPDocker**

**Please pay close attention to names that cannot be changed and those that need to be changed.**

The microservice will connect to the ATP database you created using the Oracle instantclient. The service has a hard-coded requirement for the 12.1.0.2.0 instantclient libraries. We have downloaded these libraries and placed them in the **/home/oracle/ATPDocker** directory on your VM.

As the application uses hardcoded paths we need to copy the ATP wallet file from your ATP database to the '**/home/oracle/ATPDocker/wallet\_NODEAPPDB2**' and unzip it (see lab 8 on instructions how to do this if needed). The below commands assume your ATP wallet zip file has previously been copied to the **/home/oracle/wallets** directory from previous labs. Note the '.' at the end of the cp command.

```
cd /home/oracle/ATPDocker/wallet_NODEAPPDB2
cp /home/oracle/wallets/<wallet_file>.zip .
unzip <wallet_file>.zip
```

```
$ cd /home/oracle/ATPDocker/wallet_NODEAPPDB2
$ cp /home/oracle/wallets/wallet_MELATPTRAIN01.zip .
$ unzip wallet_MELATPTRAIN01.zip
Archive: wallet_MELATPTRAIN01.zip
  inflating: cwallet.sso
  inflating: tnsnames.ora
  inflating: truststore.jks
  inflating: ojdbc.properties
  inflating: sqlnet.ora
  inflating: ewallet.p12
  inflating: keystore.jks
```

Edit the sqlnet.ora and change the wallet location to reference the TNS\_ADMIN environment variable.

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="$TNS_ADMIN")))
SSL_SERVER_DN_MATCH=yes
```

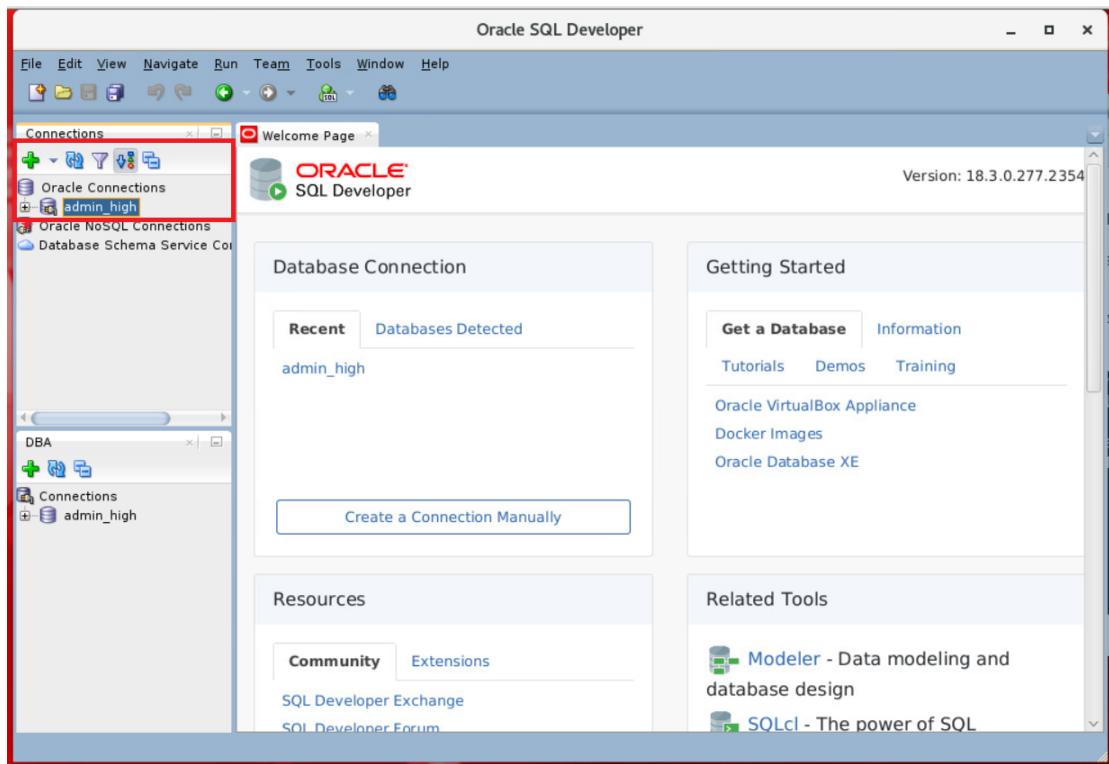
## Populating the ATP Database for the Docker/Node.js application

For this simple lab, we are going to deploy the application into the admin schema and connect as the admin schema. This is not best practice for production applications.

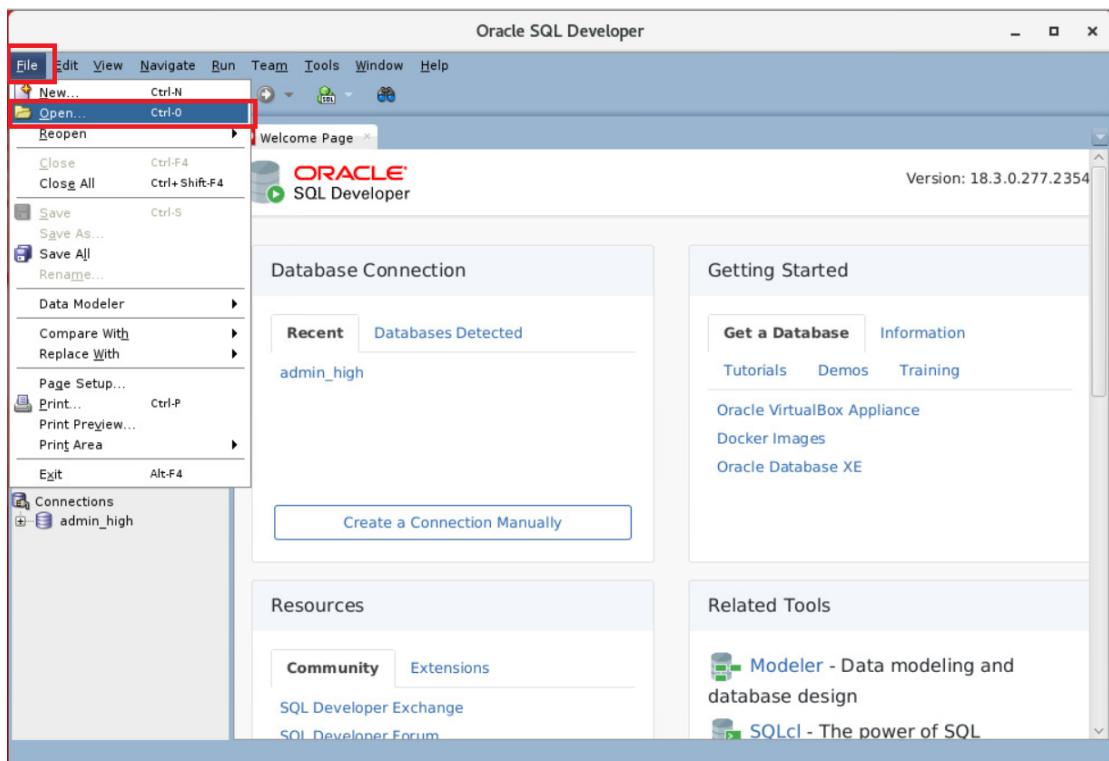
The application that will be deployed in the container accesses specific tables in the ATP database that need to be created and populated. The complete SQL to perform this task is located in the **create\_schema.sql** file located in the **/home/oracle/ATPDocker/aone** directory.

The easiest way to execute this script is through SQL Developer. Start SQL Developer if you do not already have it open. Under Connections open a connection to your ATP instance as admin. Refer to Lab 2 for more detailed instructions if you have not configured this as part of Lab2.

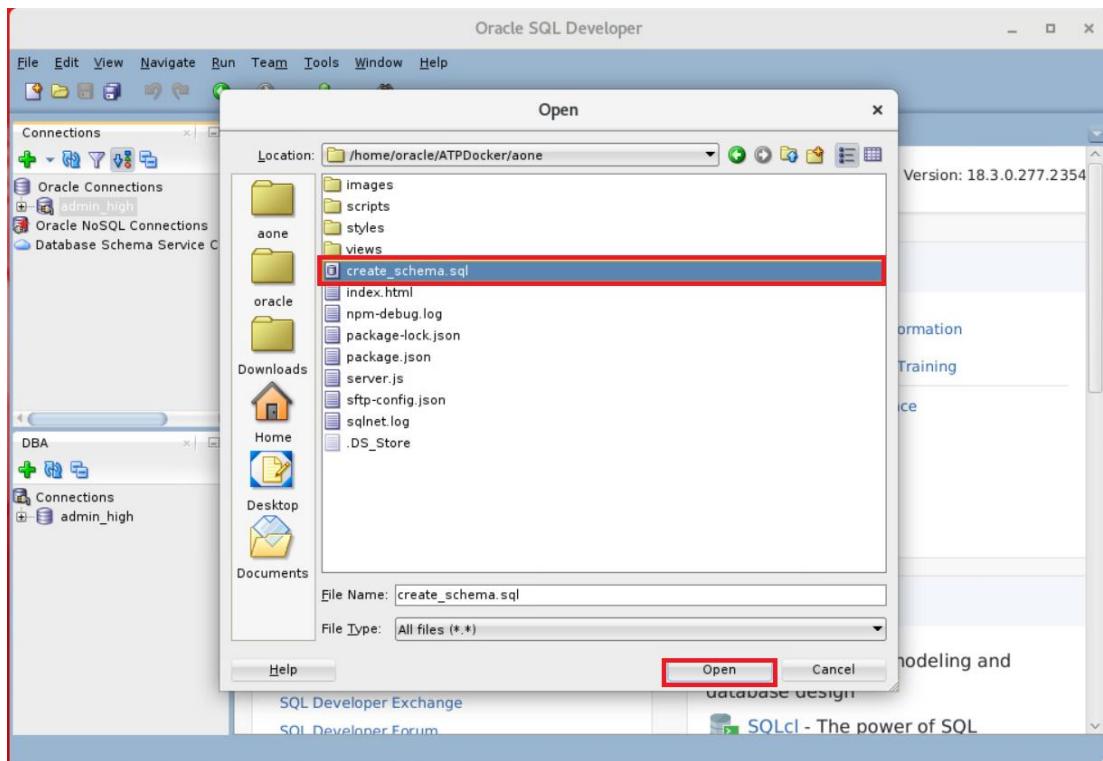
## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Select file open and navigate to **/home/oracle/ATPDocker/aone**, select **create\_schema.sql** and click **open**.

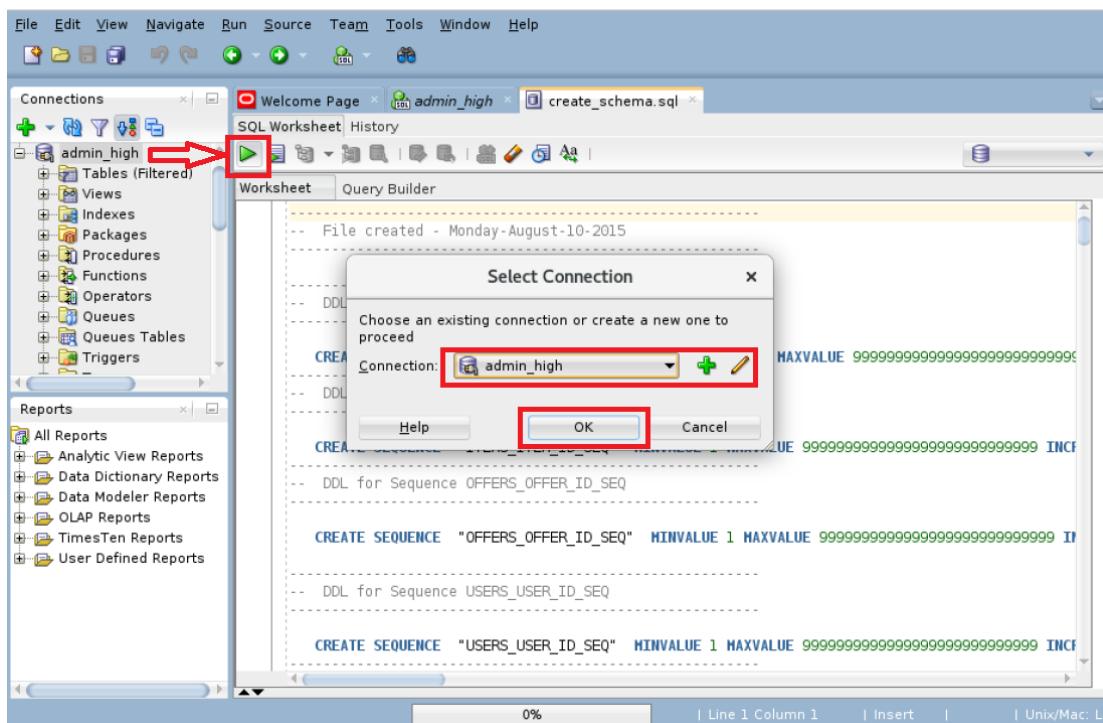


## Oracle Autonomous Transaction Processing Hands-on Lab Guide

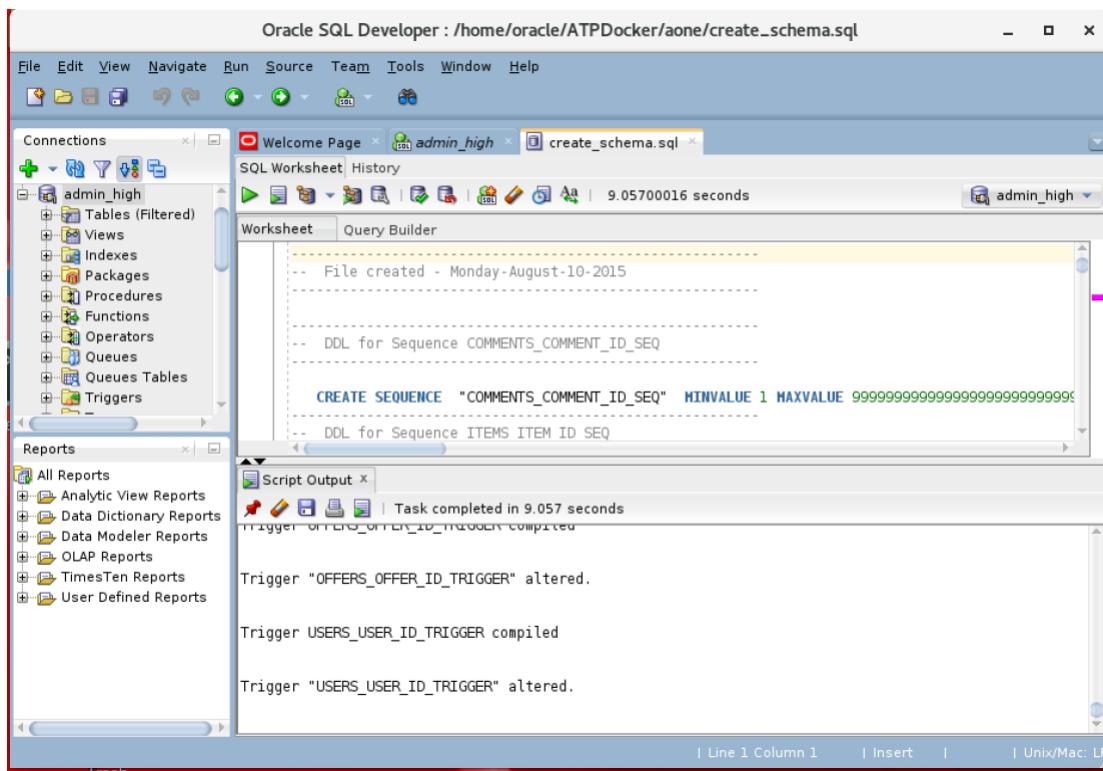


This will load the script into a SQL Worksheet.

Run the script, a pop-up window will ask you select a connection. Select the **admin\_high** connection used in previous labs.



After asking you for the admin password the script will run and create the schema required for the Node.js application.



Your database is now prepared for the application.

Alternatively, you can copy the entire contents of **create\_schema.sql** file, and paste it in the SQL Developer Query Builder box and run it. (Like in previous SQL Developer labs).

## Creating your Docker image

There is a large repository of pre-created registered Docker images that can be searched and selected for use. Docker images can either be pulled from this repository or created from configuration files. For this lab we will be using configuration files that will create our image that contains all the components needed to run the node.js application being deployed.

Before creating our image explore what images for “Oracle” already exist in the repository. In a terminal window navigate to **/home/oracle/ATPDocker** and run the Docker command to search the repository:

```
cd /home/oracle/ATPDocker  
sudo docker search "oracle"
```

You will see a list of images available

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

INDEX	NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
docker.io	oraclelinux	Official Docker builds of Oracle Linux.	545	[OK]	[OK]
docker.io	sath89/oracle-12c	Oracle Standard Edition 12c Release 1 with d...	443	[OK]	[OK]
docker.io	frolvlad/alpine-oraclejdk8	The smallest Docker image with OracleJDK 8 (...)	358	[OK]	[OK]
docker.io	alexeiled/docker-oracle-xe-11g	This is a working (hopefully) Oracle XE 11.2...	280	[OK]	[OK]
docker.io	sath89/oracle-xe-11g	Oracle xe 11g with database files mount supp...	237	[OK]	[OK]
docker.io	wnameless/oracle-xe-11g	Dockerfile of Oracle Database Express Editio...	155	[OK]	[OK]
docker.io	jaspeen/oracle-11g	Docker image for Oracle 11g database	82	[OK]	[OK]
docker.io	isuper/java-oracle	This repository contains all java releases f...	56	[OK]	[OK]
docker.io	oracle/openjdk	Docker images containing OpenJDK Oracle Linux	53	[OK]	[OK]
docker.io	airdock/oracle-jdk	Docker Image for Oracle Java SDK (8 and 7) b...	43	[OK]	[OK]
docker.io	sath89/oracle-ee-11g	Dockerfile of Oracle Database Enterprise Edi...	43	[OK]	[OK]
docker.io	cogniteev/oracle-java	Oracle JDK 6, 7, 8, and 9 based on Ubuntu 16...	24	[OK]	[OK]
docker.io	ingensi/oracle-jdk	Official Oracle JDK installed on centos.	21	[OK]	[OK]
docker.io	oracle/nosql	Oracle NoSQL on a Docker Image with Oracle L...	17	[OK]	[OK]
docker.io	sgrio/java-oracle	Docker images of Java 7/8/9/10 provided by O...	17	[OK]	[OK]
docker.io	n3ziniku5/ubuntu-oracle-jdk	Ubuntu with Oracle JDK. Check tags for versi...	16	[OK]	[OK]
docker.io	andreptb/oracle-java	Debian Jessie based image with Oracle JDK in...	7	[OK]	[OK]
docker.io	frolvlad/alpine-oraclejre8	The smallest Docker image with OracleJRE 8 (...)	7	[OK]	[OK]
docker.io	davidcaste.debian-oracle-java	Oracle Java 8 (and 7) over Debian Jessie	4	[OK]	[OK]
docker.io	martinseelner/oracle-server-jre	Oracle's Java 8 as 61 MB Docker container.	4	[OK]	[OK]
docker.io	teradatalabs/centos6-javab8-oracle	Docker image of CentOS 6 with Oracle JDK 8 i...	4	[OK]	[OK]
docker.io	publicisworldwide/oracle-core	This is the core image based on Oracle Linux...	1	[OK]	[OK]
docker.io	bitnami/oraclelinux-extras	Oracle Linux base images	0	[OK]	[OK]
docker.io	pivotaldata/oracle7-test	Oracle Enterprise Linux (OEL) image for GPDB...	0	[OK]	[OK]
docker.io	softwareplant/oracle	oracle db	0	[OK]	[OK]

If you wanted to use any of these images you would use the '**docker pull**' command, for example '**docker pull oraclelinux**' would pull the oraclelinux image from the repository and install it on your Docker containers.

In this lab we will not pull an image but build our own from configuration files.

Edit the Docker configuration file to set the github repository location.

```
cd /home/oracle/ATPDocker  
vi Dockerfile
```

Change line 34 from

```
RUN git clone https://github.com/cloudsolutionhubs/ATPDocker.git
```

to

```
RUN git clone https://github.com/melanieosc/ATPDocker.git
```

And save the file.

You are ready to create your Docker image that will contain the Node.js application. Ensure that you are in the **/home/oracle/ATPDocker** directory and run the Docker build command. We will build an image called "**aone**" based on the steps and configurations defined in the default "**Dockerfile**" in this directory (alternate configuration files could be selected with the -f parameter if required):

```
cd /home/oracle/ATPDocker  
sudo docker build -t aone .
```

Note the "dot" at the end of the Docker build command.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
[oracle@demo-long ~]$ cd /home/oracle/ATPDocker
[oracle@demo-long ATPDocker]$ sudo docker build -t aone .
Sending build context to Docker daemon    76.1MB
Step 1/18 : FROM frovlvlad/alpine-glibc:alpine-3.8
--> 29238b990a07
Step 2/18 : RUN apk update && apk add libaio libnsl &&     ln -s /usr/lib/libnsl.so.2 /usr/lib/libnsl.so.1
--> Using cache
--> 0eed448b0bc1
Step 3/18 : RUN apk add --update      nodejs      nodejs-npm      git      python      && rm -rf /var/cache/apk/*
--> Using cache
--> f06ba8fc30bb
Step 4/18 : ENV CLIENT_FILENAME instantclient-basic-linux.x64-12.1.0.2.0.zip
--> Using cache
--> 6e9c58614803
Step 5/18 : WORKDIR /opt/oracle/lib
```

This will build the Docker image (for this lab ignore any errors during the build process!). You can verify the image was built by running the following command to list all your Docker images installed on your computer and looking for **aone** on the list.

```
sudo docker images -a
```

```
[oracle@demo-long-v4 ]$ sudo docker images -a
REPOSITORY          TAG           IMAGE ID        CREATED         SIZE
aone                latest        c481031662b9   20 seconds ago  576MB -----
<none>              <none>        7c5c6d7a0d8d   21 seconds ago  576MB
<none>              <none>        9ed8b976b421   22 seconds ago  576MB
<none>              <none>        e92098b80b42   29 seconds ago  562MB
<none>              <none>        f60ecae0103d   30 seconds ago  562MB
<none>              <none>        0f8127e5b055   31 seconds ago  562MB
<none>              <none>        d60b2e54d996   32 seconds ago  562MB
<none>              <none>        0407206b1f80   33 seconds ago  562MB
<none>              <none>        9fe34b133d86   34 seconds ago  562MB
<none>              <none>        8dc938d36a30   34 seconds ago  562MB
<none>              <none>        43310280c6da   39 seconds ago  562MB
<none>              <none>        e99352ef63fa   44 seconds ago  562MB
<none>              <none>        80fd2e5cb1c2   48 seconds ago  550MB
<none>              <none>        d6e80b6738ce   56 seconds ago  173MB
<none>              <none>        5b8257b45fb    58 seconds ago  110MB
<none>              <none>        8231975ed6bb   59 seconds ago  110MB
<none>              <none>        8890c4d4d2a9   About a minute ago  110MB
<none>              <none>        27df3d8e278   About a minute ago  14.9MB
frovlvlad/alpine-glibc  alpine-3.8       29238b990a07   5 weeks ago    11.3MB
```

Display the hostname of your Lab VM. It will not match the value in the screenshot.

```
uname -n
```

```
$ uname -n
demo-long-v4
```

Now run the image. The image is started in interactive mode (**-i**), using port 3050 (**-p**) and image aone (**-t**) with bash prompt (**sh**) as the interface:

```
sudo docker run -i -p 3050:3050 -t aone sh
```

The image will start, and you will be at the Unix command prompt of the container. You have created a container service running Linux on your VM. Note that the prompt has changed and if you check the hostname you will see that this has a different value to your VM. The hostname is generated, and so will not match the value in the screenshots.

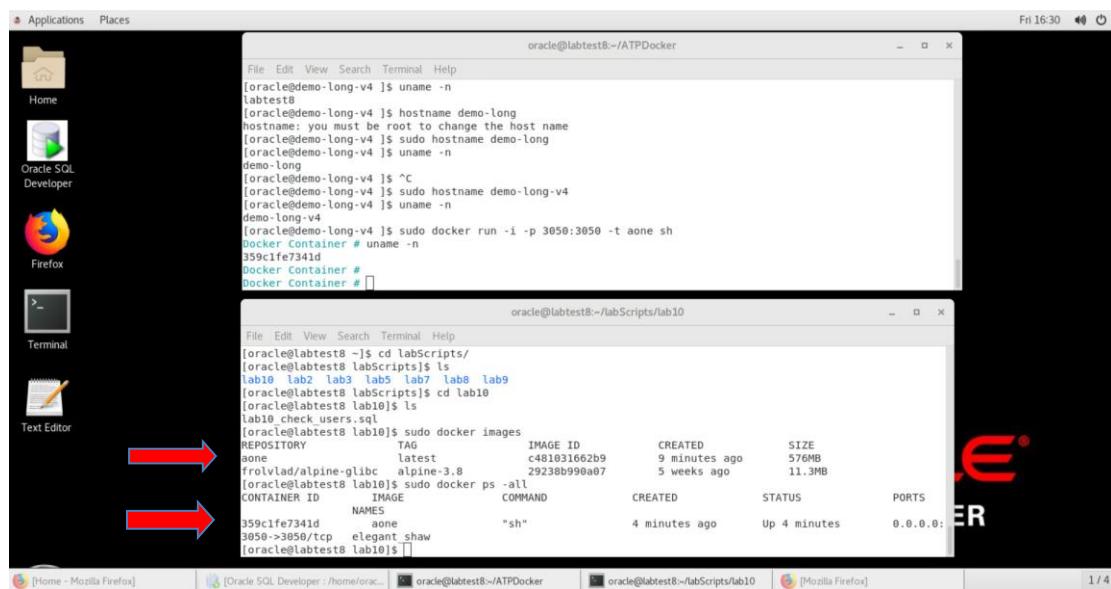
## Oracle Autonomous Transaction Processing Hands-on Lab Guide

```
[oracle@demo-long-v4 ]$ uname -n  
demo-long-v4  
[oracle@demo-long-v4 ]$ sudo docker run -i -p 3050:3050 -t aone sh  
Docker Container # uname -n  
359c1fe7341d [REDACTED] Different hostname  
Docker Container # [REDACTED]  
Docker Container # [REDACTED] Different prompt
```

Open a new terminal **window leaving the Docker image aone running** in the original terminal. In the new terminal window run the following commands to view your image information, and what Docker images are running.

You will see your **aone** image listed.

```
sudo docker images  
sudo docker ps -all
```



## Running the Node.js application in your Docker image

If you completed the previous Node.js lab the next set of steps should be familiar. At this point we have a running Docker image with all the components we manually installed and configured for the Node.js lab above. The next step is to configure the ATP database connection file (`dbconfig.js`) to contain your ATP connection information.

In the window with your running container (which is running the Bourne shell, command prompt including '#'), navigate to the directory that has the config file. You will need to know vi or another form of editing in Linux to update the file.

```
cd /opt/oracle/lib/ATPDocker/aone/scripts  
vi dbconfig.js
```

```
/opt/oracle/lib # cd /opt/oracle/lib/ATPDocker/aone/scripts  
/opt/oracle/lib/ATPDocker/aone/scripts # vi dbconfig.js
```

The file should contain your database information, below is the information for this demo database (replace items in black with your information):

```
module.exports= {
  user:"admin",
  password:"ATPwelcome-1234",
  connectString:"atpmeltrain01_high"
}
```

Once you configure this file you are ready to run the sample application provided. The node application creates a simple 'storefront' web page that pulls information from an ATP Database, and displays it on port 3050 of your web browser. To run the application, go to the directory where the application resides and run it with node:

```
cd /opt/oracle/lib/ATPDocker/aone/
node server.js
```

```
[oracle@demo-long-v4 ]$ sudo docker run -i -p 3050:3050 -t aone sh
Docker Container # cd /opt/oracle/lib/ATPDocker/aone/scripts
Docker Container # vi dbconfig.js
Docker Container # cat dbconfig.js
module.exports= {
  user:"admin",
  password:"ATPwelcome-1234",
  connectString :"melatptrain01_high"
}
Docker Container # cd /opt/oracle/lib/ATPDocker/aone
Docker Container # node server.js
aOne listening on port 3050
■
```

When the application is running you will notice an initial message indicating output on port 3050. Open a new browser window or tab and connect to:

<http://localhost:3050/>

You will see a website with information pulled from your ATP Database. Press any buttons and explore the app.

The screenshot shows a web browser window for the 'aOne' application. The URL bar shows '127.0.0.1:3050/#/'. The page has a blue header with the 'aOne' logo, 'Browse Stuff', 'Login', and 'Register' buttons. Below the header is a search bar with the placeholder 'Search'. A large graphic on the right side features four overlapping price tags in red, yellow, green, and blue, each with a string hanging from the top. The word 'SALE' is written in large white letters across the tags. To the left of the graphic is a list of five items:

Item Image	Description	Price	Status
	Bicycle 3 years ago	\$100	sold
	Samsung galaxy s6 active 3 years ago	\$255	sold
	Pool table 3 years ago	\$100	sold
	Used macbook 3 years ago	\$1650	available
	Italian Antique Hand-Paint... 4 years ago	\$70	cancelled

As you navigate the page you will see different information being pulled from the database displayed on your terminal window where you started the app. This is only informational output to show that database operations are occurring.

```
/opt/oracle/lib/ATPDocker/aone # node server.js
aOne listening on port 3050
[ { USER_NAME: 'Ashish',
  USER_GRAVATAR: 'https://www.gravatar.com/avatar/1cb1c39857f5eef49897f849251861a9.jpg?d=identicon',
  COMMENT_ID: 50,
  COMMENT_BY: 4,
  COMMENT_TEXT: 'HI, I am making an offer for this item. Hope it is in good condition. Offer subject to',
  COMMENT_CREATE_DATE: 2015-07-13T00:00:00.000Z },
{ USER_NAME: 'Ashish',
  USER_GRAVATAR: 'https://www.gravatar.com/avatar/1cb1c39857f5eef49897f849251861a9.jpg?d=identicon',
  COMMENT_ID: 61,
  COMMENT_BY: 4,
  COMMENT_TEXT: 'Hey Im putting an offer for this. let me know',
  COMMENT_CREATE_DATE: 2015-07-20T00:00:00.000Z } ]
```

### Optional – Prove that the application is using the ATP database for data storage

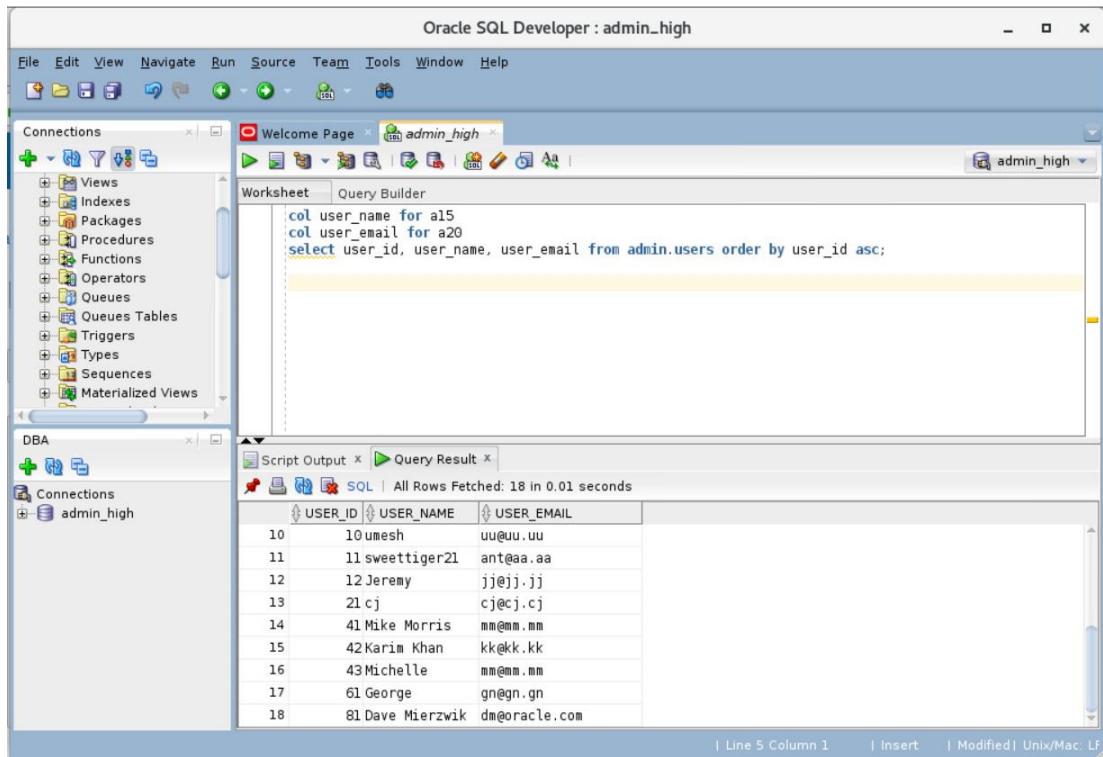
We are going to register a new user in the application and show the user in the database.

Check the current status of the registered users by starting SQL Developer, and connecting to your **admin\_high** connection. Execute the SQL to list the registered application users. This sql can be found in the file

**/home/oracle/labScripts/lab10/lab10\_check\_users.sql**

```
col user_name for a15
col user_email for a20
select user_id, user_name, user_email from admin.users order by
user_id asc;
```

## Oracle Autonomous Transaction Processing Hands-on Lab Guide



Note the number of rows returned and the highest value for USER\_ID.  
Select 'Register' in the top right-hand side of the application web page

The screenshot shows a web browser window with the URL 'localhost:3050/#/browse/61'. The page has a blue header with the text 'aOne'. On the right side of the header, there are 'Browse Stuff', 'Login', and a red-bordered 'Register' button. Below the header, there is a search bar and a list of items:

- Bicycle - \$100 (sold)
- Samsung galaxy s6 active - \$255 (sold)
- Pool table - \$100

On the right, there is a detailed view for the first item:

**Bicycle**  
Posted by Dave Mierzwik - 3 years ago sold  
**Description**  
My Schwinn Bike is ready to go  
**Offers**

Complete the form and select 'Register'. As this application only does very basic data validation use the following sample data

**Username:** atpdemo

**Email:** [atpdemo@example.com](mailto:atpdemo@example.com)

**Password:** hello

# Register

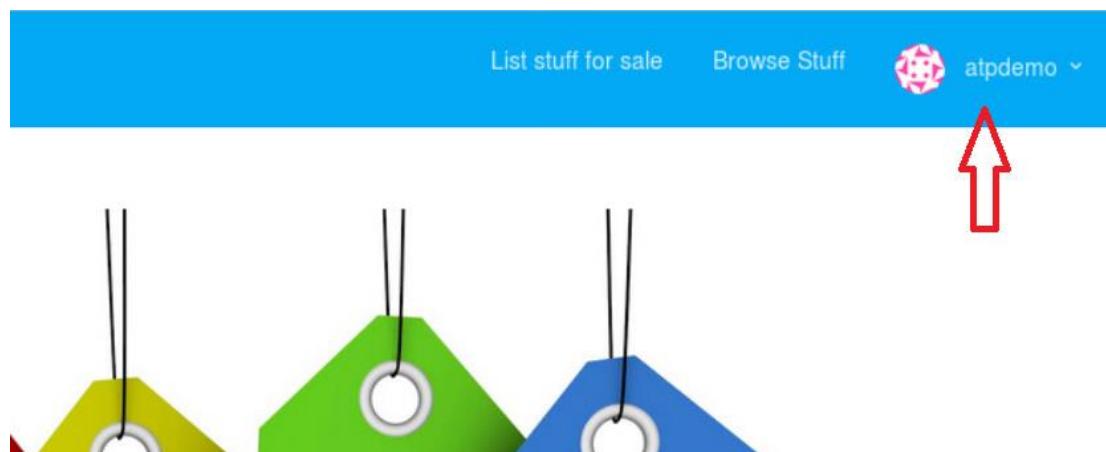
atpdemo

atpdemo@example.com

.....

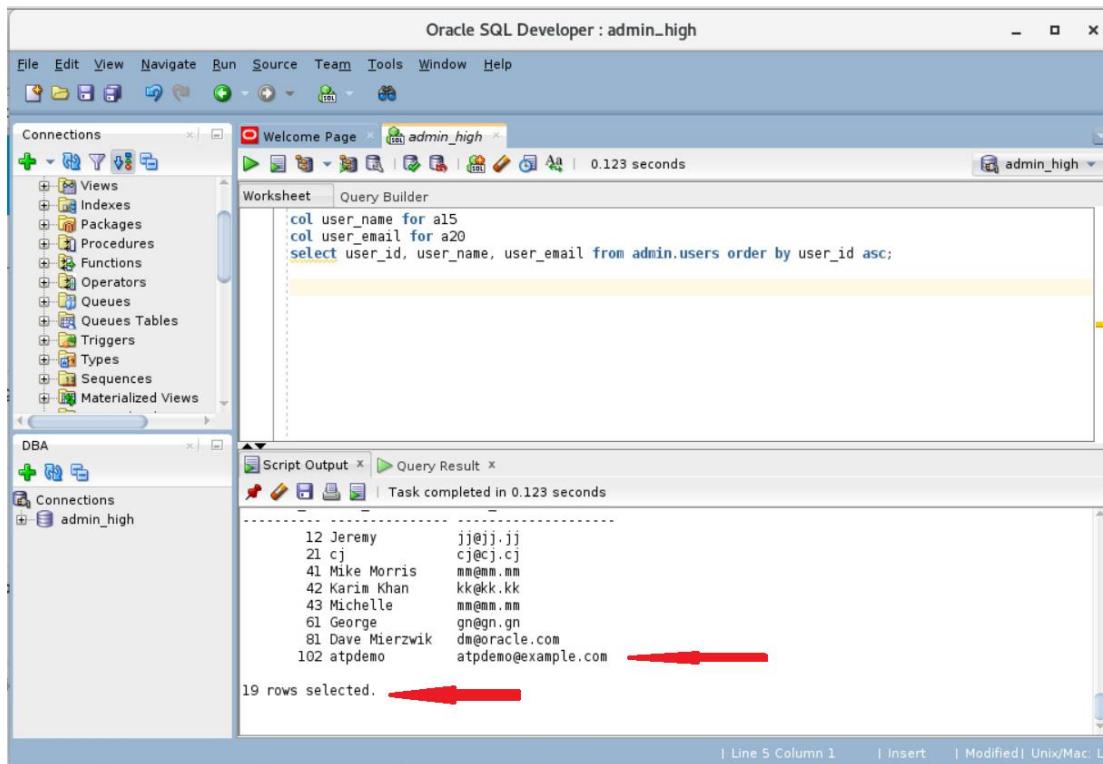
**REGISTER**

When you are returned to the store front page you will note that you are now logged in as the registered user **atpdemo**



Return to SQL Developer and execute the user query again using your **admin\_high** connection.

```
col user_name for a15
col user_email for a20
select user_id, user_name, user_email from admin.users order by
user_id asc;
```



You should see that an additional row has been added to the USERS table for the user 'atpdemo'.

## Creating and running your own customized Docker Container Image

Now that you have customized the Docker image, you can create your own container image based on the customizations you made. You can then take that image and publish it to run it on a Docker service without having to make any further customizations (for example, changing database parameters).

Open a new terminal and issue the command to list running docker containers, so you can find the CONTAINER\_ID.

```
sudo docker ps -all
```

```
[oracle@demo-long ~]$ sudo docker ps -all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
be69eb9cf89d        aone                "sh"               About an hour ago   Up About an hour   friendly_borg
[oracle@demo-long ~]$
```

To create an image of your running Docker container use the docker commit command. In this case we are going to commit changes from our running container

and create a new container with those changes. The Docker commit statement needs the CONTAINER ID (highlighted above), that is associated with the “aone” container we are currently running. Call the new container newaone, running the following command (replace the black id below with your CONTAINER\_ID from the docker –ps all command).

```
sudo docker commit be69eb9cf89d newaone
```

```
[oracle@demo-long ~]$ sudo docker commit be69eb9cf89d newaone  
sha256:7514e41b0c8ccb692db3e660a9983214e2b2eda34e3644e2c5cd75c7a62a570d  
[oracle@demo-long ~]$ █
```

To verify the image created, run the command to list the docker images.

```
sudo docker images
```

```
[oracle@demo-long ~]$ sudo docker images  
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE  
newaone              latest   7514e41b0c8c  About a minute ago  735MB  
aone                 latest   6fe15b569d29  About an hour ago  568MB  
frolvlad/alpine-glibc  alpine-3.8  29238b990a07  2 weeks ago   11.3MB  
[oracle@demo-long ~]$
```

Your “newaone” will be listed along with the original aone image. You have just created a Docker image.

Now run your new Docker image. As this image is configured to use the same ports as our original image, we need to stop the original image before we can successfully run this one.

Go back to the original window where you are running aone and hit <CTRL>C to stop the original image and completely exit the original Docker container by entering **exit**. Close the browser page you were using to display the application <http://localhost:3050/>.

```
Docker Container # node server.js  
aOne listening on port 3050  
^C  
Docker Container # exit  
[oracle@demo-long-v4 ]$
```

In the terminal where you ran the Docker commit statement start your new Docker image.

```
sudo docker run -i -p 3050:3050 -t newaone sh
```

Once in the image go to the scripts directory. Notice that in the new image the directories are still the same structure, but you no longer have to edit the dbconfig.js file, as that change was committed to our new image.

Go to scripts directory

```
cd /opt/oracle/lib/ATPDocker/aone/scripts
```

Verify changes made to dbconfig.js in aone image were saved  
`cat dbconfig.js`

Go to aone directory where the node application resides  
`cd /opt/oracle/lib/ATPDocker/aone/`

Run node application

```
node server.js
```

```
[oracle@labtest8 lab10]$ sudo docker run -i -p 3050:3050 -t newaone sh
Docker Container # cd /opt/oracle/lib/ATPDocker/aone/scripts
Docker Container # cat dbconfig.js
module.exports= {
  user:"admin",
  password:"ATPwelcome-1234",
  connectString :"melatptrain01_high"
}
Docker Container # cd /opt/oracle/lib/ATPDocker/aone/
Docker Container # node server.js
aOne listening on port 3050
```

Open a new web browser window or tab and go to: <http://localhost:3050/>

You will see the application in the window, executing from your new container.

When you have finished with the Lab, hit Ctrl-C on your terminal window to end the application.

## Optional – Clean your Docker environment.

Once you have completed with this lab, you can clean up your docker environment. The following commands can be run from the terminal window. This will maintain your Docker installation, but remove any images created. You can always recreate the images above by starting at the docker build command.

```
sudo docker image prune
sudo docker container prune
sudo docker volume prune
sudo docker network prune
```

Congratulations, you have completed the Docker lab. You created a Docker container image and implemented a Node.js application within that container which accessed an ATP database. You have created a container from a generic image downloaded from a library and customized it to use your own ATP database credentials. You created a new Docker image using this customised container. This image could be deployed on any Docker container service and access the ATP cloud database from any internet connected location.

# Want to Learn More?

## Recommended Reading

- [Autonomous Transaction Processing Made Easy](#) (eBook)

## Additional Resources

- Additional Resources: [https://cloud.oracle.com/en\\_US/atp/additional-resources](https://cloud.oracle.com/en_US/atp/additional-resources)
- Demos and Videos: [https://cloud.oracle.com/en\\_US/atp/videos](https://cloud.oracle.com/en_US/atp/videos)
- [Oracle Autonomous Transaction Processing Documentation](#)
- Join the Autonomous Transaction Processing [forum](#).
- [Getting Started with Oracle Cloud](#)

## **Oracle** Autonomous Transaction Processing Hands-on Lab Guide

<https://docs.oracle.com/en/cloud/paas/atp-cloud/atpug/load-data.html>



# Appendix



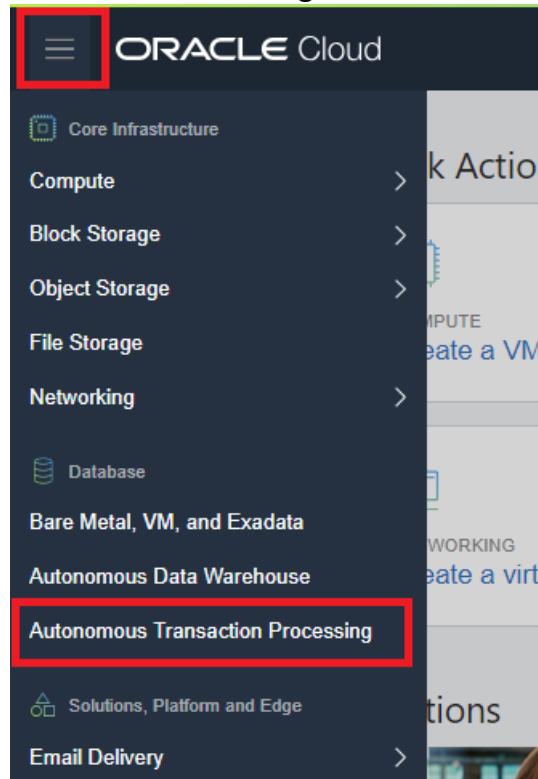
# Appendix A – Provisioning Process Walkthrough

This is a walkthrough of the process used to create an ATP instance. This was demonstrated live during your lab session.

## Creating your Autonomous Transaction Processing Database

Click on the **MENU** link at the top left of the page.

This will produce a drop-down menu, where you should select “**Autonomous Transaction Processing**”



This will take you to the management console page for ATP in the root compartment of the tenancy. The red warning icon "Forbidden" indicates that the Policies within the tenancy does not allow your user to create ATP Instances in the root compartment.

The screenshot shows the Oracle Autonomous Database console interface. On the left, there's a sidebar with 'List Scope' and 'Filters' sections. Under 'List Scope', 'COMPARTMENT' is set to 'oscemea001 (root)'. Below it is a search bar with placeholder text 'Don't see what you're looking for?'. The main area is titled 'Autonomous Databases in oscemea001 (root) Compartment'. It features a 'Create Autonomous Database' button. A table lists databases with columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. A single row is shown with a status of 'Forbidden' and a red exclamation mark icon. At the bottom right, there are links for 'No Autonomous Databases' and 'Page'.

To begin the process of creating your ATP instance in this tenancy you need to select a compartment.

Click on the pulldown menu marked **Compartment**. Expand the menu under ATP\_Workshop by clicking the '+'. Select the “**ATP\_Delegate**” compartment.

The screenshot shows the Oracle Autonomous Transaction Processing console. On the left, there's a sidebar with 'List Scope' and 'COMPARTMENT' section. The 'COMPARTMENT' dropdown is set to 'oscemea001 (root)', which is highlighted with a red box. Below it is a search bar with placeholder text 'Search compartments'. The main area shows a tree view of compartments under 'oscemea001 (root)'. 'oscemea001 (root)' is expanded, showing 'ATP\_Workshop' and 'ATP\_Delegate', which is also highlighted with a red box. Other visible compartments include 'ATP\_TTT' and 'ManagedCompartmentForPaaS'. At the bottom, there's a 'Filter state' dropdown.

*Note – Your list of compartments may be different to the one shown above. In these lab notes we will use the compartment called ATP\_Delegate in all the screenshots*

The main page will now change to show the list of ATP instances within your compartment, as shown below:

*Note: this lab uses the same tenancy and compartment for all lab attendees, therefore, it is possible that you may see ATP instances listed on this page which have already been created by other users attending this lab.*

To create a new instance, click the blue "Create Autonomous Database" button.

The screenshot shows a table titled "Autonomous Databases in ATP\_Delegate Compartment". The table has columns: Name, State, Database Name, CPU Core Count, Storage (TB), Workload Type, and Created. There are 14 rows of data, each representing a different database instance. A red box highlights the "Create Autonomous Database" button at the top left of the table area.

Autonomous Databases in ATP_Delegate Compartment						
List Scope		Name	State	Database Name	CPU Core Count	Storage (TB)
COMPARTMENT	ATP_Delegate	dd13ATP	Available	dd13ATP	1	1
	atp_maintenanceautodb (test) ATP_Workshop	dd24ATP	Available	dd24ATP	1	1
	ATP_Delegate	dd21ATP	Available	dd21ATP	1	1
	Don't see what you're looking for? <a href="#">(1)</a>	dd22ATP	Available	dd22ATP	1	1
FILTERS	DATE	dd21ATP	Available	dd21ATP	1	1
	Any state	dd20ATP	Available	dd20ATP	1	1
	WORKLOAD TYPE	dd19ATP	Available	dd19ATP	1	1
	ATP	dd18ATP	Available	dd18ATP	1	1
	dd17ATP	Available	dd17ATP	1	1	Transaction Processing
Tag Filters	dd16ATP	Available	dd16ATP	1	1	Transaction Processing
	dd15ATP	Available	dd15ATP	1	1	Transaction Processing
	dd14ATP	Available	dd14ATP	1	1	Transaction Processing

Enter the required information and click the "**Create Autonomous Transaction Processing Database**" button at the bottom of the form. For the purposes of this lab, use the information below:

**Workload Type:** Autonomous Transaction Processing

**Compartment:** Verify that the **ATP\_Delegate** compartment is selected. You will not be able to create instances outside this compartment due to Identity Management policies.

**Display Name:** Enter the display name for your ATP Instance. (Hint- use your username so you can identify your instance more easily)

**Database Name:** Enter any database name you choose that fits the requirements for ATP. The database name must consist of letters and numbers only, starting with a letter. The maximum length is 14 characters. (Hint- use your username so you can identify your instance more easily)

**CPU Count:** 1

**Storage Capacity (TB):** 1

**Administrator Password:** Enter any password you wish to use noting the specific requirements imposed by ATP. A suggested password for this lab is ATPwelcome-1234

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Create Autonomous Database help cancel

**Workload Type**

AUTONOMOUS DATA WAREHOUSE

AUTONOMOUS TRANSACTION PROCESSING  
Configures the database for a transactional workload, with a bias towards high volumes of random data access.

**Database Information**

COMPARTMENT: ATP\_Delegate

DISPLAY NAME: MELATPTRAIN01

DATABASE NAME: MELATPTRAIN01  
The name must contain only letters and numbers, starting with a letter. 14 characters max.

CPU CORE COUNT: 1      STORAGE (TB): 1  
The number of CPU cores to enable. Available cores are subject to your tenancy's service limits.  
The amount of storage to allocate.

**Administrator Credentials**

Set the password for your Autonomous Transaction Processing database ADMIN user here.

USERNAME: READ-ONLY  
ADMIN

PASSWORD:

CONFIRM PASSWORD:

**License Type**

MY ORGANIZATION ALREADY OWNS ORACLE DATABASE SOFTWARE LICENSES  
Bring my existing database software licenses to the database cloud service ([details](#)).  
 SUBSCRIBE TO NEW DATABASE SOFTWARE LICENSES AND THE DATABASE CLOUD SERVICE

**TAGS**

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

[Learn more about tagging](#)

TAG NAMESPACE: None (apply a free-form tag)      TAG KEY:       VALUE:   
+ Additional Tag

**Create Autonomous Database**

When you enter the administrator password, note the specific requirements imposed by ATP:

**Administrator Credentials**

Set the password for your Autonomous Transaction Processing database ADMIN user here.

USERNAME: READ-ONLY  
ADMIN

PASSWORD:   
•

>Password must be 12 to 30 characters and contain at least one uppercase letter, one lowercase letter, and one number. The password cannot contain the double quote ("") character or the username "admin".

CONFIRM PASSWORD:

Oracle Autonomous Transaction Processing Hands-on Lab Guide

When you have completed the required fields, scroll down and click on the blue **Create Autonomous Database** button at the bottom of the form:

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

[Learn more about tagging](#)

TAG NAMESPACE	TAG KEY	VALUE
None (apply a free-form tag)	<input type="text"/>	<input type="text"/>

+ Additional Tag

**Create Autonomous Database**

The console page will display the message “**Provisioning**” under the “State” column.

Autonomous Databases in ATP_Delegate Compartment								
Create Autonomous Database		Name	Status	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
comusername	ATP_Delegate	MEJATRDB1	Planning...	MEJATRDB1	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT
comusername	ATP_Delegate	ds01ATP	Available	ds01ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:13:00 GMT
comusername	ATP_Delegate	ds02ATP	Available	ds02ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:09:07 GMT
comusername	ATP_Delegate	ds03ATP	Available	ds03ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:05:06 GMT
comusername	ATP_Delegate	ds04ATP	Available	ds04ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:01:05 GMT
comusername	ATP_Delegate	ds05ATP	Available	ds05ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:18:14 GMT
comusername	ATP_Delegate	ds06ATP	Available	ds06ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:53:03 GMT
comusername	ATP_Delegate	ds07ATP	Available	ds07ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:49:02 GMT
comusername	ATP_Delegate	ds08ATP	Available	ds08ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:45:01 GMT
comusername	ATP_Delegate	ds09ATP	Available	ds09ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:41:00 GMT
comusername	ATP_Delegate	ds10ATP	Available	ds10ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:37:00 GMT
comusername	ATP_Delegate	ds11ATP	Available	ds11ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:33:00 GMT
comusername	ATP_Delegate	ds12ATP	Available	ds12ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 10:29:00 GMT

Click on the name of your ATP instance, as shown below

Autonomous Databases in ATP_Delegate Compartment							
Create Autonomous Database							
Name	Status	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created	
ATP_Delegate	Creating	MELAATPAND1	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT	
ATP_Delegate	Available	atp1447	1	1	Transaction Processing	Thu, 07 Mar 2019 10:13:00 GMT	
ATP_Delegate	Available	atp1447	1	1	Transaction Processing	Thu, 07 Mar 2019 10:09:07 GMT	
ATP_Delegate	Available	atp1447	1	1	Transaction Processing	Thu, 07 Mar 2019 10:05:07 GMT	
ATP_Delegate	Available	atp2247	1	1	Transaction Processing	Thu, 07 Mar 2019 10:03:00 GMT	
ATP_Delegate	Available	atp2147	1	1	Transaction Processing	Thu, 07 Mar 2019 10:18:14 GMT	
ATP_Delegate	Available	atp2047	1	1	Transaction Processing	Thu, 07 Mar 2019 10:15:33 GMT	
ATP_Delegate	Available	atp1947	1	1	Transaction Processing	Thu, 07 Mar 2019 10:05:00 GMT	
ATP_Delegate	Available	atp1847	1	1	Transaction Processing	Thu, 07 Mar 2019 10:03:05 GMT	
ATP_Delegate	Available	atp1747	1	1	Transaction Processing	Thu, 07 Mar 2019 10:01:50 GMT	
ATP_Delegate	Available	atp1647	1	1	Transaction Processing	Thu, 07 Mar 2019 10:18:34 GMT	
ATP_Delegate	Available	atp1547	1	1	Transaction Processing	Thu, 07 Mar 2019 10:18:34 GMT	
ATP_Delegate	Available	atp1447	1	1	Transaction Processing	Thu, 07 Mar 2019 10:20:47 GMT	

This will display more information about your instance and you should notice the various menu buttons that help you manage your new instance – because the instance is currently being provisioned all the management buttons are grayed out.

Autonomous Database - Autonomous Database Details

## MELATPTRAIN01

DB Connection Service Console Scale Up/Down Start Archive

Autonomous Database Information Tags

Workload Type: Transaction Processing  
Display Name: MELATPTRAIN01  
Database Name: MELATPTRAIN01  
CPU Core Count: 1  
Storage (TB): 1

Created: Fri, 08 Mar 2019 08:38:38 GMT  
Compartment: branchedatadb.clob (root)/ATL/Workshop/ATL\_DevStage  
OCID: oc1.us-phx-1afw-2001  
License Type: Bring Your Own License  
Lifecycle Stage: Provisioning...

Resources Backups

Backups are automatically created daily.

Create Manual Backup

Name	State	Type	Started	Ended
			No items found.	

(Showing 0 items) < Page 1 >

A summary of your instance status is shown in the large box on the left. In this example, the color is amber and the status is **Provisioning**



After a short while the status will change to **Available** and the “ATP” box will change color to green:



Once the Lifecycle Status is **Available**, additional summary information about your instance is populated, including the workload type. You can also see the Lifecycle Status reported in this region.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

**MELATPTRAIN01**

Autonomous Database Information Workload Type: Transaction Processing

Created: Fri, 08 Mar 2019 08:38:38 GMT  
Compartments: orclcdmelatptrain01 (root)/ATP\_Workshop/ATP\_Delegate  
OCID: a...-aut... [Show](#) [Copy](#)  
License Type: Bring Your Own License  
Lifecycle State: Available

**Resources**

**Backups**  
Backups are automatically created daily.

Name	State	Type	Started	Ended
No items found.				

Showing 0 item(s) < Page 1 >

Congratulations you have created your first ATP instance!

Return to the main page which list all your ATP instances by clicking on the Autonomous Database link at the top of the page:

**Autonomous Database** » Autonomous Database Details

**MELATPTRAIN01**

DB Connection Service Console Scale Up/Down

Autonomous Database Information Tags

**Workload Type:** Transaction Processing  
**Display Name:** MELATPTRAIN01  
**Database Name:** MELATPTRAIN01  
**CPU Core Count:** 1  
**Storage (TB):** 1

Autonomous Database

Autonomous Databases in ATP\_Delegate Compartment

Create Autonomous Database						
Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
MELATPTRAIN01	Available	MELATPTRAIN01	1	1	Transaction Processing	Fri, 08 Mar 2019 08:38:38 GMT
dd21ATP	Available	dd21ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:15:58 GMT
dd24ATP	Available	dd24ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:09:57 GMT
dd23ATP	Available	dd23ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:05:56 GMT
dd27ATP	Available	dd27ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 19:01:55 GMT
dd26ATP	Available	dd26ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:57:54 GMT
dd20ATP	Available	dd20ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:53:53 GMT
dd19ATP	Available	dd19ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:49:52 GMT
dd18ATP	Available	dd18ATP	1	1	Transaction Processing	Thu, 07 Mar 2019 18:45:51 GMT

You are now ready to start Lab 2.



# Appendix B – Creating and preparing a user to access Object Storage

## Information about how the environment was prepared

To load data from the Oracle Cloud Infrastructure Object Storage you will need a Cloud user with the appropriate privileges to read data from the Object Store. The communication between the database and the object store relies on the Swift protocol and a username/password authentication token.

This is an outline of the steps that your instructor carried out – your lab user does not have permission to do these operations, or to see all of the referenced objects.

Select/Create your object storage bucket.

Buckets *in ATP\_Workshop Compartment*

The screenshot shows a list of buckets in the ATP\_Workshop compartment. There is one bucket named "DEMO\_DATA". The bucket has a blue circular icon with a white letter "B" inside. To the right of the bucket name, it says "Created: Fri, 21 Dec 2018 11". At the top left, there is a "Create Bucket" button.

Upload the files to the object storage bucket.

DEMO\_DATA

The screenshot shows the details for the DEMO\_DATA bucket. At the top, there are buttons for "Change Compartment", "Update Visibility", "Delete", and "Apply Tag(s)". Below that, there are tabs for "Bucket Information" (selected) and "Tags". Under "Bucket Information", it shows the Namespace as "oscemea001", Storage Tier as "Standard", ETag as "984a1594-6465-4243-8c8a-3e4dde44353c", and Encryption Key as "None". It also mentions "Developer tools" for advanced operations. A note at the bottom says "Developer tools are available for advanced object operations."

Objects

The screenshot shows the list of objects in the DEMO\_DATA bucket. There are four files listed: "ATP-HOL-Long-v0.6.pdf", "channels.csv", "channels\_special.csv", and "countries.csv". At the top, there are buttons for "Upload Object" and "Restore Object".

Construct the URL that points to the location of the file staged in the OCI Object Storage. The URL is structured as follows

[https://swiftobjectstorage.<region\\_name>.oraclecloud.com/v1/<tenant\\_name>/<bucket\\_name>/<file\\_name>](https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/<tenant_name>/<bucket_name>/<file_name>)

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

Create a user, in this example, 'atp\_oss\_access' and associate the user with a group.

atp\_oss\_access

Description: ATP Workshop shared OSS access account

Create/Reset Password Edit User Capabilities Unblock Delete Apply Tag(s)

User Information Tags

OCID: ...aps47a Show Copy

Created: Fri, 21 Dec 2018 12:47:52 GMT

**Capabilities**

Local password: Yes  
API keys: Yes  
Auth tokens: Yes

### Groups

Add User to Group

G ATP\_Group OCID: ...aps47a Show Copy

Create a policy statement to allow the group to read the object storage bucket.

Allow group Atp\_group to read buckets in compartment Atp\_workshop

Create an Authentication Token (Auth Token) for this user. Make a note of it as you are only told this information once.

atp\_oss\_access

Description: ATP Workshop shared OSS access account

Create/Reset Password Edit User Capabilities Unblock Delete Apply Tag(s)

User Information Tags

OCID: ...aps47a Show Copy

Created: Fri, 21 Dec 2018 12:47:52 GMT

Status: Active  
Federated: No

**Capabilities**

Local password: Yes  
API keys: Yes  
Auth tokens: Yes

**Auth Tokens**

Generate Token

AT OCID: ...xxyyyq Show Copy Description: Token created for ATP workshops

To access data in the Object Store you must enable your database user to authenticate itself with the Object Store using your object store account and Auth Token.

You do this by creating a private CREDENTIAL object for your user that stores this information encrypted in your ATP instance using the DBMS\_CLOUD package. This encrypted connection information is only usable by your user schema.

```
set define off
begin
  DBMS_CLOUD.create_credential(
    credential_name => 'OBJ_STORE_CRED',
    username => 'atp_oss_access',
    password => '2Qn8uD}D-5fyAw6cZ9QO'
  );
end;
/
```

# Appendix C – Connecting to the database using Oracle Machine Learning Notebook

You can use the included Oracle Machine Learning OML Notebook based environment to connect to your ATP environment. OML is a browser-based environment, and so provides an easy to connect and fast environment to work with ATP. It can be used to run SQL queries and scripts, which can then be grouped together within a notebook. Notebooks can be used to build single reports, collections of reports and dashboards, and notebooks can be shared with other users.

## Creating an OML user

The OML Notebook requires you to create a user, as the ATP instance admin user is prohibited from creating notebooks and jobs.

Return to your Firefox window and open your ATP instance service console.

Select Administration and Manage Oracle ML Users

The screenshot shows the Oracle Autonomous Transaction Processing service console. On the left, there's a sidebar with 'Autonomous Transaction Processing' at the top, followed by 'Overview', 'Activity', and 'Administration'. The 'Administration' button is highlighted with a red box. Below the sidebar, it says 'DATABASE MELATPTTRAIN01'. The main content area has several sections: 'Download Client Credentials (Wallet)', 'Set Resource Management Rules', 'Set Administrator Password', 'Download Oracle Instant Client', and 'Send Feedback to Oracle'. The 'Manage Oracle ML Users' section is highlighted with a blue box. This section contains the text: 'Create new Oracle Machine Learning user accounts and manage the credentials for existing Oracle Machine Learning users.'

You will now see the User management page. Select **Create** to create your new user.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows the Oracle Machine Learning User Administration interface. At the top, it says 'ORACLE Machine Learning User Administration'. Below that is a header with 'Users' and buttons for '+ Create', 'Delete', and 'Show All Users'. There is also a search bar and a status indicator. The main area is a table with columns: User Name, Full Name, Role, Email, Created On, and Status. One row is shown: 'ADMIN' with 'System Administrator' role, 'melanie.ashworth-march@oracle.com' email, '11/25/18 4:42 PM' created on, and 'Open' status.

Enter the required information and click **Create**. For the purposes of this lab, use the information below:

**Username:** *omluser1*

**Email Address:** Enter a valid email address

**Uncheck the generate password button.** You can optionally have a secure password emailed to the email address and then reset it on first login. As access to your email is not guaranteed, to speed up this lab we are going to manually enter a password.

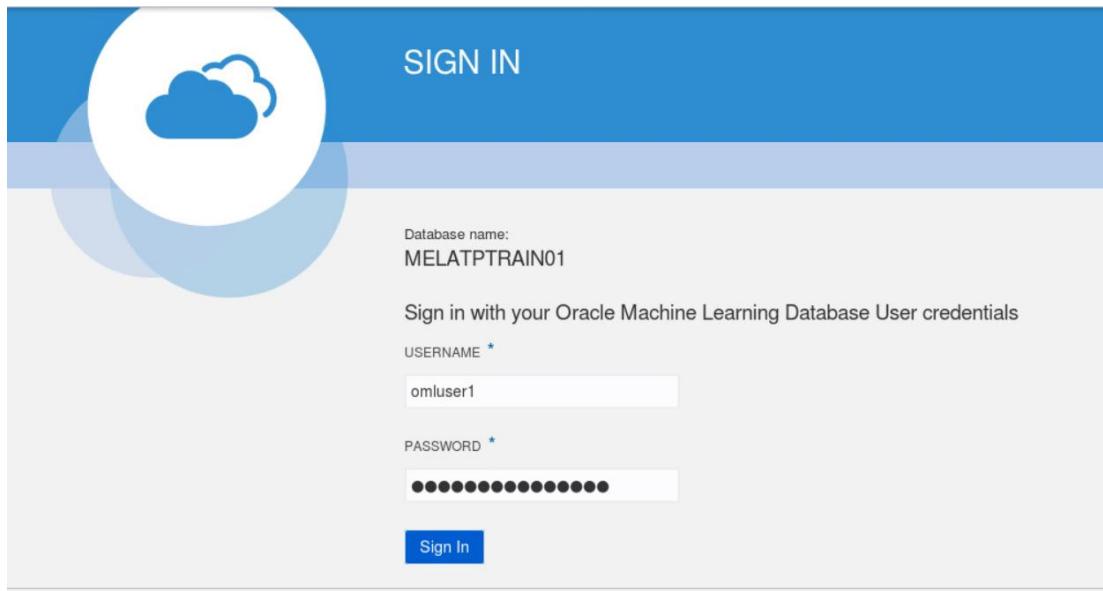
**Password:** Enter any password you wish to use noting the specific requirements imposed by ATP. A suggested password for this lab is ATPwelcome-1234

The screenshot shows the 'Create User' dialog box. It has fields for Username ('omluser1'), First Name, Last Name, and Email Address ('melanie.ashworth-march@oracle.com'). There is a checkbox for generating a password and account details, which is unchecked. Below that are fields for Password and Confirm Password, both containing 'ATPwelcome-1234'. At the bottom right are 'Create' and 'Cancel' buttons, with 'Create' highlighted by a red box.

Connect as your new OML user by selecting the Home Icon (shaped like a house) on the Users screen

The screenshot shows the Oracle Machine Learning User Administration interface again. At the top, it says 'User Created'. Below that is a header with 'Users' and buttons for '+ Create', 'Delete', and 'Show All Users'. There is also a search bar and a status indicator. The main area is a table with columns: User Name, Full Name, Role, Email, Created On, and Status. Two rows are shown: 'ADMIN' with 'System Administrator' role, 'melanie.ashworth-march@oracle.com' email, '11/25/18 4:42 PM' created on, and 'Open' status; and 'OMLUSER1' with 'Developer' role, 'melanie.ashworth-march@oracle.com' email, '1/3/19 4:37 PM' created on, and 'Open' status. A red box highlights the home icon in the top right corner of the header.

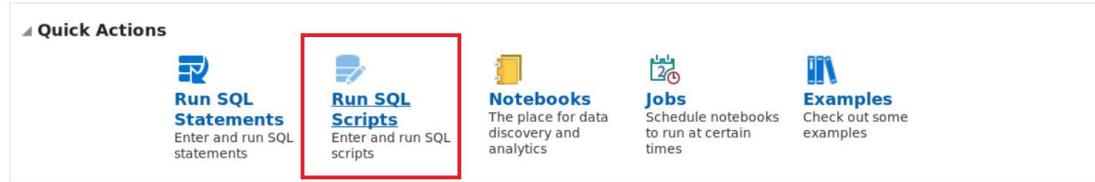
This will open a new browser tab. Sign in as **omluser1**



You are now connected as an OML Notebook user.

Now we will use the OML SQL Scripts function to run the same simple query we ran in SQL Developer.

Select **Run SQL Scripts** to open the SQL Script Scratchpad.



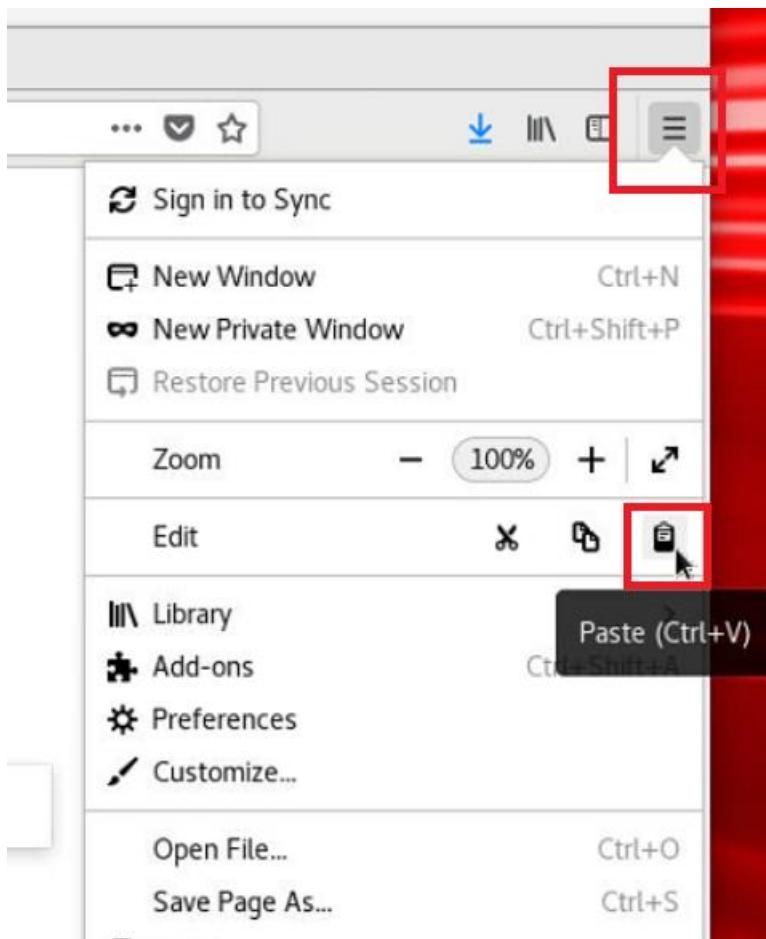
*Note: As this is the first time the Notebook Server has been started in this ATP instance, sometimes it will fail to start first time with the message 'Notebook server not available, please try again at a later time' Just try the Run SQL Scripts button again and it should start successfully.*

**Notebook server not available, please try again at a later time**

Copy this SQL statement and paste it in the gray SQL script box.

```
select country_region, count(country_name) from sh.countries group by country_region;
```

If you are having a problem pasting data into Firefox there is a 'Paste' menu item in the Firefox Menu.



Select the **Run all Paragraphs** icon to execute the script.



Select **ok** on the **Run all paragraphs** confirmation screen. The results of the query will be displayed.

## Oracle Autonomous Transaction Processing Hands-on Lab Guide

The screenshot shows the Oracle Machine Learning interface. At the top, it says "ORACLE Machine Learning" and "OMLUSER1 Project [OMLUSER1 Work...]" with a "Connected" status. Below that is the "SQL Script Scratchpad" section. It contains a code editor with the following SQL script:

```
%script
select country_region, count(country_name) from sh.countries group by country_region;
```

And the results of the query:

COUNTRY_REGION	COUNT(COUNTRY_NAME)
Asia	5
Middle East	1
Europe	10
Americas	4
Oceania	2
Africa	1

At the bottom of the scratchpad, it says "6 rows selected." and "Total 4 sec. Last updated by OMLUSER1 at January 04 2019, 1:56:07 PM." To the right of the scratchpad, there's a "READY" status bar.

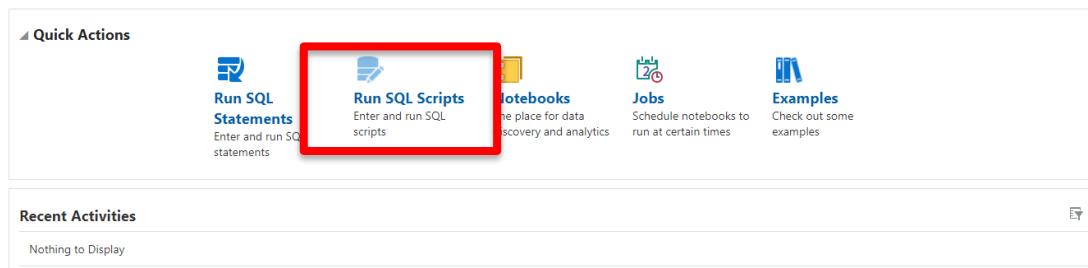
You have successfully connected and run an operation against ATP with Oracle OML.

### Selecting the Service in OML

OML makes it easier to do this because Notebooks automatically “see” all services available in the instance. Notebooks also allow re-running the same query multiple times without having to make any changes or re-posting the query.

Your OML session should still be open from the previous section, if not connect to OML as your omluser1 session.

If not already in **Run SQL Scripts**, select it from Quick Actions:



Before running a query, explore the different services that can be used to run the query by selecting the **Interpreter binding** button (next to the **default** drop down button) which as shown below:

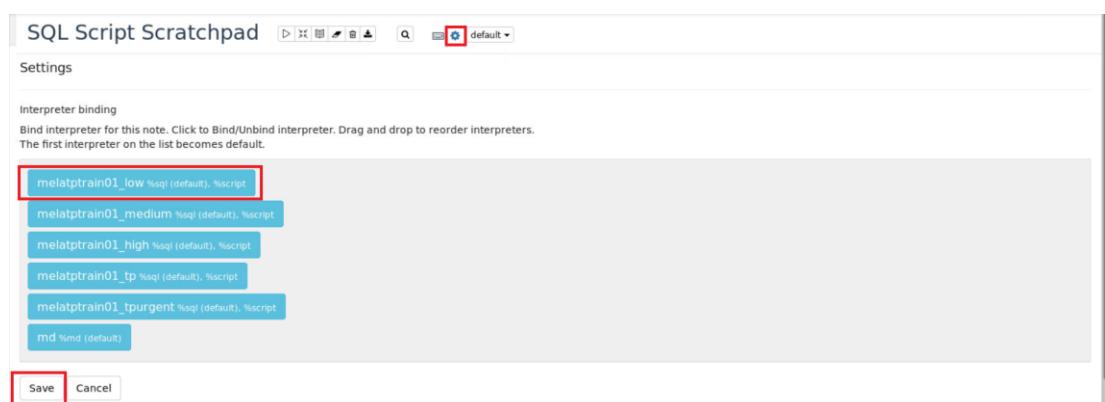


A new pane opens listing all the services available to run queries. You will see the five services already discussed previously and a new one specific to OML (md). The services are selected in list order, with the service at the top of the list being tried first. OML will try to connect to each selected service in order, and will then execute the SQL against the first successfully connected service.

The first time you run a query against a service, OML will establish a database connection. This will be reflected in the total execution time, which could include several additional seconds for the new connection. With a short query this will substantially add to the total execution time.

A blue color indicates the service is selected and will be tried, if you click on any of the services it will turn white, which de-selects it. You can drag and drop any service up or down the list to specify the order in which they will be attempted.

For the first test we will use the **\_LOW** service. If it is not at the top of the list, drag and drop it to the top of the list. Select **Save** when done.



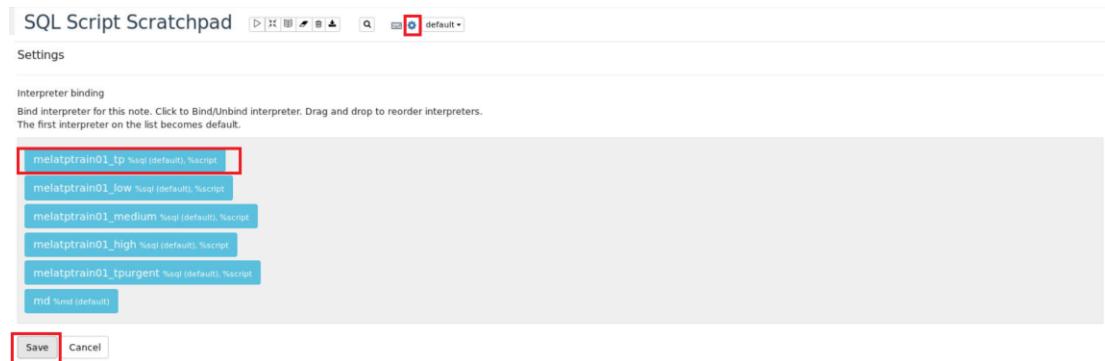
Paste this query into OML. The sql can be found in the file  
**/home/oracle/labScripts/lab2/lab2\_oml\_countries.sql**

```
Select /*+ no_result_cache */ count(*), co.country_name,
co.country_region
From sh.countries co, sh.customers cu
where co.country_id=cu.country_id
group by co.country_region, co.country_name
order by co.country_region;
```

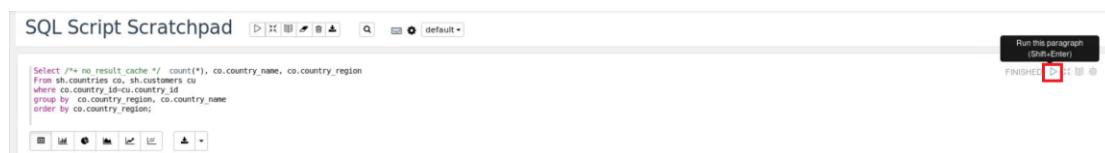
Now run the query by selecting the **Run This Paragraph** button



Now run the query in the **\_tp** service. Follow the steps above for changing the service by selecting the **Interpreter binding** button and when the list of services appears drag the **\_tp** service to the top and **Save**:



Now run the exact same query as before by selecting the **Run This Paragraph** button. As this is the first connection to the **\_tp** service the elapsed run time for this SQL will be slower, and it will include the time taken to make the initial connection.



The number of CPUs assigned to your instance, the other workloads running and the amount of parallelization that your query requires will affect how much impact selecting the different built in services has on your query run time.



Integrated Cloud Applications & Platform Services

**ORACLE**

Oracle Corporation, World Headquarters  
500 Oracle Parkway  
+1.650.506.7000  
Redwood Shores, CA 94065, USA

Worldwide Inquiries  
Phone:  
Fax: +1.650.506.7200

---

CONNECT WITH US

-  [blogs.oracle.com/oracle](http://blogs.oracle.com/oracle)
-  [facebook.com/oracle](http://facebook.com/oracle)
-  [twitter.com/oracle](http://twitter.com/oracle)
-  [oracle.com](http://oracle.com)

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. This document is provided *for* information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0519