Oracle Autonomous Database

Hands-on Workshop

Ver 11. (04/17/19)

Contents

0. Workshop Overview

Introducing Oracle Autonomous Database

Oracle redefines data management with the world's first autonomous database. **Oracle Autonomous Database** eliminates complexity, human error, and manual management, helping to ensure higher reliability, security, and more operational efficiency at the lowest cost. Compared to traditional database technology, an autonomous database cloud has greater availability greater security, and lower operating costs.

Other industry terms for autonomous database are self-driving database, self-repairing and self-securing.

- **Self-driving** means that the database can automatically provision, or deploy databases; and monitor, back-up, recover, and troubleshoot those databases. It also means to instantly grow and shrink compute or storage without downtime.
- **Self-securing** refers to adaptive AI-enabled threat detection and remediation, along with automatic data encryption. A self-security database can also apply security patches automatically.
- **Self-repairing** databases are automatically protected from downtime. With up to 99.995 percent availability, a self-repairing database experiences less than 2.5 minutes of downtime per month, including planned maintenance.

With Oracle Autonomous Database you do not need to configure or manage any hardware or install any software. Autonomous database handles database creation at the push of a button, database backups, patching and upgrading the database, and growing or shrinking the database.

Oracle Autonomous Database is built upon the Oracle Database, so that all applications and tools that support Oracle Database also support Oracle Autonomous Database. These tools and applications connect to autonomous database using standard SQL*Net connections. The tools and applications can either be in your data center or in a public cloud. Oracle Analytics Cloud and other Oracle Cloud services are preconfigured for autonomous database.

Optimized for Workload Types

Modern automobiles are specialized by workload: family car, van, pickup truck, sports car, etc. In the same way, the Autonomous Database consists of a single set of technologies available in multiple products, each tailored to a different workload:

Oracle Autonomous Data Warehouse

Autonomous Data Warehouse is a fully managed database tuned and optimized for data warehouse workloads with the market-leading performance of Oracle Database. As a data warehouse developer, business user, or data scientist, Autonomous Data Warehouse lets you use all your existing data warehouse design, data integration, analysis, and reporting tools.

Oracle Autonomous Transaction Processing

Autonomous Transaction Processing is designed to run mission-critical enterprise applications, including mixed workloads and real-time analytics, with no compromise on application performance. It provides a high-performance Oracle Database in an environment that is tuned and optimized for transaction processing workloads.

This workshop walks you through all the steps to get started using **Oracle Autonomous Database**.

Workshop Objectives

The Oracle Autonomous Database hands-on workshop is focused on the following (some topics are dependent on whether the workshop is Transaction Processing focused or Data Warehouse focused):

- **Provisioning and Connectivity**: Learn to provision a new autonomous database and connect your favorite client tools.
- **Management and Monitoring**: Learn to Manage and Monitor the service, and scale dynamically to experience the elasticity of autonomous database.
- **Data Loading and Integration**: Learn to load data into the autonomous database and integrate with data residing in Cloud storage or other RDBMs (e.g. using Oracle Data Sync).
- **Migration to Autonomous Database**: Migrate Oracle Databases on premise or Cloud to the autonomous database.
- **Build Applications using Autonomous Database (ATP)**: Learn to use your favorite development environments to connect and build applications using the autonomous database.

This is an instructor-led workshop, please follow the guidance from the instructor before attempting the lab exercises.

Lab Environment Setup

Cloud Accounts

Obtain your access to the following Cloud Services and lab accounts. Your instructor will provide this information.

- Oracle Cloud Account (includes Oracle Autonomous Database):
  - **URL :**
  - **Cloud Tenant :**
  - **User Name :**
  - **Password :**
  - **Cloud Region :**
- Lab VM Account:
  - **Remote Desktop IP :**
  - **User Name :**

  - **Password :**
- OCI Compartment to Create Autonomous Services
  - **ADB Compartment :**

Required Software

A lab VM hosted in Oracle Cloud Infrastructure is provided to you to run the hands-on lab exercises. However, to connect to the lab VM, you would need the following software:

For Microsoft Windows hosts:

- **Windows 10, Server 2012 R2, 2016**
  - Install the **Remote Desktop App** from the Microsoft Store from the link:
    - https://www.microsoft.com/en-us/store/p/microsoft-remote-desktop/9wzdncrfj3ps
- **Windows 8**
  - Use the built-in **Remote Desktop Connection** application with the following Hotfix:
    - http://www.microsoft.com/en-us/download/confirmation.aspx?id=35387
- **Windows 7, Server 2008 R2**
  - Use the built-in **Remote Desktop Connection** application with the following Hotfix:
    - https://support.microsoft.com/en-us/help/2923545/update-for-rdp-8-1-is-available-for-windows-7-sp1

For macOS, Mac OS X hosts:

- Install **Microsoft Remote Desktop 10** from the App Store

About the Lab VM

The lab VM is a preconfigured virtual machine that is available to you to assist with the lab exercises. Without the lab VM, you would need to install quite a few software components which would take valuable time away from the class.

The lab VM is hosted in the Oracle Compute Cloud and you would connect to it from your laptop/desktop using the RDP protocol.

The lab VM includes some of your favorite client tools and other required software preinstalled to help you complete the labs. Below is a partial list of software packages that are preinstalled and configured in the VM:

- Oracle SQL Developer
- Oracle Database Client
- Swingbench load generator

- Node.js & Docker

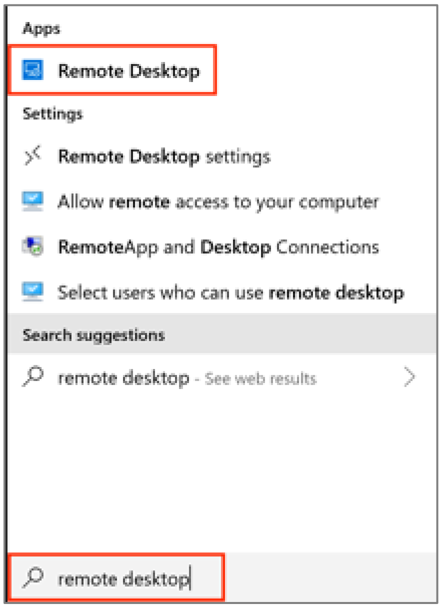
Lab VM Setup Steps

Perform the following setup before starting the lab exercises.

STEP 1: Connect to the Lab VM

- To access the lab VM, start **Microsoft Remote Desktop 10** App on macOS, or **Remote Desktop Connection** or **Remote Desktop** for Windows.

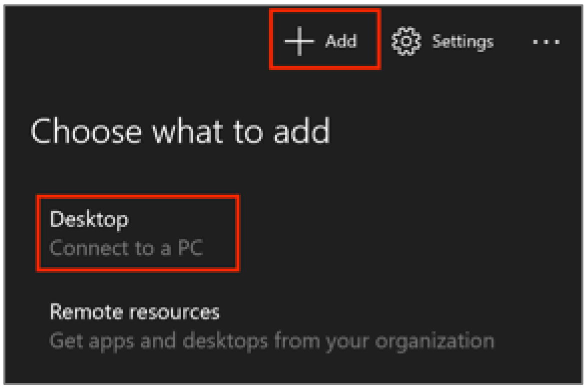
Windows


Apps
Remote Desktop
Settings
Remote Desktop settings
Allow remote access to your computer
RemoteApp and Desktop Connections
Select users who can use remote desktop
Search suggestions
remote desktop - See web results
remote desktop

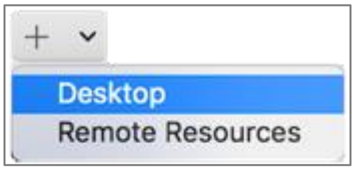
Mac

- Add a new connection for the lab VM. If this is your first time using the App, click on **Add Desktop**, or you may click on **(+)** then **Desktop**.

Windows

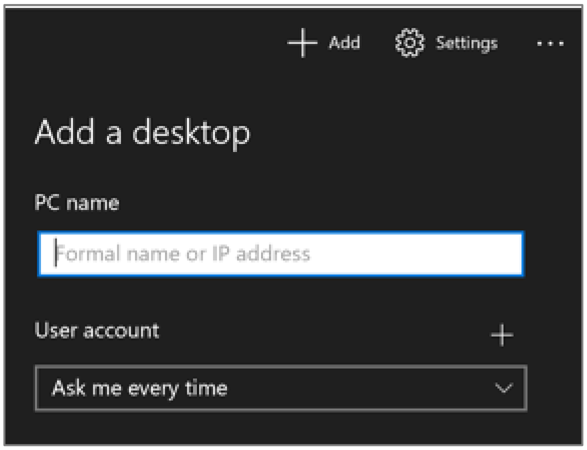


Mac



- On the **Add Desktop** pop-up, enter the **Remote Desktop IP** or the domain name of the lab VM as **PC Name** and.

Windows **Mac**



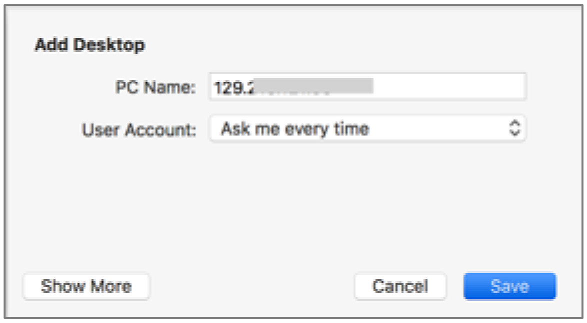
Add a desktop

PC name

Formal name or IP address

User account

Ask me every time

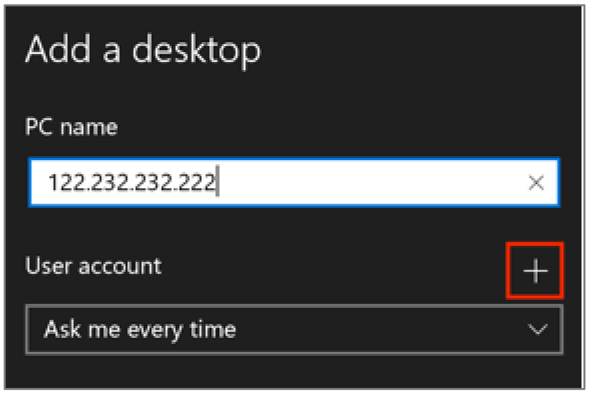

Add Desktop

PC Name: 129.2[redacted]

User Account: Ask me every time

- Click on **(+)** to add a user account, or the **User Account** and select **Add User Account**.

Windows **Mac**

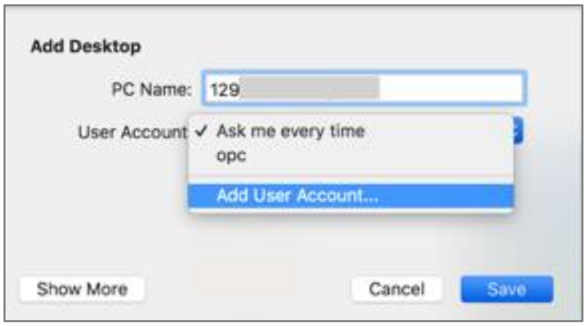

Add a desktop

PC name

122.232.232.222

User account

Ask me every time

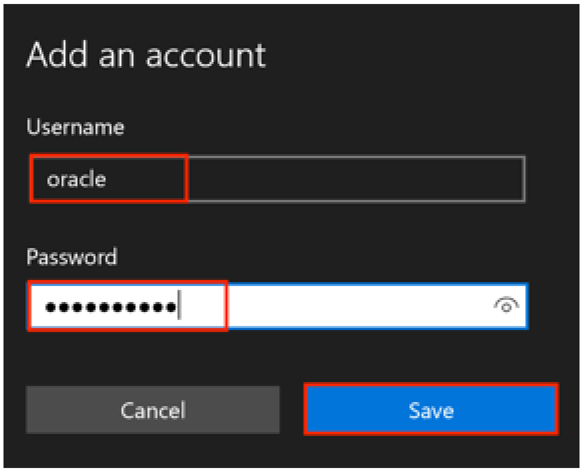

Add Desktop

PC Name: 129[redacted]

User Account

- Enter **User Name** and **Password** provided to you for the lab VM. Click **Save**.

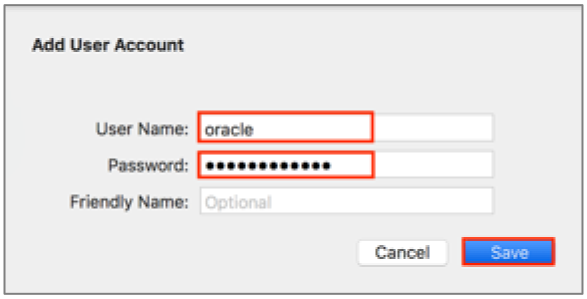
Windows **Mac**



Add an account

Username: oracle

Password: ••••••••••



Add User Account

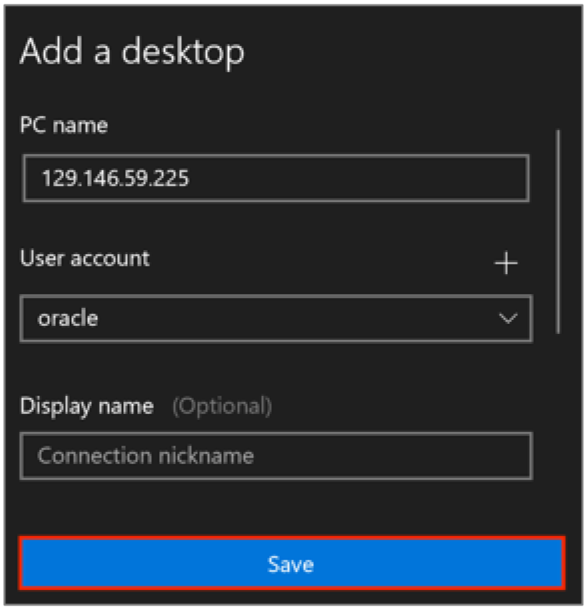
User Name: oracle

Password: ••••••••••••

Friendly Name: Optional

- Click **Save** one more time to Save the connection.

Windows **Mac**

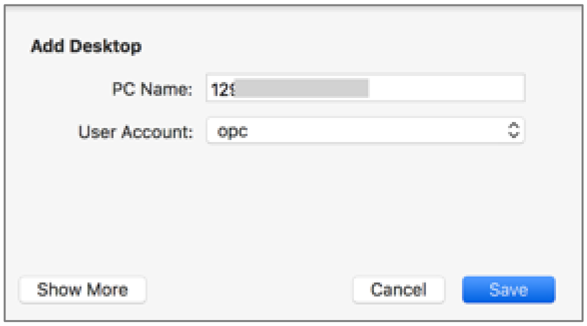


Add a desktop

PC name: 129.146.59.225

User account: oracle

Display name (Optional): Connection nickname



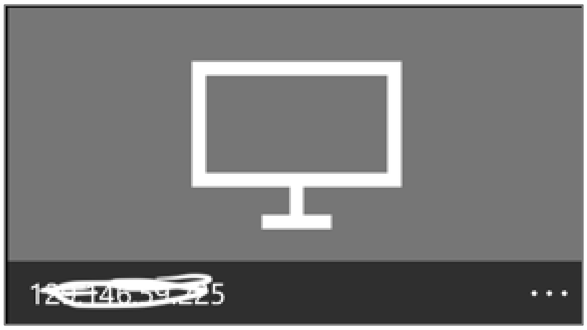
Add Desktop

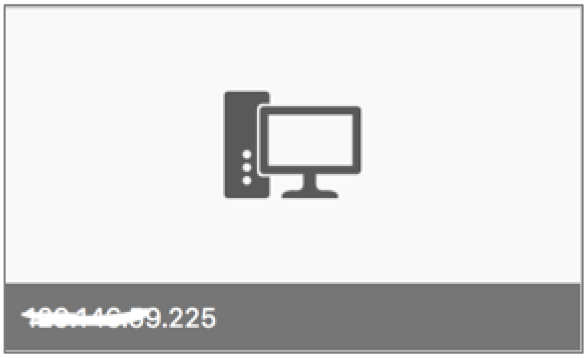
PC Name: 12[redacted]

User Account: opc

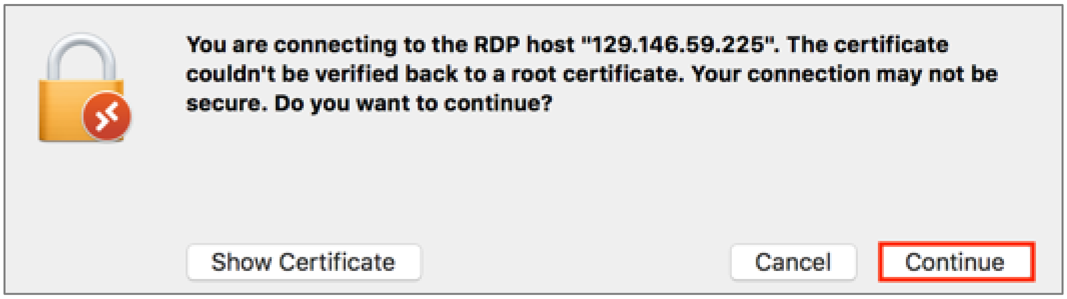
- Connect to the lab VM by **Double Clicking** the connection.

Windows **Mac**





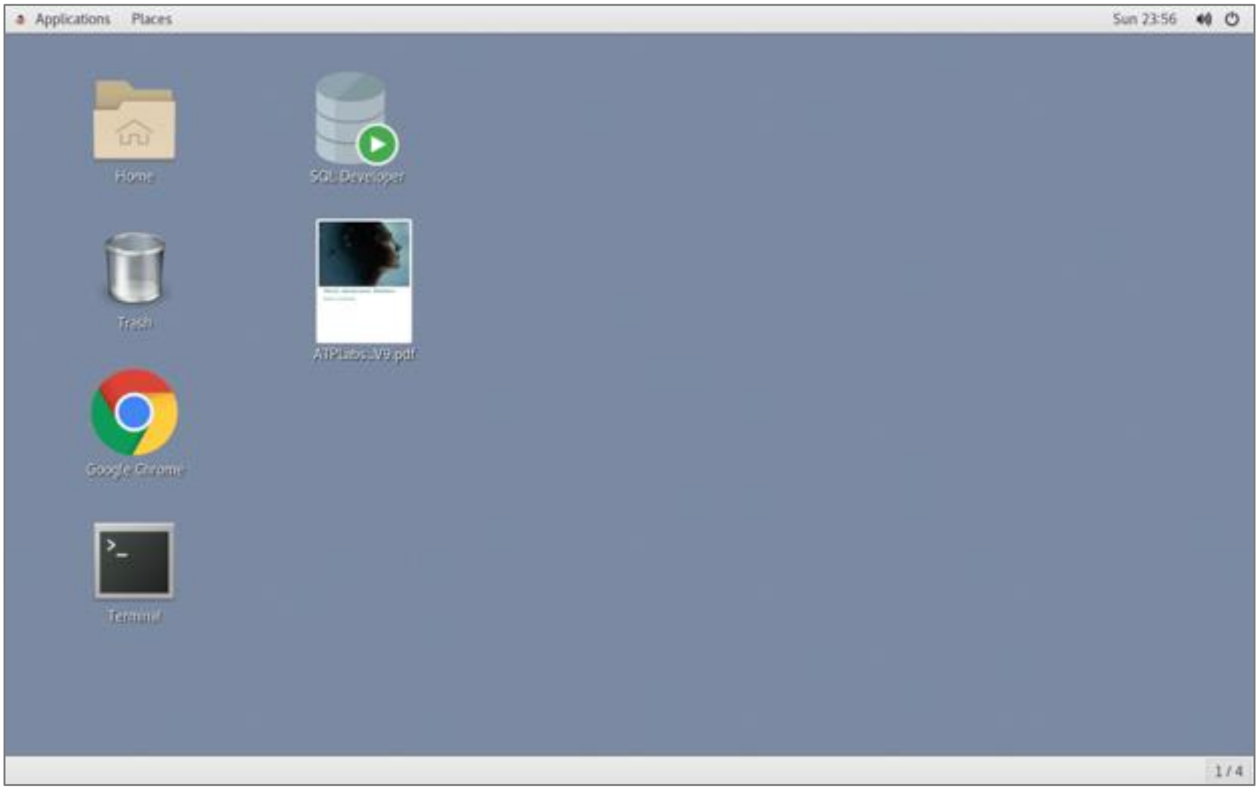
- Click **Continue** when prompted that the certificate couldn't be verified.



You are connecting to the RDP host "129.146.59.225". The certificate couldn't be verified back to a root certificate. Your connection may not be secure. Do you want to continue?

- You should now be connected to the VM and the following desktop should be displayed.

Note: If you have issues connecting, check if your firewall is blocking port **3389** as the Remote Desktop connection uses this port. This is usually the case when you are connected to your corporate VPN.



- If you see the above desktop, you have successfully connected to the lab VM.

Optional Downloads

Ideally, you should use the lab VM to run all labs. But you may be able to run some labs directly from your Desktop, for example, labs that only require only a Web browser. Therefore, it may be beneficial to have a local copy of the lab manual and/or the lab support files on your desktop. Please download the files from the locations below.

Lab Manual

https://objectstorage.us-ashburn-1.oraclecloud.com/n/oraclepartnersas/b/ATPWorkshop/o/ATPLabs.pdf

Lab Support Files

https://objectstorage.us-ashburn-1.oraclecloud.com/n/oraclepartnersas/b/ATPWorkshop/o/ATPLabfiles.zip

1. Provisioning and Connectivity

Lab 1-1: Sign-In to Oracle Cloud Infrastructure Console

This lab walks you through the steps to log in to the **Oracle Cloud Infrastructure** console.

Objectives

- Learn to sign-in to Oracle Cloud Infrastructure console.

Required Artifacts

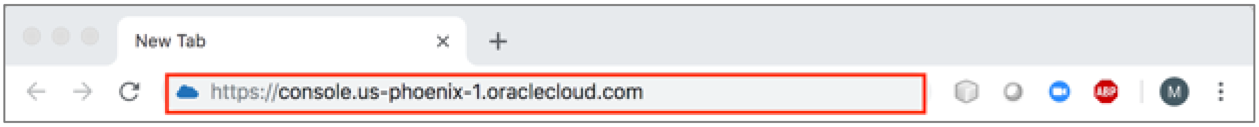
- An active Oracle Public Cloud account. You may use your own cloud account or a cloud account that you obtained through a trial or the lab account provided by the instructor.
- If you are using a laptop or a desktop, ensure that you have a supported version of web browser for Oracle Cloud. Alternatively, you may use the lab VM that has a supported browser pre-installed.

Lab Steps

STEP 1: Sign-In to Oracle Cloud Infrastructure Console

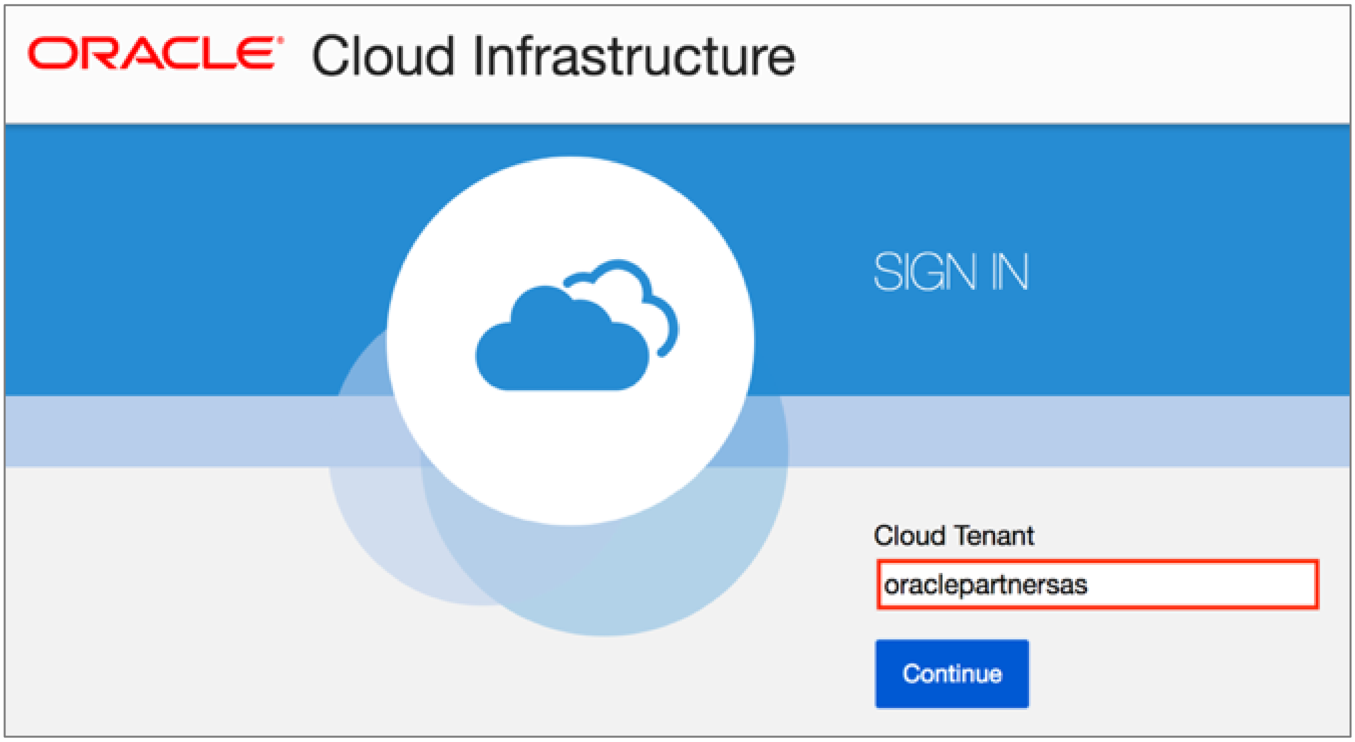
- Using a web browser on your laptop/desktop or the lab VM, browse to the ADB Cloud Account URL provided to you by your instructor.

  In the screenshot below, the URL assigned is https://console.us-phoenix-1.oraclecloud.com (yours may be different so please check your lab handout).



- On the **Oracle Cloud Infrastructure Sign In** page, enter the **Cloud Tenant** assigned to you and click **Continue**.

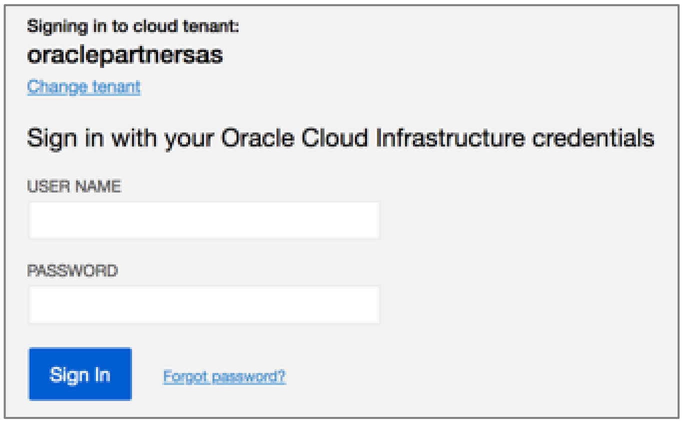
  In the screenshot below, the Cloud Tenant assigned is **oraclepartnersas** (yours may be different so please check your lab handout).



Cloud Tenant: oraclepartnersas

- You will be presented one of the following options to login, depending on whether the authentication is federated to an external identity provider such as Oracle Identity Cloud Services, or if it is done locally by Oracle Cloud Infrastructure.

  Choose to **Sign In** with your **Oracle Cloud Infrastructure** credentials as the lab accounts are not federated.



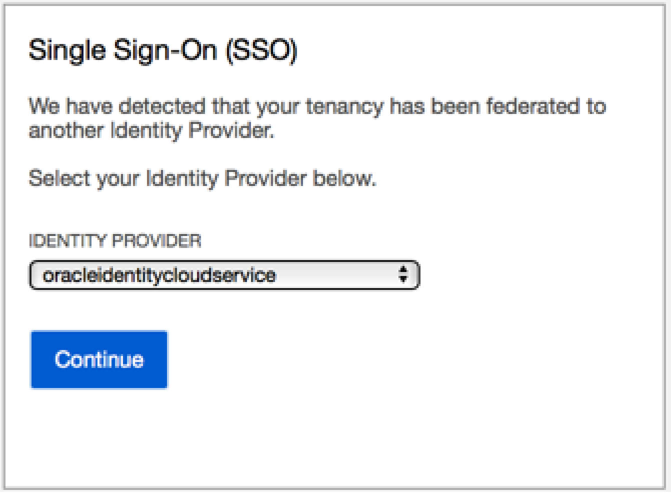
Signing in to cloud tenant:
oraclepartnersas

Sign in with your Oracle Cloud Infrastructure credentials

USER NAME

PASSWORD

(or)



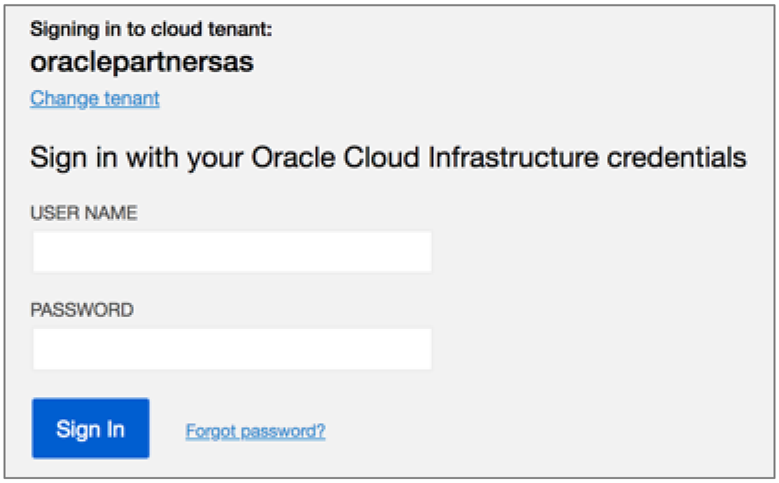
Single Sign-On (SSO)

We have detected that your tenancy has been federated to another Identity Provider.

Select your Identity Provider below.

IDENTITY PROVIDER: oracleidentitycloudservice

- Enter your Oracle Cloud **User Name** and **Password** supplied to you by the instructor and click **Sign In.**



Signing in to cloud tenant:

oraclepartnersas

Sign in with your Oracle Cloud Infrastructure credentials

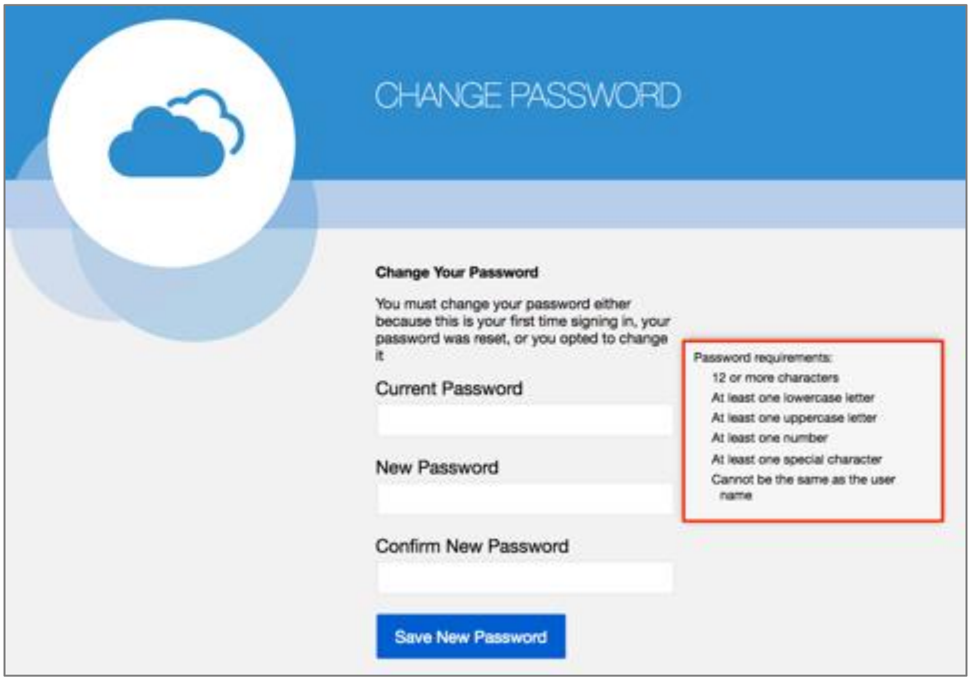
USER NAME

PASSWORD

- **Optionally**, you may be prompted to change the password, especially if you are logging in for the first time or after a password reset.

  Enter the **Current Password** supplied to you by the instructor, the **New Password**, **Confirm New Password** and click **Save New Password**.

Note the password requirements in the screenshot below.



CHANGE PASSWORD

Change Your Password

You must change your password either because this is your first time signing in, your password was reset, or you opted to change it

Current Password

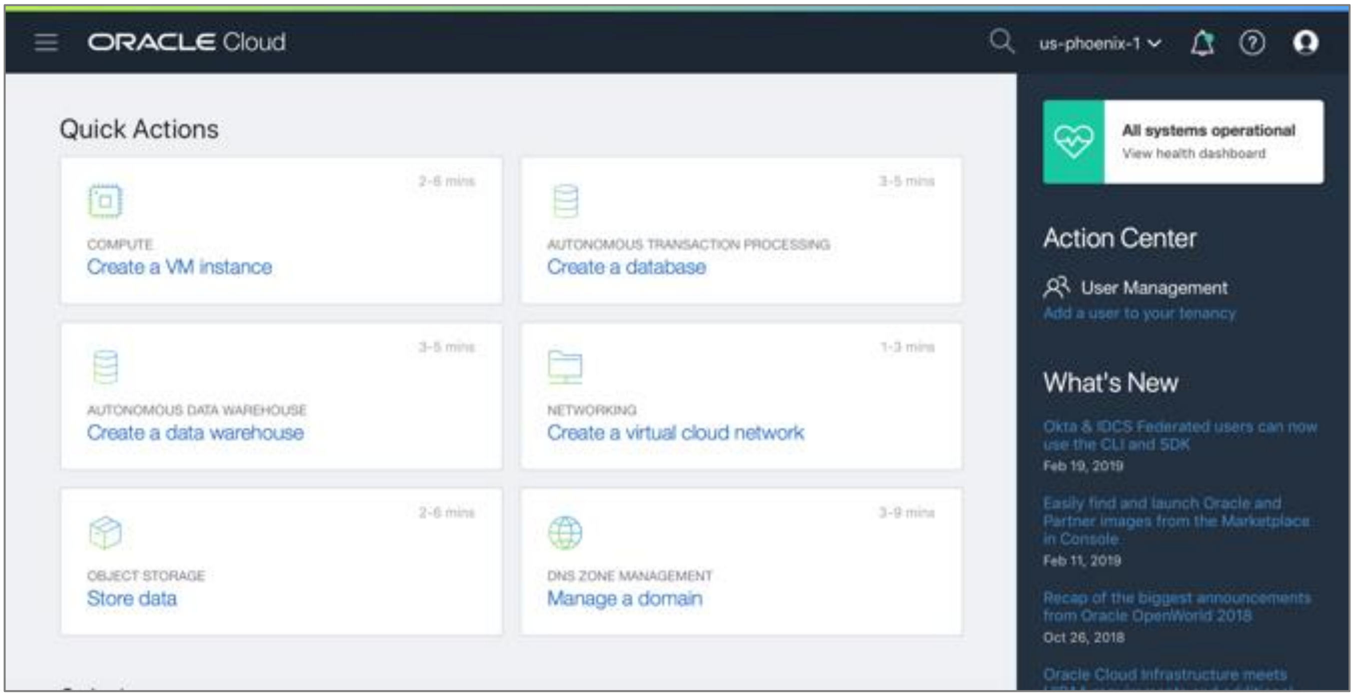
New Password

Confirm New Password

Password requirements:

- 12 or more characters
- At least one lowercase letter
- At least one uppercase letter
- At least one number
- At least one special character
- Cannot be the same as the user name

- Upon a successful login you will be presented **OCI Console Home Page**. Notice that **Quick Actions**, **Solutions** and **Learn** sections in the main pane and **Action Center** on the right.



Okta & IDCS Federated users can now use the CLI and SDK
Feb 19, 2019

Easily find and launch Oracle and Partner images from the Marketplace in Console
Feb 11, 2019

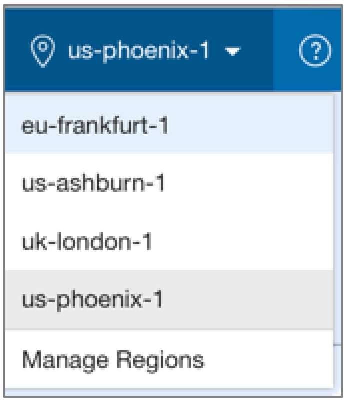
Recap of the biggest announcements from Oracle OpenWorld 2018
Oct 26, 2018

Oracle Cloud Infrastructure meets

STEP 2: Select your Cloud Data Center Region

- From the **Region** drop-down menu on the top, select the **Cloud Region** assigned to you by your instructor (check your lab handout).

IMPORTANT: The drop-down displays all **Cloud Data Center** regions that are assigned to your account. Ensure that you are in the correct region at all times as you work through the labs.



- You have successfully logged into the **Oracle Cloud Infrastructure** console.

Lab 1-2: Provisioning an Autonomous Transaction Processing Database

This lab walks you through the steps to create a new **Oracle Autonomous Transaction Processing** database.

Objectives

- Learn to provision a new Oracle Autonomous Transaction Processing database.

Required Artifacts

- An active Oracle Public Cloud account. You may use your own cloud account or a cloud account that you obtained through a trial or the lab account provided by the instructor.

- If you are using a laptop or a desktop, ensure that you have a supported version of web browser for Oracle Cloud. Alternatively, you may use the lab VM that has a supported browser pre-installed.

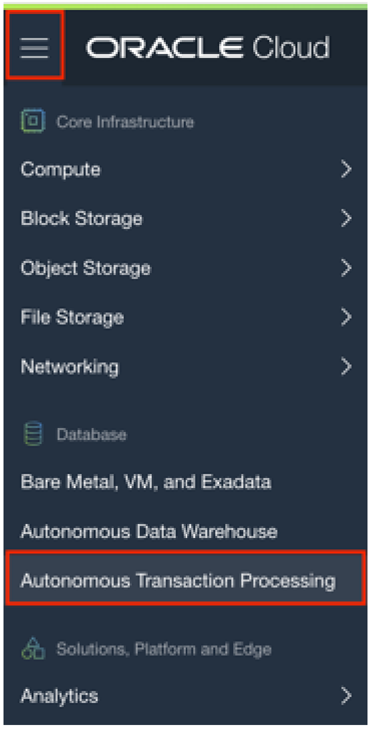
Lab Steps

STEP 1: Sign-In to Oracle Cloud Infrastructure Console

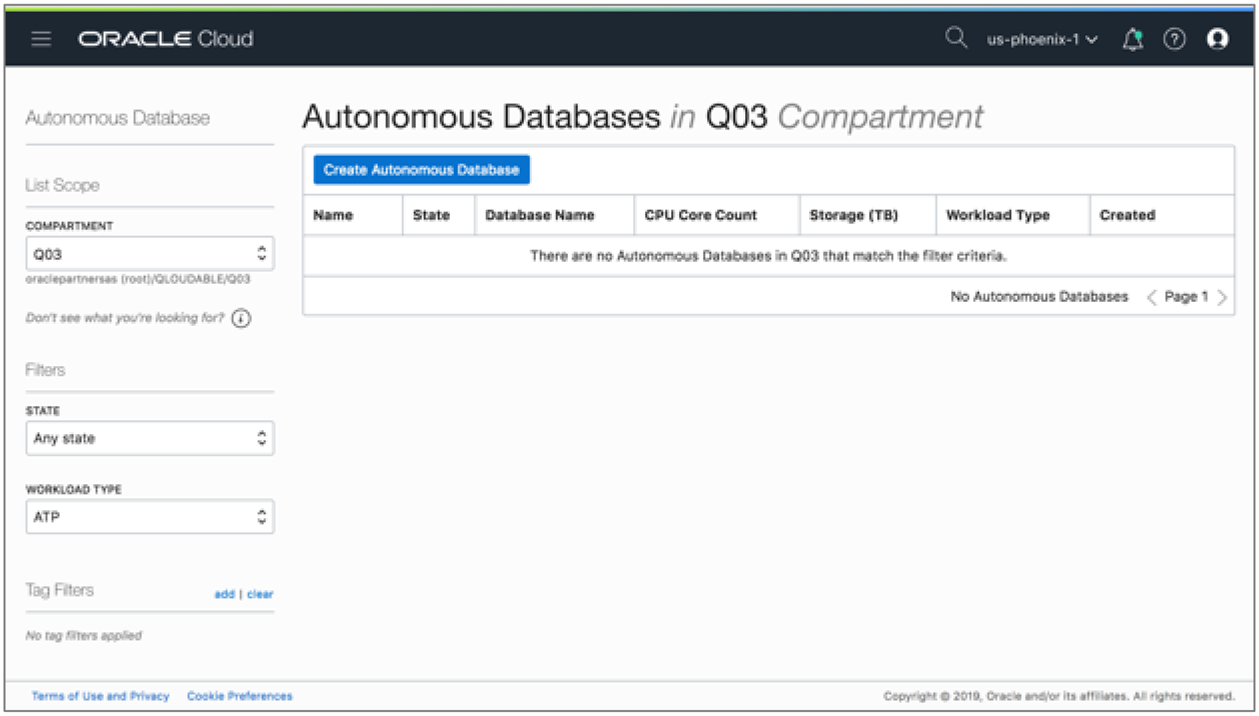
- Sign-in to Oracle Cloud Infrastructure console by following the steps outlined in **Sign-In to Oracle Cloud Infrastructure Console** lab.

STEP 2: Browse to Autonomous Database Home Page

- From the **OCI Console Home Page**, browse to **Autonomous Database Home Page** by clicking the hamburger **Menu** on the top-left and selecting **Autonomous Transaction Processing** from the **Database** section.


ORACLE Cloud
Core Infrastructure
Compute
Block Storage
Object Storage
File Storage
Networking
Database
Bare Metal, VM, and Exadata
Autonomous Data Warehouse
Autonomous Transaction Processing
Solutions, Platform and Edge
Analytics

- You will be presented the **Autonomous Database Home Page** where you will see the list of **ADB Databases** created in your compartment and for the **Region** selected.



Autonomous Databases *in* Q03 *Compartment*

COMPARTMENT: Q03

oraclepartnersas (root)/QLOUDABLE/Q03

Don't see what you're looking for?

STATE: Any state

WORKLOAD TYPE: ATP

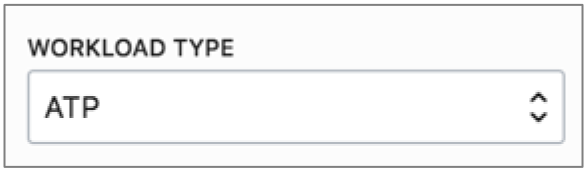
No tag filters applied

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
There are no Autonomous Databases in Q03 that match the filter criteria.						

No Autonomous Databases Page 1

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

- Verify the **Workload Type** is **ATP**.

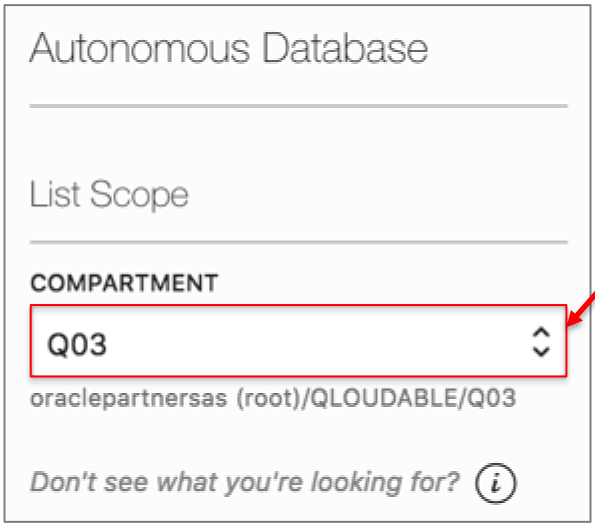


WORKLOAD TYPE: ATP

- Oracle Cloud Infrastructure allows logical isolation of users within a tenant through **Compartments**. This allows multiple users and business units to share a tenant account while being isolated from each other. More information about Compartments and Policies is provided in the OCI Identity and Access Management documentation here.

- From the **Compartment** drop-down menu on the left, choose the pre-created compartment assigned to you by your instructor (check your lab handout). Again, ensure that you are in the correct **Region** and the assigned **Compartment** at all times, especially when you work through the lab steps.

Note: The below screenshot assigns **Q03** compartment. Please change it to your assigned compartment instead.



Autonomous Database

List Scope

COMPARTMENT

Q03

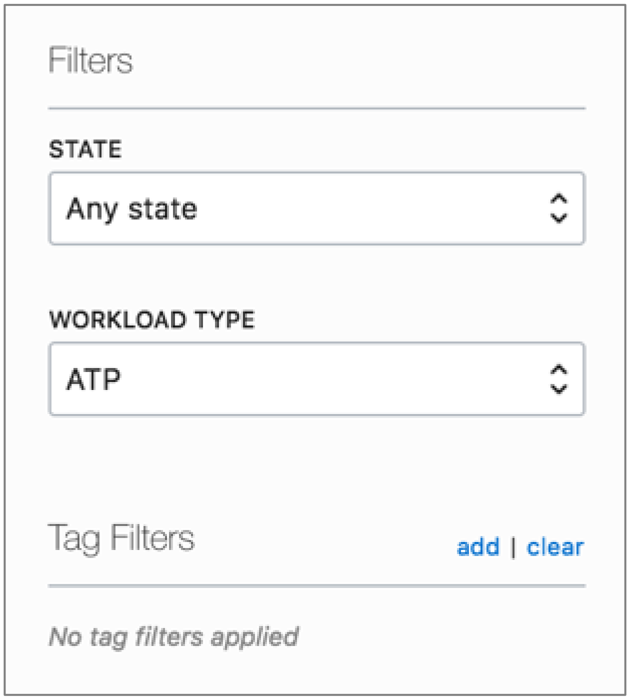
oraclepartnersas (root)/QLOUDABLE/Q03

Don't see what you're looking for?

Enter your assigned compartment

Note: If you have chosen the compartment you do not have privileges on, you will not be able to see or provision any instances in it.

- Note that the list of ATP Databases shown is filtered for the **Compartment** selected and may be further filtered using **State** and **Tags Filters**.



Filters

STATE

Any state

WORKLOAD TYPE

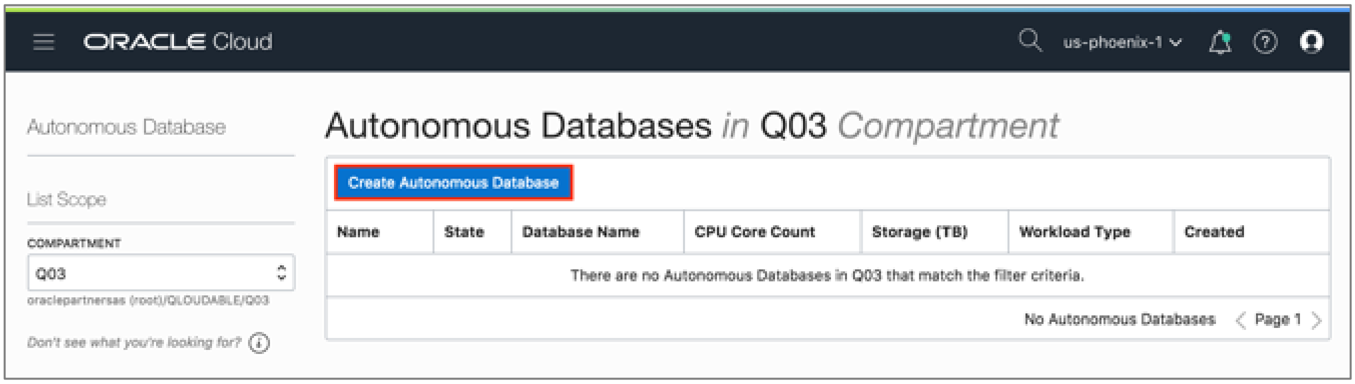
ATP

Tag Filters

No tag filters applied

STEP 3: Create an Autonomous Transaction Processing Database Instance

- Provision a new Autonomous Transaction Processing instance from the **Autonomous Database Home Page**.
- Click on **Create Autonomous Database** button to start the instance creation process.



ORACLE Cloud

us-phoenix-1

Autonomous Database

List Scope

COMPARTMENT

Q03

oraclepartnersas (root)/QLOUDABLE/Q03

Don't see what you're looking for?

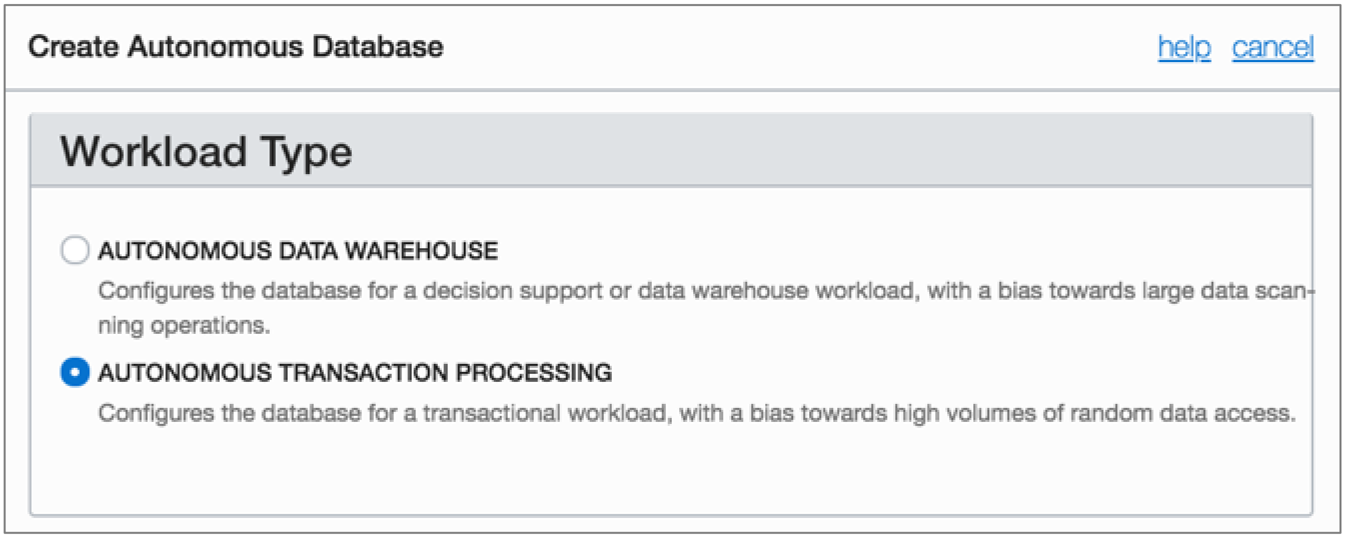
Autonomous Databases *in* Q03 *Compartment*

Create Autonomous Database

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created
There are no Autonomous Databases in Q03 that match the filter criteria.						

No Autonomous Databases ‹ Page 1 ›

- This will bring up **Create Autonomous Database** screen where you specify the configurations for the autonomous database.
- First, validate the **Workload Type** selected is **Autonomous Transaction Processing**.



Create Autonomous Database

help cancel

Workload Type

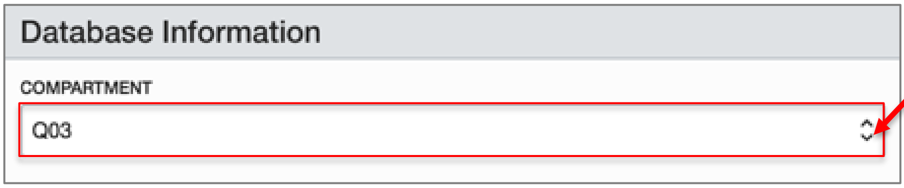
○ AUTONOMOUS DATA WAREHOUSE

Configures the database for a decision support or data warehouse workload, with a bias towards large data scanning operations.

● AUTONOMOUS TRANSACTION PROCESSING

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

- Second, verify that your assigned **Compartment** is selected (check your lab handout).



Database Information

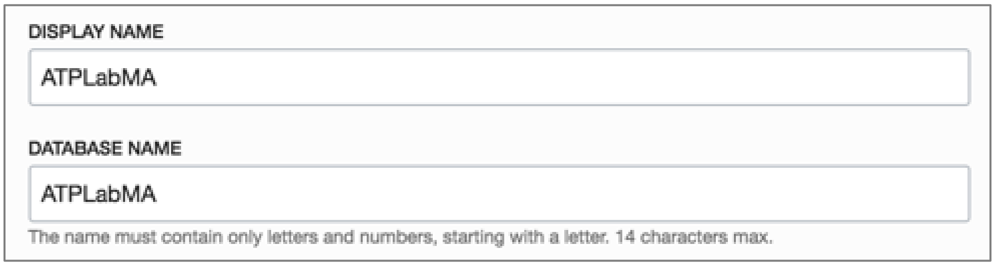
COMPARTMENT

Q03

Enter your assigned compartment

- Next, specify a **Display Name** and a unique **Database Name** for the instance.

IMPORTANT: Since the lab account may be shared by others, append a unique (and unused) integer or initials to **ATPLab** to come up with a unique name, for e.g. **ATPLabMA** (the name must contain only letters and numbers, starting with a letter, 14 characters max).



DISPLAY NAME

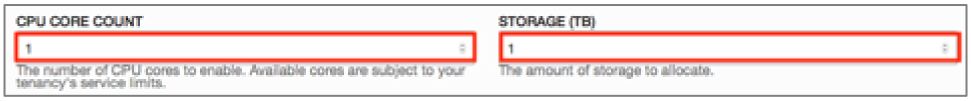
ATPLabMA

DATABASE NAME

ATPLabMA

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

- You can choose an instance shape, specified by the CPU count and storage size. Default **CPU Core Count** is **1** and **Storage** is **1 TB**. Please keep default selections at this time.



CPU CORE COUNT

1

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

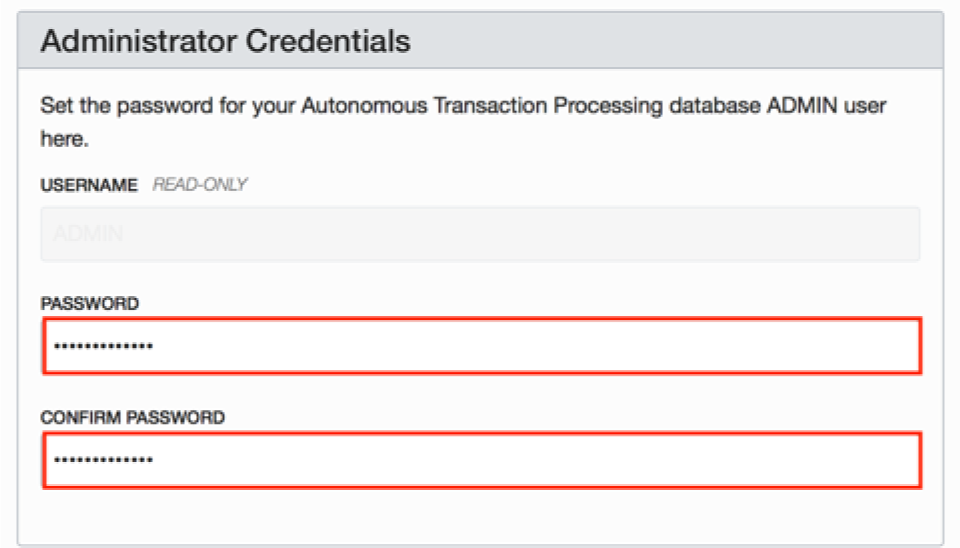
STORAGE (TB)

1

The amount of storage to allocate.

- Specify the **Password** for the instance. For this lab, we will be using the following as password:

```
WElcome_123#
```



Administrator Credentials

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

USERNAME *READ-ONLY*

ADMIN

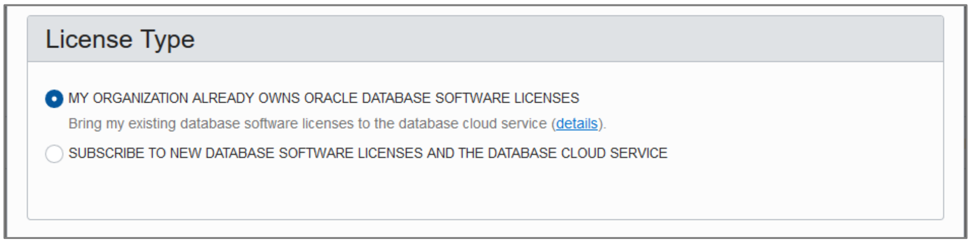
PASSWORD

••••••••••••••

CONFIRM PASSWORD

••••••••••••••

- For **License Type**, you will see the following two options:
  - My organization already owns Oracle database software licenses: Oracle allows you to bring your unused on-premise licenses to the cloud and your instances are billed at a discounted rate. This is the default option so ensure you have the right license type for this subscription.
  - Subscribe to new database software licenses and the database cloud service: Your cloud service instance should include database license. This is an all-inclusive cost and you do not need to bring any additional licenses to cloud.
- Select **My Organization Already Owns Oracle Database Software Licenses** for the purpose of this lab.



License Type

● MY ORGANIZATION ALREADY OWNS ORACLE DATABASE SOFTWARE LICENSES

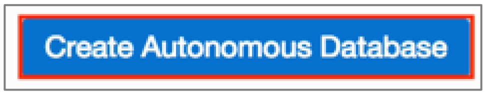
Bring my existing database software licenses to the database cloud service (details).

○ SUBSCRIBE TO NEW DATABASE SOFTWARE LICENSES AND THE DATABASE CLOUD SERVICE

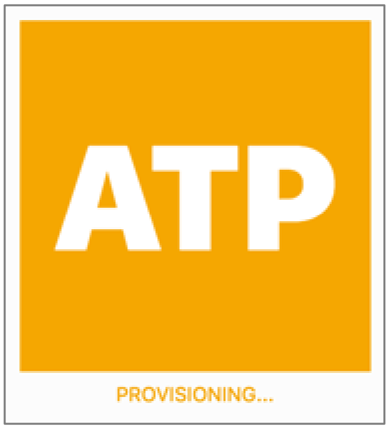
- Optionally, you may create **Tags**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources. More information about Tags and Tag Namespaces is provided in the OCI Identity and Access Management documentation here.

For lab purposes, you will not create a **Tag Namespace**.

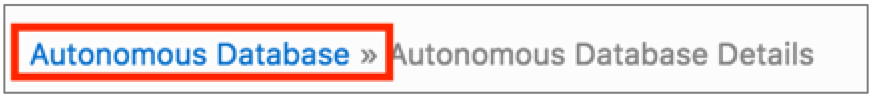
- Click on **Create Autonomous Database** to start provisioning the instance.



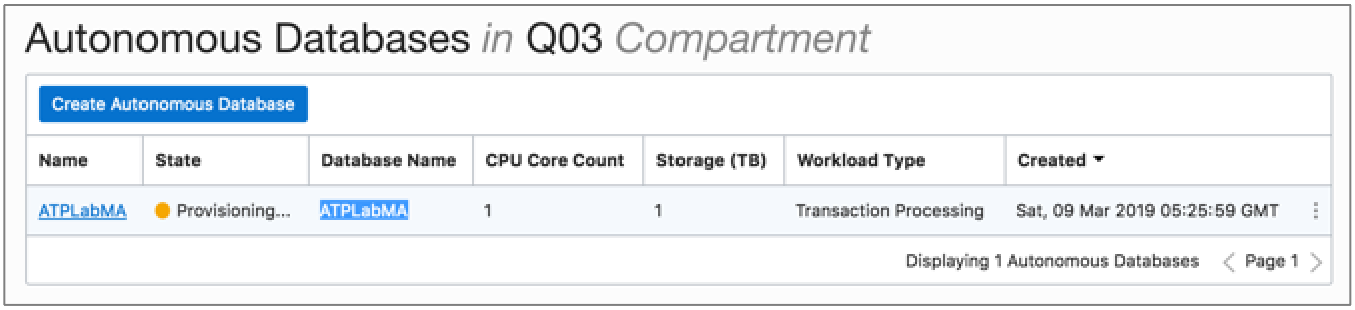
- You will see the database in **Provisioning** status as follows:



- Click on **Autonomous Database** link on the top breadcrumb to go back to **Autonomous Database Home Page**.



- Note that the Autonomous Database is in **Provisioning** state.



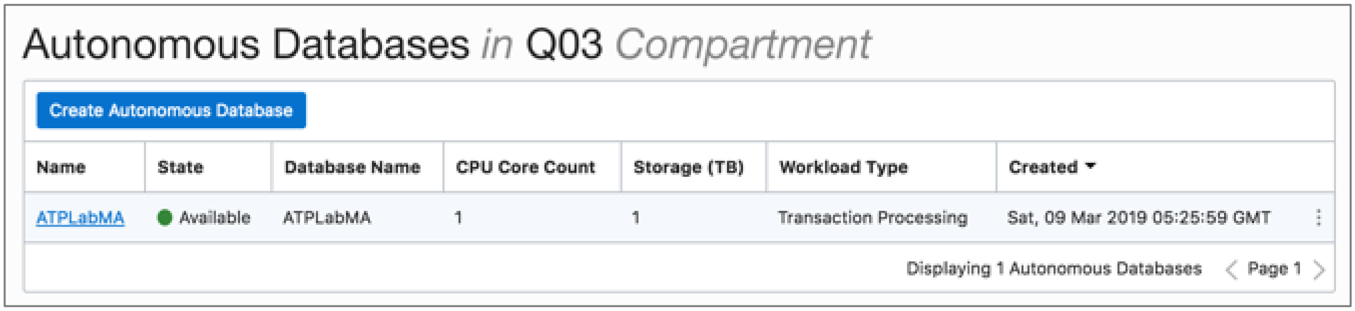
Autonomous Databases *in* Q03 *Compartment*

Create Autonomous Database

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created ▾
ATPLabMA	● Provisioning...	ATPLabMA	1	1	Transaction Processing	Sat, 09 Mar 2019 05:25:59 GMT

Displaying 1 Autonomous Databases ‹ Page 1 ›

- In a few minutes, the status will automatically change to **Available** indicating the database is ready and provisioned.



Autonomous Databases *in* Q03 *Compartment*

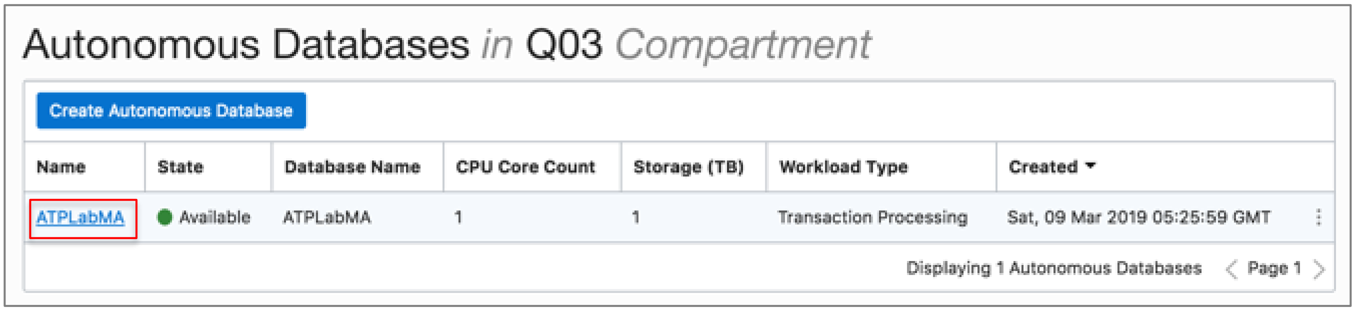
Create Autonomous Database

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created ▾
ATPLabMA	● Available	ATPLabMA	1	1	Transaction Processing	Sat, 09 Mar 2019 05:25:59 GMT

Displaying 1 Autonomous Databases ‹ Page 1 ›

STEP 4: Browse to Autonomous Database Details page

- From the **Autonomous Database Home Page**, click on your instance name that was just created to browse to **Autonomous Database Details Page**.



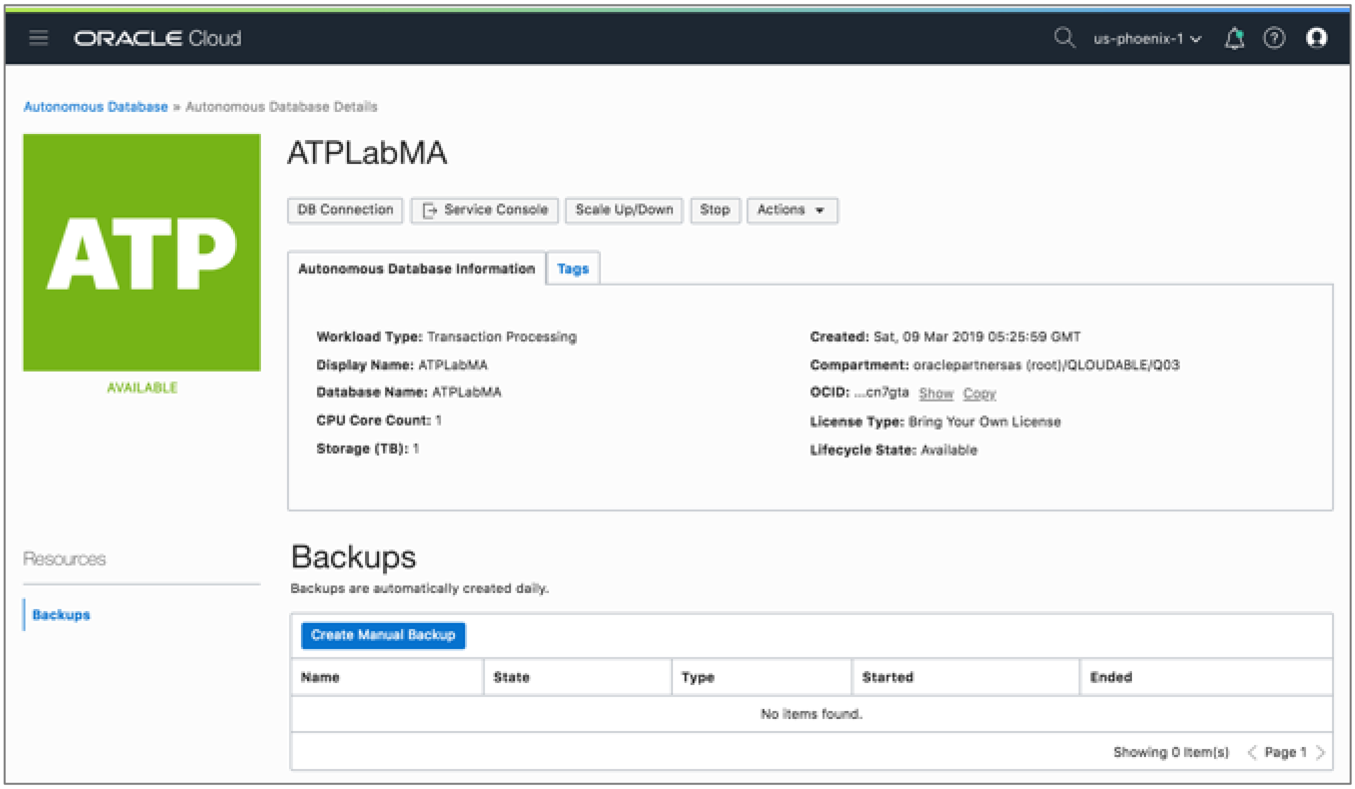
Autonomous Databases *in* Q03 *Compartment*

Create Autonomous Database

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created ▾
ATPLabMA	● Available	ATPLabMA	1	1	Transaction Processing	Sat, 09 Mar 2019 05:25:59 GMT

Displaying 1 Autonomous Databases ‹ Page 1 ›

- The **Autonomous Database Details Page** displays more information about the instance. Take a note of the various menu buttons that help you manage your autonomous database instance. Notice the **green** color of the **ATP** logo indicating the service is available.



Autonomous Database » Autonomous Database Details

ATPLabMA

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

Created: Sat, 09 Mar 2019 05:25:59 GMT
Compartment: oraclepartnersas (root)/QLOUDABLE/Q03
OCID: ...cn7gta Show Copy
License Type: Bring Your Own License
Lifecycle State: Available

Backups

Backups are automatically created daily.

Name	State	Type	Started	Ended
No items found.				

Showing 0 Item(s) Page 1

- You have successfully created your first **Autonomous Database** of type **ATP**.

Lab 1-3: Connecting to Oracle Autonomous Database

Oracle Autonomous Database is preconfigured to support Oracle Net Services with secure TCPS connections which allows the clients to connect using secure client credentials.

This lab will walk you through the steps of connecting to Oracle Autonomous Database using credentials wallets for secure connections. The tool you will use to connect is **Oracle SQL Developer**. You will also perform simple queries to validate the connection.

Objectives

- Download client credentials of autonomous database for secure connectivity.
- Connect to the autonomous database using a secure connection from Oracle SQL Developer.
- Run sample queries to validate the connection.

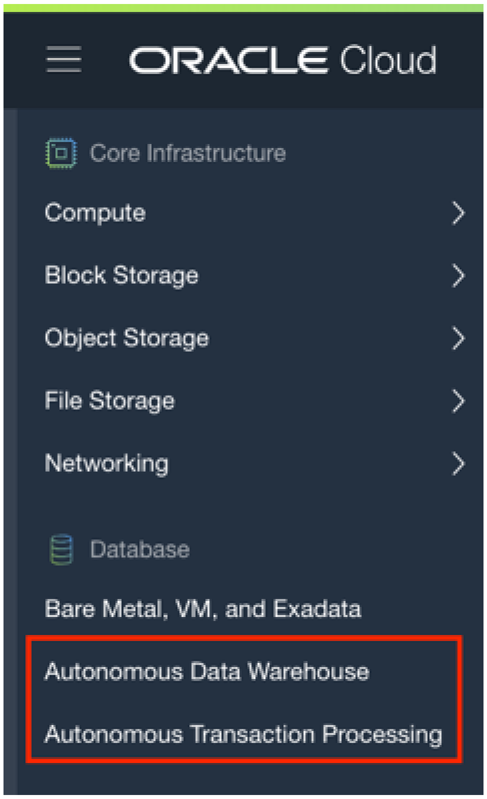
Required Artifacts

- Please ensure you have already provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** database.
- If you are not using the instructor supplied lab VM, ensure that your laptop/desktop has the following software installed:
  - **Oracle SQL Developer** (version 18.3 or above)
  - A Web browser supported for Oracle Cloud
  - Download **Lab Support Files** from the **Lab Setup** section

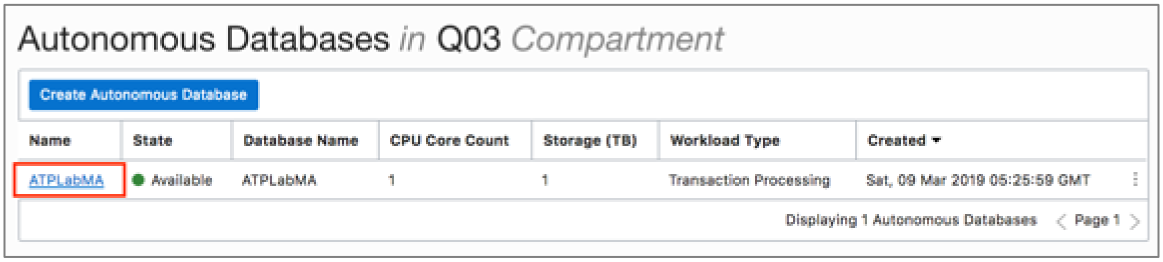
Lab Steps

STEP 1: Sign-In to Autonomous Database Service Console

- Sign in to **Oracle Cloud Infrastructure Home Page** using the credentials provided and the instructions from an earlier lab.
- Browse to your **Autonomous Database Home Page** (either **Autonomous Transaction Processing** or **Autonomous Data Warehouse**), based on what you created in the previous labs.



- From the Autonomous Database **Home Page**, browse to the Autonomous Database **Details** page by clicking on the service name.

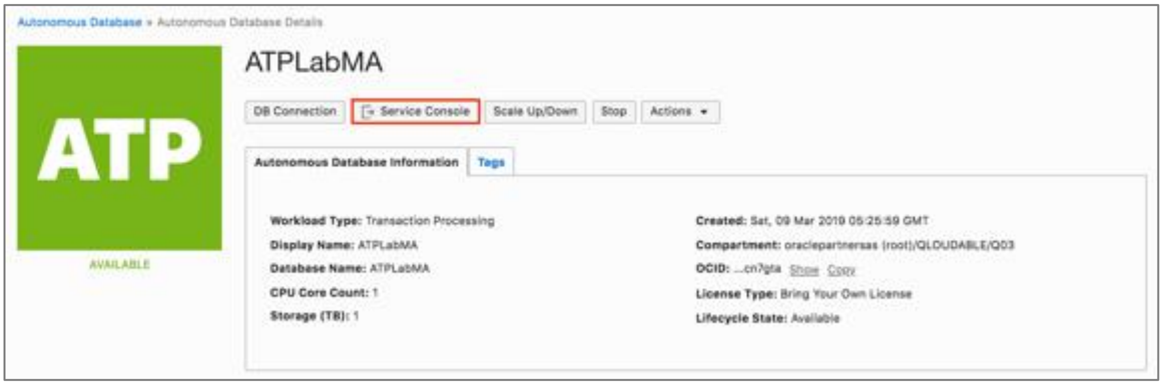


Autonomous Databases *in* Q03 *Compartment*

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created ▾
ATPLabMA	● Available	ATPLabMA	1	1	Transaction Processing	Sat, 09 Mar 2019 05:25:59 GMT

Displaying 1 Autonomous Databases ‹ Page 1 ›

- Click on **Service Console** to sign in to Autonomous Database Service Console.


Autonomous Database » Autonomous Database Details
ATP
AVAILABLE
DB Connection
Service Console
Scale Up/Down
Stop
Actions ▾
Autonomous Database Information
Tags

ATPLabMA

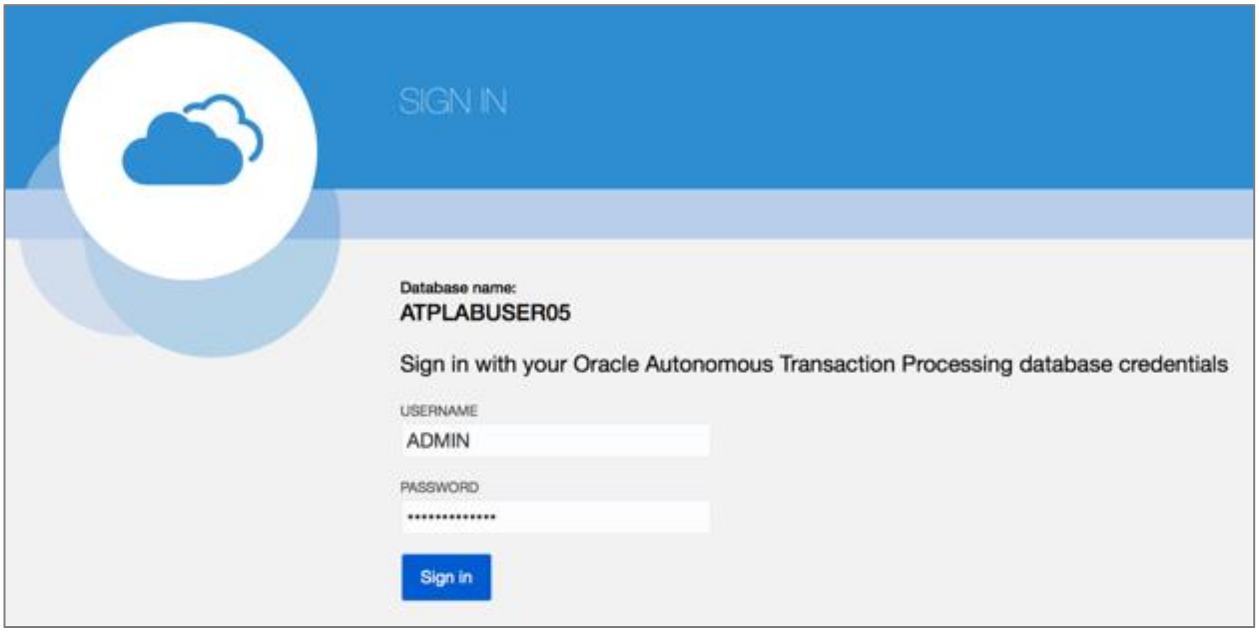
Workload Type: Transaction Processing
Display Name: ATPLabMA
Database Name: ATPLabMA
CPU Core Count: 1
Storage (TB): 1

Created: Sat, 09 Mar 2019 05:25:59 GMT
Compartment: oraclepartnersas (root)/QLOUDABLE/Q03
OCID: ...cn7gta Show Copy
License Type: Bring Your Own License
Lifecycle State: Available

- Sometimes you may be prompted to **Sign In** (due to timeout). Fill in the following User and Password and select **Sign In**:
  - Username: **ADMIN**

- Password: **<Password>** specified during ADB provisioning (e.g. `WElcome_123#`)

Note: The database is initially created with only one user i.e. the **ADMIN** user.



SIGN IN

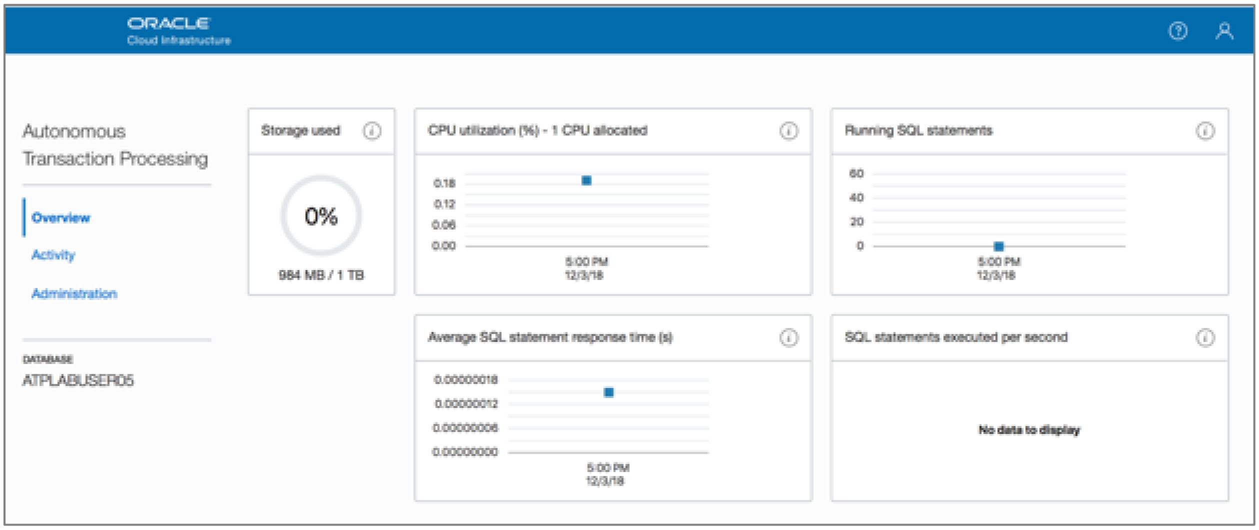
Database name:
ATPLABUSER05

Sign in with your Oracle Autonomous Transaction Processing database credentials

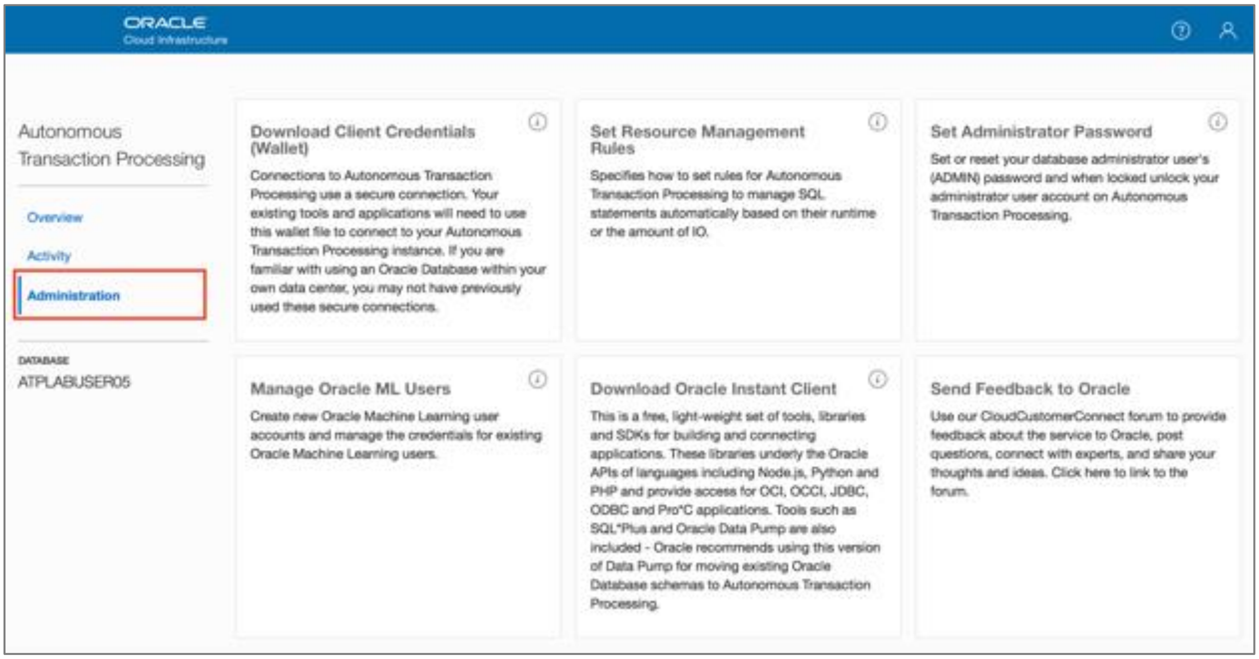
USERNAME
ADMIN

PASSWORD
••••••••••••••

- You will be placed in the **Overview** page. Notice there is no activity displayed because this is a new instance.



- Click on **Administration** link in the left menu to go to the service **Administration** page.



Autonomous Transaction Processing

DATABASE
ATPLABUSER05

Download Client Credentials (Wallet)

Connections to Autonomous Transaction Processing use a secure connection. Your existing tools and applications will need to use this wallet file to connect to your Autonomous Transaction Processing instance. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these secure connections.

Set Resource Management Rules

Specifies how to set rules for Autonomous Transaction Processing to manage SQL statements automatically based on their runtime or the amount of IO.

Set Administrator Password

Set or reset your database administrator user's (ADMIN) password and when locked unlock your administrator user account on Autonomous Transaction Processing.

Manage Oracle ML Users

Create new Oracle Machine Learning user accounts and manage the credentials for existing Oracle Machine Learning users.

Download Oracle Instant Client

This is a free, light-weight set of tools, libraries and SDKs for building and connecting applications. These libraries underly the Oracle APIs of languages including Node.js, Python and PHP and provide access for OCI, OCCI, JDBC, ODBC and Pro*C applications. Tools such as SQL*Plus and Oracle Data Pump are also included - Oracle recommends using this version of Data Pump for moving existing Oracle Database schemas to Autonomous Transaction Processing.

Send Feedback to Oracle

Use our CloudCustomerConnect forum to provide feedback about the service to Oracle, post questions, connect with experts, and share your thoughts and ideas. Click here to link to the forum.

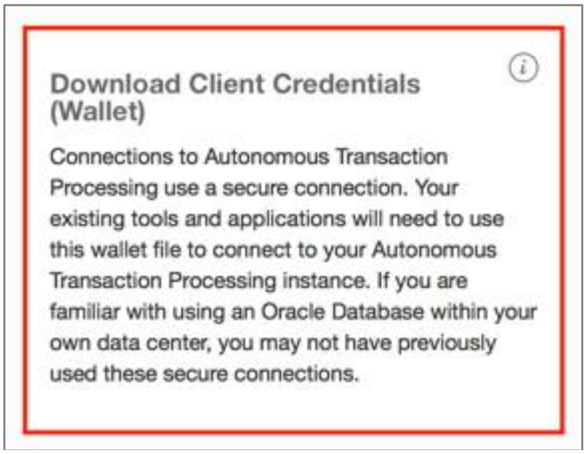
- Notice the six options on the **Administration** page:
  - **Download a Connection Wallet**: This contains the credentials files used for connectivity to the instance from client applications, tools.
  - **Set Administrator Password**: Used to change the “Admin” account password Download Oracle.
  - **Instant Client**: Points to different clients that can be used to connect to the database (like SQL*Plus).
  - **Set Resource Management Rules**: ADB has pre-created resource consumer groups which are managed here.
  - **Manage Oracle ML Users**: ML Notebook development environment that can be used with the ADB.
  - **Send Feedback to Oracle**: Email feedback to Oracle.

STEP 2: Download the Client Credentials Wallet

The connection wallet provides a secure authentication method that can be used to connect to your ADB database. This wallet must be downloaded to the client that will be connecting to the database.

The wallet is downloaded either from the autonomous database **Service Console** or the **Administration** page within the console.

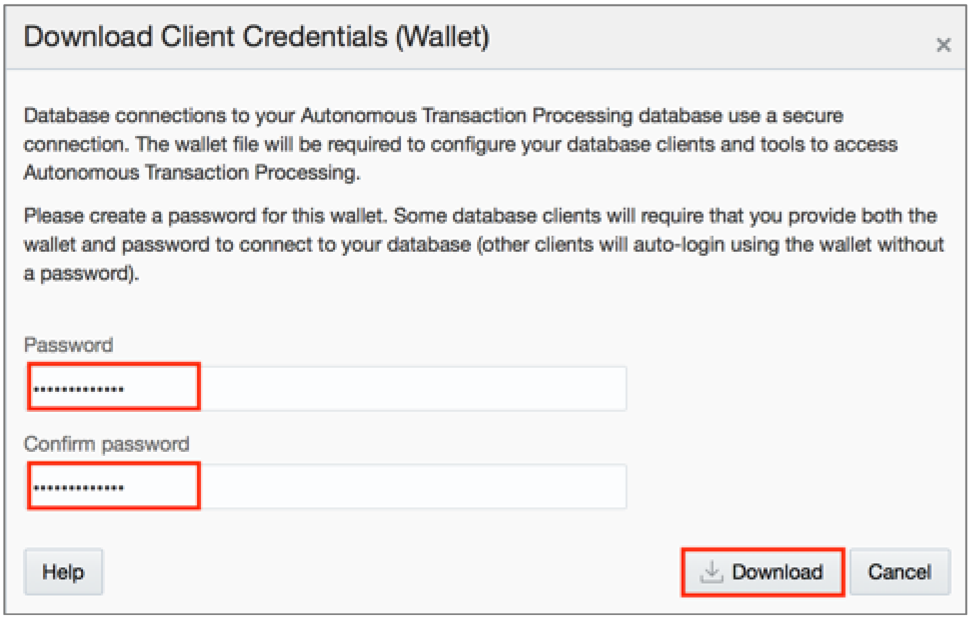
- From the service **Administration** page click on **Download Client Credentials Wallet**.



Download Client Credentials (Wallet)

Connections to Autonomous Transaction Processing use a secure connection. Your existing tools and applications will need to use this wallet file to connect to your Autonomous Transaction Processing instance. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these secure connections.

- On the **Download Client Credentials (Wallet)** menu, enter a password for the wallet and click **Download**.
  - Note that this password is separate from the **ADMIN** user password created earlier (but the same password can be used).



Download Client Credentials (Wallet)

Database connections to your Autonomous Transaction Processing database use a secure connection. The wallet file will be required to configure your database clients and tools to access Autonomous Transaction Processing.

Please create a password for this wallet. Some database clients will require that you provide both the wallet and password to connect to your database (other clients will auto-login using the wallet without a password).

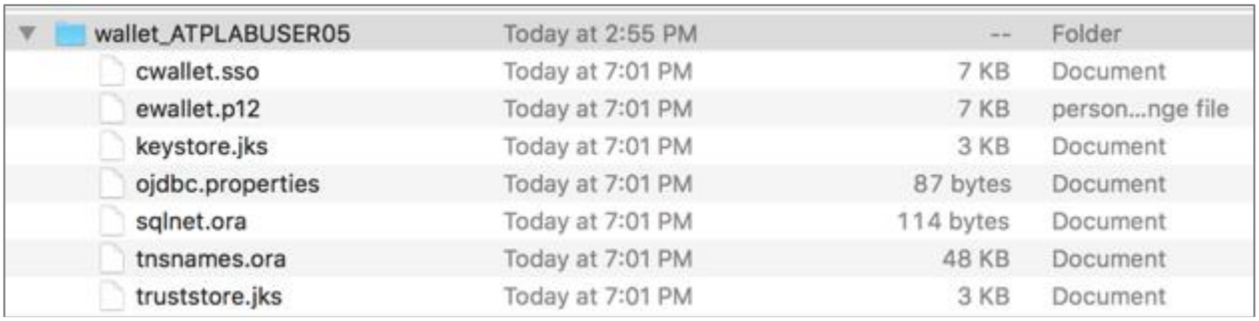
Password: ••••••••••••••

Confirm password: ••••••••••••••

- Save this file in a secure location. The credentials zip file contains the encryption wallet, Java Keystore and other relevant files to make a secure TLS 1.2 connection to your database from client applications.

- Navigate to the location in your system where the file was downloaded (typically your **Downloads** directory).

- The format of the file is always **Wallet_<dbname>.zip**. Extract the contents of the wallet into a directory (using a zip utility, usually by right clicking on the file), you will find the following files:



▼ wallet_ATPLABUSER05	Today at 2:55 PM	--	Folder
cwallet.sso	Today at 7:01 PM	7 KB	Document
ewallet.p12	Today at 7:01 PM	7 KB	person...nge file
keystore.jks	Today at 7:01 PM	3 KB	Document
ojdbc.properties	Today at 7:01 PM	87 bytes	Document
sqlnet.ora	Today at 7:01 PM	114 bytes	Document
tnsnames.ora	Today at 7:01 PM	48 KB	Document
truststore.jks	Today at 7:01 PM	3 KB	Document

- There are a few files from the list that you will work with during the hands-on labs. Some tools use the wallet file (.zip) directly whereas some use specific files contained in the wallet. Here is the description of some of these files:

  – **Wallet_<dbname>.zip** : The wallet.

  – **sqlnet.ora** : Points to the location of the wallet for sqlnet connections.

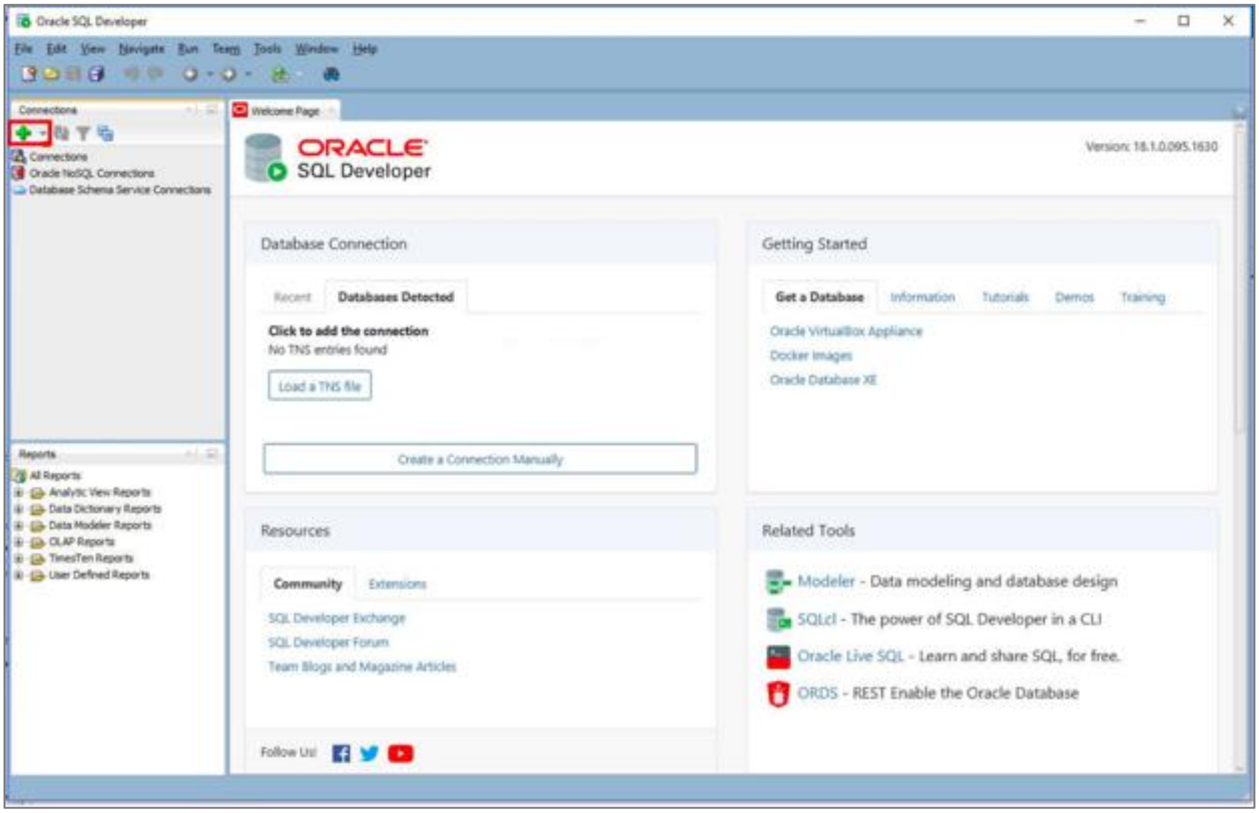
  – **tnsnames.ora** : Connection description for the database service (please note this file contains connection description for all the databases that exist in that cloud account).

  – **ojdbc.properties** : Points to the location of the wallet for JDBC connections.

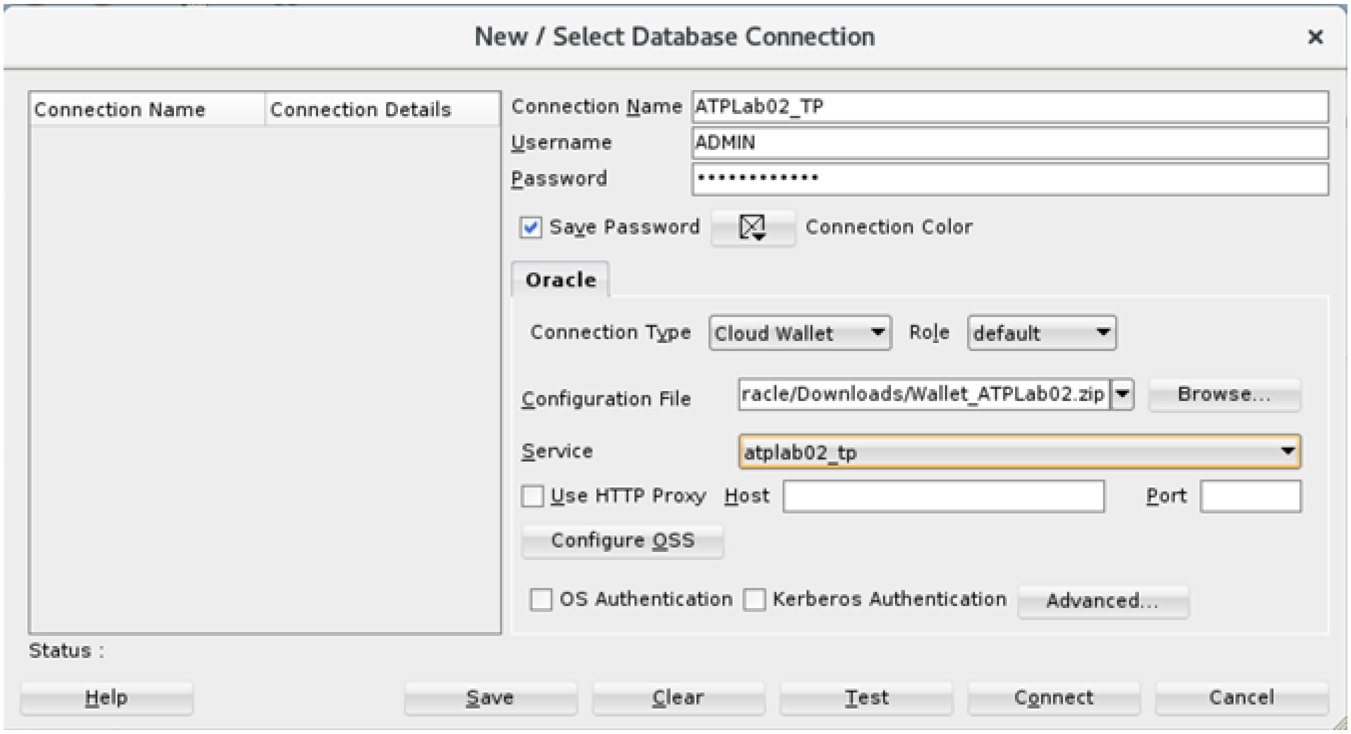
STEP 3: Connect to Autonomous Database using Oracle SQL Developer

Create a connection for your database using the default administrator account, **ADMIN**, by following these steps.

- Launch SQL Developer and click **Add Connection** on top left.



- Enter the following in New database connection
  - **Connection Name**: Name for your connection
  - **Username**: ADMIN
  - **Password**: ADMIN user's password (e.g. WElcome_123#)
  - **Save Password**: Checked
  - **Connection Type**: Cloud Wallet
  - **Role**: Default
  - **Configuration File**: Click on **Browse** and select the wallet file you downloaded
  - **Service**: <Database_Name>_TP**.** The service name is the **Database Name** followed by suffix of either **TP, TP_URGENT, LOW, MEDIUM, or HIGH**. These suffixes determine degree of parallelism used and are relevant for a DSS workload. For OLTP workloads it's safe to select any of them.
  - In the screenshot below, the <Database_Name> is **ATPLab02** and the Service being connected to is **atplab02_tp**.



New / Select Database Connection

Connection Name: ATPLab02_TP

Username: ADMIN

Password: •••••••••••••

☑ Save Password

Connection Type: Cloud Wallet

Role: default

Configuration File: racle/Downloads/Wallet_ATPLab02.zip

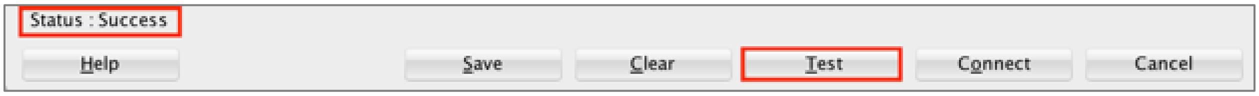
Service: atplab02_tp

☐ Use HTTP Proxy Host: Port:

☐ OS Authentication ☐ Kerberos Authentication

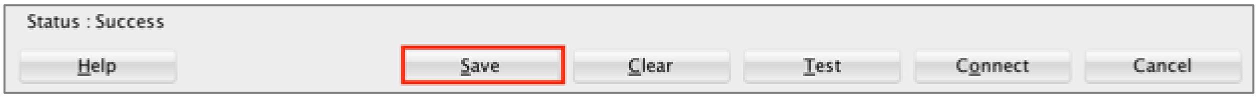
Status :

- Test your connection by clicking **Test**. The **Status** bar will show **Success** if it is a successful connection.



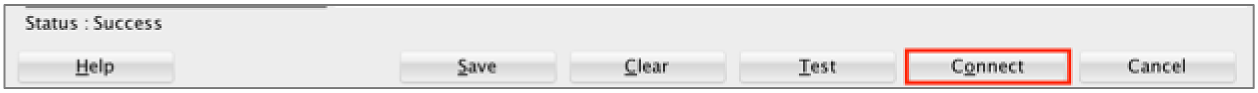
Status : Success

- Save the Connection by clicking **Save**.



Status : Success

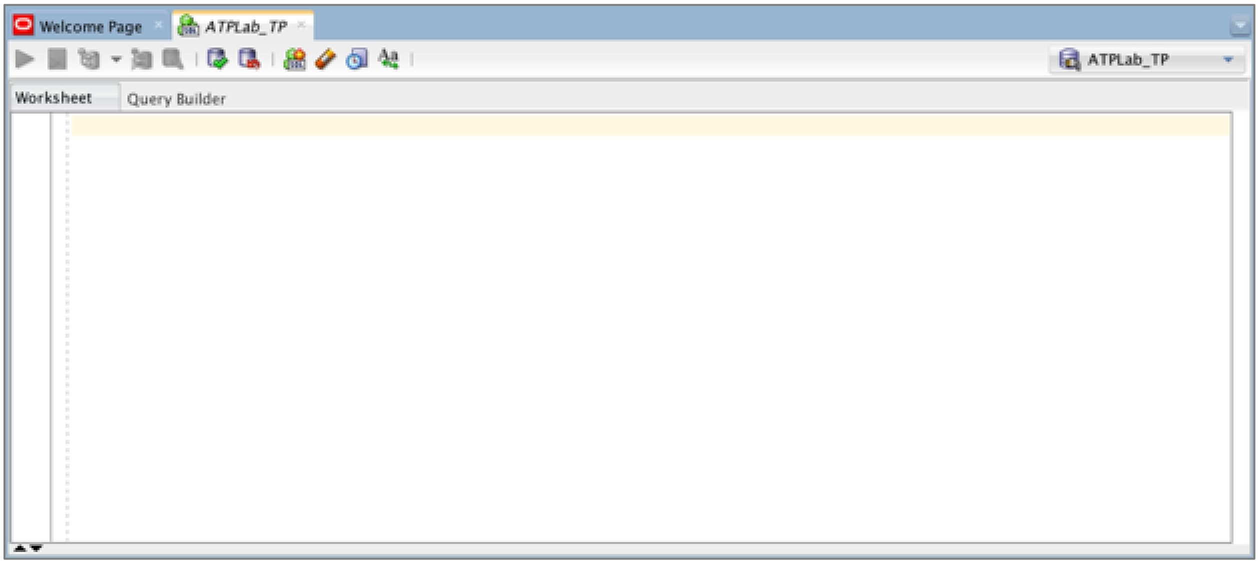
- Click on **Connect**.



Status : Success

- Upon a successful connection you will see a **SQL Developer Worksheet**.

Note: If you do not see a **Worksheet** for your connection, just click the **Worksheet** drop-down on the top and select your connection to force open a worksheet.



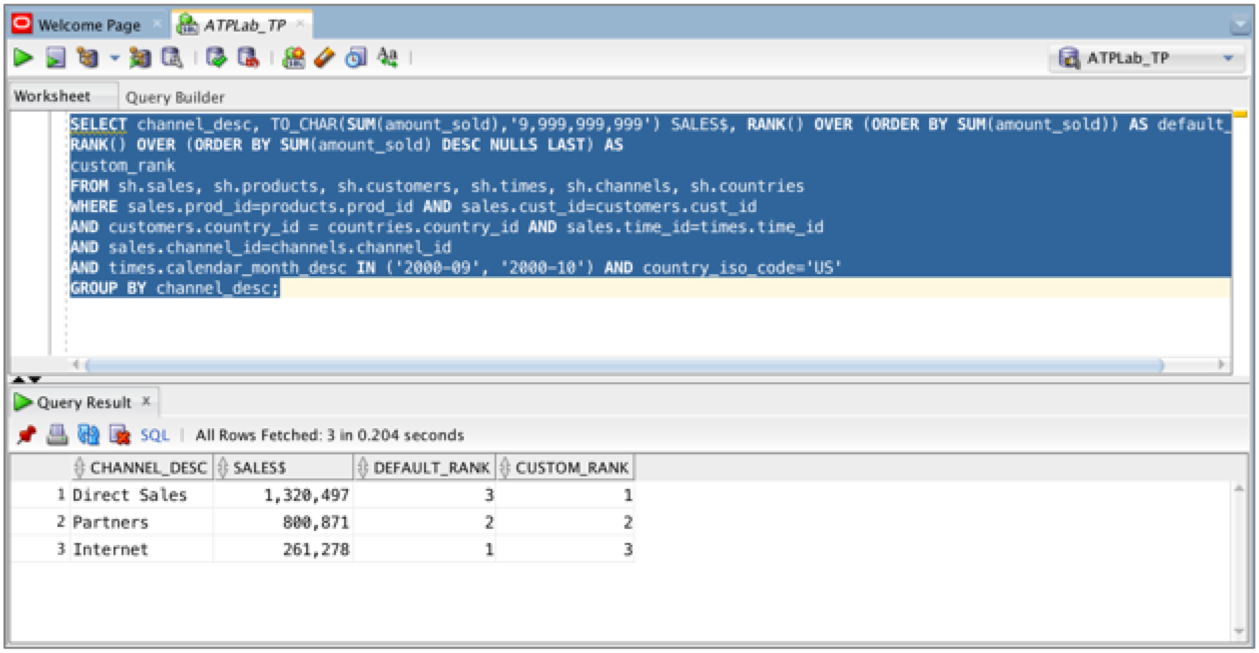
- Run a test query. The autonomous database you created contains the sample **Sales History** (SH) schema, we will use this schema to run a test query to make sure everything is working correctly.

- Copy the contents of the file **/home/oracle/labfiles/select_sql1.sql** from the lab VM (or from a locally downloaded location on your computer) and paste it in the **SQL Worksheet**. Below is the SQL that you would be copying.

Note: Do not copy/paste directly from below because sometimes copy/paste from PDF has issues.

```
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;
```

- Click **F5** or the **Run Script** button. Verify the query executes and results are displayed.



```
SELECT channel_desc, TO_CHAR(SUM(amount_sold),'9,999,999,999') SALES$, RANK() OVER (ORDER BY SUM(amount_sold)) AS default_
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;
```

All Rows Fetched: 3 in 0.204 seconds

	CHANNEL_DESC	SALES$	DEFAULT_RANK	CUSTOM_RANK
1	Direct Sales	1,320,497	3	1
2	Partners	800,871	2	2
3	Internet	261,278	1	3

- You have successfully connected SQL Developer to autonomous database and validated the connection.

2. Data Loading and Integration

Lab 2-1: Loading Data Using SQL Developer Import Data Wizard

Traditionally transaction processing systems ingest data through routine transactions or DML operations; Data Warehouses normally perform bulk data loads using Oracle Database tools, and Oracle or other 3rd party data integration tools.

In general, you load data from files local to your client computer or from files stored in a cloud-based object store.

Oracle SQL Developer provides the ability to import data into tables in autonomous database or any version of the Oracle Database using the **Import Data Wizard**. The import wizard allows you to import data from a delimited format file or a Microsoft Excel XLS file.

Objectives

- Learn to use the SQL Developer Import Data Wizard.

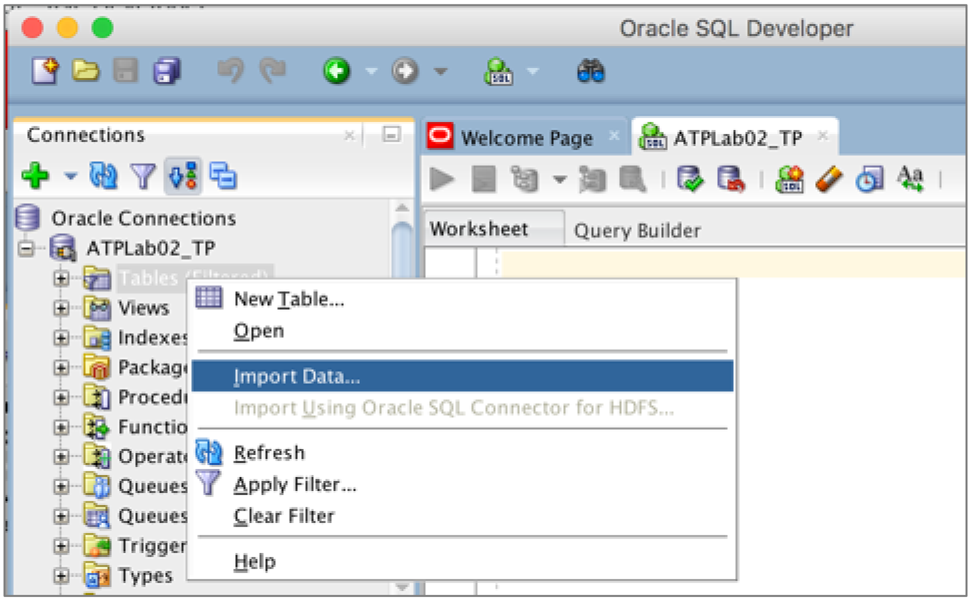
Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** database.
- If you are not using the instructor supplied lab VM, ensure that your laptop/desktop has the following software installed:
  - **Oracle SQL Developer** (version 18.3 or above)
  - A Web browser supported for Oracle Cloud
  - Download **Lab Support Files** from the **Lab Setup** section

Lab Steps

STEP 1: Load a Local File to ADB using SQL Developer

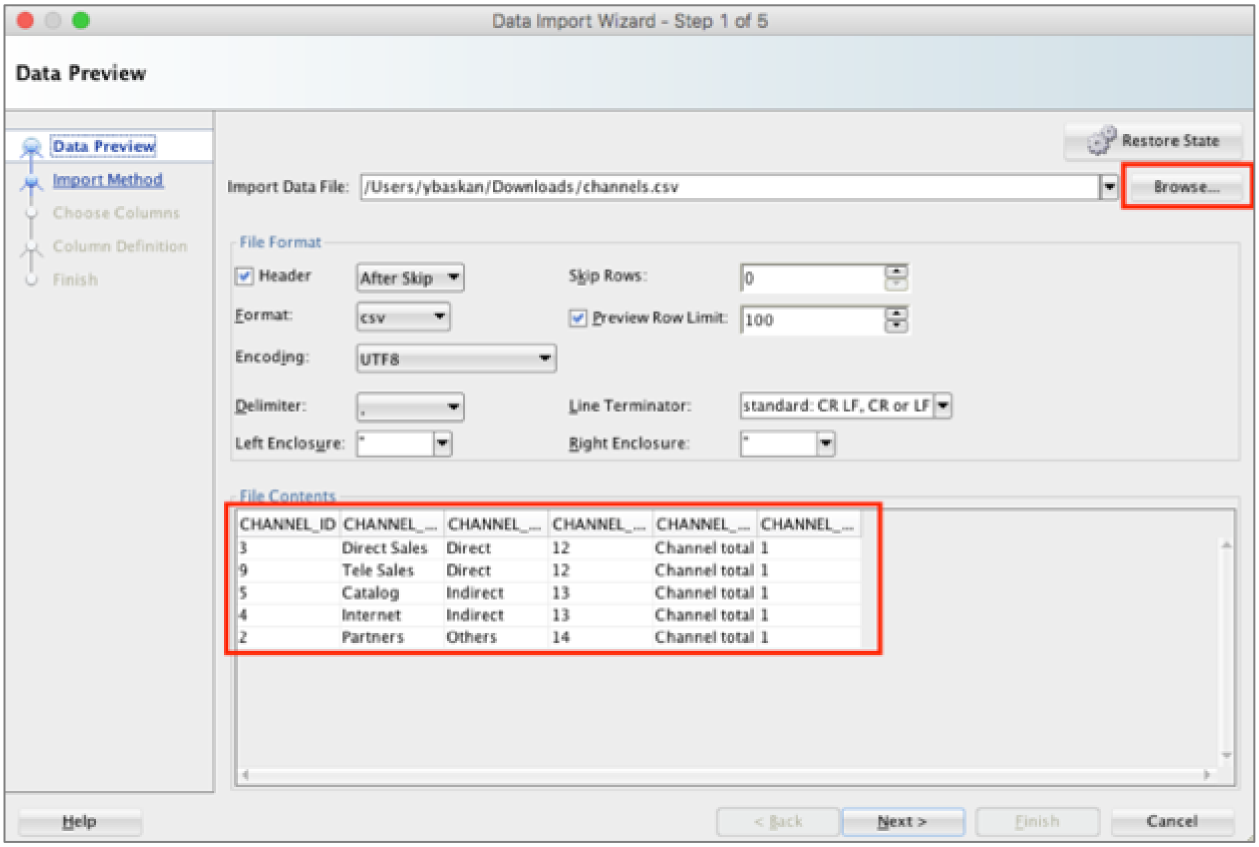
- Start **SQL Developer** (from your desktop or the lab VM).
- Connect to your autonomous database (ATP or ADW) using the connection configured earlier and right-click **Tables** and click **Import Data**.



Note: SQL Developer **Import Data Wizard** is accessible by right-clicking an existing table on the database explorer menu, and also by right-clicking on the **Tables** tree of the database explorer (as above), where you can import data and also create the table in a one-step operation.

- This will open the **Data Import Wizard**.

  Click **Browse** and locate **`channels.csv`** from the folder **`/home/oracle/labfiles`** (or from a locally downloaded location on your computer). When you select the file, you will see the file contents on the screen.

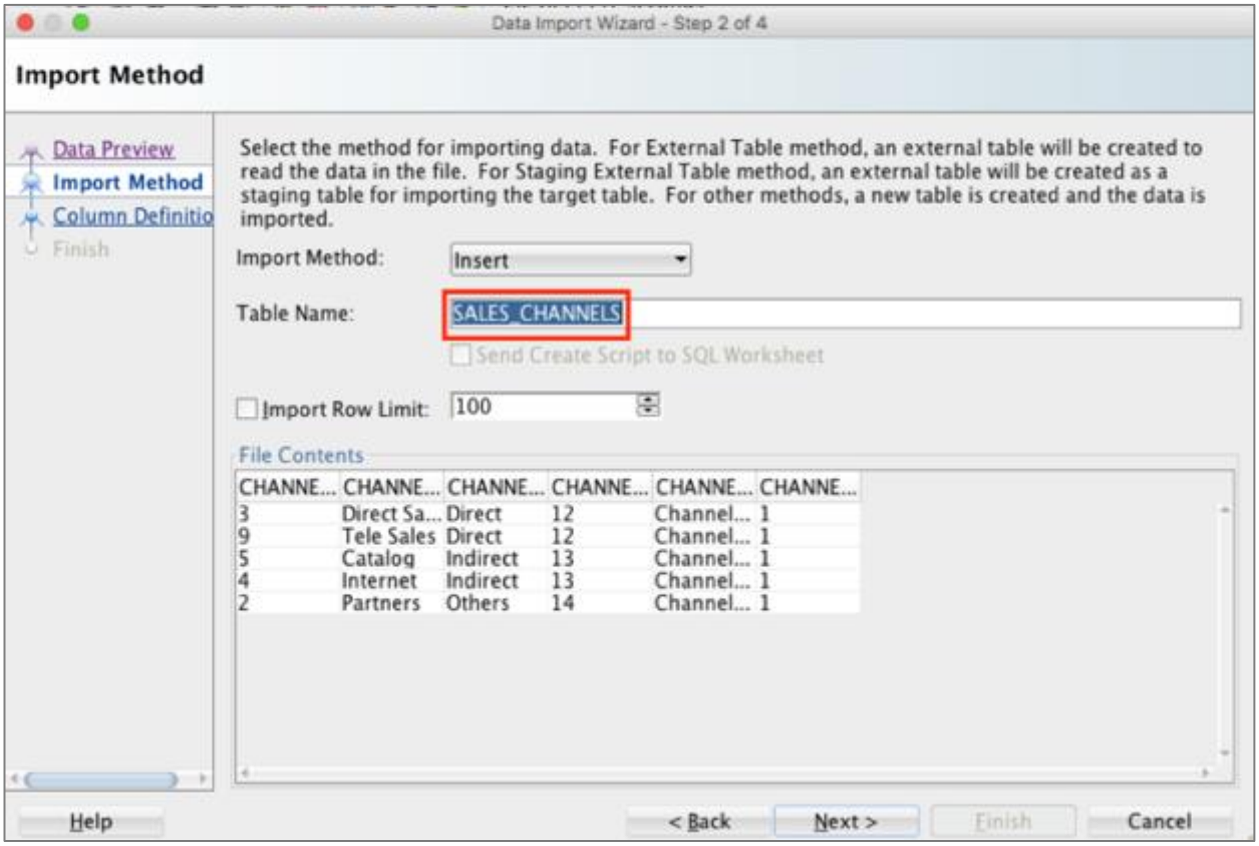


Import Data File: /Users/ybaskan/Downloads/channels.csv

☑ Header After Skip
Skip Rows: 0
Format: csv
☑ Preview Row Limit: 100
Encoding: UTF8
Delimiter: ,
Line Terminator: standard: CR LF, CR or LF
Left Enclosure: "
Right Enclosure: "

CHANNEL_ID	CHANNEL_...	CHANNEL_...	CHANNEL_...	CHANNEL_...	CHANNEL_...
3	Direct Sales	Direct	12	Channel total	1
9	Tele Sales	Direct	12	Channel total	1
5	Catalog	Indirect	13	Channel total	1
4	Internet	Indirect	13	Channel total	1
2	Partners	Others	14	Channel total	1

- Click **Next**. In the next screen enter **SALES_CHANNELS** as the table name that will be created and the data loaded into.



Import Method

Select the method for importing data. For External Table method, an external table will be created to read the data in the file. For Staging External Table method, an external table will be created as a staging table for importing the target table. For other methods, a new table is created and the data is imported.

Import Method: Insert

Table Name: SALES_CHANNELS

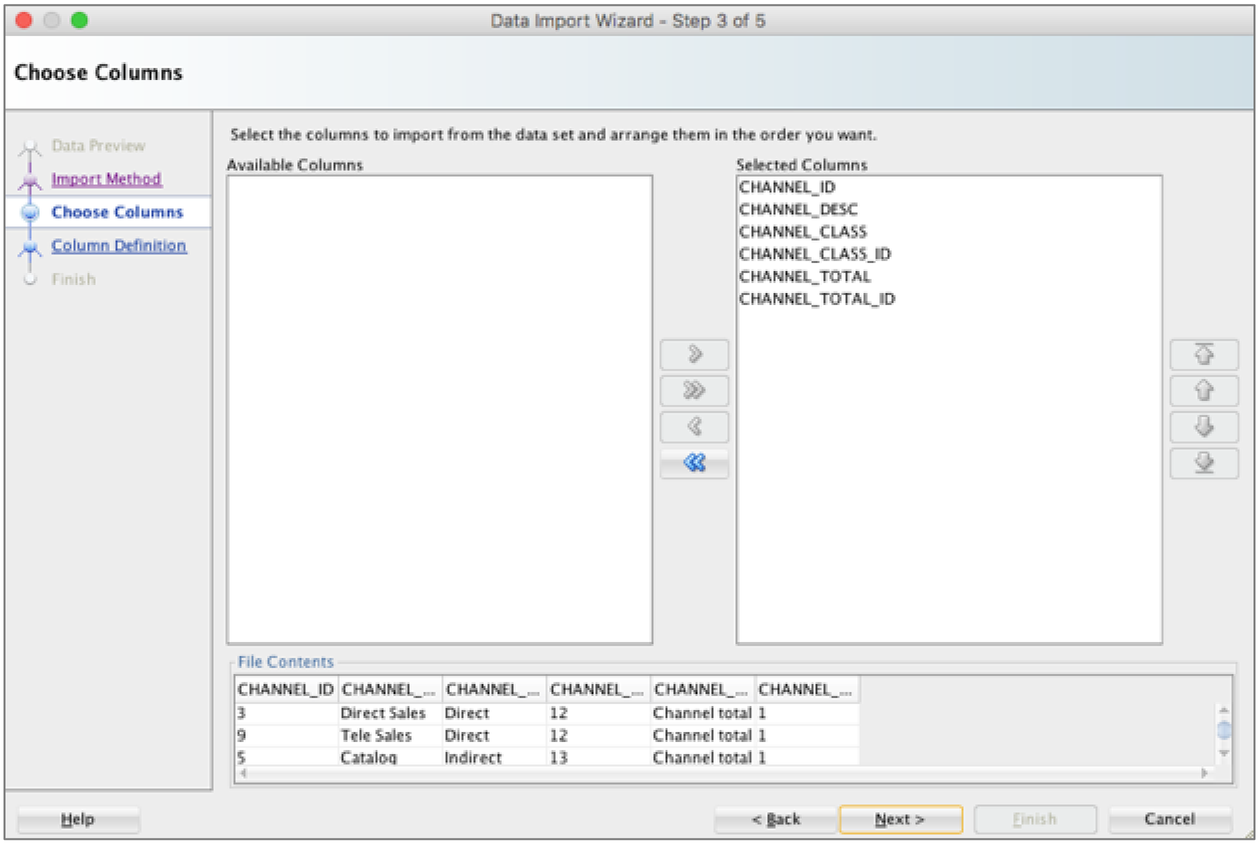
☐ Send Create Script to SQL Worksheet

☐ Import Row Limit: 100

File Contents

CHANNE...	CHANNE...	CHANNE...	CHANNE...	CHANNE...	CHANNE...
3	Direct Sa...	Direct	12	Channel...	1
9	Tele Sales	Direct	12	Channel...	1
5	Catalog	Indirect	13	Channel...	1
4	Internet	Indirect	13	Channel...	1
2	Partners	Others	14	Channel...	1

- 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.



Select the columns to import from the data set and arrange them in the order you want.

Available Columns

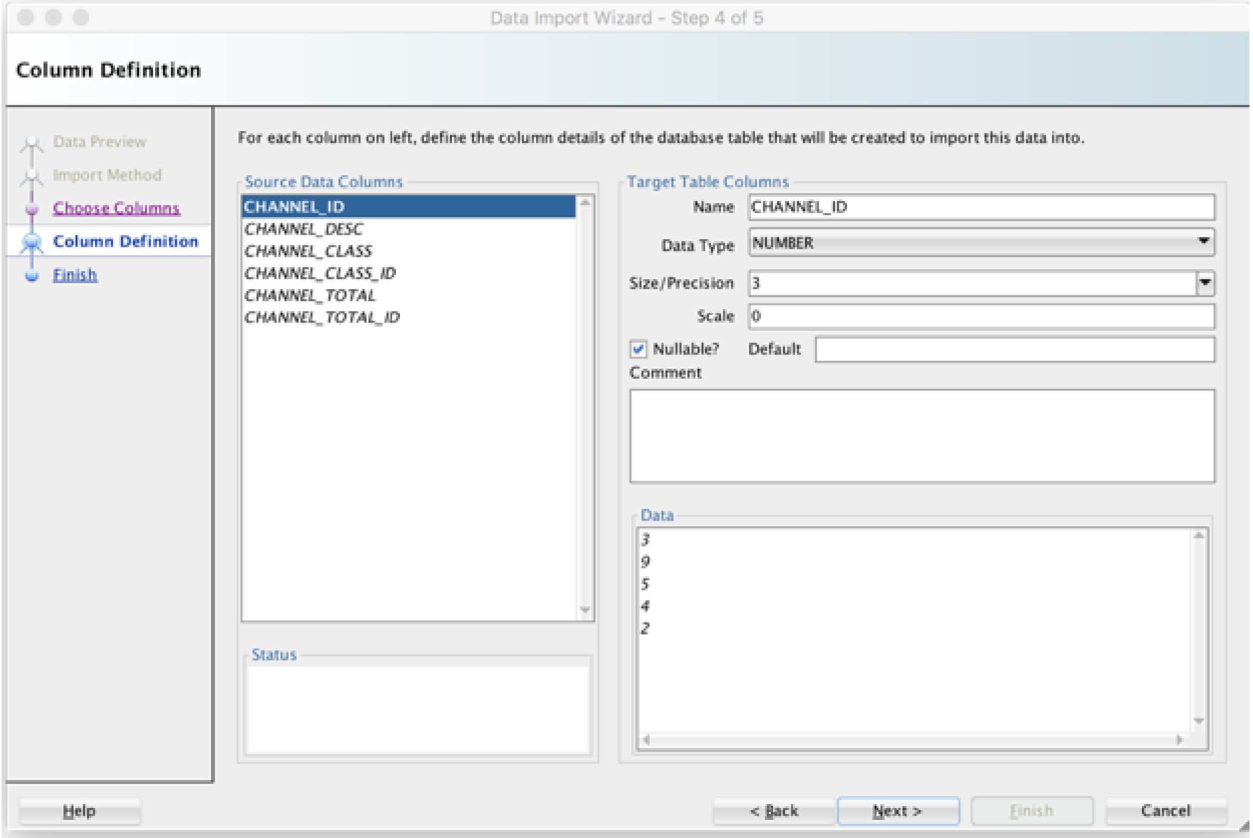
Selected Columns

CHANNEL_ID
CHANNEL_DESC
CHANNEL_CLASS
CHANNEL_CLASS_ID
CHANNEL_TOTAL
CHANNEL_TOTAL_ID

File Contents

CHANNEL_ID	CHANNEL_...	CHANNEL_...	CHANNEL_...	CHANNEL_...	CHANNEL_...
3	Direct Sales	Direct	12	Channel total	1
9	Tele Sales	Direct	12	Channel total	1
5	Catalog	Indirect	13	Channel total	1

- Click **Next**. The next screen allows you to look at the data types for each column, which you can change if there is a need. For this exercise leave the data types as default.



For each column on left, define the column details of the database table that will be created to import this data into.

Source Data Columns

CHANNEL_ID
CHANNEL_DESC
CHANNEL_CLASS
CHANNEL_CLASS_ID
CHANNEL_TOTAL
CHANNEL_TOTAL_ID

Status

Target Table Columns

Name: CHANNEL_ID

Data Type: NUMBER

Size/Precision: 3

Scale: 0

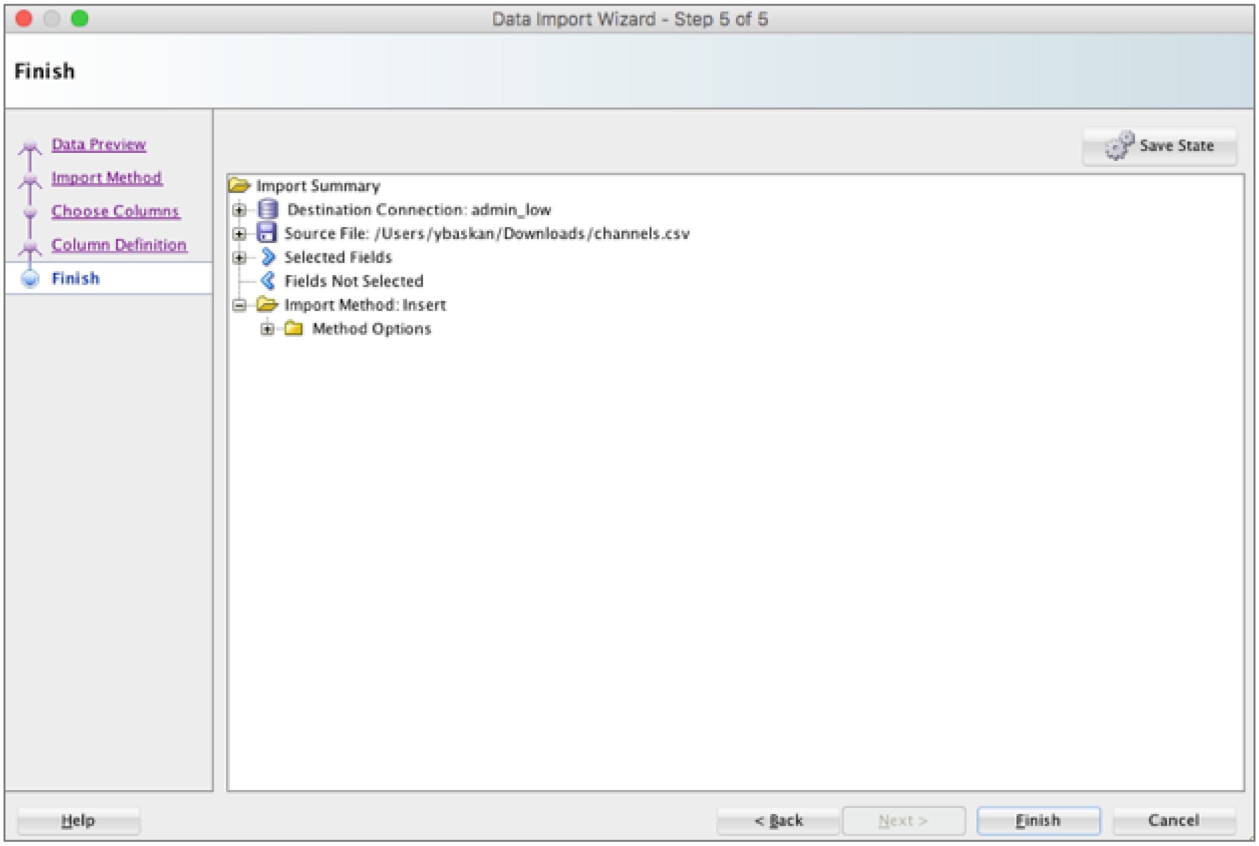
☑ Nullable? Default:

Comment:

Data

3
9
5
4
2

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

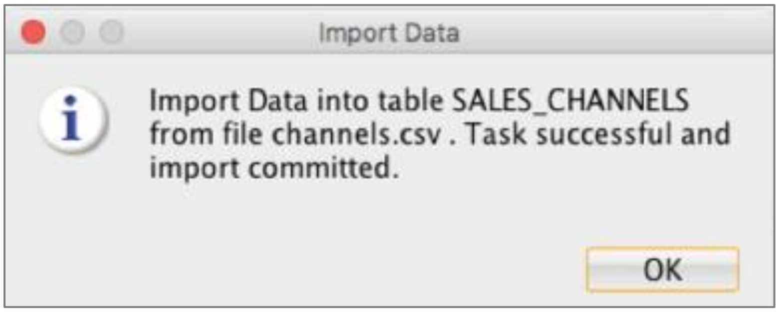


Data Import Wizard - Step 5 of 5

Finish

Import Summary
Destination Connection: admin_low
Source File: /Users/ybaskan/Downloads/channels.csv
Selected Fields
Fields Not Selected
Import Method: Insert
Method Options

- 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.

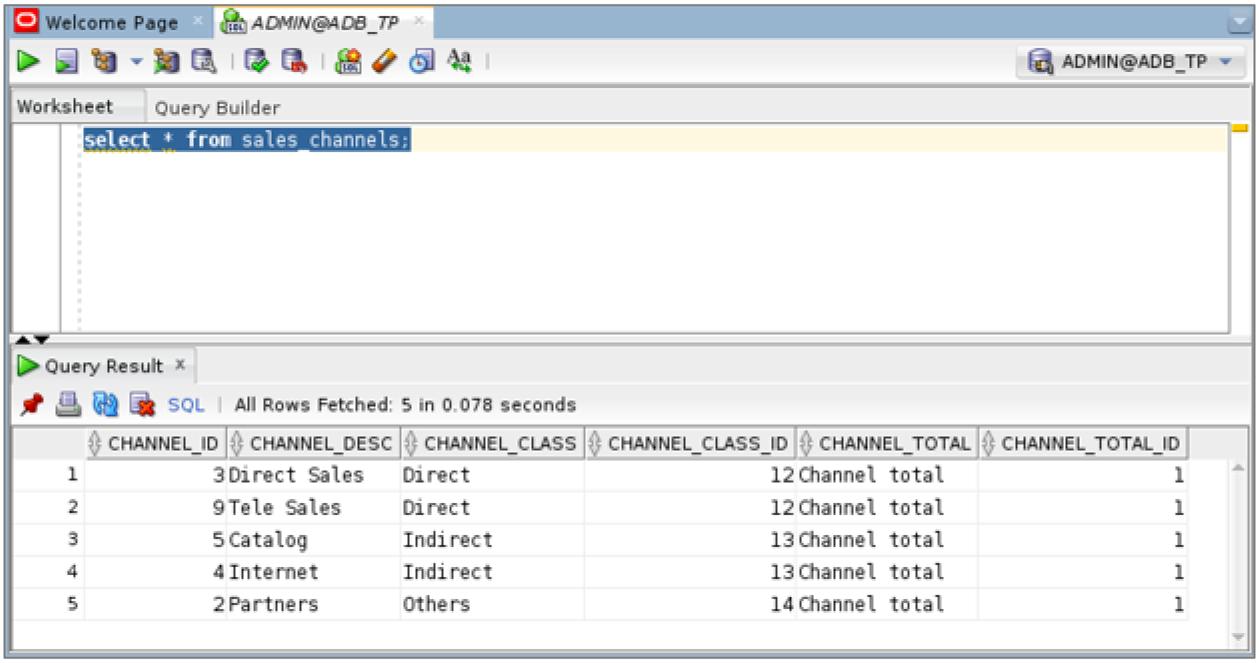


Import Data

Import Data into table SALES_CHANNELS from file channels.csv . Task successful and import committed.

- Your source file is now loaded into the autonomous database. You can run a query on the table to validate rows returned.

```
SELECT * FROM SALES_CHANNELS;
```



```
select * from sales_channels;
```

All Rows Fetched: 5 in 0.078 seconds

	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

- You have successfully loaded data to your autonomous database using Oracle SQL Developer Data Import Wizard.

Lab 2-2: Loading Data from Object Storage

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 data into the autonomous database. Oracle also provides support for loading files that are located locally in your data center (as you have seen in the previous step), but when using this method of data loading you should factor in the transmission speeds across the Internet which may be significantly slower.

Objectives

- Learn to upload files to the OCI Object Storage
- Learn to load data from the Object Store into the autonomous database

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** database.
- If you are not using the instructor supplied lab VM, ensure that your laptop/desktop has the following software installed:
  - **Oracle SQL Developer** (version 18.3 or above)
  - A Web browser supported for Oracle Cloud
  - Download **Lab Support Files** from the **Lab Setup** section

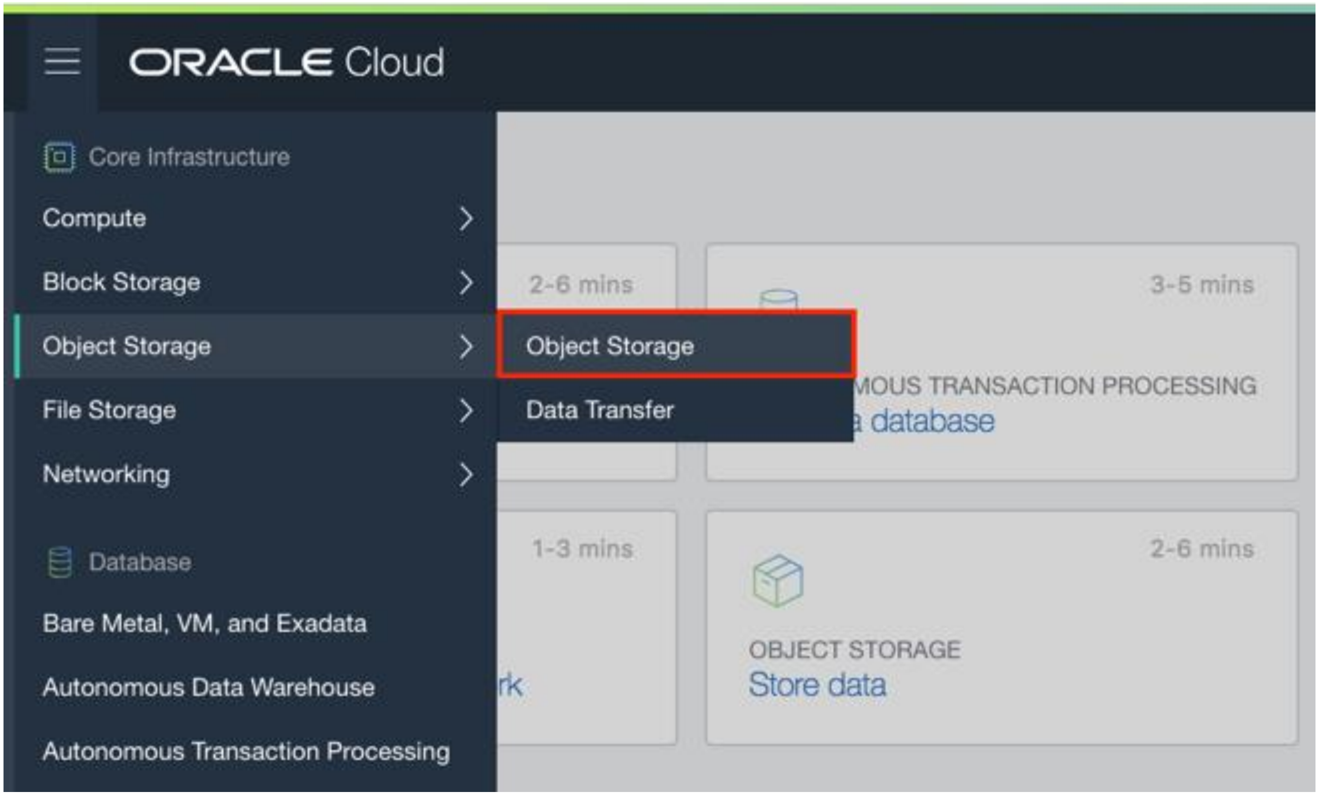
Lab Steps

STEP 1: Create an Object Storage Bucket

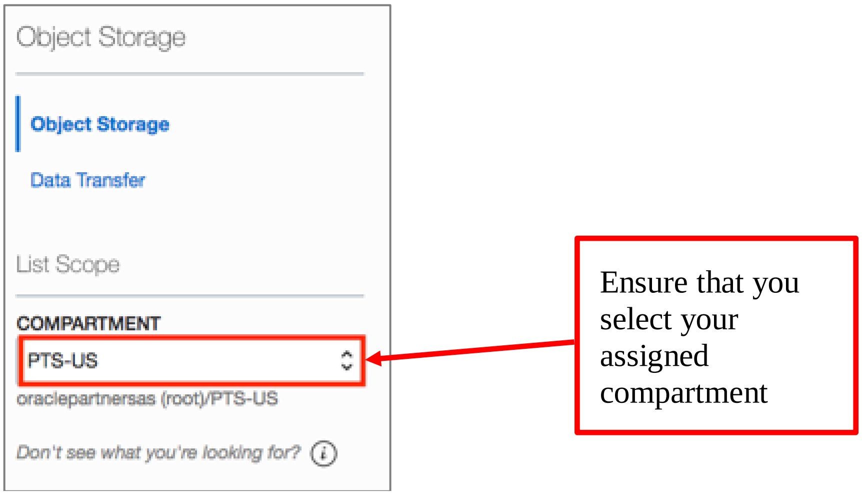
One of the first steps is to create a **Storage Bucket** in **OCI Object Storage** and upload files you plan on loading to the storage bucket.

> **Note:** In OCI Object Storage, a **Bucket** is the terminology for a container of multiple files (similar to a folder).

- Login to your OCI console with the instructions from earlier labs and click on the hamburger menu on the top-left and select **Object Storage -> Object Storage**.

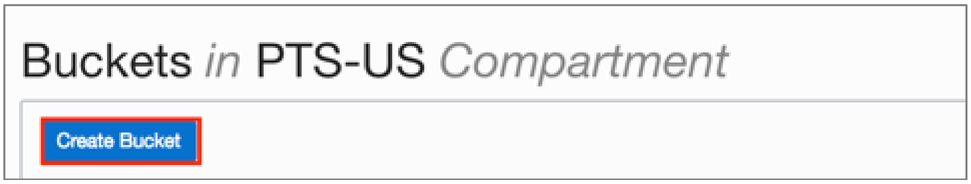


- To create a new **Bucket,** select a **Compartment** from the drop-down (choose your assigned compartment).



Ensure that you select your assigned compartment

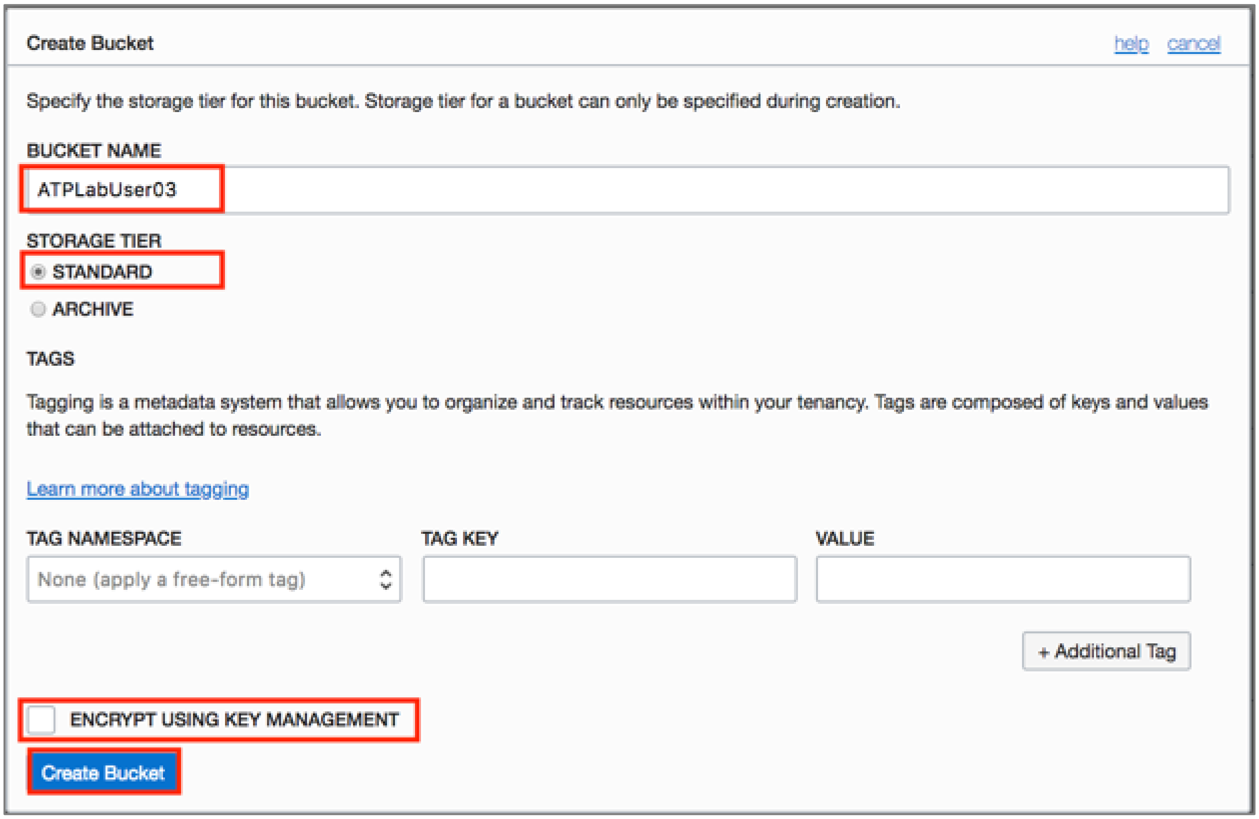
- Click on **Create Bucket** to create the object storage bucket.



- In the **Create Bucket** dialog box, enter the following:
  - **Bucket Name**: ATPLabUser<XX> (where <XX> is a unique number that no one has used)

  - **Storage Tier**: Standard
  - **Encrypt Using Key Management**: Unchecked
- Click on **Create Bucket.**

> **Note:** Bucket names must be unique within the namespace and cannot be nested. The name cannot be changed from the Console. The name can contain letters, numbers, dashes, and periods. Buckets do not have Oracle-assigned Oracle Cloud Identifiers (OCIDs).



Create Bucket

Specify the storage tier for this bucket. Storage tier for a bucket can only be specified during creation.

BUCKET NAME: ATPLabUser03

STORAGE TIER

● STANDARD

○ ARCHIVE

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:

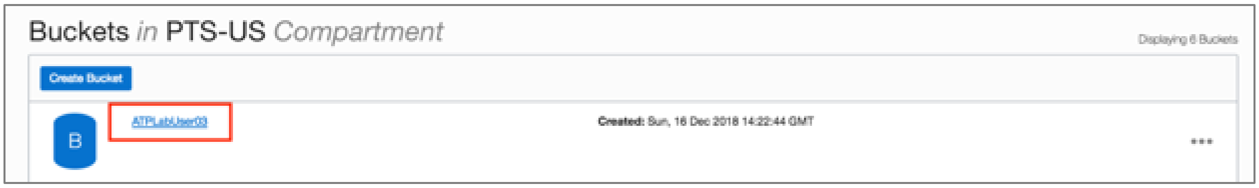
VALUE:

□ ENCRYPT USING KEY MANAGEMENT

STEP 2: Upload the Customers Sample Data File

You can now upload the **customers.csv** file to the bucket.

- Click on your **bucket name** to view details:



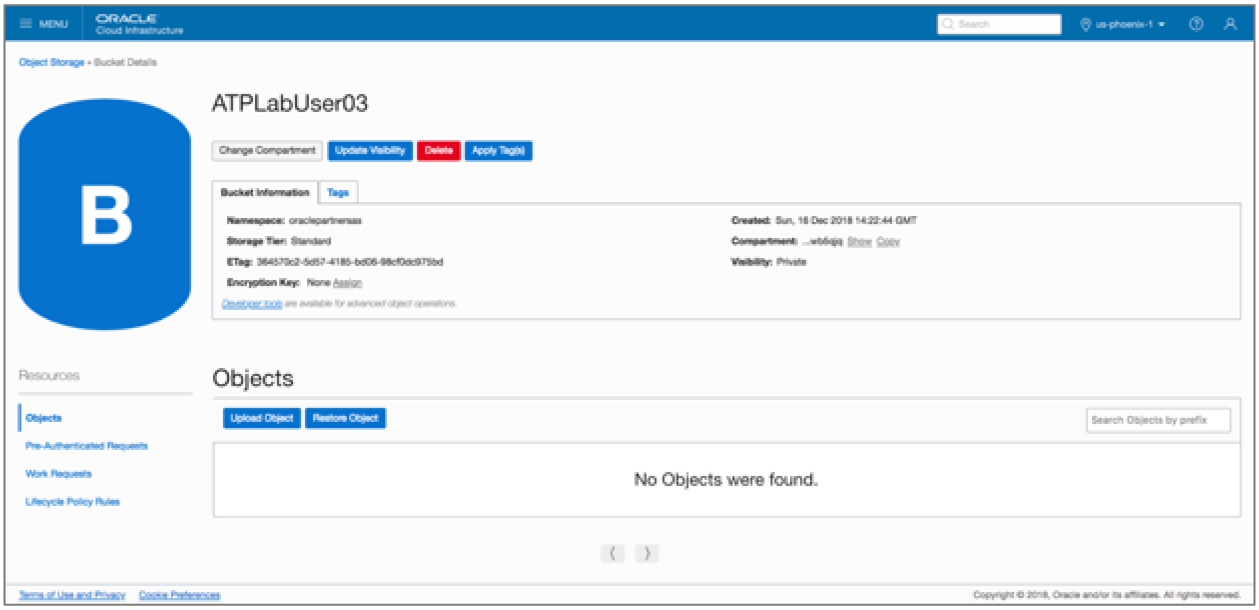
Buckets *in* PTS-US *Compartment*

Displaying 6 Buckets

ATPLabUser03

Created: Sun, 16 Dec 2018 14:22:44 GMT

- You will be presented the **Bucket Details** page.



ATPLabUser03

Namespace: oraclepartnersas

Storage Tier: Standard

ETag: 384570c2-5d57-4185-bd06-98cf0dc975bd

Encryption Key: None Assign

Developer tools are available for advanced object operations.

Created: Sun, 16 Dec 2018 14:22:44 GMT

Compartment: ...wb6qjq Show Copy

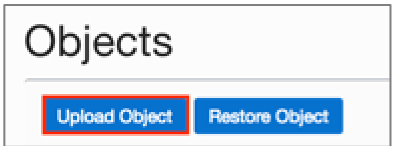
Visibility: Private

Objects

No Objects were found.

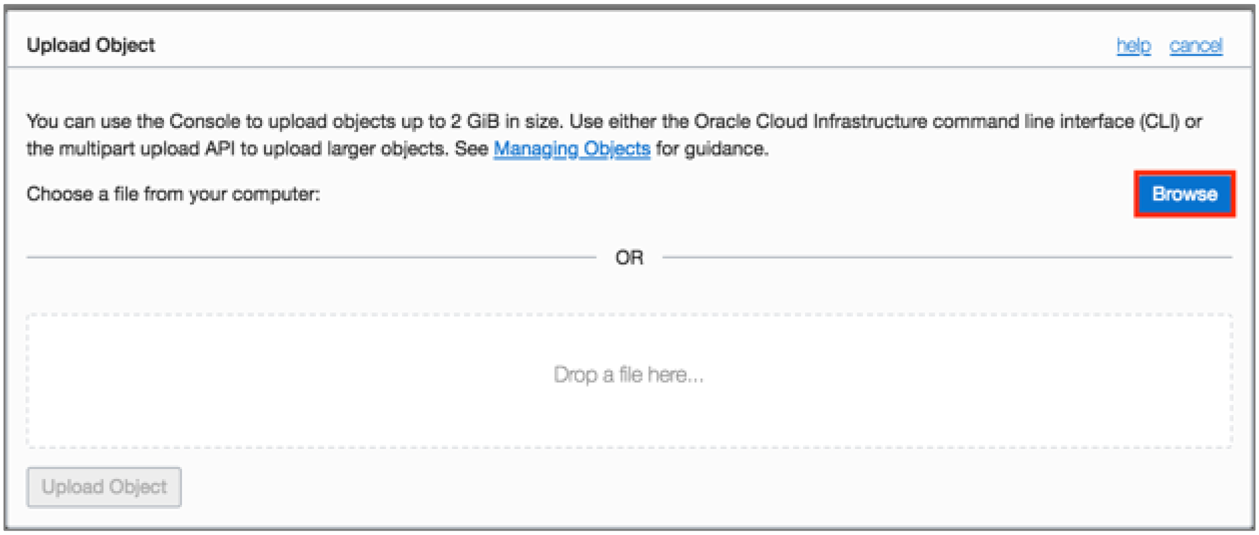
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Click on the **Upload Object** button.



- Use the **Browse** button to locate the `customers.csv` file located in `/home/oracle/labfiles` folder in the lab VM (or from a locally downloaded location on your computer).

Note: Alternatively, you may also drag-and-drop the file into this pane.

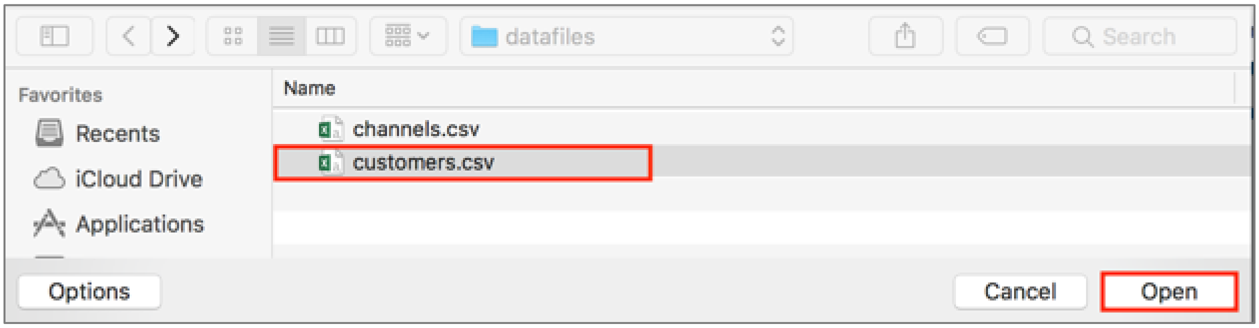


Upload Object

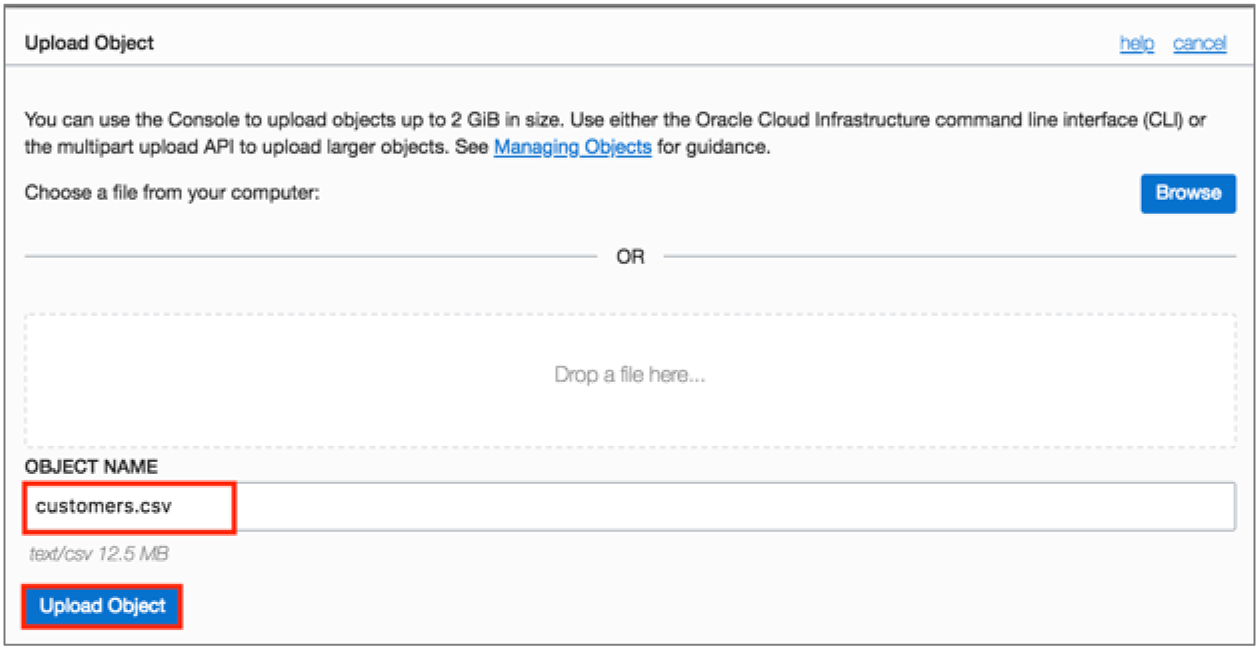
You can use the Console to upload objects up to 2 GiB in size. Use either the Oracle Cloud Infrastructure command line interface (CLI) or the multipart upload API to upload larger objects. See Managing Objects for guidance.

Choose a file from your computer:

- Select the **customers.csv** file and click **Open**.



- Click **Upload Object**



Upload Object

You can use the Console to upload objects up to 2 GiB in size. Use either the Oracle Cloud Infrastructure command line interface (CLI) or the multipart upload API to upload larger objects. See Managing Objects for guidance.

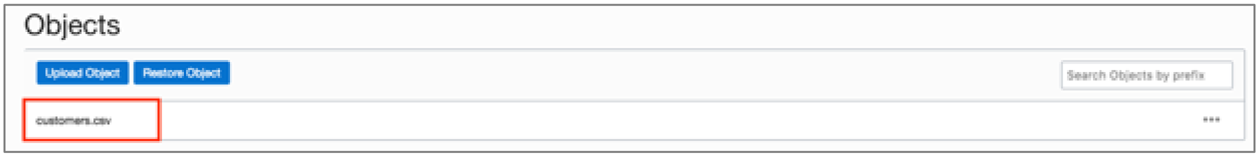
Choose a file from your computer:

OBJECT NAME

customers.csv

text/csv 12.5 MB

- Once the file is uploaded, it will be listed under **Objects** as follows:



STEP 3: Construct the URL of the File

Construct the URL that points to the location of the customers.csv file staged in the OCI Object Storage. The URL is structured as follows. The values for you to specify are in bold:

```
https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/<tenant_name>/<bucket_n
ame>/<file_name>
```

- **region_name** : The region you have created the Object Storage **Bucket** (i.e. the **Region** supplied to you by the instructor). Typically, this would be **us-phoenix-1**, **us-ashburn-1**, etc.
- **tenant_name** : The **Tenant** name supplied to you by the instructor (**in lower case**)
- **bucket_name** : The bucket name

- **file_name** : **customers.csv**

The below example of the constructed URL, the region name is **us-phoenix-1**, the tenant name is **oraclepartnersas**, and the bucket name is **ATPLabUser03**, hence the URL of the **customers.csv** file is:

```
https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas/ATPLabUser03/customers.csv
```

- Your URL would be different, so please modify accordingly and **Save** the URL to a notepad. We will use this URL at a later step.

STEP 4: Generate an Auth Token

An **Auth Token** is an Oracle-generated token string that you can use to authenticate with third-party APIs that do not support Oracle Cloud Infrastructure's signature-based authentication. For example, use an Auth Token to authenticate with a Swift client with the Object Storage Service.

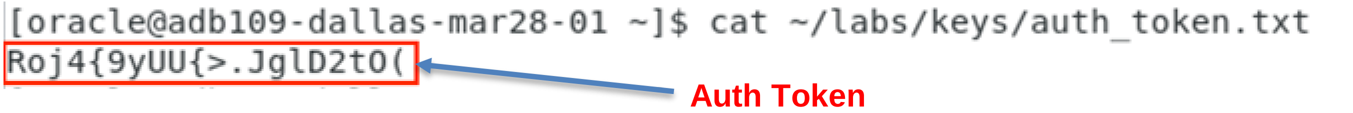
The auth token is associated with the user's Console login. Auth tokens never expire. A user can have up to two auth tokens at a time.

> **Note:** The token is always an Oracle-generated string and you can't change it to a string of your choice.

- Your instructor may have pre-generated an Auth Token for your OCI login. This will be the case when you are sharing the login with other students.
- To check if the Auth Token has been generated. Start a **Terminal** session in the lab VM.

- Check the contents of **auth_token.txt** file. If it is seeded with a value, then you would use this as the Auth Token in the later steps.

```
$ cat ~/labs/keys/auth_token.txt
```



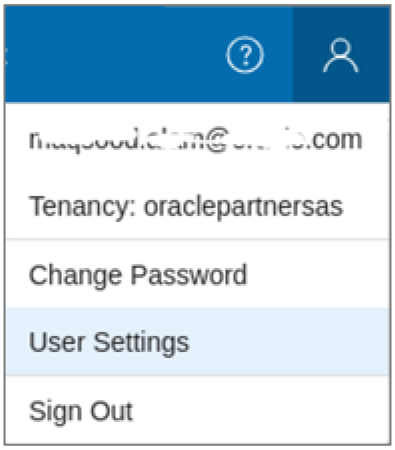
```
[oracle@adb109-dallas-mar28-01 ~]$ cat ~/labs/keys/auth_token.txt
Roj4{9yUU{>.JglD2tO(
```

Auth Token

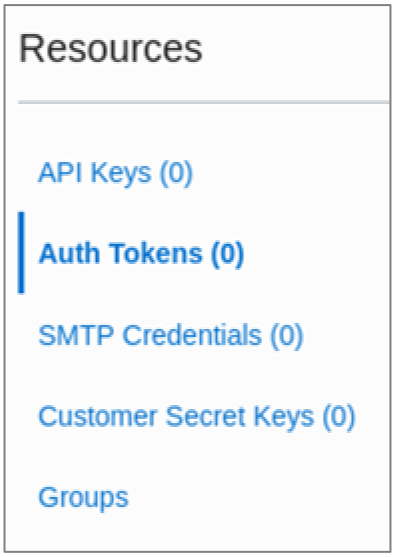
STOP!

If you have located the Auth Token from the above steps, skip the remaining instructions and proceed directly to Step 5.

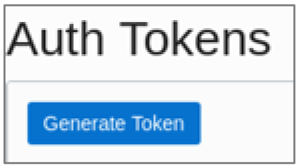
- If the Auth Token has not been generated for you, create a new auth token as follows:
- In the **top-right** corner of the **Console**, open the **User** menu and then click **User Settings** to view the details.



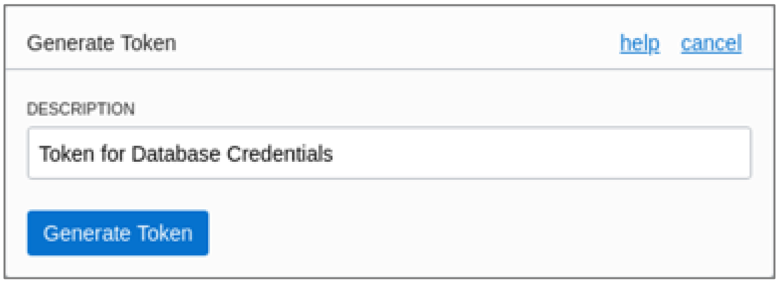
- Click on **Auth Tokens** under **Resources**.



- On the **Auth Tokens** page, click **Generate Token**. Note that you can only have two Auth Tokens and you need to delete an existing one before generating another one.



- Enter a friendly description for the auth token. Avoid entering confidential information.

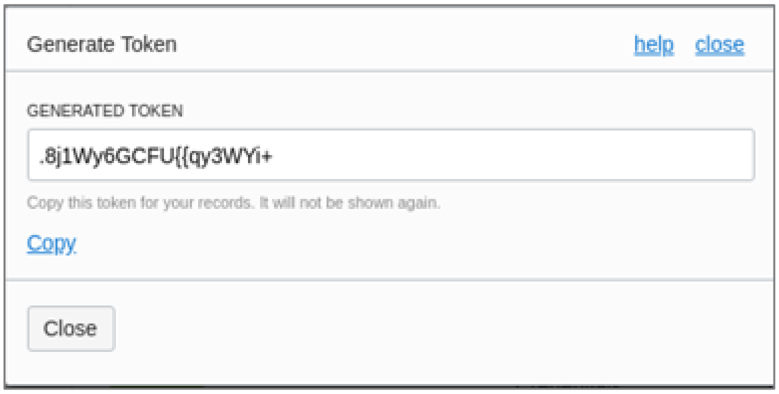


Generate Token

DESCRIPTION

Token for Database Credentials

- Click **Generate Token**. The new auth token is displayed.



Generate Token

GENERATED TOKEN

.8j1Wy6GCFU{{qy3WYi+

Copy this token for your records. It will not be shown again.

- **Copy** the auth token immediately to a secure location from where you can retrieve it later, because you won't see the auth token again in the Console.

- Close the **Generate Token** dialog.

STEP 5: Create Database Credential

Create a database credential object that holds the credentials of the object store (i.e. where your data is staged). The credential information is stored encrypted in the database and only usable for your user schema.

- Open **SQL Developer** and connect to your autonomous database as the **ADMIN** user.

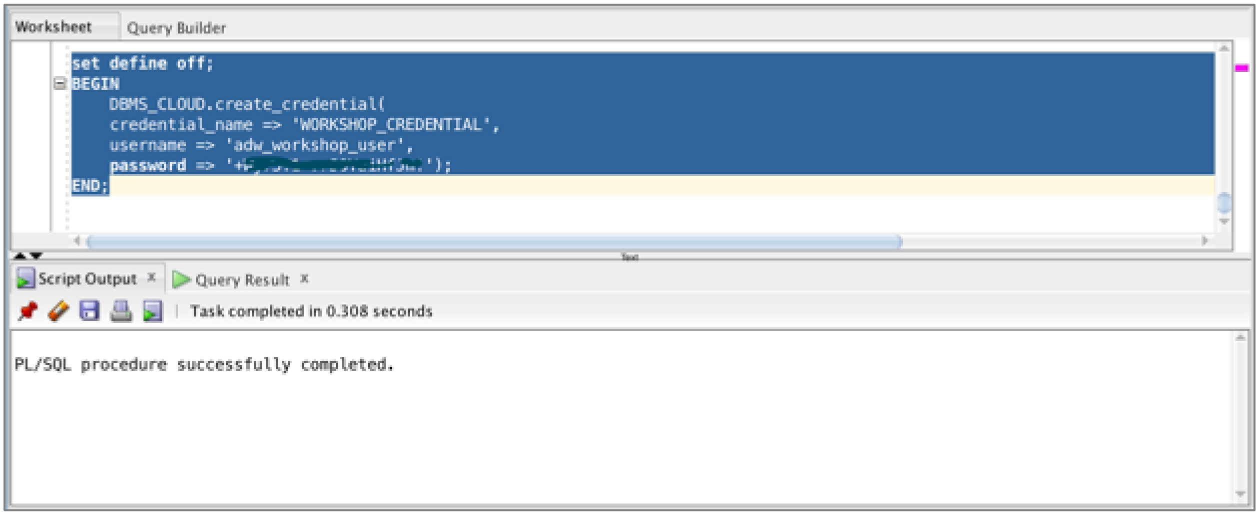
- In the worksheet, run the following **CREATE_CREDENTIAL** statement to create the **WORKSHOP_CREDENTIAL** credential object while replacing the **<Auth Token>** with the values from the previous step. Also, the **<OCI Lab User>** is the OCI User that you used to sign-in to the OCI Console (**NOT** the ADMIN user).

> **Note:** Run the following SQL statements by selecting them both and executing them using **Run Script (F5)**

```
SET DEFINE OFF;
BEGIN
    DBMS_CLOUD.create_credential(
    credential_name => 'WORKSHOP_CREDENTIAL',
    username => '<OCI Lab User>',
    password => '<Auth Token>');
END;
```

<OCI Lab User> is the OCI user provided to you by the instructor.

- Verify the credential object got successfully created.



```
set define off;
BEGIN
    DBMS_CLOUD.create_credential(
    credential_name => 'WORKSHOP_CREDENTIAL',
    username => 'adw_workshop_user',
    password => '[redacted]');
END;
```

Task completed in 0.308 seconds

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

- Now you are ready to load data from the Object Store using the credentials just created.

STEP 6: Load Data from the Object Store to the Autonomous Database

Autonomous database provides a new PL/SQL package called **DBMS_CLOUD** to load data from files from the cloud storage into your database. The DBMS_CLOUD package supports loading data files from the following Cloud sources: Oracle Cloud Infrastructure Object Storage, Oracle Cloud Infrastructure Object Storage Classic and Amazon AWS S3.

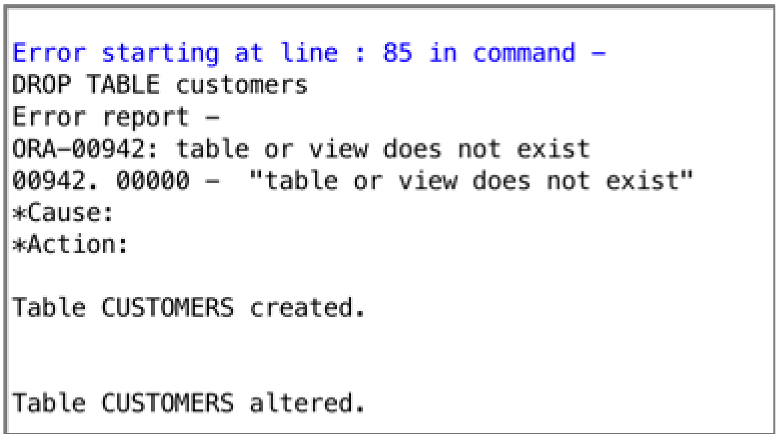
- The **COPY_DATA** procedure of **DBMS_CLOUD** package requires that target tables must already exist in your autonomous database.

- Connect as the ADMIN user in SQL Developer and create the target **CUSTOMERS** table.

- Copy the table creation DDL from the file **`/home/oracle/labfiles/create_customers.sql`** (or from a locally downloaded location on your computer).

```
DROP TABLE customers;

CREATE TABLE customers (
cust_id NUMBER NOT NULL,
cust_first_name VARCHAR2(20) NOT NULL,
cust_last_name VARCHAR2(40) NOT NULL,
cust_gender CHAR(1) NOT NULL,
cust_year_of_birth NUMBER(4) NOT NULL,
cust_marital_status VARCHAR2(20),
cust_street_address VARCHAR2(40) NOT NULL,
cust_postal_code VARCHAR2(10) NOT NULL,
cust_city VARCHAR2(30) NOT NULL,
cust_city_id NUMBER NOT NULL,
cust_state_province VARCHAR2(40) NOT NULL,
cust_state_province_id NUMBER NOT NULL,
country_id NUMBER NOT NULL,
cust_main_phone_number VARCHAR2(25) NOT NULL,
cust_income_level VARCHAR2(30),
cust_credit_limit NUMBER,
cust_email VARCHAR2(50),
cust_total VARCHAR2(14) NOT NULL,
cust_total_id NUMBER NOT NULL,
cust_src_id NUMBER,
cust_eff_from DATE,
cust_eff_to DATE,
cust_valid VARCHAR2(1));

ALTER TABLE customers ADD CONSTRAINT customers_pk PRIMARY KEY (cust_id);
```

- Verify the table was successfully created (ignore the ORA-00942 during DROP).



```
Error starting at line : 85 in command -
DROP TABLE customers
Error report -
ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:

Table CUSTOMERS created.


Table CUSTOMERS altered.
```

- Now run the **DBMS_CLOUD.COPY_DATA** procedure to copy the data staged in your object store to your autonomous database tables.

- Modify the SQL provided below by replacing the highlighted values as appropriate:

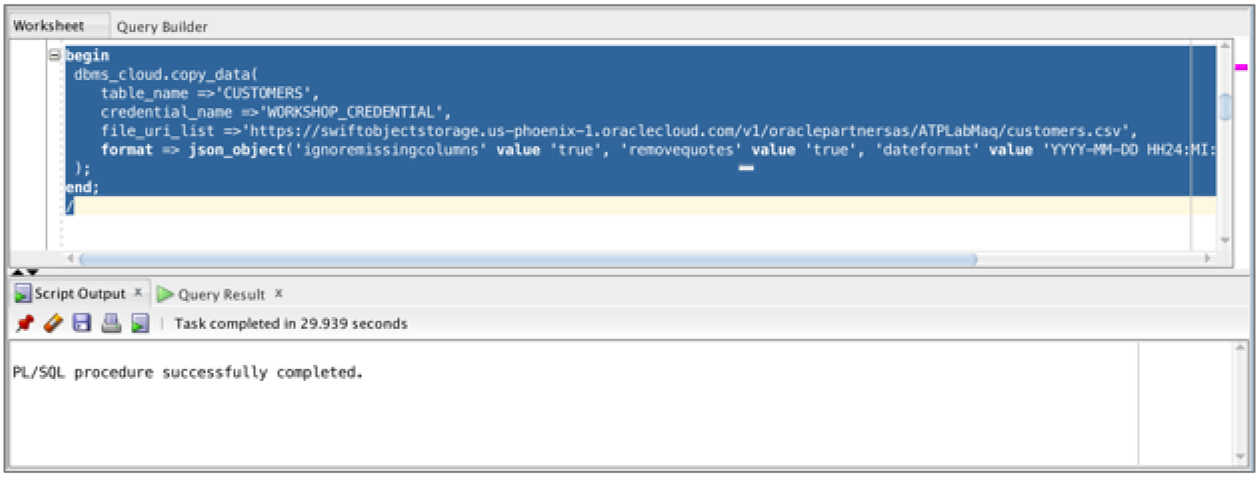
  - **table_name** : CUSTOMERS

  - **credential_name** : WORKSHOP_CREDENTIAL

  - **file_uri_list** : <your_file_uri_list> => This is the URL you saved earlier in **Construct the URL of the File** section.

```
begin
dbms_cloud.copy_data(
table_name=>'CUSTOMERS',
credential_name=>'WORKSHOP_CREDENTIAL',
file_uri_list=>'<your_file_uri_list>',
format=>json_object('ignoremissingcolumns' value 'true','removequotes' value 'true',
'dateformat' value 'YYYY-MM-DD HH24:MI:SS','blankasnull' value 'true')
);
end;
/
```

- Execute the above PL/SQL block and verify the **COPY_DATA** was successful.



```
begin
 dbms_cloud.copy_data(
    table_name =>'CUSTOMERS',
    credential_name =>'WORKSHOP_CREDENTIAL',
    file_uri_list =>'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas/ATPLabMaq/customers.csv',
    format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true', 'dateformat' value 'YYYY-MM-DD HH24:MI:
 );
end;
/
```

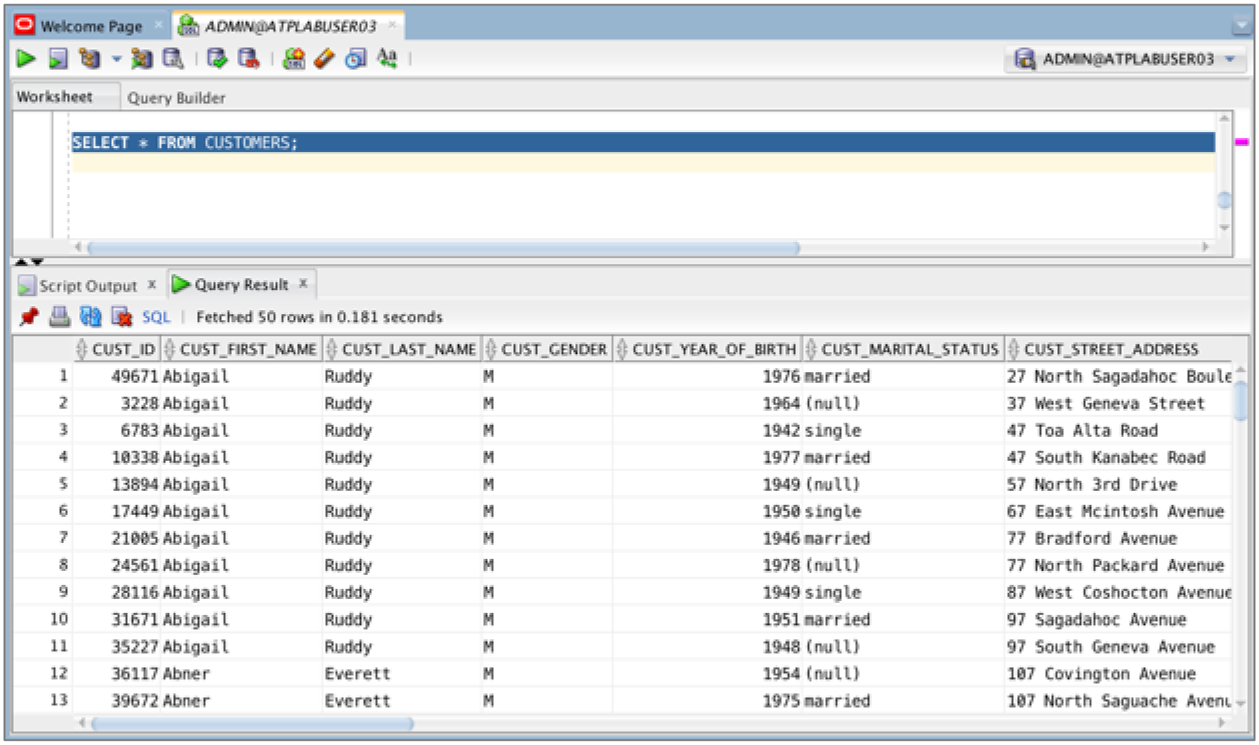
Task completed in 29.939 seconds

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

STEP 7: Verify the Data Loads

- Query the **CUSTOMERS** table and validate rows loaded:

```
SELECT * FROM CUSTOMERS;
```



```
SELECT * FROM CUSTOMERS;
```

Fetched 50 rows in 0.181 seconds

	CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_GENDER	CUST_YEAR_OF_BIRTH	CUST_MARITAL_STATUS	CUST_STREET_ADDRESS
1	49671	Abigail	Ruddy	M	1976	married	27 North Sagadahoc Boule
2	3228	Abigail	Ruddy	M	1964	(null)	37 West Geneva Street
3	6783	Abigail	Ruddy	M	1942	single	47 Toa Alta Road
4	10338	Abigail	Ruddy	M	1977	married	47 South Kanabec Road
5	13894	Abigail	Ruddy	M	1949	(null)	57 North 3rd Drive
6	17449	Abigail	Ruddy	M	1950	single	67 East Mcintosh Avenue
7	21005	Abigail	Ruddy	M	1946	married	77 Bradford Avenue
8	24561	Abigail	Ruddy	M	1978	(null)	77 North Packard Avenue
9	28116	Abigail	Ruddy	M	1949	single	87 West Coshocton Avenue
10	31671	Abigail	Ruddy	M	1951	married	97 Sagadahoc Avenue
11	35227	Abigail	Ruddy	M	1948	(null)	97 South Geneva Avenue
12	36117	Abner	Everett	M	1954	(null)	107 Covington Avenue
13	39672	Abner	Everett	M	1975	married	107 North Saguache Avenu

- Data loads performed by DBMS_CLOUD are logged in the following dictionary tables:
    - **DBA_LOAD_OPERATIONS** : Shows all load operations.
    - **USER_LOAD_OPERATIONS** : Shows the load operations in your schema.
- Query thee above tables to see information about ongoing and completed data loads. For example:

```
SELECT table_name, status, rows_loaded, logfile_table, badfile_table
FROM user_load_operations WHERE type = 'COPY';
```

- Examine the results. The log and bad files are accessible as tables.

```
TABLE_NAME       STATUS         ROWS_LOADED  LOGFILE_TABLE   BADFILE_TABLE
----------       ------------   -----------  -------------   -------------
CUSTOMERS        COMPLETED            55500  COPY$1_LOG      COPY$1_BAD
```

- Validate using **LOGFILE_TABLE** and **BADFILE_TABLE** (COPY$1_LOG and COPY$1_BAD from above).

```
SELECT * FROM copy$1_log;
SELECT * FROM copy$1_bad;
```

- You have successfully loaded data to your autonomous database using the Object Storage.

Lab 2-3: Loading Data Using SQL Loader

You can use **Oracle SQL Loader** to load data from local files on your client machine to the autonomous database.

> **Note:** SQL Loader may be suitable for loading small volumes of data, as the load performance depends on the network bandwidth between your client and the ADB. For large data loads, Oracle recommends loading data from the OCI **Object Storage**.

Objectives

- Generate SQL*Loader scripts using **SQL Developer**
- Load a Local CSV file to the autonomous database using **SQL Loader** scripts

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** database.
- This lab needs to be completed using the lab VM.
- Ensure that you have previously downloaded the secure credentials wallet to **~/Downloads** directory

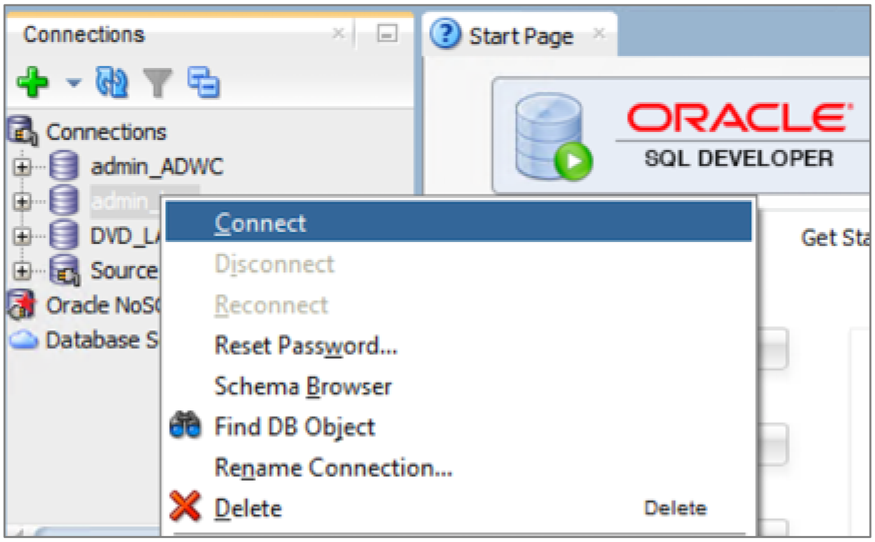
Lab Steps

STEP 1: Generate SQL Loader Scripts Using SQL Developer

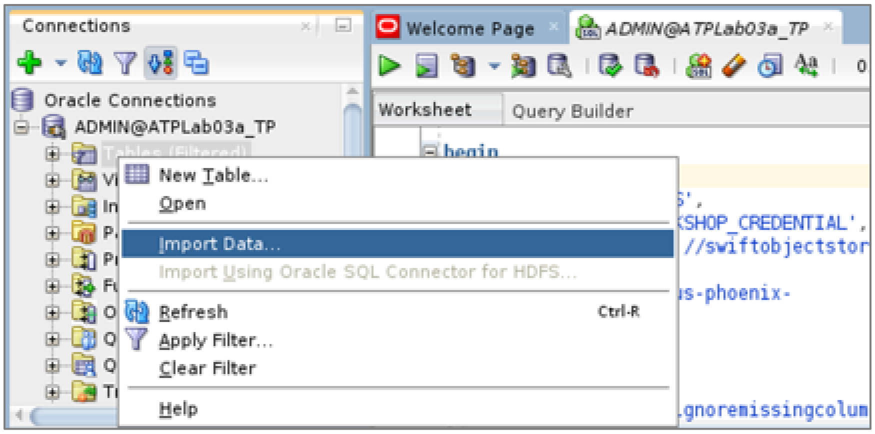
SQL Loader requires a control file that specifies the data definition and structure of input data. You may manually create the control file or use **Oracle SQL Developer** to generate them. We will use the latter method of control file generation in this lab.

- Sign in to the **lab VM** with the login credentials provided to you by the instructor.
- From the lab VM, start **SQL Developer** and connect to your autonomous database as the **ADMIN** user.

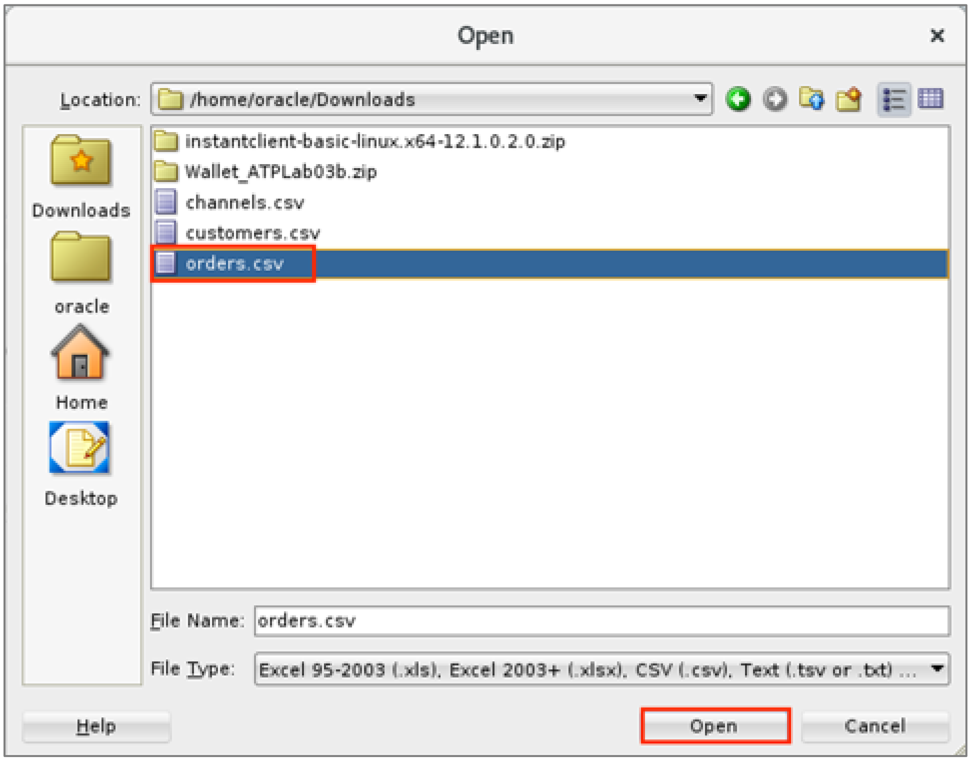
> **Note:** You may use an existing connection from a previous lab.



- In the left pane, right-click **Tables** and select **Import Data**.



- The **Data Import** wizard will pop-up. Click **Browse** and locate the **`orders.csv`** file from **`/home/oracle/labfiles`** (or from a locally downloaded location on your computer) and click **Open**.



Open

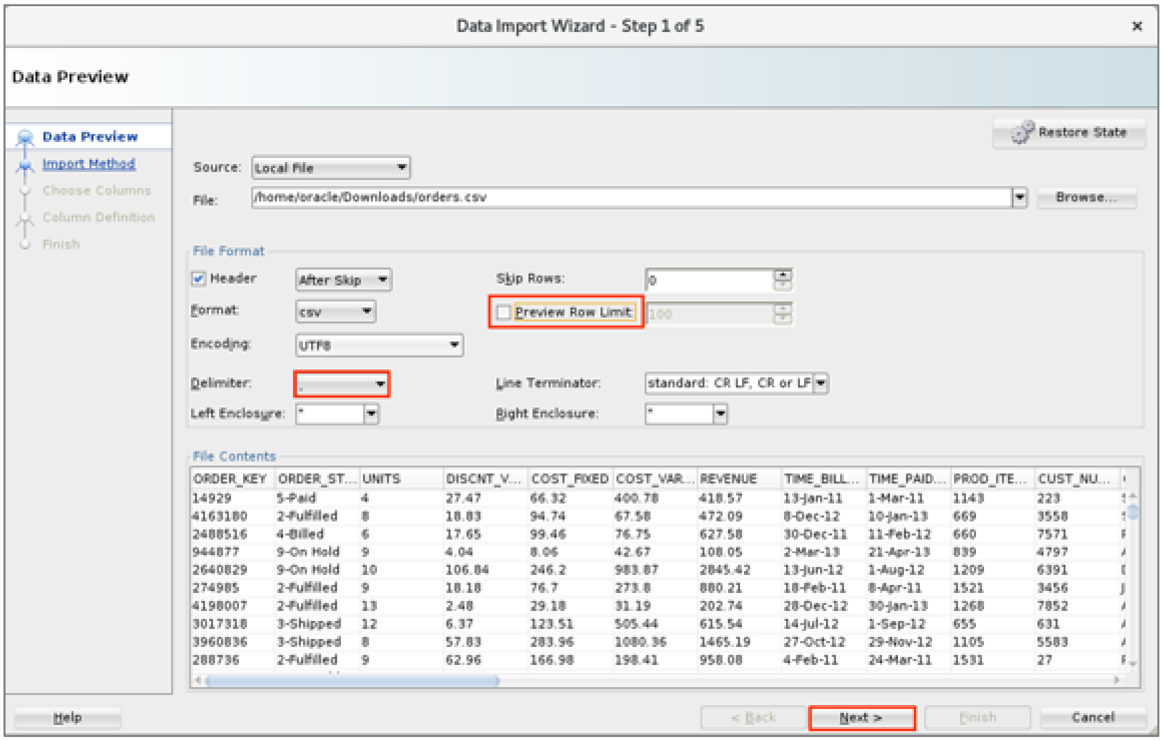
Location: /home/oracle/Downloads

- instantclient-basic-linux.x64-12.1.0.2.0.zip
- Wallet_ATPLab03b.zip
- channels.csv
- customers.csv
- orders.csv

File Name: orders.csv

File Type: Excel 95-2003 (.xls), Excel 2003+ (.xlsx), CSV (.csv), Text (.tsv or .txt) ...

- Uncheck the **Preview Row Limit** and ensure the **Delimiter** is “**,**” and click **Next**.



Data Import Wizard - Step 1 of 5

Data Preview

Source: Local File

File: /home/oracle/Downloads/orders.csv

File Format

☑ Header After Skip

Skip Rows: 0

Format: csv

□ Preview Row Limit: 100

Encoding: UTF8

Delimiter: ,

Line Terminator: standard: CR LF, CR or LF

Left Enclosure: "

Right Enclosure: "

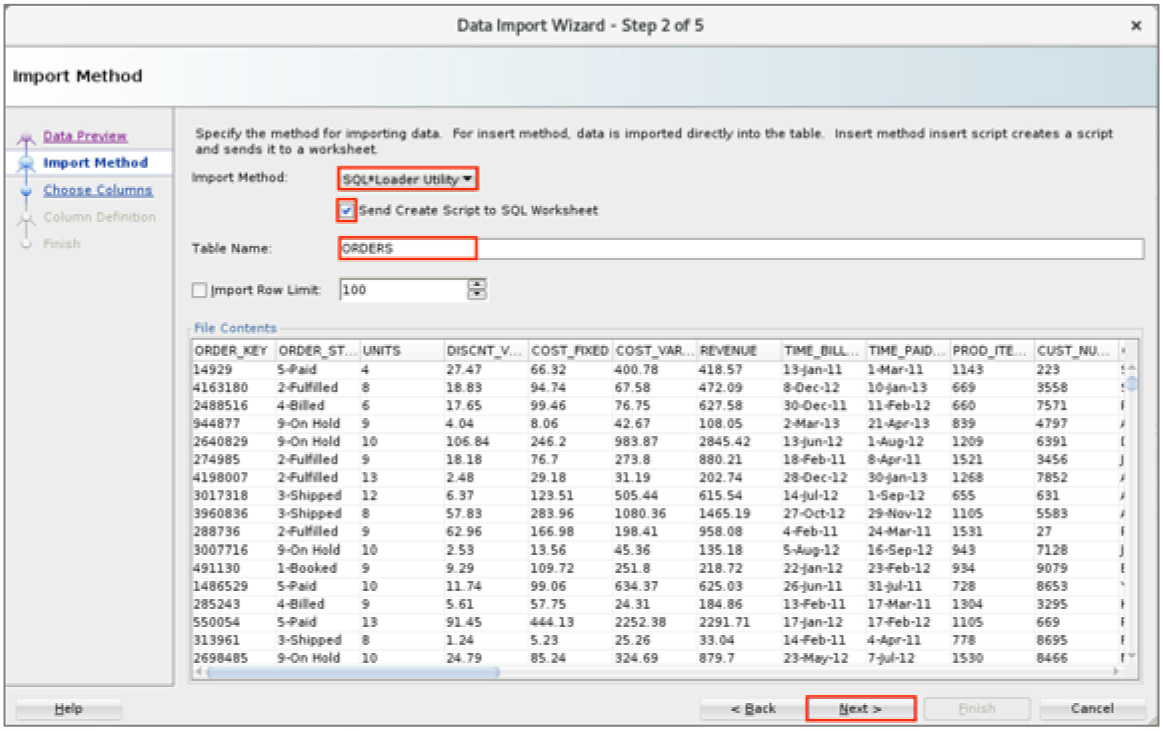
File Contents

ORDER_KEY	ORDER_ST...	UNITS	DISCNT_V...	COST_FIXED	COST_VAR...	REVENUE	TIME_BILL...	TIME_PAID...	PROD_ITE...	CUST_NU...
14929	5-Paid	4	27.47	66.32	400.78	418.57	13-Jan-11	1-Mar-11	1143	223
4163180	2-Fulfilled	8	18.83	94.74	67.58	472.09	8-Dec-12	10-Jan-13	669	3558
2488516	4-Billed	6	17.65	99.46	76.75	627.58	30-Dec-11	11-Feb-12	660	7571
944877	9-On Hold	9	4.04	8.06	42.67	108.05	2-Mar-13	21-Apr-13	839	4797
2640829	9-On Hold	10	106.84	246.2	983.87	2845.42	13-Jun-12	1-Aug-12	1209	6391
274985	2-Fulfilled	9	18.18	76.7	273.8	880.21	18-Feb-11	8-Apr-11	1521	3456
4198007	2-Fulfilled	13	2.48	29.18	31.19	202.74	28-Dec-12	30-Jan-13	1268	7852
3017318	3-Shipped	12	6.37	123.51	505.44	615.54	14-Jul-12	1-Sep-12	655	631
3960836	3-Shipped	8	57.83	283.96	1080.36	1465.19	27-Oct-12	29-Nov-12	1105	5583
288736	2-Fulfilled	9	62.96	166.98	198.41	958.08	4-Feb-11	24-Mar-11	1531	27

- In the **Import Method** choose SQL Loader Utility. Enter a table name where you like the data to be loaded, e.g. **ORDERS**.

Note: The Table need not be pre-created at this time. The **Create Table DDL** will be generated as a part of this process and you can run it to create the table.

- Ensure that **Send Create Script to SQL Worksheet** is **Checked**. Click **Next**.



Data Import Wizard - Step 2 of 5

Import Method

Specify the method for importing data. For insert method, data is imported directly into the table. Insert method insert script creates a script and sends it to a worksheet.

Import Method: SQL*Loader Utility

☑ Send Create Script to SQL Worksheet

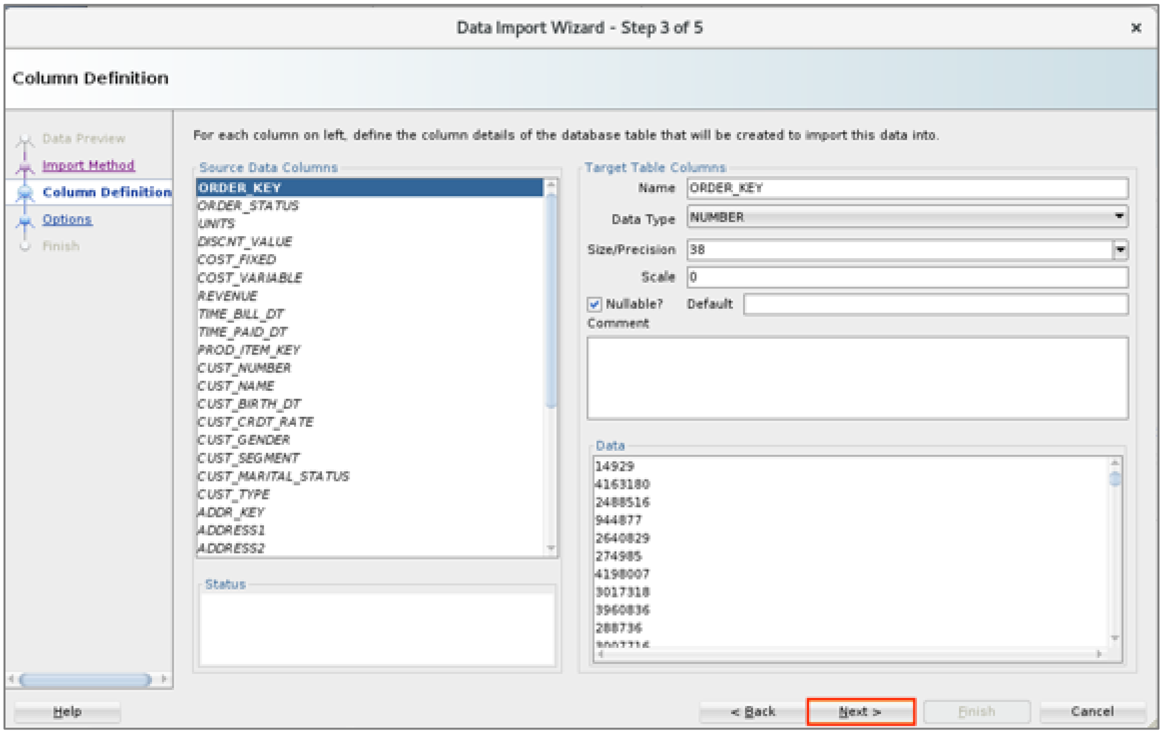
Table Name: ORDERS

☐ Import Row Limit: 100

File Contents

ORDER_KEY	ORDER_ST...	UNITS	DISCNT_V...	COST_FIXED	COST_VAR...	REVENUE	TIME_BILL...	TIME_PAID...	PROD_ITE...	CUST_NU...
14929	5-Paid	4	27.47	66.32	400.78	418.57	13-Jan-11	1-Mar-11	1143	223
4163180	2-Fulfilled	8	18.83	94.74	67.58	472.09	8-Dec-12	10-Jan-13	669	3558
2488516	4-Billed	6	17.65	99.46	76.75	627.58	30-Dec-11	11-Feb-12	660	7571
944877	9-On Hold	9	4.04	8.06	42.67	108.05	2-Mar-13	21-Apr-13	839	4797
2640829	9-On Hold	10	106.84	246.2	983.87	2845.42	13-Jun-12	1-Aug-12	1209	6391
274985	2-Fulfilled	9	18.18	76.7	273.8	880.21	18-Feb-11	8-Apr-11	1521	3456
4198007	2-Fulfilled	13	2.48	29.18	31.19	202.74	28-Dec-12	30-Jan-13	1268	7852
3017318	3-Shipped	12	6.37	123.51	505.44	615.54	14-Jul-12	1-Sep-12	655	631
3960836	3-Shipped	8	57.83	283.96	1080.36	1465.19	27-Oct-12	29-Nov-12	1105	5583
288736	2-Fulfilled	9	62.96	166.98	198.41	958.08	4-Feb-11	24-Mar-11	1531	27
3007716	9-On Hold	10	2.53	13.56	45.36	135.18	5-Aug-12	16-Sep-12	943	7128
491130	1-Booked	9	9.29	109.72	251.8	218.72	22-Jan-12	23-Feb-12	934	9079
1486529	5-Paid	10	11.74	99.06	634.37	625.03	26-Jun-11	31-Jul-11	728	8653
285243	4-Billed	9	5.61	57.75	24.31	184.86	13-Feb-11	17-Mar-11	1304	3295
550054	5-Paid	13	91.45	444.13	2252.38	2291.71	17-Jan-12	17-Feb-12	1105	669
313961	3-Shipped	8	1.24	5.23	25.26	33.04	14-Feb-11	4-Apr-11	778	8695
2698485	9-On Hold	10	24.79	85.24	324.69	879.7	23-May-12	7-Jul-12	1530	8466

- The next screen allows you to select the columns you like to include as part of the load and also as part of the table column list for the DDL. For this lab, keep all settings as default, the effect of which is to include all columns. Click **Next**.



Data Import Wizard - Step 3 of 5

Column Definition

For each column on left, define the column details of the database table that will be created to import this data into.

Source Data Columns

ORDER_KEY
ORDER_STATUS
UNITS
DISCNT_VALUE
COST_FIXED
COST_VARIABLE
REVENUE
TIME_BILL_DT
TIME_PAID_DT
PROD_ITEM_KEY
CUST_NUMBER
CUST_NAME
CUST_BIRTH_DT
CUST_CRDT_RATE
CUST_GENDER
CUST_SEGMENT
CUST_MARITAL_STATUS
CUST_TYPE
ADDR_KEY
ADDRESS1
ADDRESS2

Status

Target Table Columns

Name: ORDER_KEY

Data Type: NUMBER

Size/Precision: 38

Scale: 0

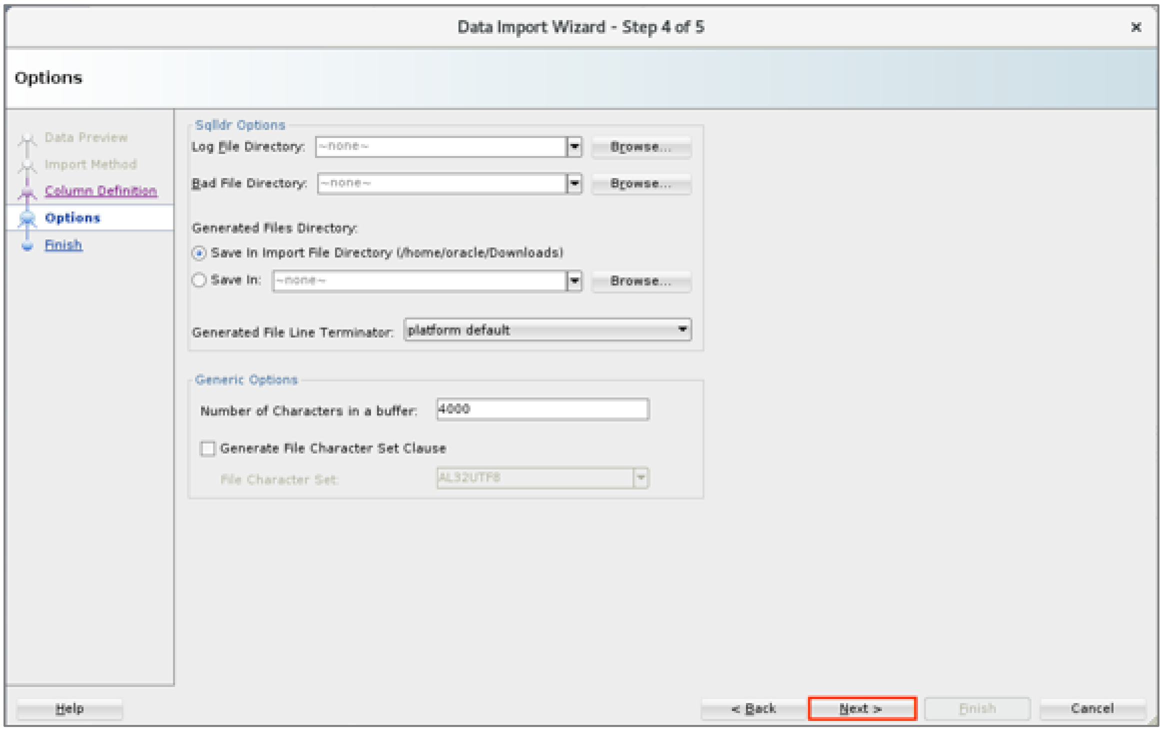
☑ Nullable? Default:

Comment:

Data

```
14929
4163180
2488516
944877
2640829
274985
4198007
3017318
3960836
288736
```

- This screen allows you to set SQL Loader options. For this exercise just note the location where the scripts will be generated. Leave the rest of the options as default. Click **Next.**



Data Import Wizard - Step 4 of 5

Options

Sqlldr Options

Log File Directory: --none--

Bad File Directory: --none--

Generated Files Directory:

● Save In Import File Directory (/home/oracle/Downloads)

○ Save In: --none--

Generated File Line Terminator: platform default

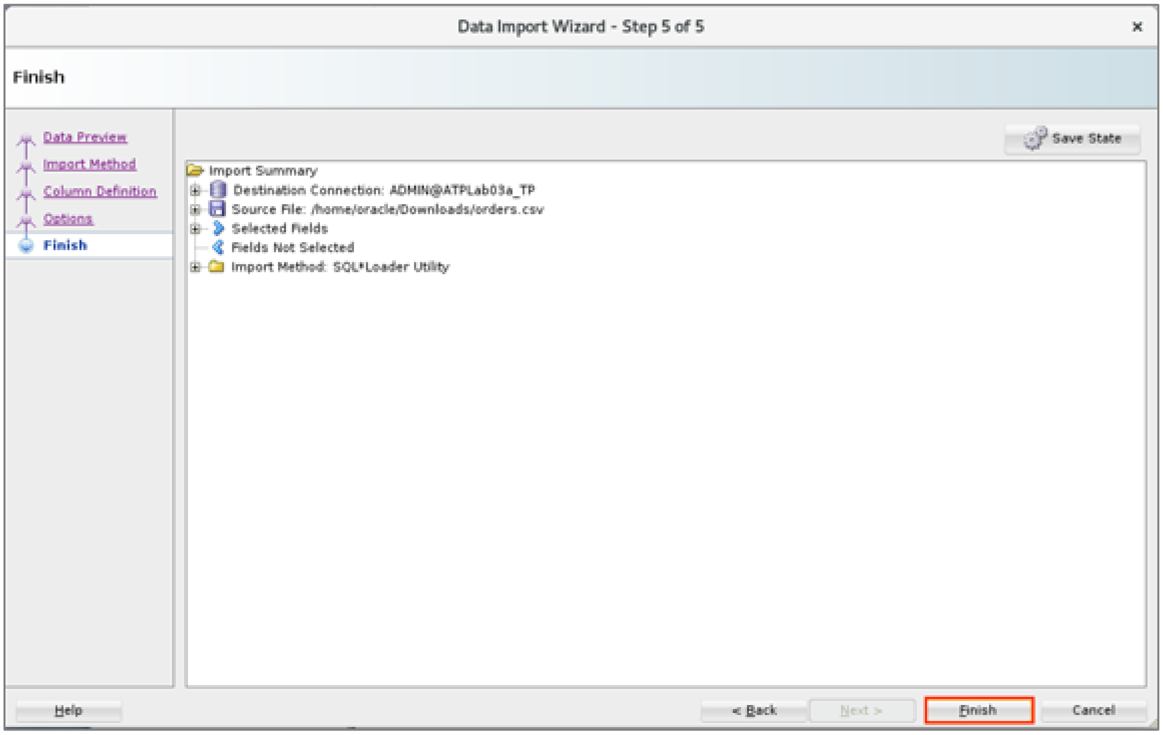
Generic Options

Number of Characters in a buffer: 4000

□ Generate File Character Set Clause

File Character Set: AL32UTF8

- On the summary screen, click **Finish**.

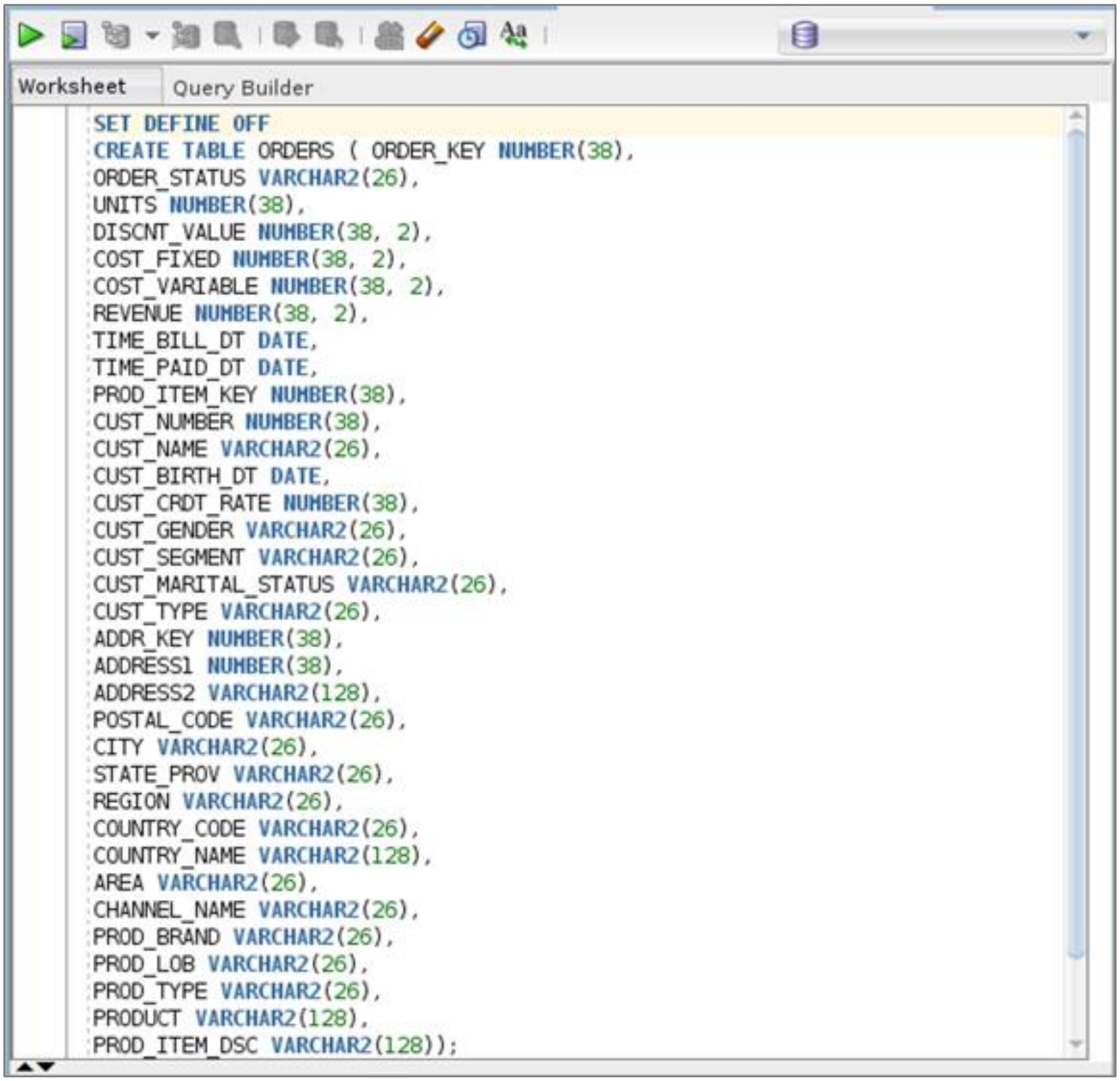


Data Import Wizard - Step 5 of 5

Finish

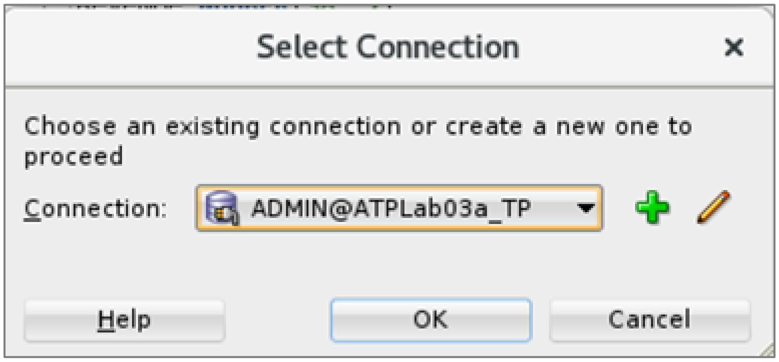
Import Summary
- Destination Connection: ADMIN@ATPLab03a_TP
- Source File: /home/oracle/Downloads/orders.csv
- Selected Fields
- Fields Not Selected
- Import Method: SQL*Loader Utility

- A new SQL Worksheet is created with the create table DDL command.



```
SET DEFINE OFF
CREATE TABLE ORDERS ( ORDER_KEY NUMBER(38),
ORDER_STATUS VARCHAR2(26),
UNITS NUMBER(38),
DISCNT_VALUE NUMBER(38, 2),
COST_FIXED NUMBER(38, 2),
COST_VARIABLE NUMBER(38, 2),
REVENUE NUMBER(38, 2),
TIME_BILL_DT DATE,
TIME_PAID_DT DATE,
PROD_ITEM_KEY NUMBER(38),
CUST_NUMBER NUMBER(38),
CUST_NAME VARCHAR2(26),
CUST_BIRTH_DT DATE,
CUST_CRDT_RATE NUMBER(38),
CUST_GENDER VARCHAR2(26),
CUST_SEGMENT VARCHAR2(26),
CUST_MARITAL_STATUS VARCHAR2(26),
CUST_TYPE VARCHAR2(26),
ADDR_KEY NUMBER(38),
ADDRESS1 NUMBER(38),
ADDRESS2 VARCHAR2(128),
POSTAL_CODE VARCHAR2(26),
CITY VARCHAR2(26),
STATE_PROV VARCHAR2(26),
REGION VARCHAR2(26),
COUNTRY_CODE VARCHAR2(26),
COUNTRY_NAME VARCHAR2(128),
AREA VARCHAR2(26),
CHANNEL_NAME VARCHAR2(26),
PROD_BRAND VARCHAR2(26),
PROD_LOB VARCHAR2(26),
PROD_TYPE VARCHAR2(26),
PRODUCT VARCHAR2(128),
PROD_ITEM_DSC VARCHAR2(128));
```

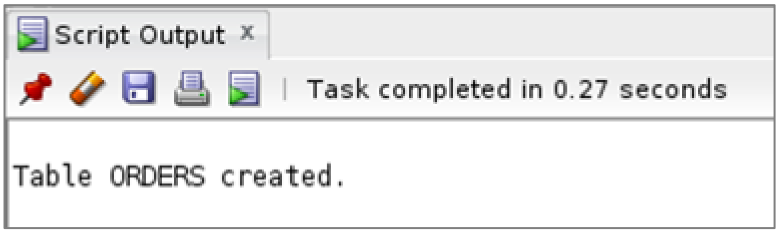
- Execute the **Create Table** script. Click **F5** or the **Run Script** button.
- You will be presented with **Select a Connection** dialog. Ensure that it points to the right autonomous database connection and click **OK**.



Choose an existing connection or create a new one to proceed

Connection: ADMIN@ATPLab03a_TP

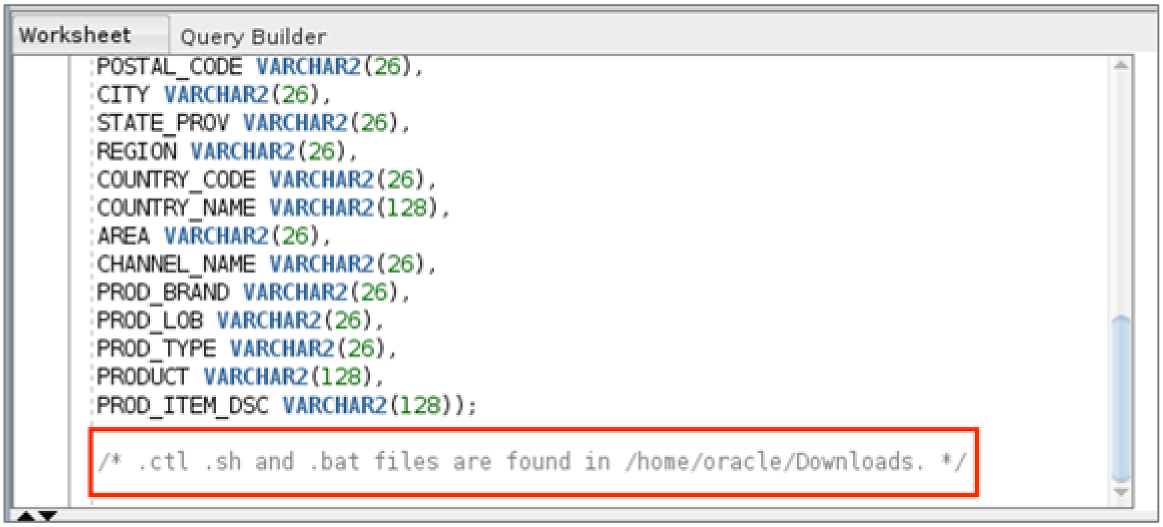
- Verify the table creation was successful.



Task completed in 0.27 seconds

```
Table ORDERS created.
```

- Scroll to the bottom of the **Worksheet** and note the location of the generated scripts.



```
POSTAL_CODE VARCHAR2(26),
CITY VARCHAR2(26),
STATE_PROV VARCHAR2(26),
REGION VARCHAR2(26),
COUNTRY_CODE VARCHAR2(26),
COUNTRY_NAME VARCHAR2(128),
AREA VARCHAR2(26),
CHANNEL_NAME VARCHAR2(26),
PROD_BRAND VARCHAR2(26),
PROD_LOB VARCHAR2(26),
PROD_TYPE VARCHAR2(26),
PRODUCT VARCHAR2(128),
PROD_ITEM_DSC VARCHAR2(128));

/* .ctl .sh and .bat files are found in /home/oracle/Downloads. */
```

- Note that your scripts were generated in **/home/oracle/Downloads** folder.

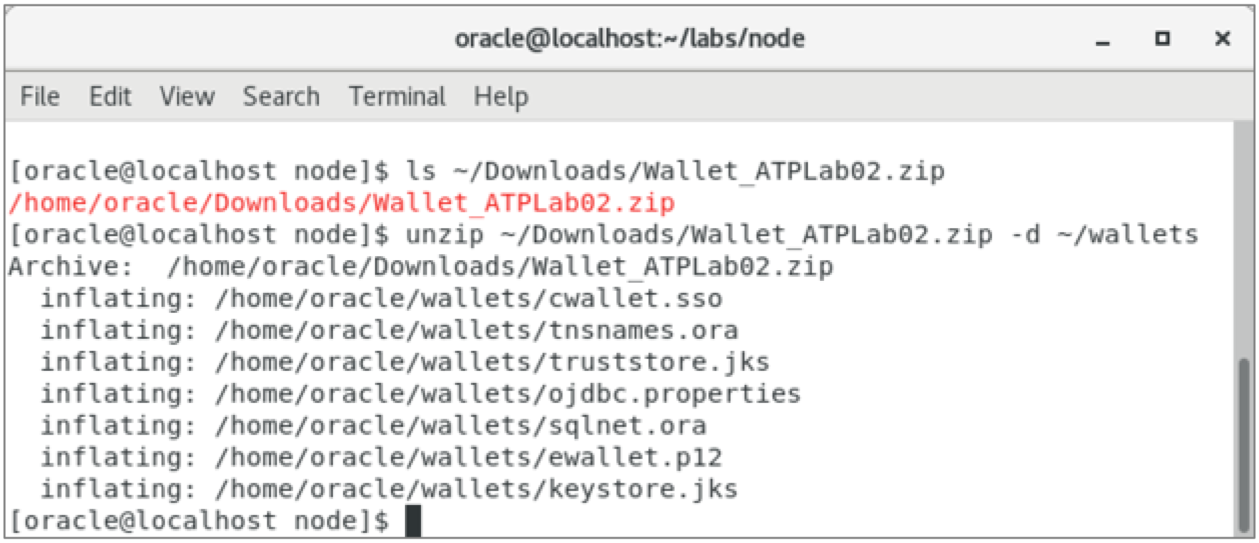
STEP 2: Setup Oracle Client to Connect using the Wallet

- Start a **Terminal** session in the lab VM.

- Unzip the Credential Wallet zip file that was downloaded earlier to the **~/wallets** folder. The below example assumes that the **full path** to your downloaded wallet zip file is **<your_wallet_file.zip>**.

```
$ unzip <your_wallet_file.zip> -d ~/wallets
```

Note: If prompted to replace existing files in **~/wallets** folder, enter [y]es for all.


oracle@localhost:~/labs/node
File Edit View Search Terminal Help

```
[oracle@localhost node]$ ls ~/Downloads/Wallet_ATPLab02.zip
/home/oracle/Downloads/Wallet_ATPLab02.zip
[oracle@localhost node]$ unzip ~/Downloads/Wallet_ATPLab02.zip -d ~/wallets
Archive:  /home/oracle/Downloads/Wallet_ATPLab02.zip
  inflating: /home/oracle/wallets/cwallet.sso
  inflating: /home/oracle/wallets/tnsnames.ora
  inflating: /home/oracle/wallets/truststore.jks
  inflating: /home/oracle/wallets/ojdbc.properties
  inflating: /home/oracle/wallets/sqlnet.ora
  inflating: /home/oracle/wallets/ewallet.p12
  inflating: /home/oracle/wallets/keystore.jks
[oracle@localhost node]$
```

- Copy the **tnsnames.ora** file from your wallet folder to **$ORACLE_HOME/network/admin**.

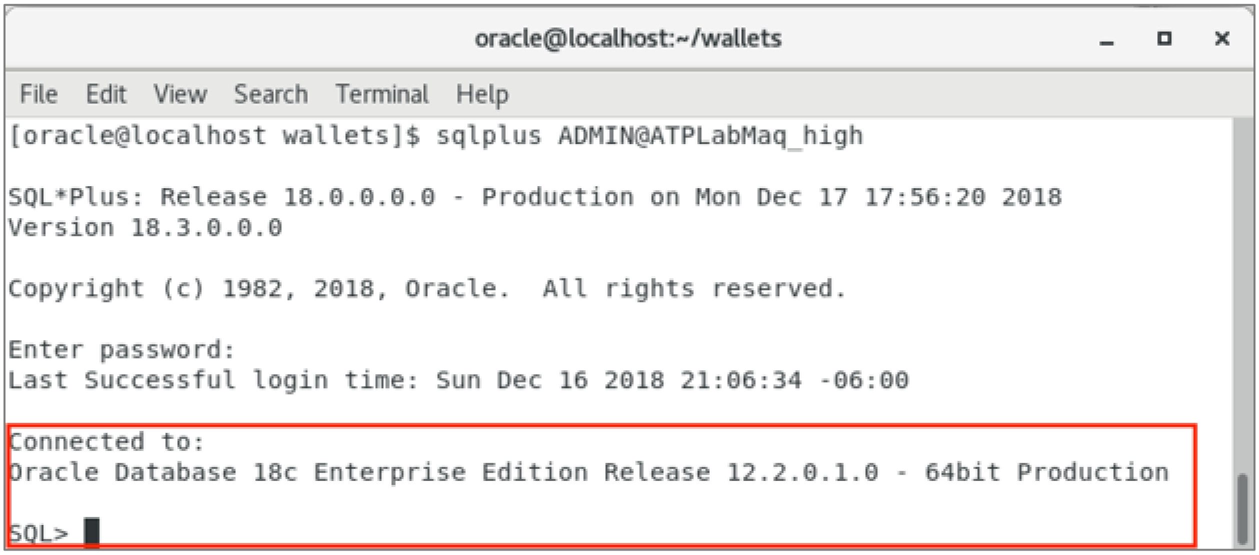
```
$ cp ~/wallets/tnsnames.ora $ORACLE_HOME/network/admin
```

Note: In a real-world scenario, you would want to append the entries from the wallet's **tnsnames.ora** file to **$ORACLE_HOME/network/admin/tnsnames.ora**, not overwrite it as you just did.

- Verify connectivity to your ADB service using **sqlplus**. The **<ADB_Service_Name>** in the below example is one of the predefined service names of your ADB sevice (e.g. **_high**, **_tp**, **_low**, etc.)

```
$ sqlplus ADMIN@<ADB_Service_Name>
```

- Provide your **ADMIN** password and verify that you were able to connect successfully.


oracle@localhost:~/wallets
File Edit View Search Terminal Help

```
[oracle@localhost wallets]$ sqlplus ADMIN@ATPLabMaq_high

SQL*Plus: Release 18.0.0.0.0 - Production on Mon Dec 17 17:56:20 2018
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle.  All rights reserved.

Enter password:
Last Successful login time: Sun Dec 16 2018 21:06:34 -06:00

Connected to:
Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL>
```

- Exit from the SQL Plus session.

```
SQL> EXIT;
```

STEP 3: Run the Generated SQL Loader Scripts

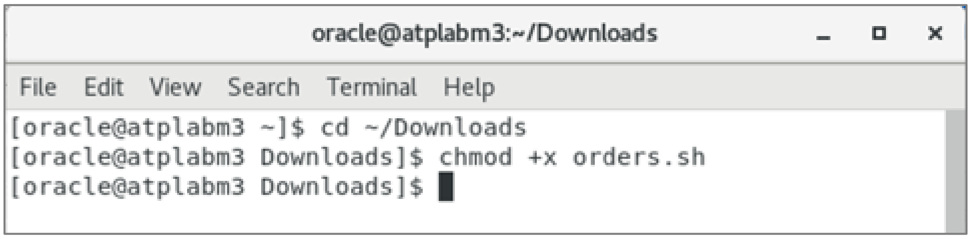
Let's use the generated scripts from the previous step to load data using SQL Loader.

- In the **Terminal** window, change directory to **~/Downloads** as this was the location of the generated scripts.

```
$ cd ~/Downloads
```

- Make the **orders.sh** file executable as by default it is not.

```
$ chmod +x orders.sh
```


oracle@atplabm3:~/Downloads
File Edit View Search Terminal Help

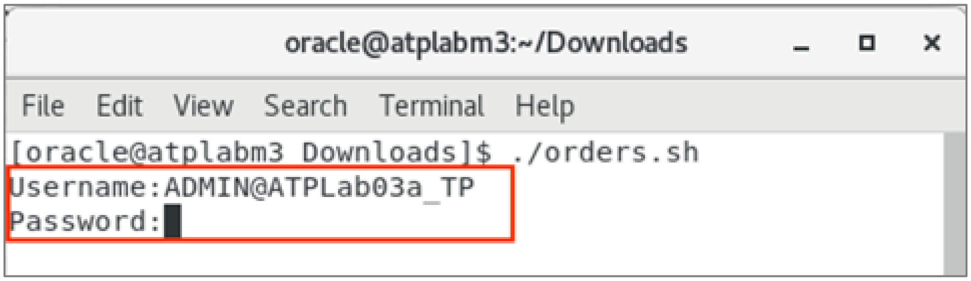
```
[oracle@atplabm3 ~]$ cd ~/Downloads
[oracle@atplabm3 Downloads]$ chmod +x orders.sh
[oracle@atplabm3 Downloads]$
```

- Staying in the **~/Downloads** folder, run the **orders.sh** script to start the SQL Loader process.

```
$ ./orders.sh
```

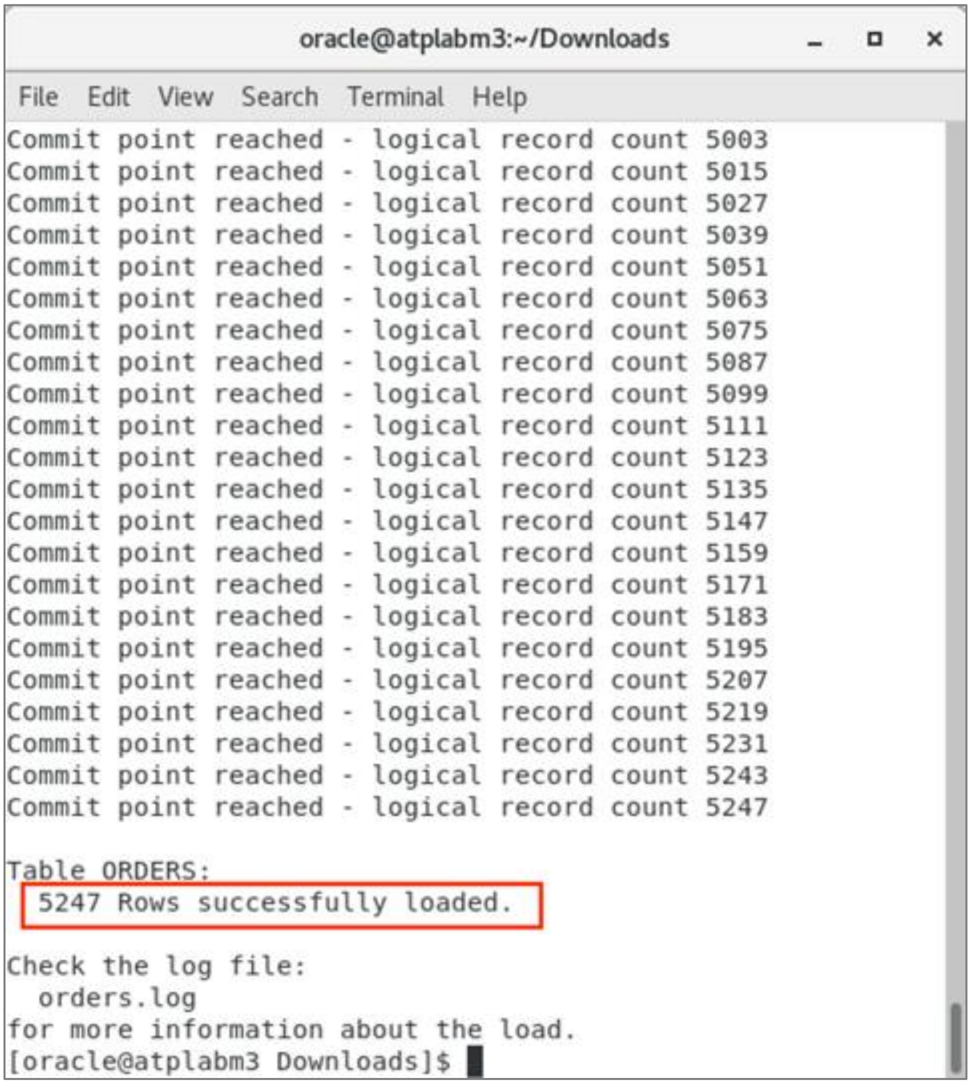
- When prompted, enter your **Username** (i.e. the **ADMIN** user) and the database **Password.** Ensure that you specify the **TNS Alias** (i.e. the connect string) of your service along with the username follows:

```
ADMIN@<YourConnect_String>
```


oracle@atplabm3:~/Downloads
File Edit View Search Terminal Help

```
[oracle@atplabm3 Downloads]$ ./orders.sh
Username:ADMIN@ATPLab03a_TP
Password:
```

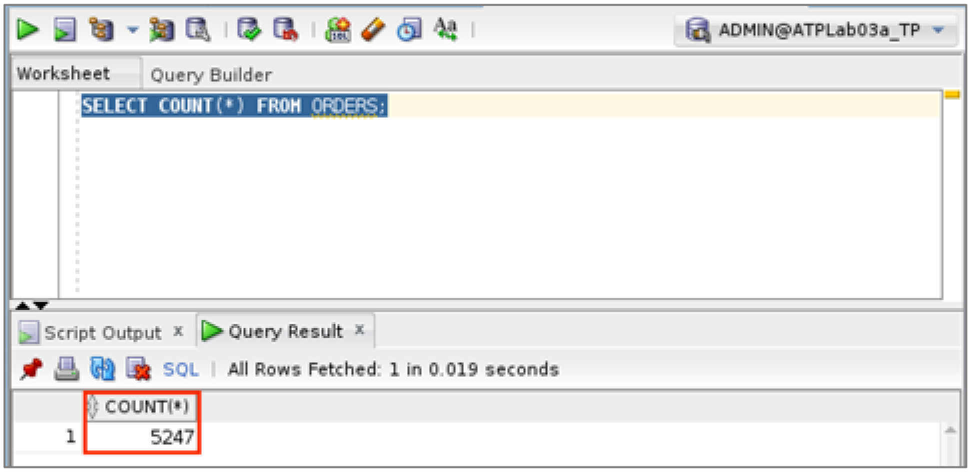
- Verify that the file gets loads successfully.


oracle@atplabm3:~/Downloads
File Edit View Search Terminal Help

```
Commit point reached - logical record count 5003
Commit point reached - logical record count 5015
Commit point reached - logical record count 5027
Commit point reached - logical record count 5039
Commit point reached - logical record count 5051
Commit point reached - logical record count 5063
Commit point reached - logical record count 5075
Commit point reached - logical record count 5087
Commit point reached - logical record count 5099
Commit point reached - logical record count 5111
Commit point reached - logical record count 5123
Commit point reached - logical record count 5135
Commit point reached - logical record count 5147
Commit point reached - logical record count 5159
Commit point reached - logical record count 5171
Commit point reached - logical record count 5183
Commit point reached - logical record count 5195
Commit point reached - logical record count 5207
Commit point reached - logical record count 5219
Commit point reached - logical record count 5231
Commit point reached - logical record count 5243
Commit point reached - logical record count 5247

Table ORDERS:
  5247 Rows successfully loaded.

Check the log file:
  orders.log
for more information about the load.
[oracle@atplabm3 Downloads]$
```

- Also verify the load using **SQL Worksheet** and running the following **COUNT(*)** query:

```
SELECT COUNT(*) FROM ORDERS;
```



```
SELECT COUNT(*) FROM ORDERS;
```

All Rows Fetched: 1 in 0.019 seconds

	COUNT(*)
1	5247

- You have successfully completed this lab.

3. Migrating to the Autonomous Database

Lab 3-1: Migrating to Autonomous Database using Data Pump

Oracle Data Pump offers a very fast and easy method for moving data and metadata between Oracle Databases, including the Oracle Autonomous Database.

Data Pump Import is a utility for loading an export dump file into a target system, using a dump file previously exported using **Data Pump Export**.

Data Pump Import lets you import data from Data Pump files residing on the Oracle Cloud Infrastructure Object Storage, Oracle Cloud Infrastructure Object Storage Classic, and AWS S3. You can save your data to your Cloud Object Store and use Oracle Data Pump to load data to the autonomous database.

Objectives

- Use **SQL Plus** to connect to the ADB Service to validate Oracle Client installation and credential setup.
- Use **Data Pump Import** to import a previously exported schema dump into the ADB.

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** database.
- Ensure that the **Object Storage Bucket** and the **Database Credential** object was created from the earlier labs.
- This lab needs to be completed using the lab VM.

Lab Steps

STEP 1: Setup Oracle Client to Connect using the Wallet

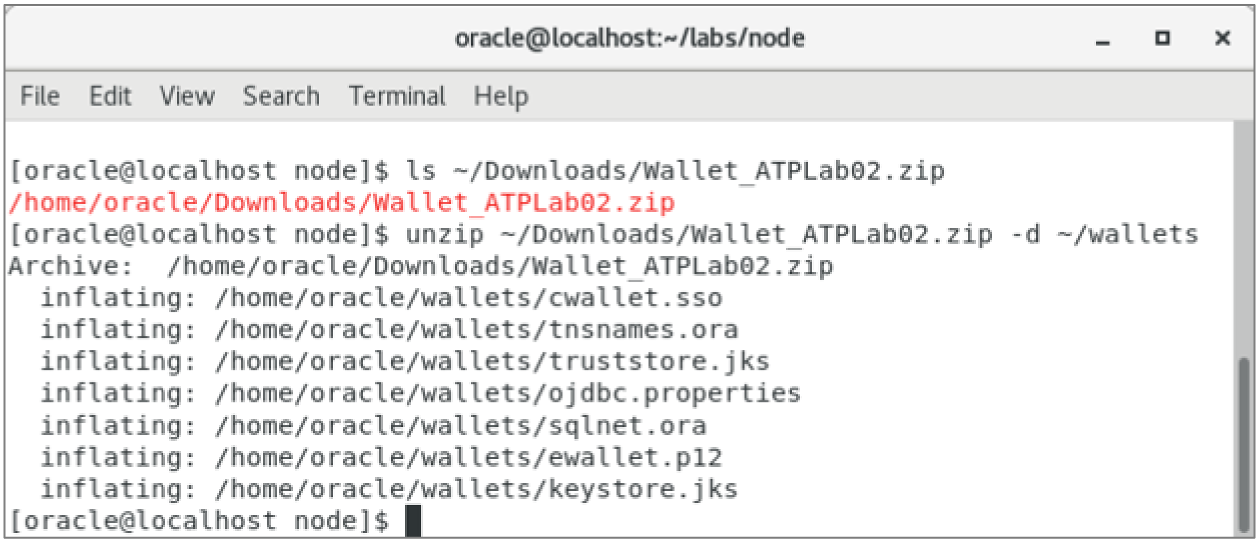
IMPORTANT: If you have completed **Loading Data using SQL Loader** lab, proceed directly to **STEP 2**.

- Sign in to the **lab VM** with the login credentials provided to you by the instructor.
- Start a **Terminal** session in the lab VM.

- Unzip the Credential Wallet zip file that was downloaded earlier to the **~/wallets** folder. The below example assumes that the **full path** to your downloaded wallet zip file is **<your_wallet_file.zip>**.

```
$ unzip <your_wallet_file.zip> -d ~/wallets
```

Note: If prompted to replace existing files in **~/wallets** folder, enter [y]es for all.



```
[oracle@localhost node]$ ls ~/Downloads/Wallet_ATPLab02.zip
/home/oracle/Downloads/Wallet_ATPLab02.zip
[oracle@localhost node]$ unzip ~/Downloads/Wallet_ATPLab02.zip -d ~/wallets
Archive:  /home/oracle/Downloads/Wallet_ATPLab02.zip
  inflating: /home/oracle/wallets/cwallet.sso
  inflating: /home/oracle/wallets/tnsnames.ora
  inflating: /home/oracle/wallets/truststore.jks
  inflating: /home/oracle/wallets/ojdbc.properties
  inflating: /home/oracle/wallets/sqlnet.ora
  inflating: /home/oracle/wallets/ewallet.p12
  inflating: /home/oracle/wallets/keystore.jks
[oracle@localhost node]$ 
```

- Copy the **tnsnames.ora** file from your wallet folder to **$ORACLE_HOME/network/admin**.

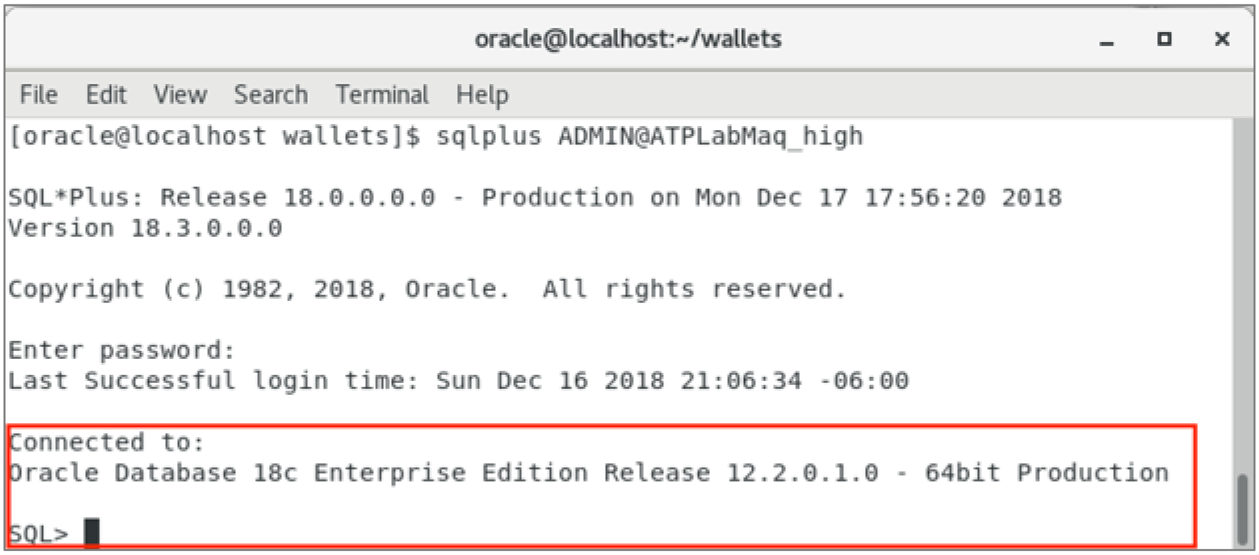
```
$ cp ~/wallets/tnsnames.ora $ORACLE_HOME/network/admin
```

Note: In a real-world implementation, you would want to append the entries from the wallet's tnsnames.ora file to $ORACLE_HOME/network/admin/tnsnames.ora, not overwrite it.

- Verify connectivity to your ADB service using **sqlplus**. The **<ADB_Service_Name>** in the below example is one of the predefined service names of your ADB sevice (e.g. **_high**, **_tp**, **_low**, etc.)

```
$ sqlplus ADMIN@<ADB_Service_Name>
```

- Provide your **ADMIN** password and verify that you were able to connect successfully.



```
[oracle@localhost wallets]$ sqlplus ADMIN@ATPLabMaq_high

SQL*Plus: Release 18.0.0.0.0 - Production on Mon Dec 17 17:56:20 2018
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle.  All rights reserved.

Enter password:
Last Successful login time: Sun Dec 16 2018 21:06:34 -06:00

Connected to:
Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL>
```

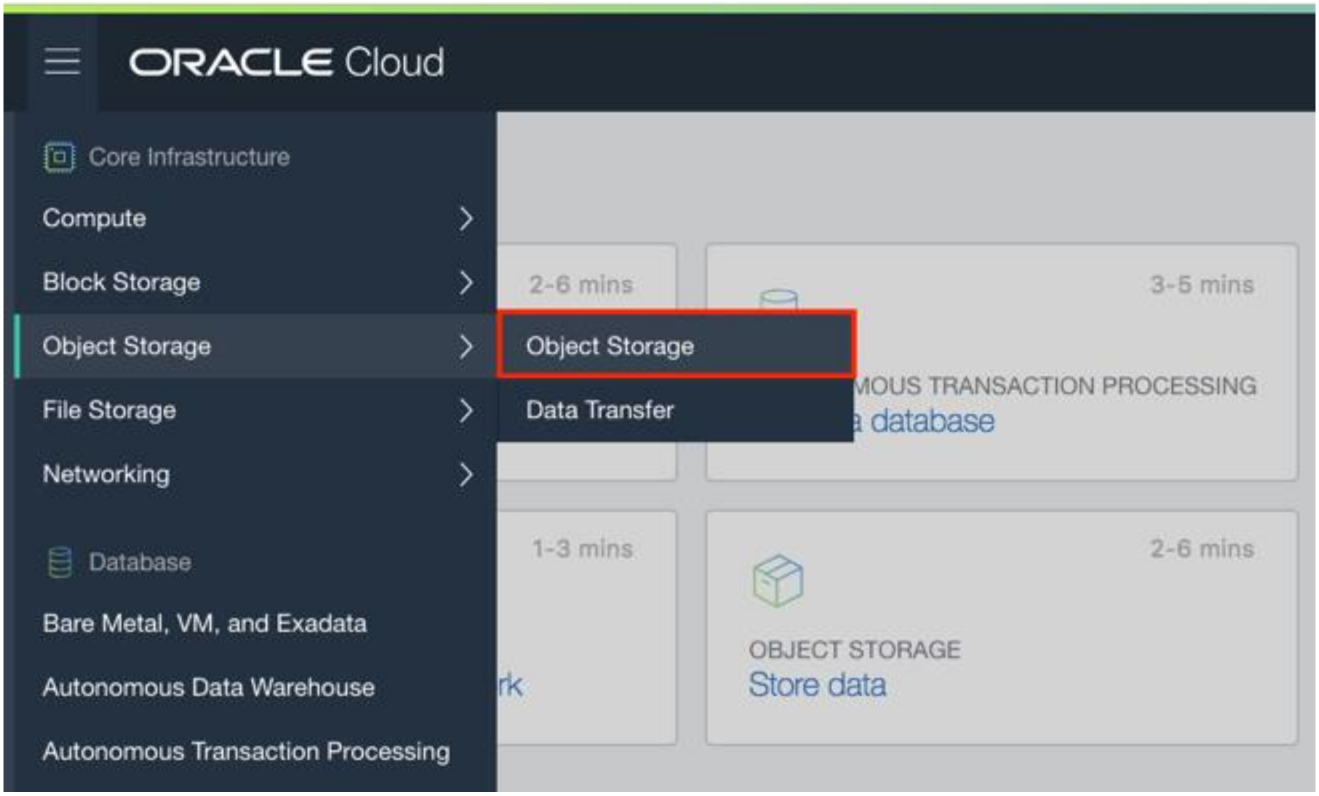
- Exit from the SQL Plus session.

```
SQL> EXIT;
```

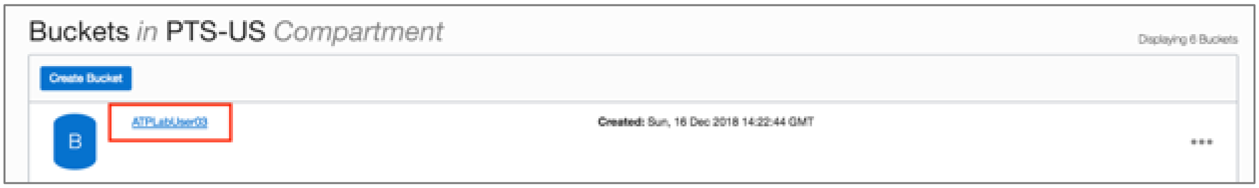
STEP 2: Upload the Data Pump Export File to the Object Storage Bucket

Follow the steps below to upload the **soe_export.dmp** file to the **Object Storage** bucket created in one of the earlier labs.

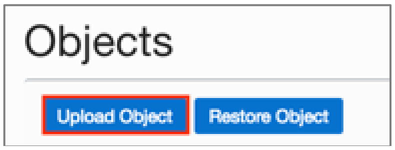
- Login to your OCI console, click on the hamburger menu on the top-left and select **Object Storage -> Object Storage**.



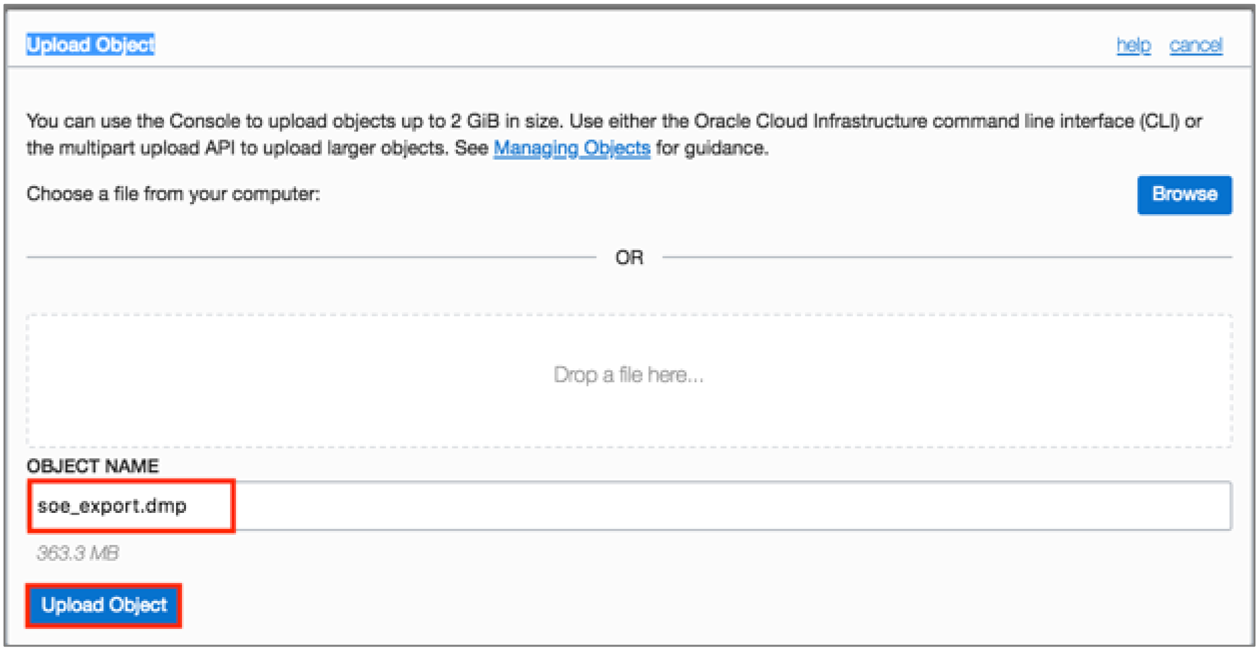
- Locate the bucket you created earlier. Click on your **bucket name** to open **Bucket Details.**


Buckets in PTS-US Compartment
Displaying 6 Buckets
Create Bucket
B
ATPLabUser03
Created: Sun, 16 Dec 2018 14:22:44 GMT
•••

- On the **Bucket Details** page click on the **Upload Object** button.



- Use the **Browse** button to locate the **`soe_export.dmp`** file from **`/home/oracle/labfiles`** directory, select the file and click on **Open**. Alternatively, you may also drag-and-drop the file into this pane.



You can use the Console to upload objects up to 2 GiB in size. Use either the Oracle Cloud Infrastructure command line interface (CLI) or the multipart upload API to upload larger objects. See Managing Objects for guidance.

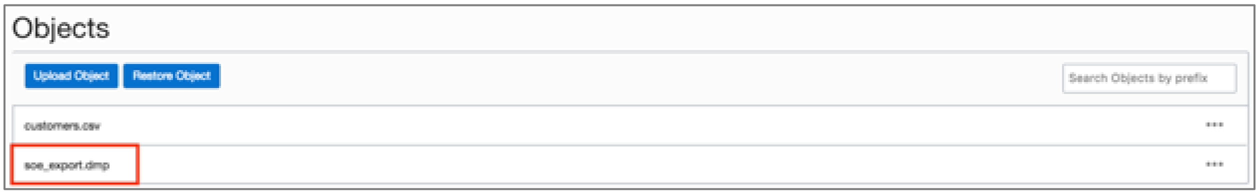
Choose a file from your computer:

OBJECT NAME

soe_export.dmp

363.3 MB

- Once the file is uploaded, it will be listed under **Objects** as follows:



- Construct the URL of **soe_export.dmp** file. The URL is structured as follows :

```
https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/<tenant_name>/<bucket_n
ame>/<file_name>
```

- **region_name** : The region you have created your Object Storage bucket. Typically, this would be **us-phoenix-1**, **us-ashburn-1**, etc.

- **tenant_name** : <OCI Tenancy Name>

- **bucket_name** : <Object Storage Bucket Name>
- **file_name** : **soe_export.dmp**
- In the example URL below, the region name is **us-phoenix-1**, tenant name is **oraclepartnersas**, bucket name is **ATPLabUser03**, and finally, the file name is **soe_export.dmp**.

```
https://swiftobjectstorage.us-phoenix-
1.oraclecloud.com/v1/oraclepartnersas/ATPLabUser03/soe_export.dmp
```

- Your URL would be different, so please modify accordingly.

STEP 3: Importing Data Using Oracle Data Pump

In this step you will import the **soe_export.dmp** file from the Oracle Object Storage into the ADB.

- Go back to your **Terminal** session in the **Lab VM**.
- Invoke the Data Pump command using the following command line, replacing **<Your_Admin_Password>**, **<Your_Connect_String>** and **<SOE_Export_File_URL>** as appropriate (Hint: You just built <SOE_Export_File_URL> in the previous step).

```
$ impdp userid=ADMIN/<Your_Admin_Password>@<Your_Connect_String>
credential=WORKSHOP_CREDENTIAL schemas=SOE directory=DATA_PUMP_DIR
dumpfile=<SOE_Export_File_URL> logfile=DATA_PUMP_DIR:soe_import.log
```

- This will start the import process. Verify the import output is similar to the following (ignore the **ORA-39082** and **ORA-31685** errors):

```
[oracle@abdlab000 Downloads]$ impdp userid=ADMIN/*******@ATPLab99_TP
credential=WORKSHOP_CREDENTIAL schemas=SOE directory=DATA_PUMP_DIR
dumpfile=https://swiftobjectstorage.us-ashburn-
1.oraclecloud.com/v1/showitbuildit1/USER99/soe_export.dmp
logfile=DATA_PUMP_DIR:soe_import.log

Import: Release 18.0.0.0.0 - Production on Sun Jan 20 04:40:36 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

Master table "ADMIN"."SYS_IMPORT_SCHEMA_01" successfully loaded/unloaded
Starting "ADMIN"."SYS_IMPORT_SCHEMA_01":  userid=ADMIN/********@ATPLab99_TP
credential=WORKSHOP_CREDENTIAL schemas=SOE directory=DATA_PUMP_DIR
dumpfile=https://swiftobjectstorage.us-ashburn-
1.oraclecloud.com/v1/showitbuildit1/USER99/soe_export.dmp
logfile=DATA_PUMP_DIR:soe_import.log
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
ORA-31685: Object type SYSTEM_GRANT:"SOE"."MANAGE SCHEDULER" failed due to insufficient
privileges. Failing sql is:
GRANT MANAGE SCHEDULER TO "SOE"
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PASSWORD_HISTORY
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "SOE"."INVENTORIES"                        15.15 MB  894904 rows
. . imported "SOE"."ADDRESSES"                          32.86 MB  450000 rows
. . imported "SOE"."LOGON"                              14.95 MB  714896 rows
. . imported "SOE"."ORDER_ITEMS"                        68.00 MB 1283887 rows
. . imported "SOE"."PRODUCT_DESCRIPTIONS"               220.2 KB    1000 rows
. . imported "SOE"."ORDERS"                             38.46 MB  428938 rows
. . imported "SOE"."ORDERENTRY_METADATA"                5.609 KB       4 rows
. . imported "SOE"."PRODUCT_INFORMATION"                188.1 KB    1000 rows
. . imported "SOE"."CARD_DETAILS"                       19.01 MB  450000 rows
. . imported "SOE"."WAREHOUSES"                         35.66 KB    1000 rows
. . imported "SOE"."CUSTOMERS"                          32.23 MB  300000 rows
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_SPEC
Processing object type SCHEMA_EXPORT/PACKAGE/COMPILE_PACKAGE/PACKAGE_SPEC/ALTER_PACKAGE_SPEC
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_BODY
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/INDEX/FUNCTIONAL_INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_INDEX/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Processing object type SCHEMA_EXPORT/POST_SCHEMA/PROCACT_SCHEMA
ORA-39082: Object type PACKAGE BODY:"SOE"."ORDERENTRY" created with compilation warnings
Job "ADMIN"."SYS_IMPORT_SCHEMA_01" completed with 2 error(s) at Sun Jan 20 04:44:19 2019
elapsed 0 00:03:06
[oracle@abdlab000 Downloads]$
```

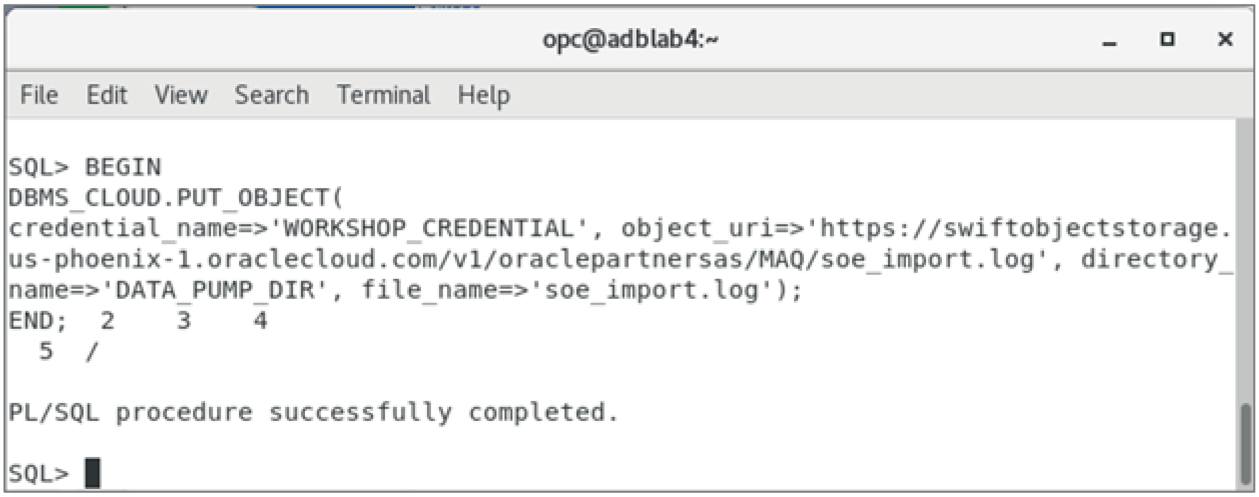
STEP 4: Verify Import Logs

The log files for Data Pump Import operations are stored in the directory DATA_PUMP_DIR; this is the only directory you can specify for the data pump directory parameter.

To access the log file, you would first need to copy the file to your Cloud Object Storage using the procedure DBMS_CLOUD.PUT_OBJECT and then download it from the Object Storage.

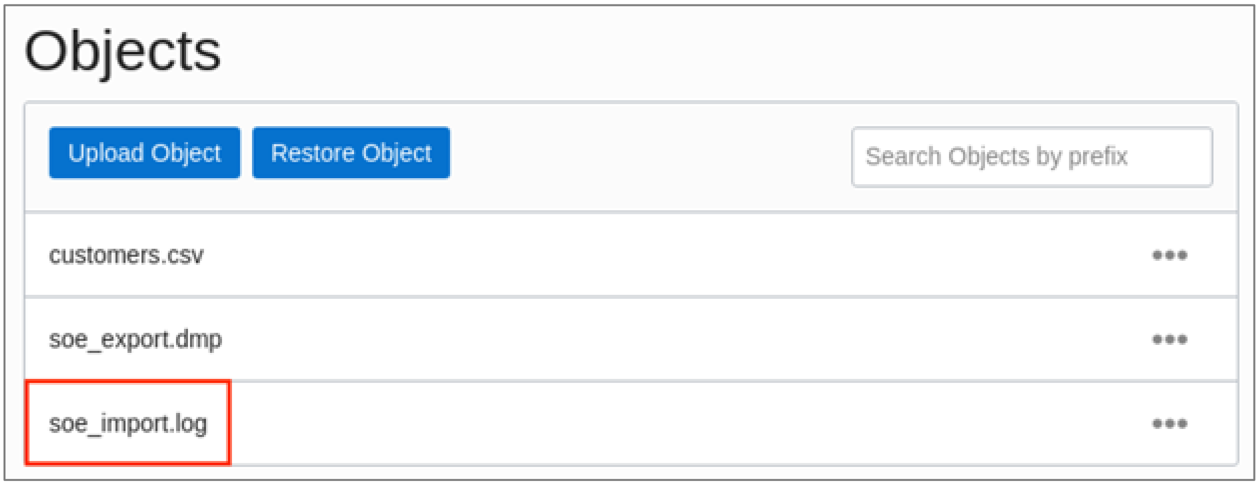
- Start a **SQL Plus** session and run the following PL/SQL block, replacing **<region_name>** with your **Region**, **<tenant_name>** with your **Tenant** and **<bucket_name>** with the name of your cloud storage **Bucket**:

```
BEGIN
DBMS_CLOUD.PUT_OBJECT(
credential_name=>'WORKSHOP_CREDENTIAL',
object_uri=>'https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/<te
nant_name>/<bucket_name>/soe_import.log', directory_name=>'DATA_PUMP_DIR',
file_name=>'soe_import.log');
END;
/
```

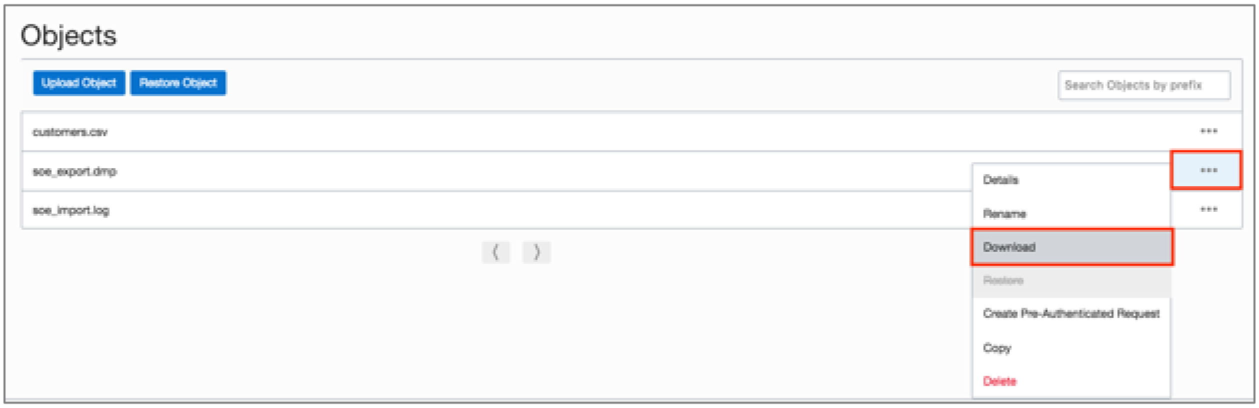


```
SQL> BEGIN
DBMS_CLOUD.PUT_OBJECT(
credential_name=>'WORKSHOP_CREDENTIAL', object_uri=>'https://swiftobjectstorage.
us-phoenix-1.oraclecloud.com/v1/oraclepartnersas/MAQ/soe_import.log', directory_
name=>'DATA_PUMP_DIR', file_name=>'soe_import.log');
END;  2    3    4
  5  /

PL/SQL procedure successfully completed.

SQL> 
```

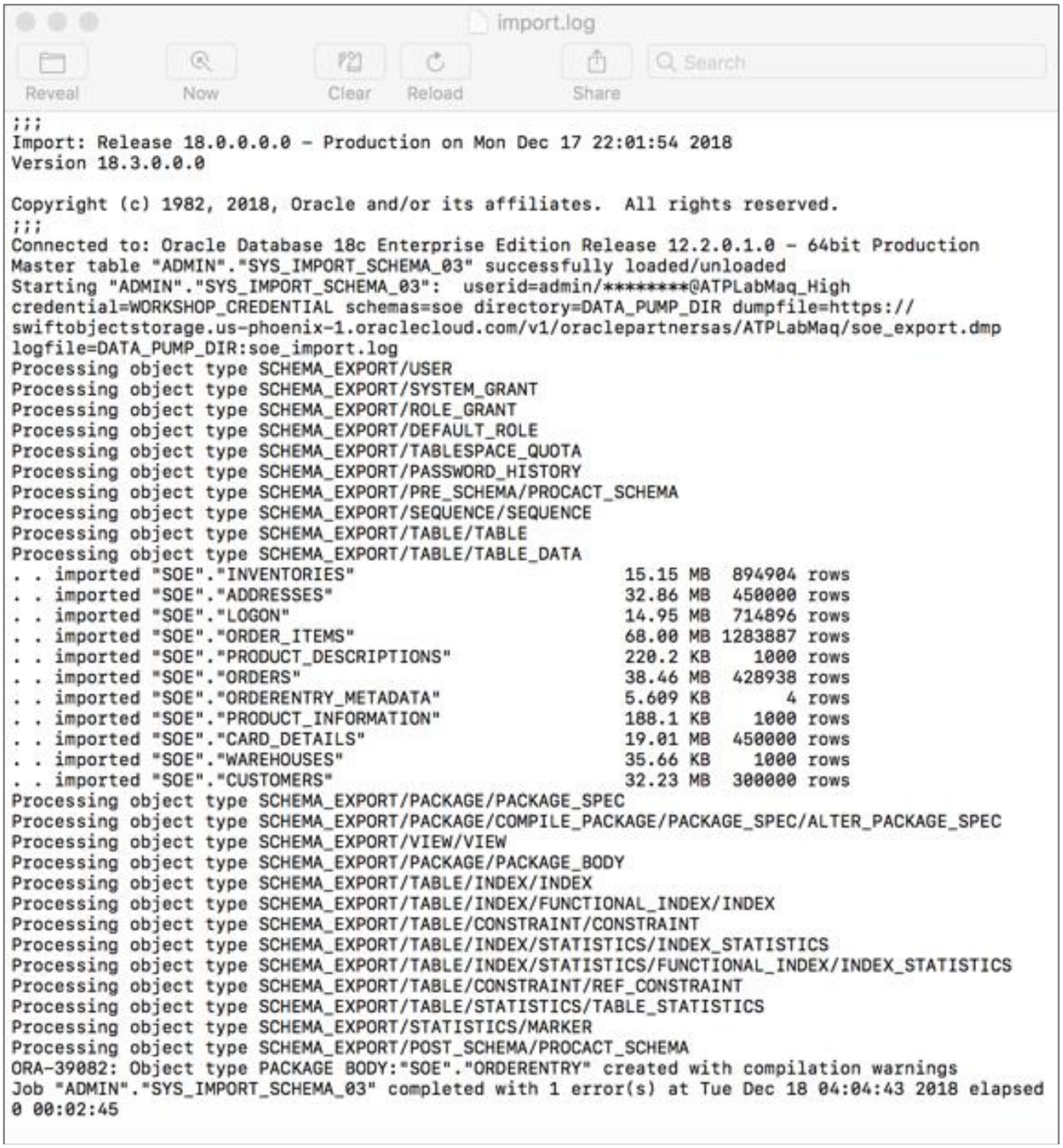
- Verify that the **import.log** file was copied to the bucket. Go to the **Bucket Details** page and locate **soe_import.log** (Hint: You may have to refresh the page).



- Download the import log by clicking on the elipses **(…)** and selecting **Download** as follows:



- View the file in your favorite editor.



```
;;;
Import: Release 18.0.0.0.0 - Production on Mon Dec 17 22:01:54 2018
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 12.2.0.1.0 - 64bit Production
Master table "ADMIN"."SYS_IMPORT_SCHEMA_03" successfully loaded/unloaded
Starting "ADMIN"."SYS_IMPORT_SCHEMA_03":  userid=admin/********@ATPLabMaq_High
credential=WORKSHOP_CREDENTIAL schemas=soe directory=DATA_PUMP_DIR dumpfile=https://
swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas/ATPLabMaq/soe_export.dmp
logfile=DATA_PUMP_DIR:soe_import.log
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PASSWORD_HISTORY
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "SOE"."INVENTORIES"                          15.15 MB  894904 rows
. . imported "SOE"."ADDRESSES"                            32.86 MB  450000 rows
. . imported "SOE"."LOGON"                                14.95 MB  714896 rows
. . imported "SOE"."ORDER_ITEMS"                          68.00 MB 1283887 rows
. . imported "SOE"."PRODUCT_DESCRIPTIONS"                 220.2 KB    1000 rows
. . imported "SOE"."ORDERS"                               38.46 MB  428938 rows
. . imported "SOE"."ORDERENTRY_METADATA"                  5.609 KB       4 rows
. . imported "SOE"."PRODUCT_INFORMATION"                  188.1 KB    1000 rows
. . imported "SOE"."CARD_DETAILS"                         19.01 MB  450000 rows
. . imported "SOE"."WAREHOUSES"                           35.66 KB    1000 rows
. . imported "SOE"."CUSTOMERS"                            32.23 MB  300000 rows
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_SPEC
Processing object type SCHEMA_EXPORT/PACKAGE/COMPILE_PACKAGE/PACKAGE_SPEC/ALTER_PACKAGE_SPEC
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_BODY
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/INDEX/FUNCTIONAL_INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_INDEX/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Processing object type SCHEMA_EXPORT/POST_SCHEMA/PROCACT_SCHEMA
ORA-39082: Object type PACKAGE BODY:"SOE"."ORDERENTRY" created with compilation warnings
Job "ADMIN"."SYS_IMPORT_SCHEMA_03" completed with 1 error(s) at Tue Dec 18 04:04:43 2018 elapsed
0 00:02:45
```

- You have now successfully migrated a database to ADB using Oracle Data Pump.

4. Developing Applications with Autonomous Database

Lab 4-1: Building Node.js Applications with ADB

Node.js is an open source, cross-platform runtime environment for building mid-tier and networking applications using the popular JavaScript language.

The **node-oracledb** add-on for Node.js powers high performance Oracle Database applications. The node-oracledb driver connects to Oracle Database for fast and functional applications. It is an open source project with Apache 2.0 license. It is maintained by Oracle and is under active development.

The objective of this lab is to get a Node.js environment running in your lab VM that is capable of connecting and running database operations against the Oracle Autonomous Database. Javascript is a very popular language with developers and it's not our objective to teach coding in Javascript, rather to emphasize that Oracle Autonomous Database supports working with Javascript.

Objectives

- Learn to connect a Node.js application to the Oracle ADB database.
- Run the supplied example applications that selects data from the Oracle Autonomous Database using Node.js.

Required Artifacts

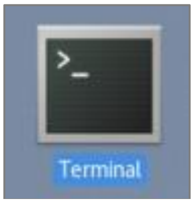
- A previously provisioned Oracle **Autonomous Database**, either **Autonomous Transaction Processing** or **Autonomous Data Warehouse**.
- Ensure that you have setup the Oracle Client to connect using the Wallet, using instructions from **Migrating to Autonomous Database using Oracle Data Pump** lab or **Loading Local Data Files using SQL Loader** lab.
- Access to a lab VM with the following software preinstalled:
  - Oracle Instant Client 18c (or the full Client)
  - Node.js
  - Node.js libraries: oracledb, async, app, express

Lab Steps

STEP 1: Validate Node.js Environment

The lab VM is preinstalled with the required software to run Node.js applications that connects to Oracle and to the autonomous database. In this step you would validate the lab environment configuration.

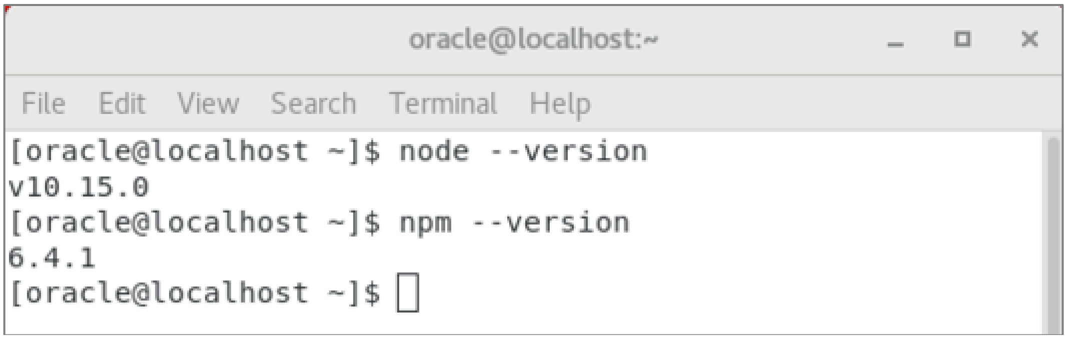
- Login to the lab VM and open a new **Terminal** window.



- Run the following commands to verify Node.js installation:

```
$ node --version
$ npm --version
```

- The above commands will print the version of **node** and **npm** (Node Package Manager) installed and they should match the output below:



```
[oracle@localhost ~]$ node --version
v10.15.0
[oracle@localhost ~]$ npm --version
6.4.1
[oracle@localhost ~]$ 
```

- Also check if **oracledb**, **app**, and **async** packages are installed by running the following **npm** commands and checking for errors:

```
$ npm view oracledb
$ npm view app
$ npm view async
```

Note: The **app** package provides the microservices application framework for Node.js, whereas **async** provides higher order functions and common patterns for running asynchronous code.

- Once you have checked the installations, let's try building our first **hello world** node application.

- Change directory to **~/labs/node**.

```
$ cd ~/labs/node
```

- Copy the **`app.js`** file from **`/home/oracle/labfiles`**.

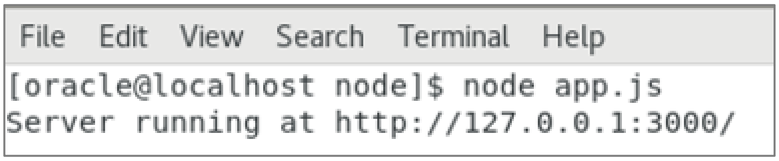
```
$ cp ~/labfiles/app.js .
```

- Using **gedit** or your favorite editor browse the **app.js** file.

```
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}/`);
});
```

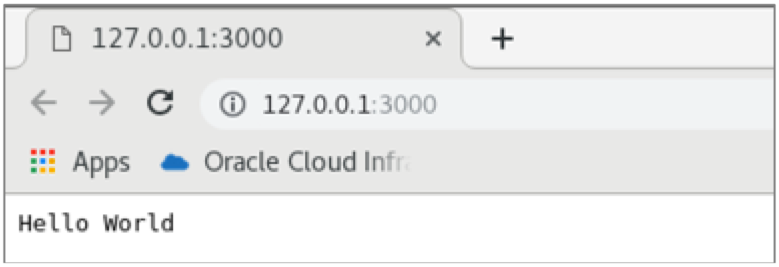
- Run your **app.js** application using "**node app.js**" command as follows (it is important that you stay in the **~/labs/node** directory):

```
$ node app.js
```



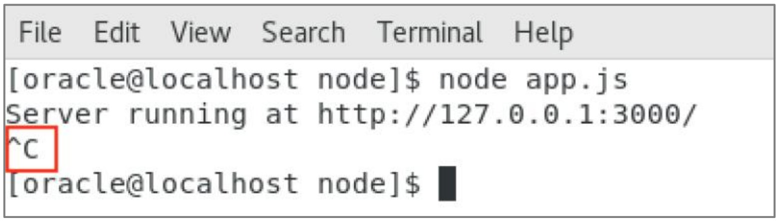
```
[oracle@localhost node]$ node app.js
Server running at http://127.0.0.1:3000/
```

- Now, point your browser to **http://localhost:3000**, and you should see the **Hello World** message.



Hello World

- Ensure the above test is successful. Stop the Node.js application by clicking **CTRL-C** and proceed to the next step.



```
[oracle@localhost node]$ node app.js
Server running at http://127.0.0.1:3000/
^C
[oracle@localhost node]$ 
```

STEP 2: Validate Connectivity to ADB Using Node.js

The credential wallet files downloaded in the earlier labs will be used to connect a sample Node.js application to the autonomous database.

- If you have not completed **Migrating to ADB Using Data Pump** lab, please complete at least the **Setup Oracle Client to Connect using the Wallet** step.
- Next, create a file that will store the ADB database credentials. Other node.js scripts will include this file and use it to initiate the connection to ADB.
- Ensure you are in directory **~/labs/node**.

```
$ cd ~/labs/node
```

- Using your favorite editor, create a file named **dbconfig.js** with the following contents:

```
module.exports= {
dbuser: '<ADMIN_User>',
dbpassword: '<ADMIN_Password>',
connectString: '<ADB_Service>'
}
```

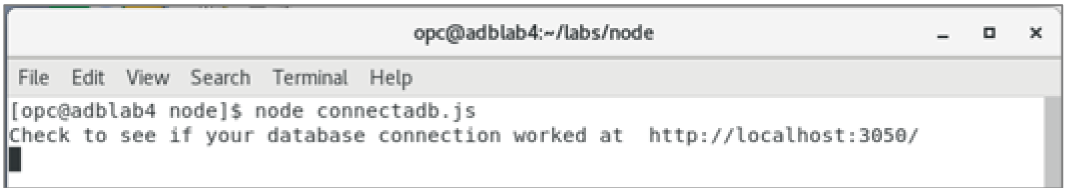
- In the above file replace the following:
  - **<ADMIN_User>** : ADMIN
  - **<ADMIN_Password>** : <Your ADMIN Password>
  - **<ADB_Service>** : <Your ADB Service Name>
- Next, we need to test the connectivity to ADB. Let's use a pre-built node.js application that will connect to the ADB database, and if successful, will display the username that was used to connect to the database on a browser at URL **http://localhost:3050/**.
- Copy the **connectadb.js** file from **/home/oracle/labfiles**.

```
$ cp ~/labfiles/connectadb.js .
```

- Open the **connectadb.js** file using your favorite editor. Note the first few lines in the file includes Node.js libraries like **oracledb** and **http**. It also includes the previously defined **dbconfig.js** script which holds the database credentials.
- To run the above code, at the command prompt and staying in the same folder as **connectadb.js**, run the following command:

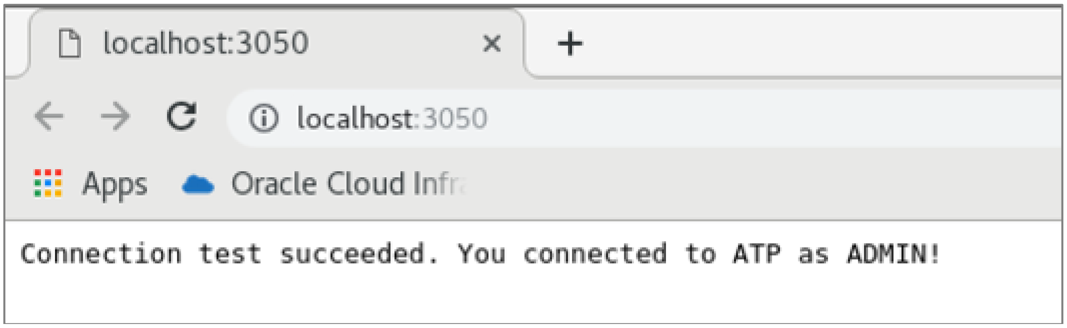
```
$ node connectadb.js
```

- You should see the following output:



```
[opc@adblab4 node]$ node connectadb.js
Check to see if your database connection worked at  http://localhost:3050/
```

- Go to your browser and to the URL **http://localhost:3050/** and observe the output is as expected.



```
Connection test succeeded. You connected to ATP as ADMIN!
```

- If the above test is successful, stop the node.js application by hitting **CTRL-C** and proceed to the next step.

STEP 3: Build a Node.js Application to Select Data from ADB

Let's build a Node.js application that will login to ADB, execute a simple SELECT query, and print the data returned on the terminal.

- Copy the **`adbselect.js`** file from **`/home/oracle/labfiles`**.

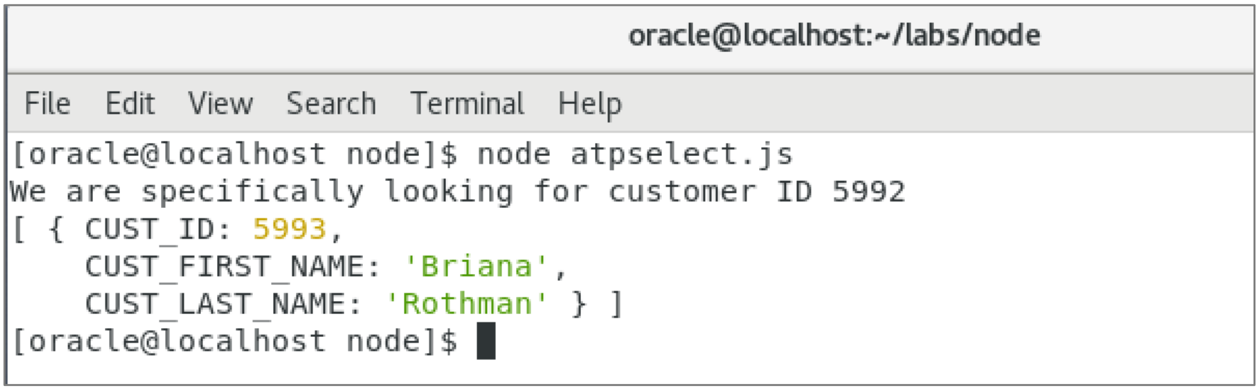
```
$ cp ~/labfiles/adbselect.js .
```

- Open the **adbselect.js** file in your favorite editor and note the library calls at the beginning and some Oracle specific calls such as **connection.execute** and **doRelease**.

- To run the above code, at the command prompt and staying in the same folder as **adbselect.js**, run the following command:

```
$ node adbselect.js
```

- Observe the results and note that **CUST_ID** returned below is **5993**, as it was part of the **WHERE** clause supplied in the select query above.


oracle@localhost:~/labs/node
File Edit View Search Terminal Help

```
[oracle@localhost node]$ node atpselect.js
We are specifically looking for customer ID 5992
[ { CUST_ID: 5993,
    CUST_FIRST_NAME: 'Briana',
    CUST_LAST_NAME: 'Rothman' } ]
[oracle@localhost node]$
```

STEP 4: Build a Node.js Application to Select Data from ADB

In this step you will build on the previous steps and manipulate JSON data into Oracle Objects for updating, analysis, inserts and deletes from the database.

- Copy the **`adbselectjson.js`** file from **`/home/oracle/labfiles`**.

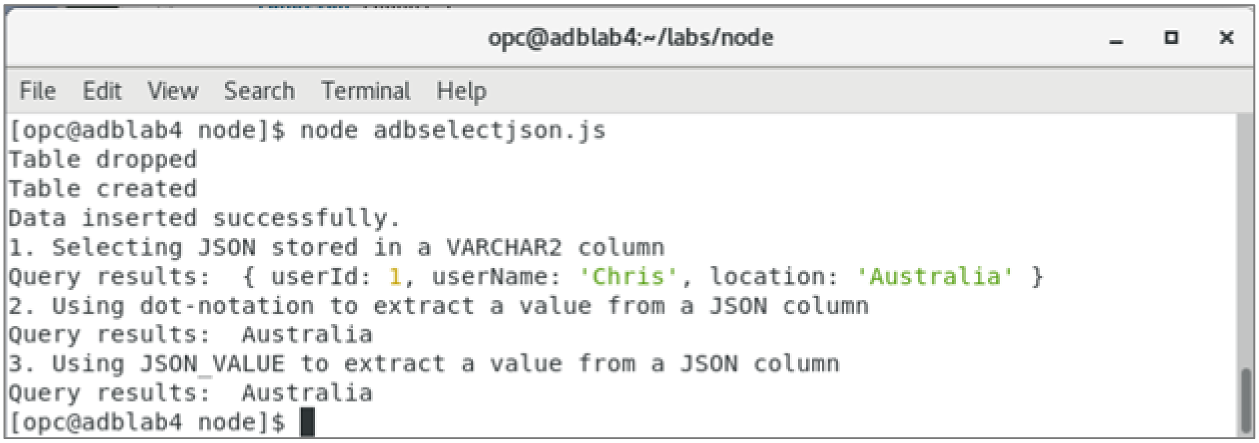
```
$ cp ~/labfiles/adbselectjson.js .
```

- Inspect **adbselectjson.js** application using your favorite editor.

- To run the script, at the command prompt and staying in the same folder as **adbselectjson.js**, run the following command:

```
$ node adbselectjson.js
```

- Observe the results.



```
[opc@adblab4 node]$ node adbselectjson.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
[opc@adblab4 node]$ 
```

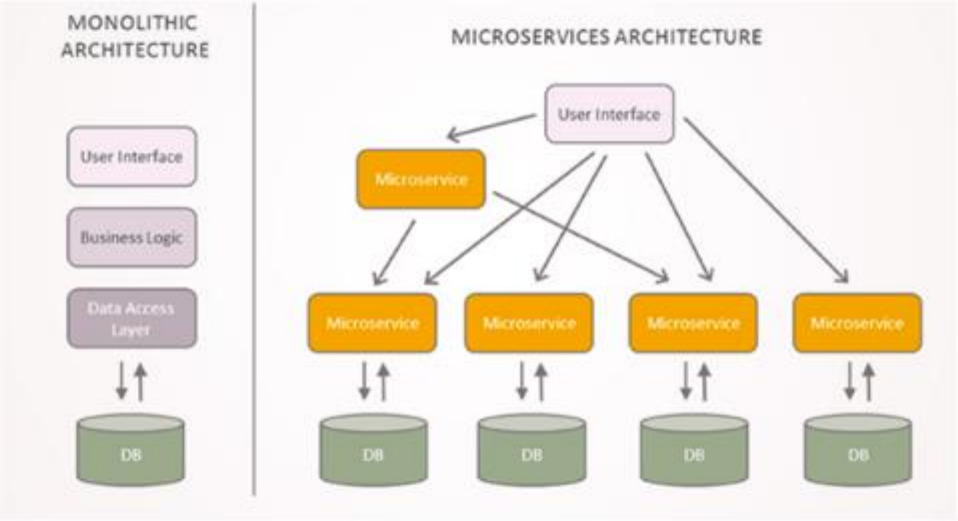
- The output will show output of the several JSON manipulation functions that were defined in the code. Please review the code to see what the different functions do.

- You have completed all steps of this lab. Again, the objective was to set up a working node.js environment with sample code that can be deployed against an ADB database and we have successfully demonstrated how that is done.

Lab 4-2: Building Microservices with Docker

Containers allow us to package applications along with all their dependencies and provide a light-weight run time environment that provides isolation similar to a virtual machine, but without the added overhead of a full-fledged operating system.

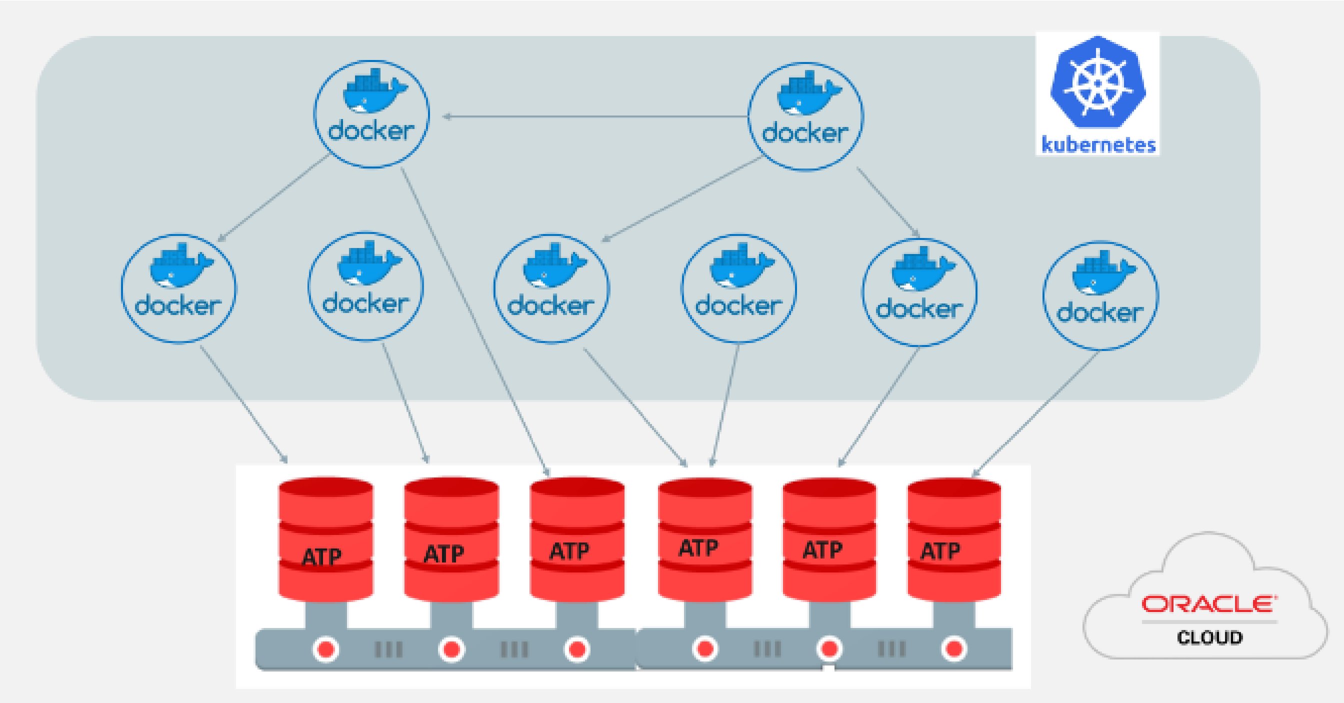
The topic of containers and microservices is a subject on its own but suffice to say that breaking up large, complex software into more manageable pieces that run isolated from each other has many advantages. It's easier to deploy, diagnose and provides better availability since failure of any microservice limits downtime to a portion of the application.



It's important to have a similar strategy in the backend for the database tier. But the issue is if you run multiple databases for each application then you end up having to maintain a large fleet of databases and the maintenance costs can go through the roof. Add to that having to manage security and availability for all of them.

This is where the Oracle’s autonomous database cloud service comes in. It is based on a pluggable architecture similar to application containers where one container database holds multiple pluggable databases. Each of these pluggable databases or PDBs are completely isolated from each other, can be deployed quickly and can be managed as a whole so that you incur the cost of managing a single database while deploying multiple micro services onto these PDBs.

The Autonomous cloud service takes it a step further. It is self-managing, self-securing and highly available. There is no customer involvement in backing it up, patching it or even tuning it for most part. You simply provision, connect and run your applications.



Objectives

- Build a docker container that runs node.js microservices.
- Configure the microservice to use an Oracle Autonomous Transaction Processing Database.

Required Artifacts

- A previously provisioned Oracle **Autonomous Database**, either **Autonomous Transaction Processing** or **Autonomous Data Warehouse**.
- A previously downloaded **Client credential wallet** of your autonomous database.
- A **lab VM** with the Docker Container platform preinstalled and a downloaded Oracle Instant Client zip file.

Lab Steps

STEP 1: Download and Configure the Sample Microservice Application

A sample microservice application for an online marketplace use case (i.e. buy/sell products) has been prebuilt for this lab. The application is named **aone** and it is a node.js app and requires a database schema with seed data to be deployed in an Oracle Database.

In this step, you will download and configure the sample application, readying it to be deployed in the docker container.

- Login to the lab VM and start a **Terminal** session.

- Change directory to **~/labs/docker**

```
$ cd ~/labs/docker
```

- Copy the sample docker microservice application named **ATPDocker** from **`/home/oracle/labfiles`** directory.

```
$ cp ~/labfiles/ATPDocker.tar .
```

- Run the following **tar** command to extract the application source in the **~/labs/docker** folder:

```
$ tar xzvf ATPDocker.tar
```

- Extract the previously downloaded credentials wallet zip file of your ATP service to **~/labs/docker/ATPDocker/wallet_NODEAPPDB2** folder, using the following unzip command, replacing <Path_to_Your_Wallet_Zip> and <Wallet_Zip_File>.zip with the path and the name of your credentials wallet zip file, respectively.

Note: Ensure that you use the same folder names/structure as above as the application is configured to run with those values.

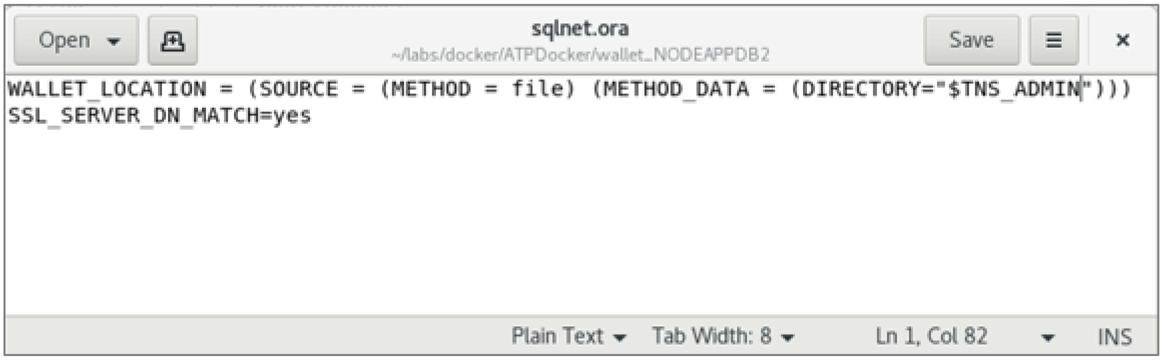
```
$ unzip <Path_to_Your_Wallet_Zip>/<Wallet_Zip_File>.zip -d
~/labs/docker/ATPDocker/wallet_NODEAPPDB2/
```

- Edit sqlnet.ora in **`~/labs/docker/ATPDocker/wallet_NODEAPPDB2`** folder:

```
$ cd ~/labs/docker/ATPDocker/wallet_NODEAPPDB2/
$ gedit sqlnet.ora
```

- Replace the contents of the file with the following text. This tells the driver to look for the wallet in the path setup in variable TNS_ADMIN. Save the file and exit (×).

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



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

- Copy the previously downloaded Oracle Instant Client from **~/Downloads/instantclient-basic-linux.x64-12.1.0.2.0.zip** to **~/labs/docker/ATPDocker** folder. The Oracle Instant Client will be installed in the Docker container during container instantiation.

```
$ cp ~/Downloads/instantclient-*.zip ~/labs/docker/ATPDocker/
```

- The **dbconfig.js** file located in **~/labs/docker/ATPDocker/aone/scripts** holds the ATP database user, password and the TNS Alias of the ATP service, which is used by the **aone** application.

- Edit the **dbconfig.js** file and replace the <User>, <Password>, and <ATP_TP_Alias>, as appropriate, with the info related to your ATP service.

```
$ gedit ~/labs/docker/ATPDocker/aone/scripts/dbconfig.js
```

```
module.exports= {
user:"<User>",
password:"<Password>",
connectString :"<ATP_TP_Alias>"
}
```

STEP 2: Create the Application Schema

The SQL script to create the schema for **aone** application is in the **/home/oracle/labs/docker/ATPDocker/aone/create_schema.sql**

- Configure the **TNS_ADMIN** variable to enable SQL Plus to connect to your ATP service.

```
$ export TNS_ADMIN=~/labs/docker/ATPDocker/wallet_NODEAPPDB2
```

- Connect to the ATP Service using SQL Plus and run the **create_schema.sql** script to create the schema objects, replacing the <User>, <Password>, and <ATP_TP_Alias>, as appropriate.

```
$ cd ~/labs/docker/ATPDocker/aone/
$ sqlplus <User>/<Password>@<ATP_TP_Alias> @create_schema.sql
```

- Verify the objects were created and exit from the SQL Plus session.

```
$ sqlplus ADMIN/******@ATPLab06_TP @create_schema.sql

SQL*Plus: Release 18.0.0.0.0 - Production on Sun Jan 6 14:19:20 2019
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle.  All rights reserved.

Last Successful login time: Wed Jan 09 2019 19:21:19 -06:00

Connected to:

Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

Sequence created.

Sequence created.
```

```
.

.

Table altered.

Table altered.

Trigger created.

Trigger altered.

Trigger created.

Trigger altered.

Trigger created.

Trigger altered.

Trigger created.

Trigger altered.
SQL> exit;
Disconnected from Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
[oracle@localhost aone]$
```

STEP 3: Build the Docker Image

Now you are ready to build the Docker image. An image is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.

- Change directory to **~/labs/docker/ATPDocker**

```
$ cd ~/labs/docker/ATPDocker
```

- Build the Docker image.

Note: Don't forget to include the "**.**" at the end of the command below. It means to use the **Dockerfile** in the current folder. Also, the **-t** option is to give your image a tag of **aone**.

```
$ sudo docker build -t aone .
```

- Verify the Docker container was built successfully. Note the **successfully built** message at the end.

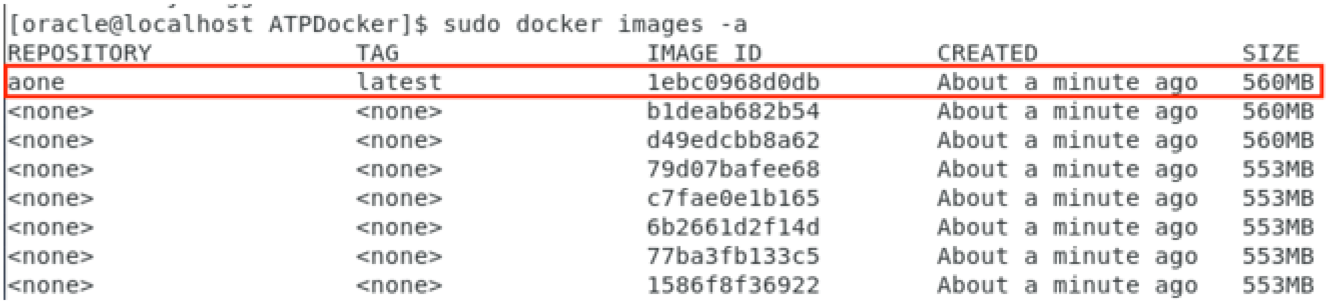
```
Sending build context to Docker daemon  76.11MB
Step 1/18 : FROM frolvlad/alpine-glibc:alpine-3.8
Trying to pull repository docker.io/frolvlad/alpine-glibc ...
alpine-3.8: Pulling from docker.io/frolvlad/alpine-glibc
cd784148e348: Pull complete
```

```
3651dac5ddfb: Pull complete
.
.
Step 19/19 : CMD [ "node", "/opt/oracle/lib/aone/server.js" ]
 ---> Using cache
 ---> cf2666e8fc40
Successfully built cf2666e8fc40
Successfully tagged aone:latest
```

- Your image should be ready in less than a minute. The entire image is about 560 MB. Check the image status using **docker images -a**.

```
$ sudo docker images -a
```

- Docker creates multiple image files as it builds each layer. Your final image would show at the top of the list and will have the tag you chose.



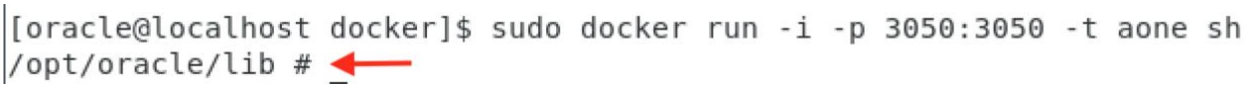
```
[oracle@localhost ATPDocker]$ sudo docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
aone                latest              1ebc0968d0db        About a minute ago   560MB
<none>              <none>              b1deab682b54        About a minute ago   560MB
<none>              <none>              d49edcbb8a62        About a minute ago   560MB
<none>              <none>              79d07bafee68        About a minute ago   553MB
<none>              <none>              c7fae0e1b165        About a minute ago   553MB
<none>              <none>              6b2661d2f14d        About a minute ago   553MB
<none>              <none>              77ba3fb133c5        About a minute ago   553MB
<none>              <none>              1586f8f36922        About a minute ago   553MB
```

STEP 4: Launch the Docker Image

A container is launched by running an image. A container is a runtime instance of an image, i.e. the instantiation of user processes and memory states when the image gets executed.

- Launch the docker image as follows:

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



```
[oracle@localhost docker]$ sudo docker run -i -p 3050:3050 -t aone sh
/opt/oracle/lib # _
```

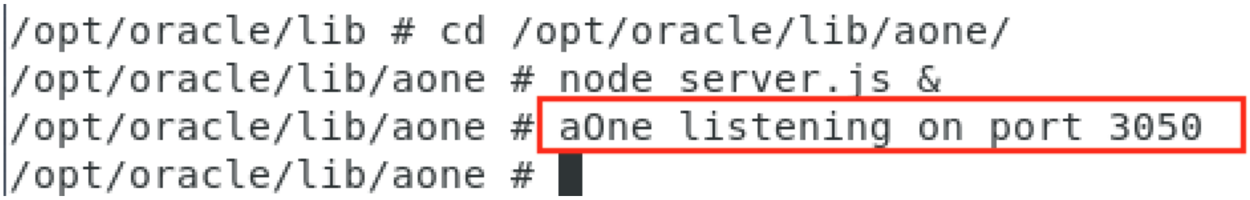
- Observe that you get the OS prompt (**/opt/oracle/lib #**) in the docker image, noted by the arrow above.

STEP 5: Run the Node.js Aone Application

- Within the docker container navigate to aone folder and run server.js script.

```
$ cd /opt/oracle/lib/aone/
$ node server.js &
```

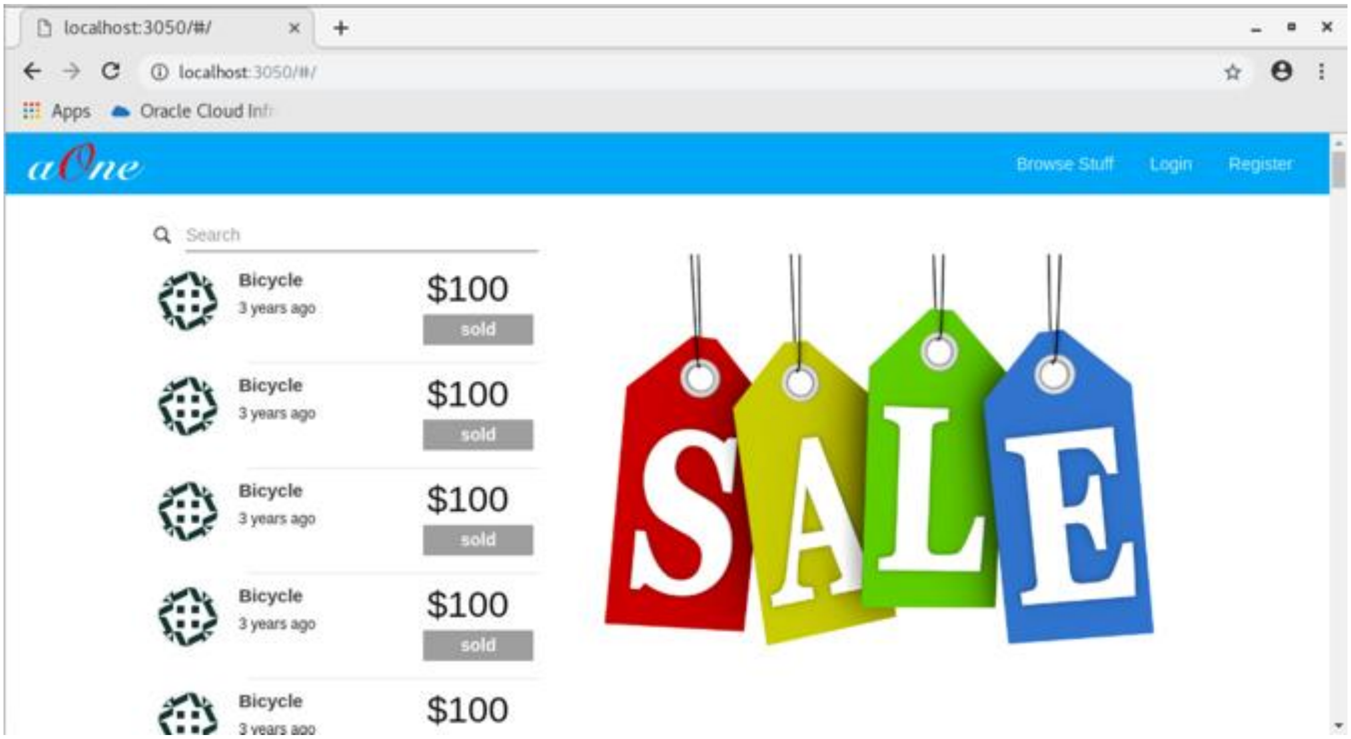
- You should get a response similar to this:



```
/opt/oracle/lib # cd /opt/oracle/lib/aone/
/opt/oracle/lib/aone # node server.js &
/opt/oracle/lib/aone # aOne listening on port 3050
/opt/oracle/lib/aone # █
```

- To check the app on the browser, you have bridged port 3050 on the container to your localhost.

- Open browser on your localhost and go to **http://localhost:3050**. This is what you see if your app ran successfully (note the items and their prices are pulled from the autonomous database).



Bicycle 3 years ago $100 sold

Bicycle 3 years ago $100 sold

Bicycle 3 years ago $100 sold

Bicycle 3 years ago $100 sold

Bicycle 3 years ago $100

- Exit the Docker container.

```
/opt/oracle/lib # exit
```

- You just built and provisioned an entire application stack consisting of a microservice and an enterprise grade, self-managing database. You can push your docker image to a public/private docker repository and it can be pushed to any container orchestration service either on-premise or with any cloud provider. Your database is autonomous – it provisions quickly, backs up, patches and tunes itself as needed.

- You have successfully completed this lab.

Lab 4-3: Using Python with Autonomous Database

Python is a popular general-purpose and dynamic scripting language. With the rise of various programming frameworks, Python is a common language for Web application development. If you want to use Python with an Oracle Autonomous Database, this tutorial helps you to get started by giving some examples.

If you are new to Python review Appendix: Python Primer to gain an understanding of the language.

Objectives

- Run basic Python code that interacts with an Oracle Autonomous Database

Required Artifacts

- A previously provisioned **Oracle Autonomous Database**, either **Autonomous Transaction Processing** or **Autonomous Data Warehouse**.
- Ensure that you have setup the Oracle Client to connect using the Wallet, using instructions from the lab **Migrating to Autonomous Database using Oracle Data Pump** or **Loading Local Data Files using SQL Loader**.
- A **lab VM** with **Python** preinstalled, along with **cx_Oracle** extension.

Lab Steps

Sample python code is located in the lab VM in **/home/oracle/labfiles** folder. You will be using these sample files to run some basic scenarios, mainly connecting to the autonomous database and running queries.

STEP 1: Connecting to Autonomous Database

In this step, you will run a sample application that connects to the autonomous database using wallet and TNS configuration, which was performed in the previous labs.

- Login to the lab VM and start a **Terminal** session.

- Change directory to **~/labs/python**.

```
$ cd ~/labs/python
```

- Copy the sample python script from **/home/oracle/labfiles** folder that validates Oracle database connectivity and prints the database version.

```
$ cp ~/labfiles/python_connect.py ~/labs/python
```

- Review the sample Python code using your favorite editor.

```
$ gedit python_connect.py
```

- The **cx_Oracle** module is imported to provide the API for accessing the Oracle database. The **os** module is imported as the Python script needs to access the OS environment variables (to capture User, Password and TNS Alias). Many inbuilt and third-party modules can be included in this way in Python scripts.

```
import os
import cx_Oracle
```

- The **connect()** method initiates the connection to the Oracle database. It takes the user name, password and the TNS Alias. Other connection methods such as Oracle's Easy Connect is also supported.

  In Python, you would use **os.environ.get** to access the OS environment variables.

```
conn = cx_Oracle.connect(os.environ.get(), os.environ.get(), os.environ.get())
```

- Python treats everything as an object. The **conn** object has a **version** attribute, which is a string.

```
print conn.version
```

- The **close()** method closes the connection. Any connections not explicitly closed will be automatically released when the script ends.

```
conn.close()
```

- Exit the editor.

- Next, initialize the Linux shell variables with the database user name, password and the TNS alias. These variables are used within the Python scripts to connect to the database.

- Set the user name for the connection. Typically, this is the **ADMIN** user.

```
$ export ORAUSER=<User>
```

- Set the password for the user. The password for **ADMIN** should be set as "**WElcome_123#**" if you followed the lab instructions verbatim.

```
$ export ORAPASS=<Password>
```

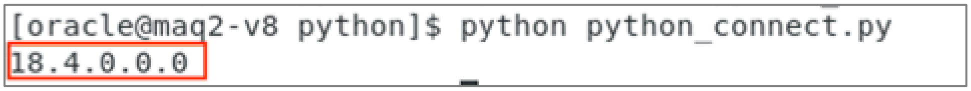
- Set the **TNS Alias** that was setup earlier to connect to the autonomous database. Typically, you would be connecting to the **_TP** service of Autonomous Transaction Processing or **_HIGH** or Autonomous Data Warehouse.

```
$ export ORATNS=<TNS Alias>
```

- Now you are ready to run the script from the command prompt, as follows:

```
$ python python_connect.py
```

- Observe the results. If the connection succeeds, the version number is printed. An exception is thrown if the connection fails.



```
[oracle@maq2-v8 python]$ python python_connect.py
18.4.0.0.0
```

STEP 2: Connecting to Autonomous Database

A common task when developing Web applications is to query a database and display the results in a Web browser. There are a number of functions you can use to query an Oracle database, but the basics of querying are always the same:

- Parse the statement for execution
- Bind data values (optional)
- Execute the statement
- Fetch the results from the database

A sample Python script is placed in your lab VM that runs a simple SQL on the autonomous database and display the results.

- First, ensure that you are in the **~/labs/python** directory.

```
$ cd ~/labs/python
```

- Copy the sample python script **python_resultset.py** from **/home/oracle/labfiles**.

```
$ cp ~/labfiles/python_resultset.py ~/labs/python
```

- Review the sample Python code using your favorite editor.

```
$ gedit python_resultset.py
```

- Notice that the variable **sql** is initialized with a SQL query. Also note that the bind variable **:country_code** can be passed to the SQL.

```
sql="""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 = :country_code
        GROUP BY channel_desc"""
```

- The **cursor()** method opens a cursor for statements to use.

```
cursor = conn.cursor()
```

- The **cursor.execute()** method parses and executes the SQL statement. Note that the SQL is passed as a variable **sql** and the bind variable **country_code** is set to **US**.

```
cursor.execute(sql, country_code = 'US')
```

- The **fetchall()** method fetches all rows returned by the SQL.

```
cursor.fetchall()
```

- Exit the editor.

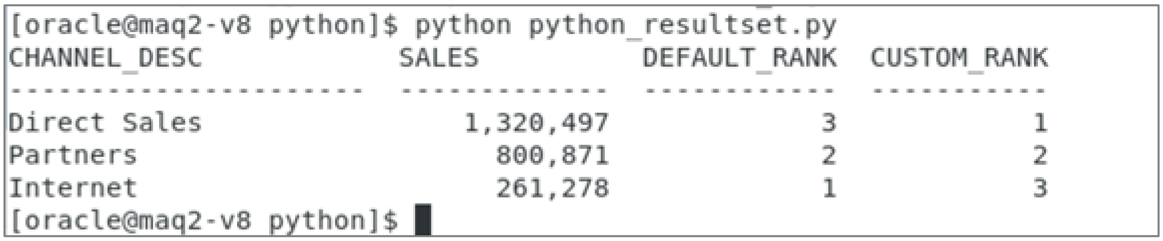
- If you are using a different terminal than the previous step, initialize the Linux shell variables with the database user name, password and the TNS alias. These variables are used within the Python scripts to connect to the database.

```
$ export ORAUSER=<User>
$ export ORAPASS=<Password>
$ export ORATNS=<TNS Alias>
```

- Now you are ready to run the script from the command prompt, as follows:

```
$> python python_resultset.py
```

- Observe the results. If the script succeeds, you will see the following results.



```
[oracle@maq2-v8 python]$ python python_resultset.py
CHANNEL_DESC            SALES          DEFAULT_RANK  CUSTOM_RANK
----------------------- -------------  ------------  -----------
Direct Sales                1,320,497             3            1
Partners                      800,871             2            2
Internet                      261,278             1            3
[oracle@maq2-v8 python]$ 
```

5. Managing the Autonomous Database

Lab 5-1: Start, Stop and Scale Autonomous Database

This section outlines the management activities that you would typically perform on Oracle Autonomous Database. Tasks such as starting and stopping the service and scaling the service are covered.

Objectives

- Start and Stop the ADB service
- Scale the ADB service

Required Artifacts

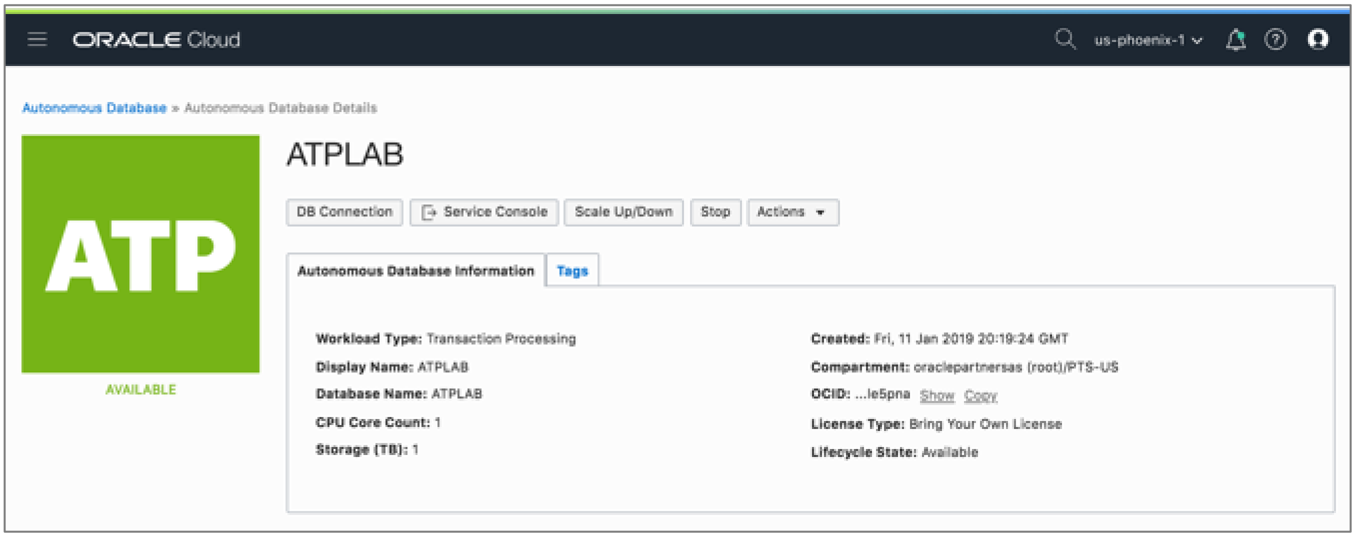
- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** instance.

Lab Steps

STEP 1: Stopping ATP Service

- Sign in to your ADB **Service Console** and browse to the **Autonomous Database Details** page of your service.

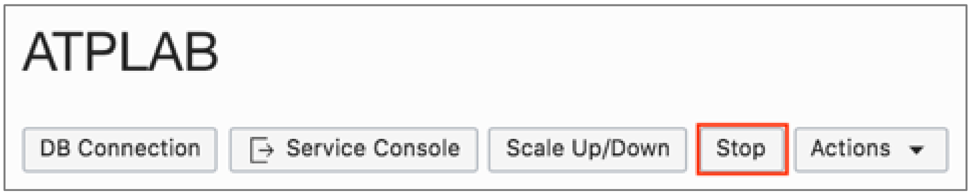
Note: Please ensure that you ONLY access the service you have created in the previous labs and not the ones created by others.



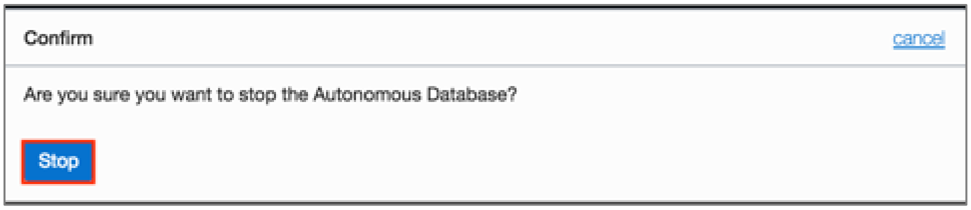
Workload Type: Transaction Processing
Display Name: ATPLAB
Database Name: ATPLAB
CPU Core Count: 1
Storage (TB): 1

Created: Fri, 11 Jan 2019 20:19:24 GMT
Compartment: oraclepartnersas (root)/PTS-US
OCID: ...le5pna
License Type: Bring Your Own License
Lifecycle State: Available

- Click **Stop** to stop the service.



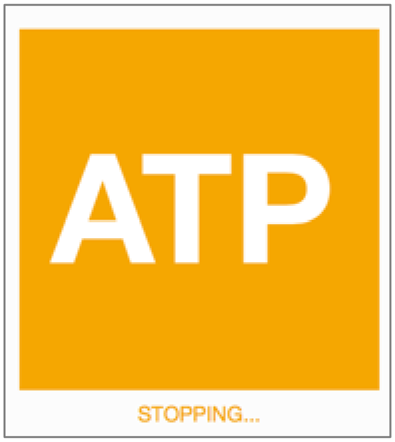
- Click **Stop** again when prompted for confirmation.


Confirm

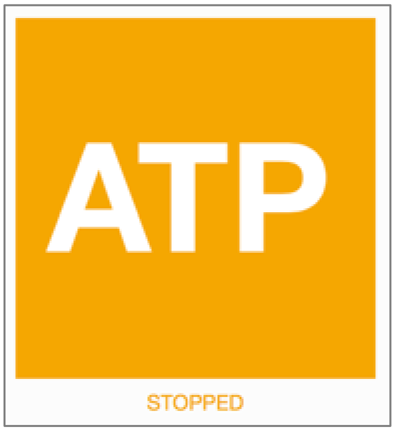
cancel

Are you sure you want to stop the Autonomous Database?

- The ADB service will take a few seconds to stop. Notice the status of **STOPPING**.

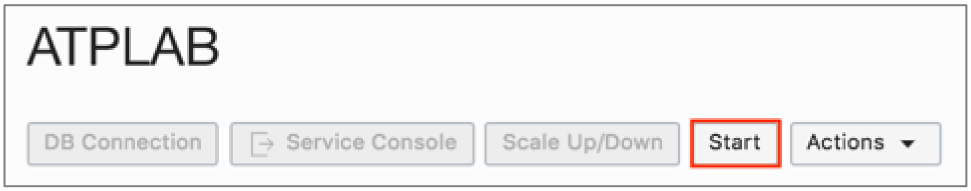


- When the service is stopped, the status will change to **STOPPED**.



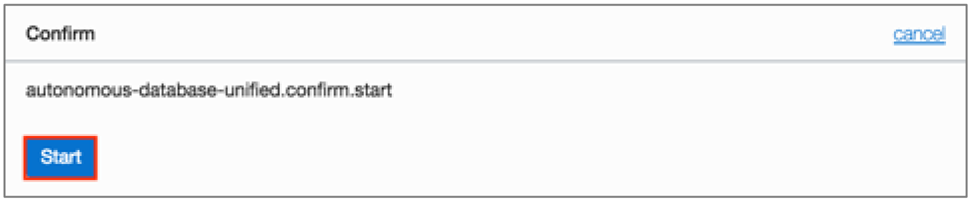
STEP 2: Starting ADB Service

- From the **Details** page of your ADB service, click **Start** to start your service.



ATPLAB

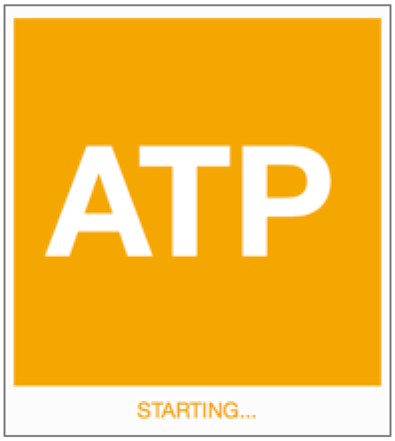
- Click **Start** again when prompted for confirmation.



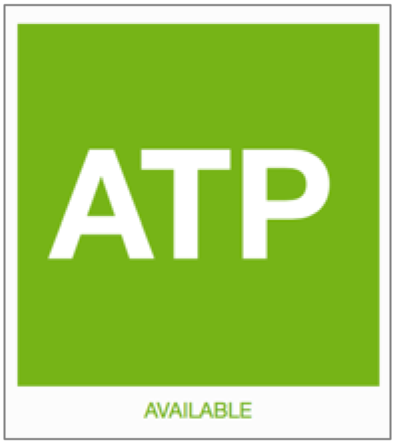
Confirm

autonomous-database-unified.confirm.start

- The ADB service will take a few seconds to start. For example, if you provisioned ATP service, you would notice the status of **STARTING** as follows:

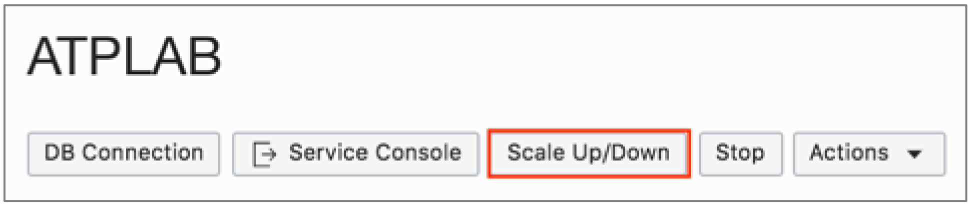


- When the service is started, the status will change to **AVAILABLE**.



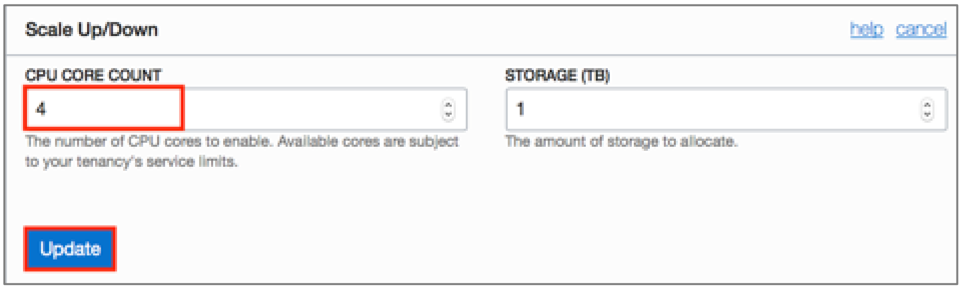
STEP 3: Scaling ADB Service

- From the **Details** page of your ADB service, click **Scale Up/Down** to scale your service.



- In the **Scale Up/Down** pop up, modify the **CPU CORE COUNT** to **4** and click **Update**.

Note: Please do not enter a CPU core count beyond **4** as the instance is shared by all students and you will run into resource limitations.


Scale Up/Down

help cancel

CPU CORE COUNT: 4

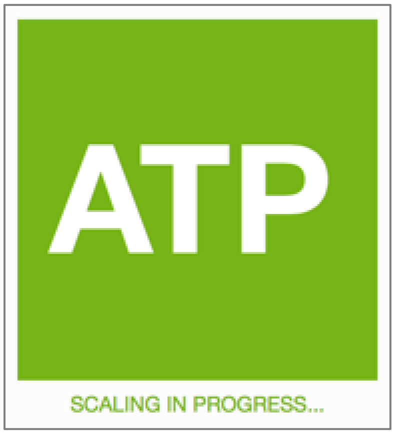
The number of CPU cores to enable. Available cores are subject to your tenancy's service limits.

STORAGE (TB): 1

The amount of storage to allocate.

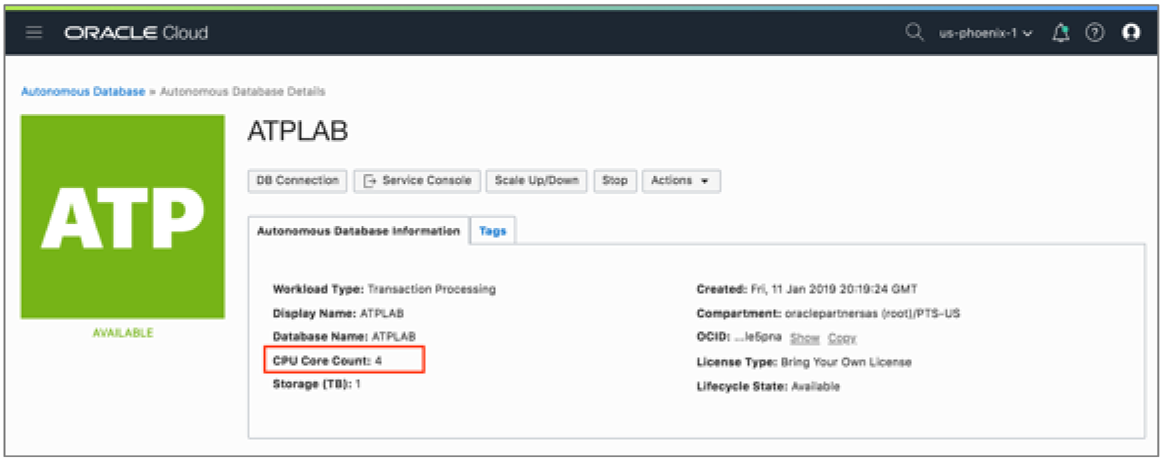
- The ADB service will take a few seconds to scale. Notice the status of **SCALING IN PROGRESS**.

Note that the **ATP** square remains green during the scaling process. No interruption to the service occurs during all scaling operations.



- When the scaling task is complete, the status will change to **AVAILABLE**.

- Note the new **CPU Core Count** is reflected in the service details.



ATPLAB

Workload Type: Transaction Processing
Display Name: ATPLAB
Database Name: ATPLAB
CPU Core Count: 4
Storage (TB): 1

Created: Fri, 11 Jan 2019 20:19:24 GMT
Compartment: oraclepartnersas (root)/PTS-US
OCID: ...le5pna Show Copy
License Type: Bring Your Own License
Lifecycle State: Available

- You have successfully completed the objectives of this lab.

Lab 5-2: Using REST APIs to Manage ADB

Often times, you would prefer to interact with your cloud services programmatically over REST rather than log into the cloud console and click through screens. Besides, by creating your own deployment and management scripts you can save and reuse your deployments, set gold standards and in fact store entire application infrastructure stacks as version-controlled code.

The Oracle Cloud Infrastructure APIs are typical REST APIs that use HTTPS requests and responses and support HTTPS and SSL protocol TLS 1.2, the most secure industry standards.

Also, All Oracle Cloud Infrastructure API requests must be signed for authentication purposes. To create and sign your API requests, you must:

- Form the HTTPS request (SSL protocol TLS 1.2 is required).
- Create the signing string, which is based on parts of the request.
- Create the signature from the signing string, using your private key and the RSASHA256 algorithm.
- Add the resulting signature and other required information to the Authorization header in the request.

While these seem like a lot of steps, they are meant to avoid using username/passwords and are based on the draft-cavage-httpsignatures-08 specification for secure communication over the internet.

Let's take a look at how to generate REST calls to the Oracle Cloud Infrastructure using a popular scripting language like node.js. You may use similar concepts to build scripts in Python, Golang, Ruby, Perl, Java, C#, bash or even curl!

Objectives

- Learn how to generate REST calls to the Oracle Cloud Infrastructure using node.js scripts.

Required Artifacts

- A previously provisioned Oracle **Autonomous Transaction Processing** or **Autonomous Data Warehouse** Database.
- Access to a lab VM with the following software preinstalled:
  - Node.js
  - Node.js libraries: oracledb, async, app, express

Lab Steps

STEP 1: Setup the Sample Node.js Application

A sample set of Node.js scripts that performs a simple create, lookup, delete operations on the ADB Cloud Services using REST has been prebuilt for this lab. In this step, you will copy the sample scripts from a location where it was previously downloaded.

- Login to the lab VM and start a **Terminal** session.

- Change directory to **~/labs/rest**

```
$ cd ~/labs/rest
```

- Copy the sample node.js scripts from **/home/oracle/labfiles** folder.

```
$ cp ~/labfiles/ATPRest.tar .
```

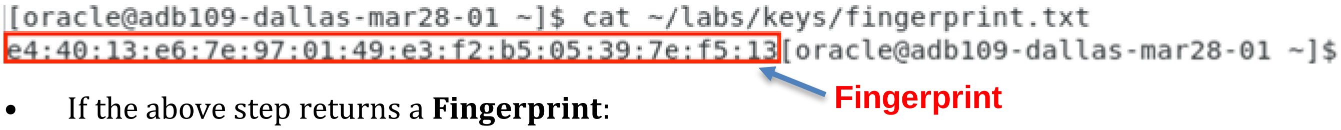
- Run the following **tar** command to extract the application source in the **~/labs/rest** folder:

```
$ tar xzvf ATPRest.tar
```

STEP 2: Generate SSH Key Pair in PEM Format and the API Fingerprint

- First, check if the **Fingerprint** has already been generated for you. Check the contents of **~/labs/keys/fingerprint.txt** file. If it is seeded with a value, then you would use this pre-generated **Fingerprint** in the later steps.

```
$ cat ~/labs/keys/fingerprint.txt
```



```
[oracle@adb109-dallas-mar28-01 ~]$ cat ~/labs/keys/fingerprint.txt
e4:40:13:e6:7e:97:01:49:e3:f2:b5:05:39:7e:f5:13[oracle@adb109-dallas-mar28-01 ~]$
```

Fingerprint

- If the above step returns a **Fingerprint**:
  - Copy the **Fingerprint** and paste it in a notepad as you will need later it to sign your API requests.
  - Copy the pre-generated **Private Key** file to **~/labs/rest** folder as you would need the private key in the later steps as well.

```
$ cp ~/labs/keys/oci_api_key.pem ~/labs/rest
```

STOP!

If you have copied the Fingerprint and the Private Key from above steps, skip the remaining instructions and proceed directly to Step 3.

- If the Fingerprint and Private Key has not been generated, follow the below steps:

- Generate a private key in PEM format that is suitable for openssl.

```
$ openssl genrsa -out ~/labs/rest/oci_api_key.pem 2048
```

Note: PEM stands for **Privacy Enhanced Mail** and is a widely used X.509 encoding format used for security certificates.

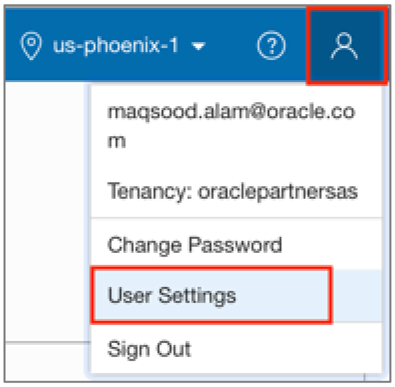
- Modify the permissions of the generated key.

```
$ chmod go-rwx ~/labs/rest/oci_api_key.pem
```

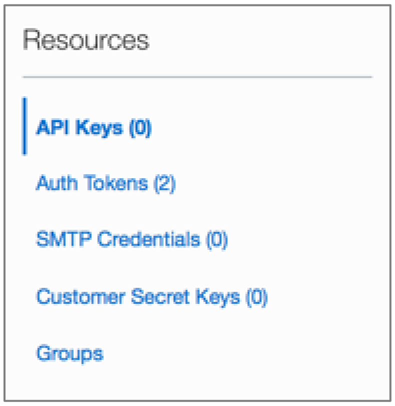
- Generate a PEM public key using the private key just created.

```
$ openssl rsa -pubout -in ~/labs/rest/oci_api_key.pem -out \
~/labs/rest/oci_api_key_public.pem
```

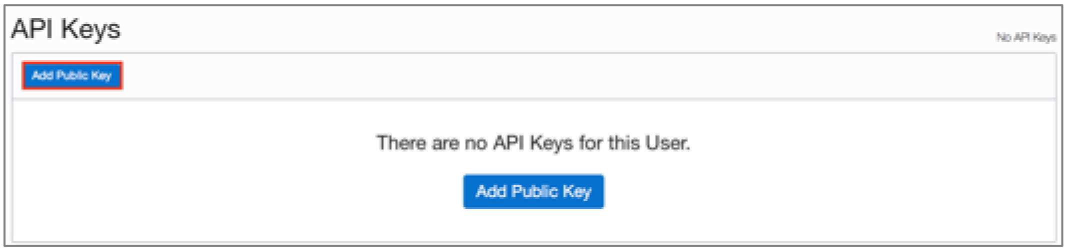
- Now you are ready to generate the **API Keys/Fingerprint** using the keys you just generated.
- Sign in to **Oracle Cloud Infrastructure Home Page**. Click on the top right user drop-down and select **User Settings**.



- On the **User Details** page, click on **API Keys** from the left **Resources** menu.



- Click **Add Public Keys**.



API Keys

There are no API Keys for this User.

- Go to the Terminal windows and cat the **oci_api_key_public.pem** file.

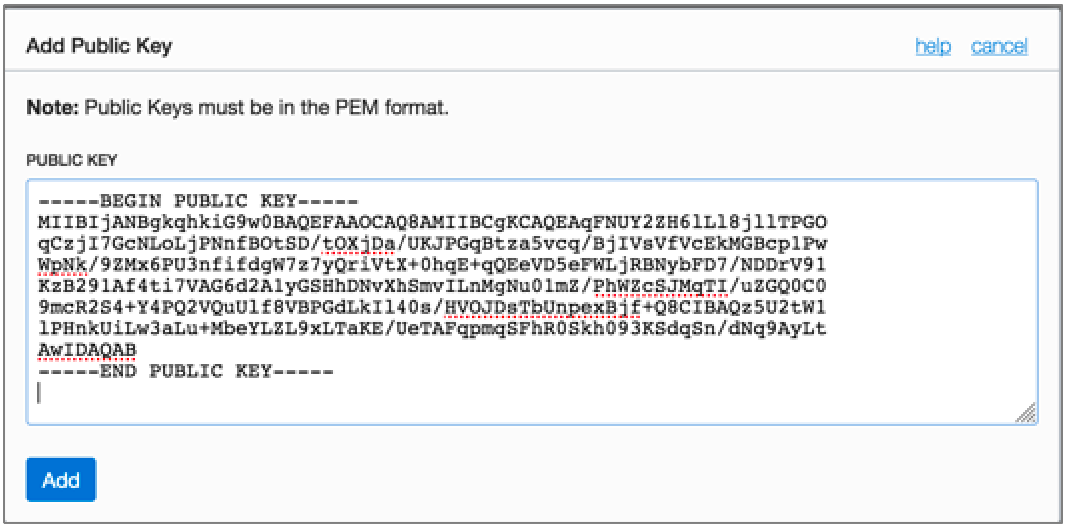
```
$ cat ~/labs/rest/oci_api_key_public.pem
```

- From the terminal, select and **COPY** the output from **BEGIN PUBLIC KEY** to **END PUBLIC KEY**.

Note: Below is an example output (**DO NOT COPY** from here):

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtMG9BPWCRdAo3+fzempt
eo7VSznMjap3Qy39QxvO3tCszp9kXir8LlRYq3h02UnBsXYBjVdFBVi1qaET976g
0TF8bqSQHwo7iu+E1L8g50OUUkkSqCBTyU18WgLWJGBVGKUAu7UmH19eGZDUH24o
66vuRTL1REjFDIJz7vdUPou5AUwUE90VcKnTAneMtOuIQ0WaHhVpVzic5C+2uKVn
uHcb11JSfZnM8MSvmGoiWyeFi8cKDj9gI2ahKaou0F8obOmNoMD31r2zd1KAiDfp
AGlMq4ZRtKCQyx2cdqyUVVEBPV9XHMIP8HJ/SawNO47eNzf1CQIN0XLY0TPcbDh4
YwIDAQAB
-----END PUBLIC KEY-----
```

- In the **Add Public Key** dialog, paste your **oci_api_key_public.pem** key and click on **Add**.



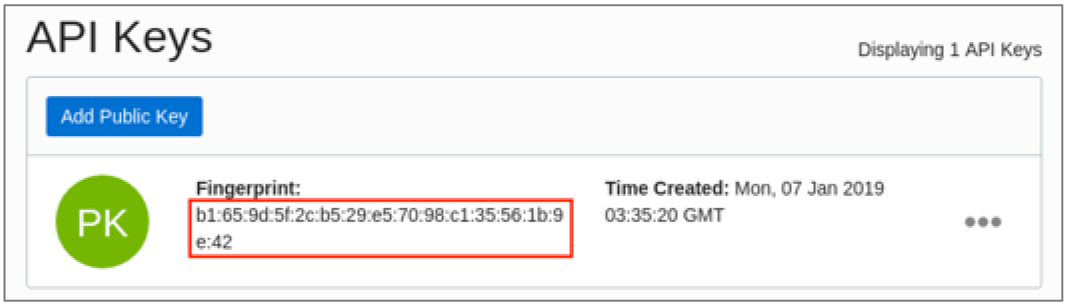
Add Public Key

Note: Public Keys must be in the PEM format.

PUBLIC KEY

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqFNUY2ZH6lL18j11TPGO
qCzjI7GcNLoLjPNnfBOtSD/tOXjDa/UKJPGqBtza5vcq/BjIVsVfVcEkMGBcplPw
WpNk/9ZMx6PU3nfifdgW7z7yQriVtX+0hqE+qQEeVD5eFWLjRBNybFD7/NDDrV91
KzB291Af4ti7VAG6d2AlyGSHhDNvXhSmvILnMgNu01mZ/PhWZcSJMqTI/uZGQ0C0
9mcR2S4+Y4PQ2VQuUlf8VBPGdLkI140s/HVOJDsTbUnpexBjf+Q8CIBAQz5U2tW1
lPHnkUiLw3aLu+MbeYLZL9xLTaKE/UeTAFqpmqSFhR0Skh093KSdqSn/dNq9AyLt
AwIDAQAB
-----END PUBLIC KEY-----
```

- The service generates a **Fingerprint**. Copy the fingerprint and paste it in a notepad as you will need it later to sign your API requests.



API Keys

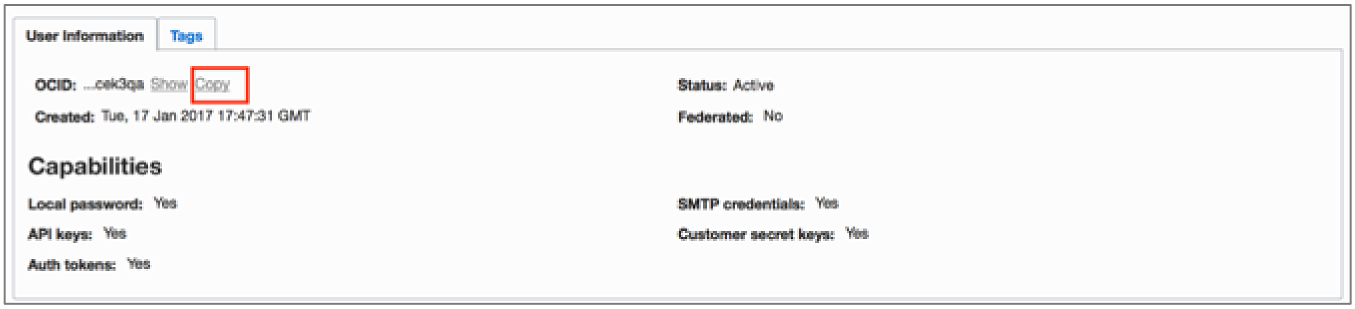
Displaying 1 API Keys

Fingerprint: b1:65:9d:5f:2c:b5:29:e5:70:98:c1:35:56:1b:9e:42

Time Created: Mon, 07 Jan 2019 03:35:20 GMT

STEP 3: Get OCI Authentication Info

- In this step you will:
  - Get the **User OCID**
  - Get the **Tenancy OCID**
  - Get the **Compartment OCID**
- Ensure you are logged in to the **Oracle Cloud Infrastructure Home Page**.
- Get the **User OCID**. On the same page, locate the **User Information** section and **COPY** the user **OCID** and paste it in a notepad. You will need this info later as well.



OCID: ...cek3qa

Created: Tue, 17 Jan 2017 17:47:31 GMT

Status: Active

Federated: No

Capabilities

Local password: Yes

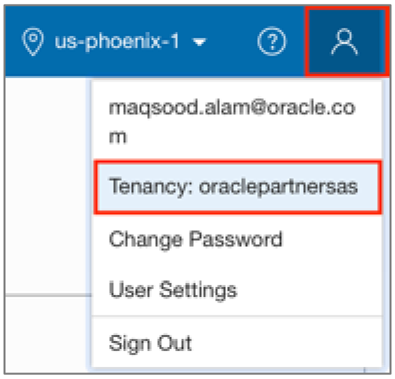
API keys: Yes

Auth tokens: Yes

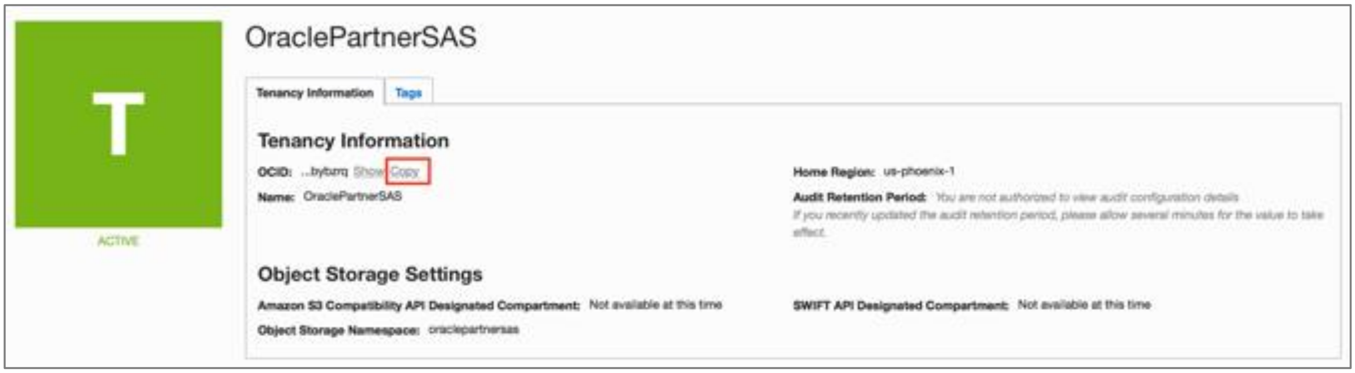
SMTP credentials: Yes

Customer secret keys: Yes

- Get the **Tenancy OCID**. Click on the top right user drop-down and select **Tenancy**.



- **Copy** the **Tenancy OCID** and also paste it in a notepad.



T

ACTIVE

OraclePartnerSAS

Tenancy Information | Tags

Tenancy Information

OCID: ...bybzrq Show Copy

Name: OraclePartnerSAS

Home Region: us-phoenix-1

Audit Retention Period: *You are not authorized to view audit configuration details*
If you recently updated the audit retention period, please allow several minutes for the value to take effect.

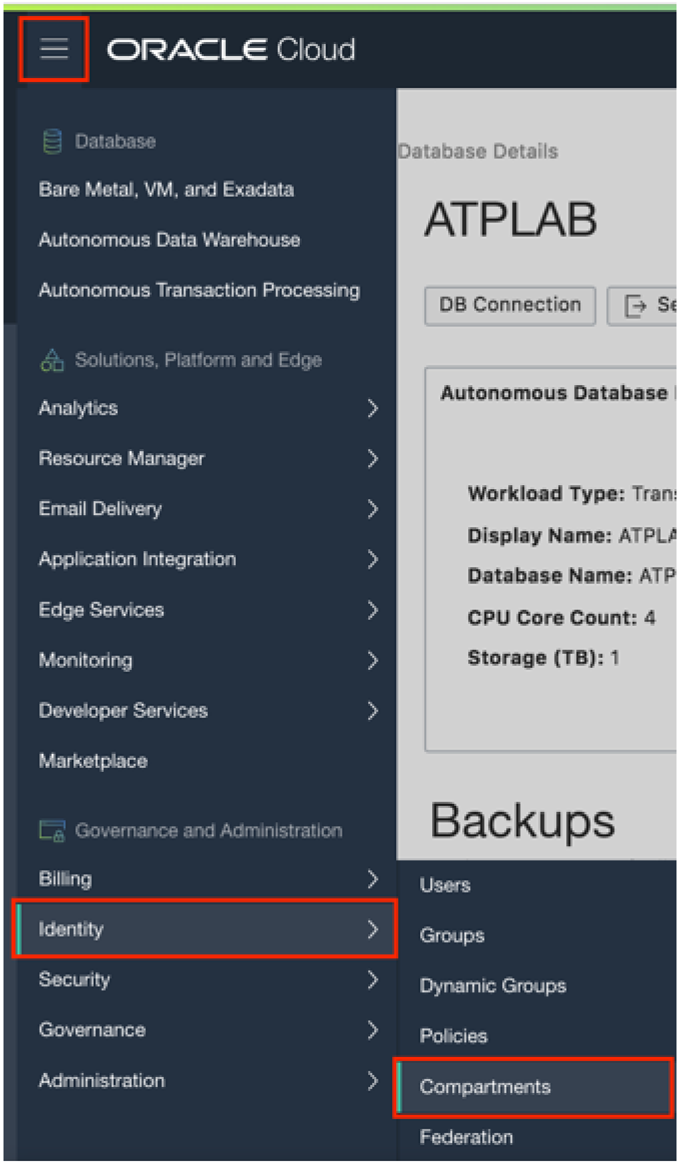
Object Storage Settings

Amazon S3 Compatibility API Designated Compartment: Not available at this time

SWIFT API Designated Compartment: Not available at this time

Object Storage Namespace: oraclepartnersas

- Get the **Compartment OCID**. Click on the top left **Menu** and select **Identity -> Compartments**.



Workload Type: Trans

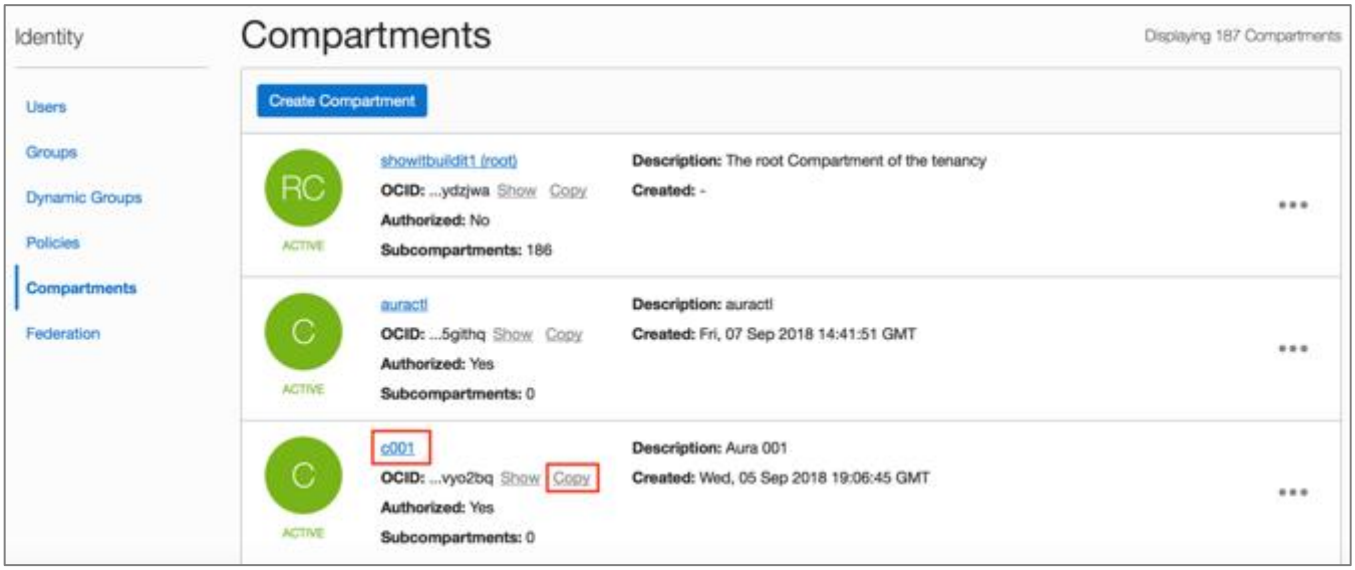
Display Name: ATPLA

Database Name: ATP

CPU Core Count: 4

Storage (TB): 1

- On the **Compartment** section, locate the compartment assigned to you by your instructor in the lab handout. Click **Copy** to copy the **Compartment OCID** and paste the values to the notepad as well.



Compartments

Displaying 187 Compartments

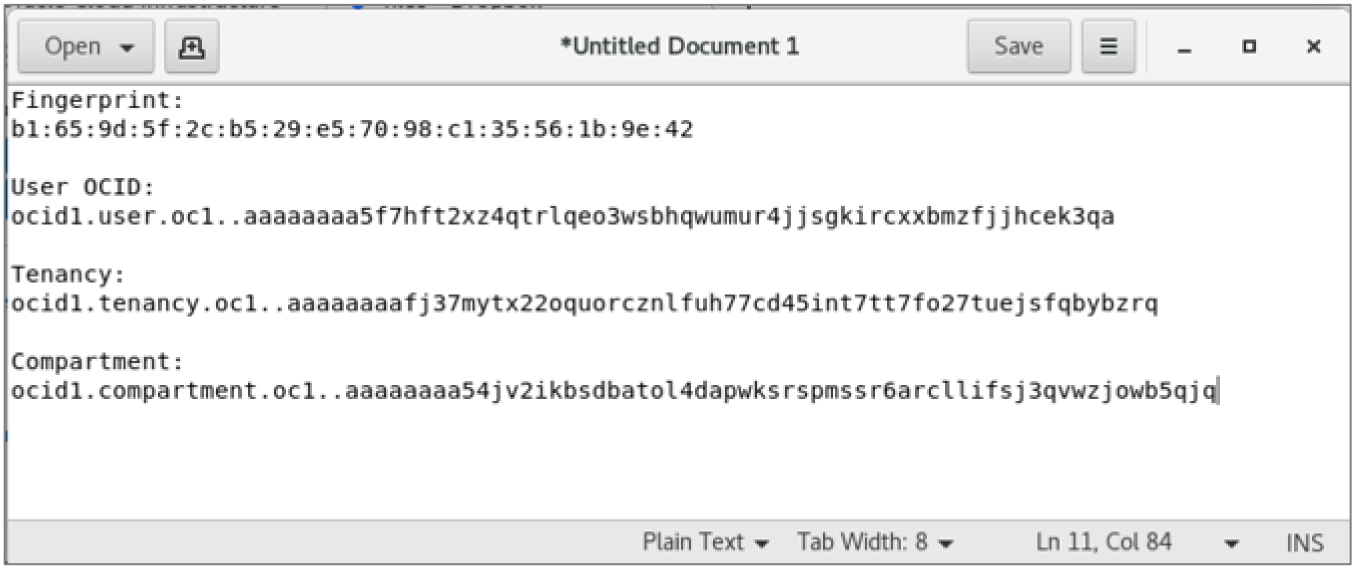
showitbuildit1 (root)
OCID: ...ydzjwa Show Copy
Authorized: No
Subcompartments: 186
Description: The root Compartment of the tenancy
Created: -

auractl
OCID: ...5githq Show Copy
Authorized: Yes
Subcompartments: 0
Description: auractl
Created: Fri, 07 Sep 2018 14:41:51 GMT

c001
OCID: ...vyo2bq Show Copy
Authorized: Yes
Subcompartments: 0
Description: Aura 001
Created: Wed, 05 Sep 2018 19:06:45 GMT

- The **Fingerprint, Tenancy OCID, User OCID, Compartment OCID** copied earlier, along with user's **Private Key** make up a unique signature that is used to sign the REST requests.

- Your notepad may look something similar to this:



```
Fingerprint:
b1:65:9d:5f:2c:b5:29:e5:70:98:c1:35:56:1b:9e:42

User OCID:
ocid1.user.oc1..aaaaaaaa5f7hft2xz4qtrlqeo3wsbhqwumur4jjsgkircxxbmzfjjhcek3qa

Tenancy:
ocid1.tenancy.oc1..aaaaaaaafj37mytx22oquorcznlfuh77cd45int7tt7fo27tuejsfqbybzrq

Compartment:
ocid1.compartment.oc1..aaaaaaaa54jv2ikbsdbatol4dapwksrspmssr6arcllifsj3qvwzjowb5qjq
```

Note: It is extremely important that you do not share this with anyone or expose it over an unencrypted network.

STEP 4: Create an ATP Database using REST

- Change directory to the folder where Node.js scripts were uncompressed earlier.

```
$ cd ~/labs/rest/ATPRest
```

- Edit the **auth.js** script. This module has all the user authentication information used to generate the signature and other header information including compartments.

- You will need to replace the values of the following variables with the values you copied to the notepad:
  - **tenancyId** : The **Tenancy OCID** copied earlier.
  - **authUserId** : The **User OCID** copied earlier.
  - **keyFingerprint** : The **Fingerprint** copied earlier.
  - **compartments** : The **Compartment OCID** copied earlier.
  - **privateKeyPath** : The location of your private key file, mainly **/home/oracle/labs/rest/oci_api_key.pem**.
- Your **auth.js** file should look similar to:

```
var tenancyId=
"ocid1.tenancy.oc1..aaaaaaaawrgt5au6hbledhhyas2secm3q2atqiuvihck45rbi3jyc5tfyfga";
var authUserId=
"ocid1.user.oc1..aaaaaaaavscszlxxcf2wnq73nxpguxtubvpxaklqbspmuuml7xxp26mbxmgq";
var keyFingerprint = "0d:33:d8:eb:9a:48:2a:15:cd:36:2e:f5:20:fe:b3:d3";
var Compartment =
"ocid1.compartment.oc1..aaaaaaaai75jrzzbfe6t43fd6yivpqngk2tufisqnohacf3s26p6uhcgz7b
q";
var privateKeyPath = "/home/oracle/labs/rest/oci_api_key.pem";
```

- Browse the **region.js** script. This module lists all the API endpoints for OCI. You do not need to change anything here unless a new service gets added or Oracle makes a change to the URLs.
- Browse the **headers.js** scripts. This module builds API signing keys and generates https headers required for your REST calls depending upon whether it's a GET, PUT, POST or DELETE call. You do not need to modify anything here.
- It also has an option getUser method used in every REST call to get user information from the Identity and Access Management services. You may use that example to generate other IAM REST calls.
- The other scripts, mainly **createAutonomousDatabase.js, listAutonomousDatabase.js, createVCN.js, getAutonomousDatabase.js, deleteAutonomousDatabase.js,** are the scripts you would need to run. Make sure the variable in each of these scripts are set right before you run them.
- Edit the **createAutonomousDatabase.js** script and replace the following to match your requirements:
  - **displayName** : ATP Database Display Name (Append your initials or a number for uniqueness)
  - **dbName** : ATP Database Name (Append your initials or a number for uniqueness)
  - **adminPassword** : Password for the ADMIN user

  - **cpuCoreCount** : CPU Core Count
  - **dataStorageInTbs** : Storage in TB for the Service
  - **host region** : Region for the Service

- For example, here is how **createAutonomousDatabase.js** will look after your edits:

```
function createATP(callback) {
var body = JSON.stringify({
  "compartmentId" : auth.Compartment,
  "displayName" : "ATPLab10",
  "dbName" : "ATPLab10",
  "adminPassword" : "AVeryLongPassword321!",
  "cpuCoreCount" : 1,
  "dataStorageSizeInTBs" : 1
});
```

Ensure uniqueness by appending your initials to **dbName** and **displayName**

- Modify the region as identified by the **host** variable of the **options** structure. The valid values for **host** for your **Region** are (use the **Region** that was allocated to you by the instructor):

  - us-phoenix-1 : **regions.dbPhoenixRegion**
  - us-ashburn-1 : **regions.dbAshburnRegion**
  - us-frankfurt-1 : **regions.dbFrankfurtRegion**
  - us-london-1 : **regions.dbLondonRegion**

- Modify the value of the **host** variable according to the **Region** allocated to you:

```
var options = {
        host: <region>,
        path: '/20160918/autonomousDatabases',
        method: 'POST',
        headers: {
            "Content-Type": "application/json"
        }
    };
```

- Now, let's create an ATP Service running the **createAutonomousDatabase.js**:

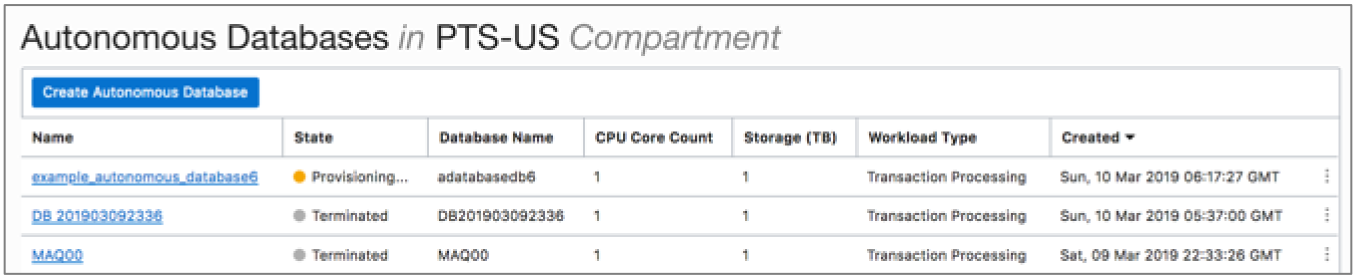
```
$ node createAutonomousDatabase.js
```

- Note the output from the script doesn't have any errors. Also note that the last state of the REST call was **PROVISIONING.**

```
[oracle@localhost ATPRest]$ node createAutonomousDatabase.js
{ capabilities:
   { canUseConsolePassword: true,
     canUseApiKeys: true,
     canUseAuthTokens: true,
     canUseSmtpCredentials: true,
     canUseCustomerSecretKeys: true },
  identityProviderId: null,
  externalIdentifier: null,
  timeModified: '2019-01-07T03:35:20.069Z',
  id:
   'ocid1.user.oc1..aaaaaaaa5f7hft2xz4qtrlqeo3wsbhqwumur4jjsgkircxxbmzfjjhcek3qa',
  compartmentId:
   'ocid1.tenancy.oc1..aaaaaaaafj37mytx22oquorcznlfuh77cd45int7tt7fo27tuejsfqbybzrq',
  name: 'maqsood.alam@oracle.com',
  description: 'Maqsood Alam',
  timeCreated: '2017-01-17T17:47:31.861Z',
  freeformTags: {},
  definedTags: {},
  lifecycleState: 'ACTIVE' }

CREATING ATP Service:
{ additionalDatabaseStatus: null,
  autonomousPodId: null,
  compartmentId:
   'ocid1.compartment.oc1..aaaaaaaa54jv2ikbsdbatol4dapwksrspmssr6arcllifsj3qvwzjowb5qjq',
  connectionStrings: null,
  cpuCoreCount: 1,
  dataStorageSizeInTBs: 1,
  dbName: 'ATPLab10',
  dbVersion: null,
  definedTags: {},
  displayName: 'ATPLab10',
  freeformTags: {},
  id:

'ocid1.autonomousdatabase.oc1.phx.abyhqljs56zurehbnt6anuaim7gofsu4qeuild6372v3exojxi2h3n3xcbf
a',
  isDedicated: false,
  licenseModel: 'BRING_YOUR_OWN_LICENSE',
  lifecycleDetails: null,
  lifecycleState: 'PROVISIONING',
  serviceConsoleUrl: null,
  timeCreated: '2019-01-07T04:37:54.540Z' }
[oracle@localhost ATPRest]$
```

- Verify the service has started provisioning. Browse to the **Autonomous Transaction Processing** page and locate the service with the ATP Service name you provisioned.

Note: If you do not see your service listed, ensure that you have selected the correct **Region**.



Autonomous Databases *in* PTS-US *Compartment*

Name	State	Database Name	CPU Core Count	Storage (TB)	Workload Type	Created ▾
example_autonomous_database6	● Provisioning...	adatabasedb6	1	1	Transaction Processing	Sun, 10 Mar 2019 06:17:27 GMT
DB 201903092336	● Terminated	DB201903092336	1	1	Transaction Processing	Sun, 10 Mar 2019 05:37:00 GMT
MAQ00	● Terminated	MAQ00	1	1	Transaction Processing	Sat, 09 Mar 2019 22:33:26 GMT

- You have successfully created the autonomous database using REST API.

- You may build similar scripts using Python, Java, golang, Perl, C#, bash and Curl.

- You have successfully completed this lab.

6. Performance Monitoring

Lab 6-1: Database Services and Resource Management in ATP

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

- **<database_name>_low**
- **<database_name>_medium**
- **<database_name>_high**
- **<database_name>_tp**
- **<database_name>_tpurgent**

These services map to the **LOW, MEDIUM, HIGH, TP and TPURGENT** resource manager consumer groups. As a user you need to pick the database service based on your performance and concurrency requirements.

For example, a user connecting with the ***_low** service uses the consumer group **LOW**. The **TP** services are focused on high concurrency low parallelism operations whereas low, medium and high focus on increasing levels of serialized execution of highly parallelized operations. The basic characteristics of these consumer groups are:

- **HIGH** : A high priority application connection service for reporting and batch operations. All operations run in parallel 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 reporting or batch processing operations. This connection service does not run with parallelism.
- **TP** : A typical application connection service for transaction processing operations. This connection service does not run with parallelism.
- **TPURGENT** : The highest priority application connection service for time critical transaction processing operations. This connection service supports manual parallelism.

In this section you will connect to **TP** database services and observe resource allocations.

Objectives

- Observe Resource Manager allocations for consumer groups.

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** instance.

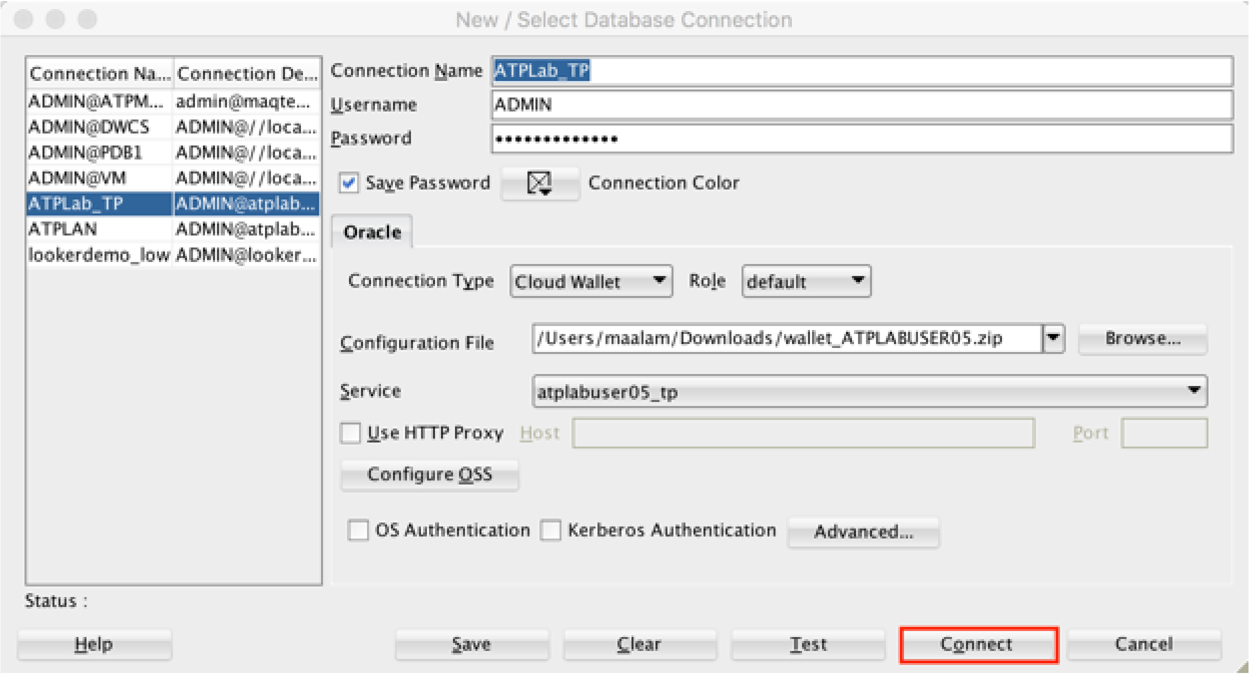
- **Oracle SQL Developer** installed, or access to a lab VM with Oracle SQL Developer.

Lab Steps

STEP 1: Query Resource Manager Allocations

Oracle Database Resource Manager enables you to manage multiple workloads within a database that are contending for system and database resources. Oracle Autonomous Database utilizes the Resource Manager extensively to control resource allocation to user sessions, such as for CPU and Parallelism.

- In a previous lab you created a connection to the ***_TP** service from SQL Developer using the **ADMIN** user. Let's connect to this TP service.



New / Select Database Connection

Connection Na...	Connection De...
ADMIN@ATPM...	admin@maqte...
ADMIN@DWCS	ADMIN@//loca...
ADMIN@PDB1	ADMIN@//loca...
ADMIN@VM	ADMIN@//loca...
ATPLab_TP	ADMIN@atplab...
ATPLAN	ADMIN@atplab...
lookerdemo_low	ADMIN@looker...

Connection Name: ATPLab_TP
Username: ADMIN
Password: **************
☑ Save Password
Connection Color

Oracle

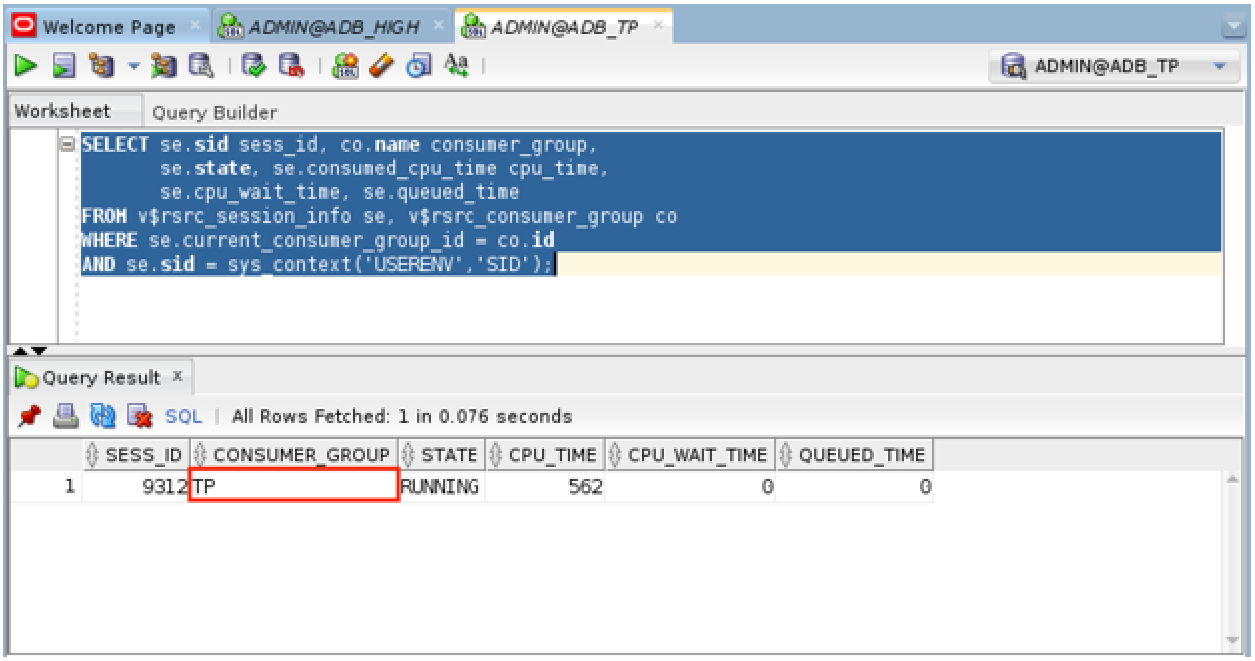
Connection Type: Cloud Wallet
Role: default
Configuration File: /Users/maalam/Downloads/wallet_ATPLABUSER05.zip
Service: atplabuser05_tp
□ Use HTTP Proxy Host Port
□ OS Authentication □ Kerberos Authentication

Status :

- Run the following SQL to check the **Consumer Group** of your connected session.

```
SELECT se.sid sess_id, co.name consumer_group,
       se.state, se.consumed_cpu_time cpu_time,
       se.cpu_wait_time, se.queued_time
FROM v$rsrc_session_info se, v$rsrc_consumer_group co
WHERE se.current_consumer_group_id = co.id
AND se.sid = sys_context('USERENV','SID');
```

- Notice that you are connected to the **TP** consumer group which is to be expected.



```sql
SELECT se.sid sess_id, co.name consumer_group,
       se.state, se.consumed_cpu_time cpu_time,
       se.cpu_wait_time, se.queued_time
FROM v$rsrc_session_info se, v$rsrc_consumer_group co
WHERE se.current_consumer_group_id = co.id
AND se.sid = sys_context('USERENV','SID');
```

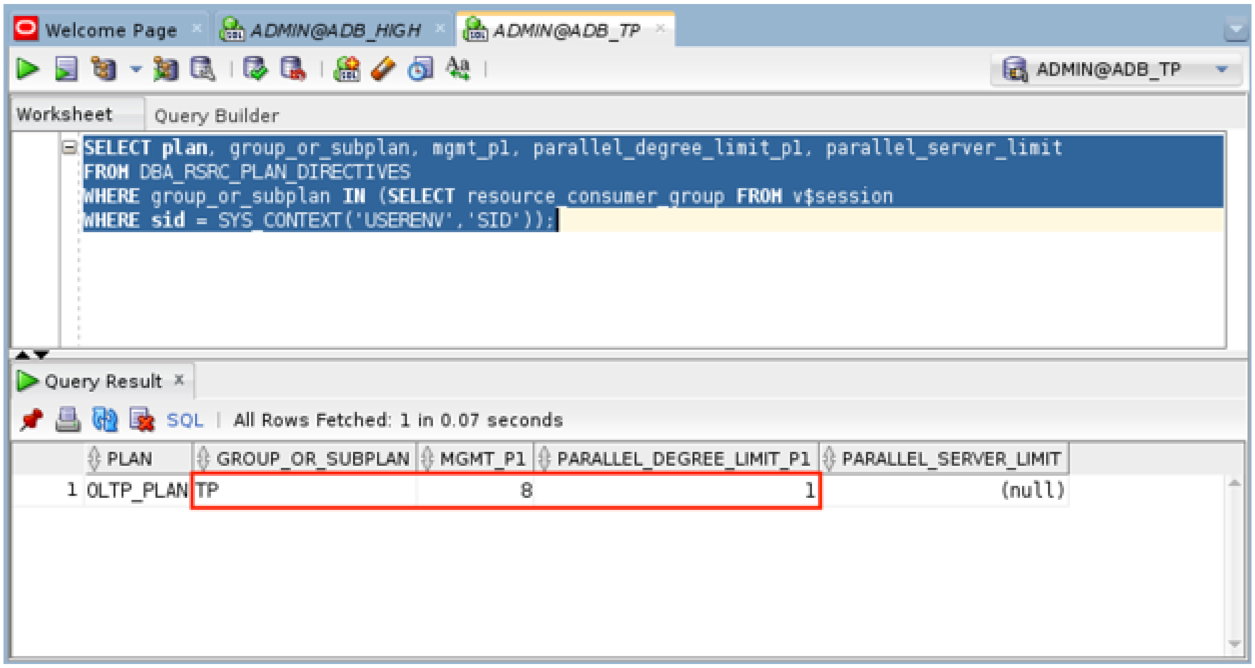
All Rows Fetched: 1 in 0.076 seconds

	SESS_ID	CONSUMER_GROUP	STATE	CPU_TIME	CPU_WAIT_TIME	QUEUED_TIME
1	9312	TP	RUNNING	562	0	0

- Next, check the resource allocations of your connected session.

```sql
SELECT plan, group_or_subplan, mgmt_p1, parallel_degree_limit_p1,
parallel_server_limit
FROM DBA_RSRC_PLAN_DIRECTIVES
WHERE group_or_subplan IN (SELECT resource_consumer_group FROM v$session
WHERE sid = SYS_CONTEXT('USERENV','SID'));
```

- Note that the TP consumer group is allotted **8 CPU** shares and a **Parallel Degree Limit** of **1** (no parallel).



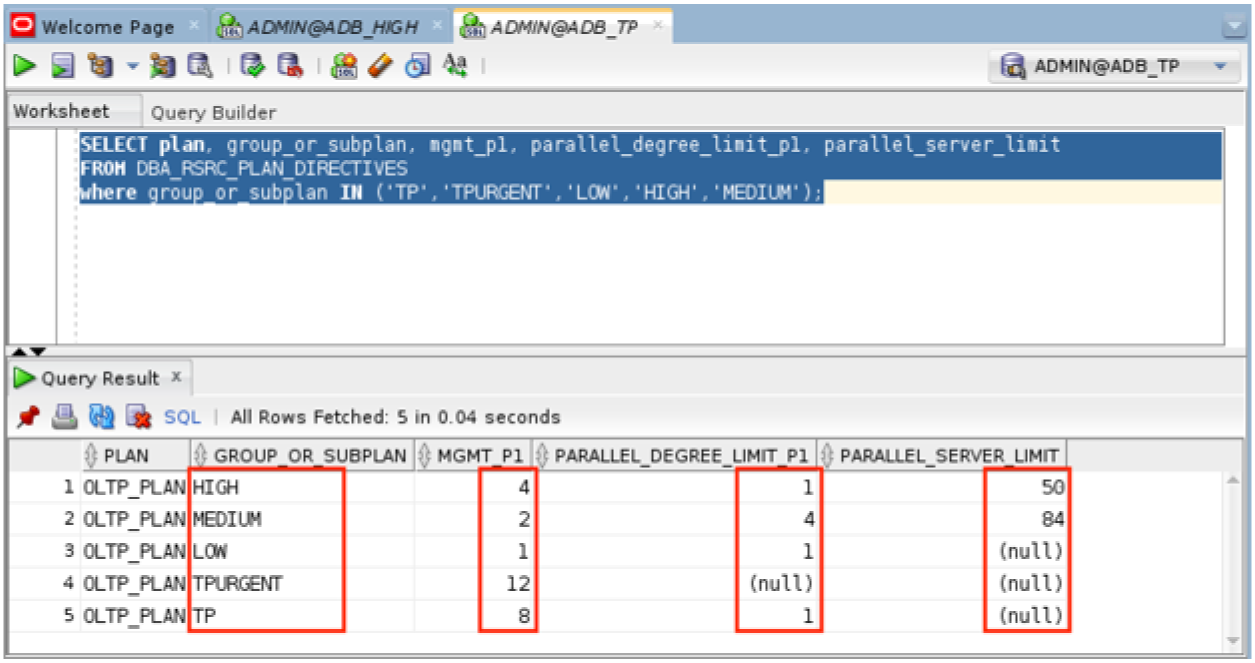
```sql
SELECT plan, group_or_subplan, mgmt_p1, parallel_degree_limit_p1, parallel_server_limit
FROM DBA_RSRC_PLAN_DIRECTIVES
WHERE group_or_subplan IN (SELECT resource_consumer_group FROM v$session
WHERE sid = SYS_CONTEXT('USERENV','SID'));
```

All Rows Fetched: 1 in 0.07 seconds

	PLAN	GROUP_OR_SUBPLAN	MGMT_P1	PARALLEL_DEGREE_LIMIT_P1	PARALLEL_SERVER_LIMIT
1	OLTP_PLAN	TP	8	1	(null)

- Check the resource allocations of other **Consumer Groups**.

```
SELECT plan, group_or_subplan, mgmt_p1, parallel_degree_limit_p1,
parallel_server_limit
FROM DBA_RSRC_PLAN_DIRECTIVES
where group_or_subplan IN ('TP','TPURGENT','LOW','HIGH','MEDIUM');
```



```
SELECT plan, group_or_subplan, mgmt_p1, parallel_degree_limit_p1, parallel_server_limit
FROM DBA_RSRC_PLAN_DIRECTIVES
where group_or_subplan IN ('TP','TPURGENT','LOW','HIGH','MEDIUM');
```

All Rows Fetched: 5 in 0.04 seconds

	PLAN	GROUP_OR_SUBPLAN	MGMT_P1	PARALLEL_DEGREE_LIMIT_P1	PARALLEL_SERVER_LIMIT
1	OLTP_PLAN	HIGH	4	1	50
2	OLTP_PLAN	MEDIUM	2	4	84
3	OLTP_PLAN	LOW	1	1	(null)
4	OLTP_PLAN	TPURGENT	12	(null)	(null)
5	OLTP_PLAN	TP	8	1	(null)

- You have successfully completed this lab.

Lab 6-2: Scalability and Performance

You have seen by now that Oracle Autonomous Database provides the capability to dynamically scale your compute capacity. The expectation is that when you scale up and add more CPUs, you will get higher performance and throughput.

In this lab you will scale the CPUs and observe the impact on workload performance.

Objectives

- Scale CPUs and observe the impact on query performance using SQL Developer.

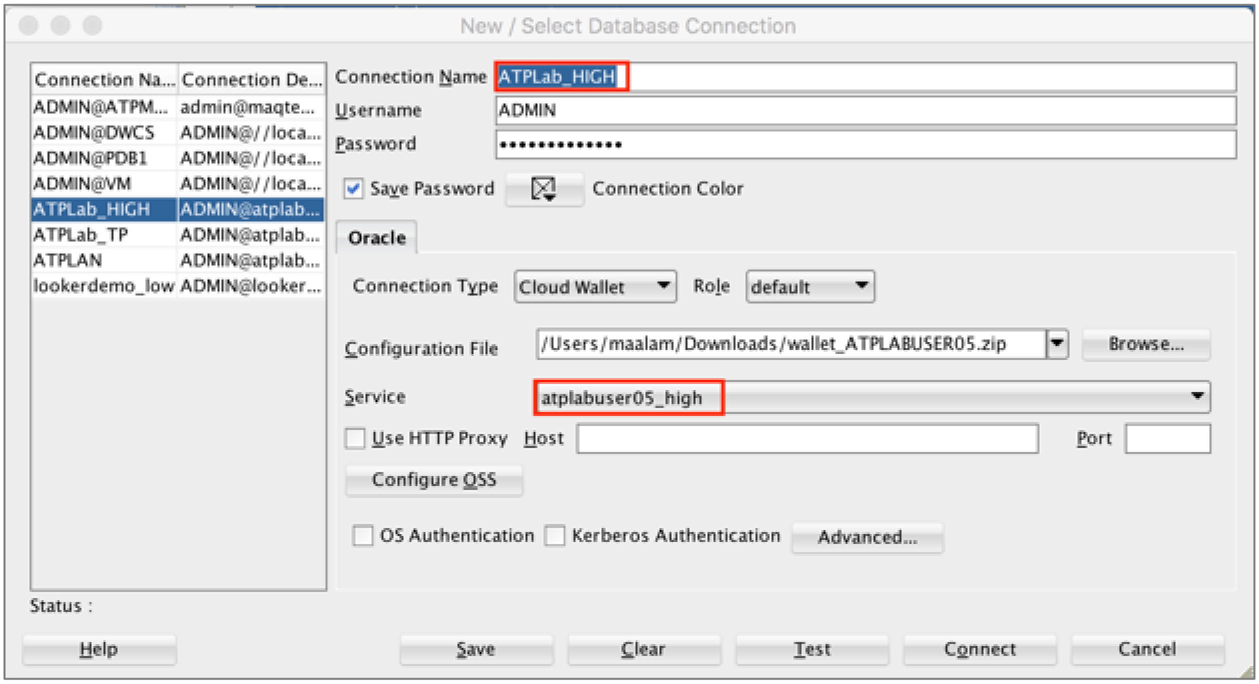
Required Artifacts

- Ensure you have provisioned an Oracle **Autonomous Transaction Processing** service.
- Ensure that **Oracle SQL Developer** installed, or you have access to a lab VM.

Lab Steps

STEP 1: Create a Connection to the *_HIGH Service

- In the previous labs you have created a connection in SQL Developer to connect to the ***_TP** service or the ***_LOW** service. Now, create a new connection to connect to the ***_HIGH** service as the **ADMIN** user, using the instructions from an earlier lab.



New / Select Database Connection

Connection Na...	Connection De...
ADMIN@ATPM...	admin@maqte...
ADMIN@DWCS	ADMIN@//loca...
ADMIN@PDB1	ADMIN@//loca...
ADMIN@VM	ADMIN@//loca...
ATPLab_HIGH	ADMIN@atplab...
ATPLab_TP	ADMIN@atplab...
ATPLAN	ADMIN@atplab...
lookerdemo_low	ADMIN@looker...

Connection Name: ATPLab_HIGH
Username: ADMIN
Password: ••••••••••••••
☑ Save Password
Connection Color

Oracle

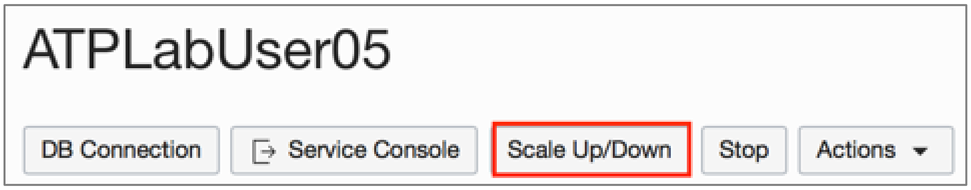
Connection Type: Cloud Wallet
Role: default
Configuration File: /Users/maalam/Downloads/wallet_ATPLABUSER05.zip
Service: atplabuser05_high
□ Use HTTP Proxy Host Port
□ OS Authentication □ Kerberos Authentication

Status :

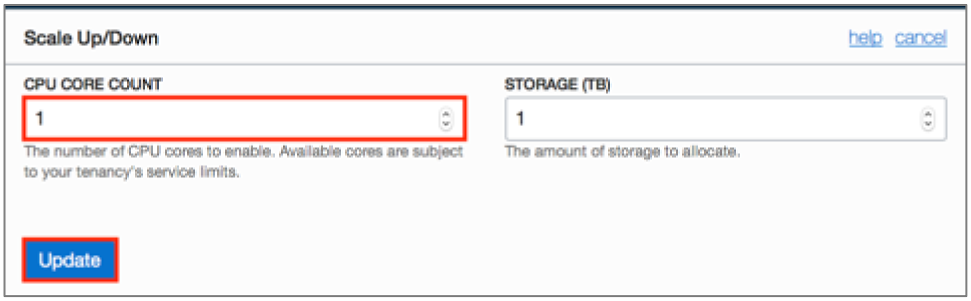
- Click on **Test**, then **Save**, and then **Cancel** to save the connection and exit the dialog.

STEP 2: Scale Down CPUs and Run a Sample Query

- In one of the previous labs you may have scaled up the CPUs. Let's scale them down and run a sample query to record execution time (skip this step if your service is running with **1** CPUs already).
- Browse to the **Autonomous Transaction Processing Database Details** page of your ATP service and click **Scale Up/Down** to scale your service.



- In the **Scale Up/Down** pop up, modify the **CPU Core Count** to **1** and click **Update**.



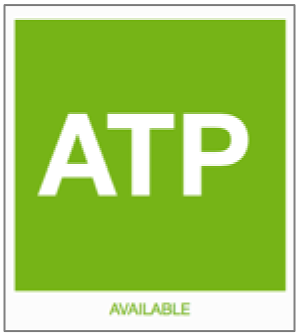
CPU CORE COUNT: 1

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

STORAGE (TB): 1

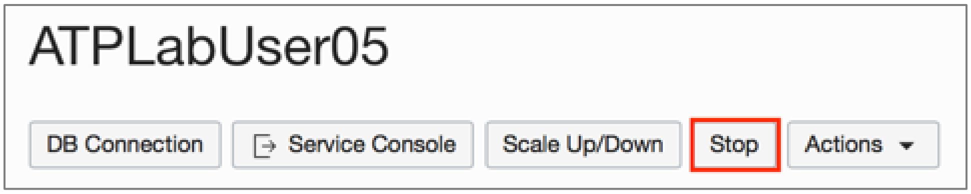
The amount of storage to allocate.

- Wait for the scaling task to complete.

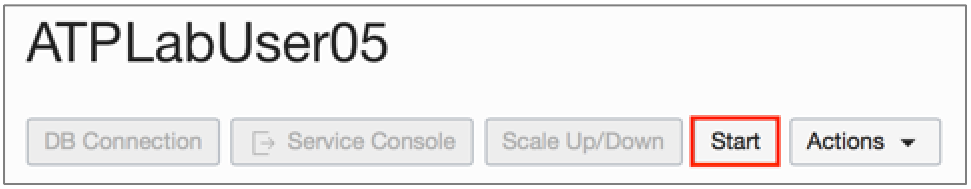


IMPORTANT: To get a good performance baseline, we need to restart the ADB service. This is to clear the buffer cache of cached query results and to get realistic query runtime when there is IO involved. This is needed for apples-to-apples comparison.

- Click **Stop** to stop the service.

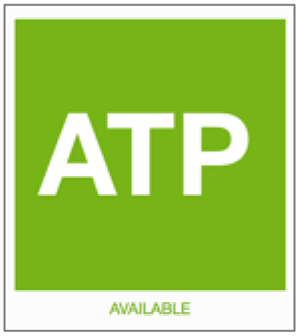


- Click **Start** to start your service.



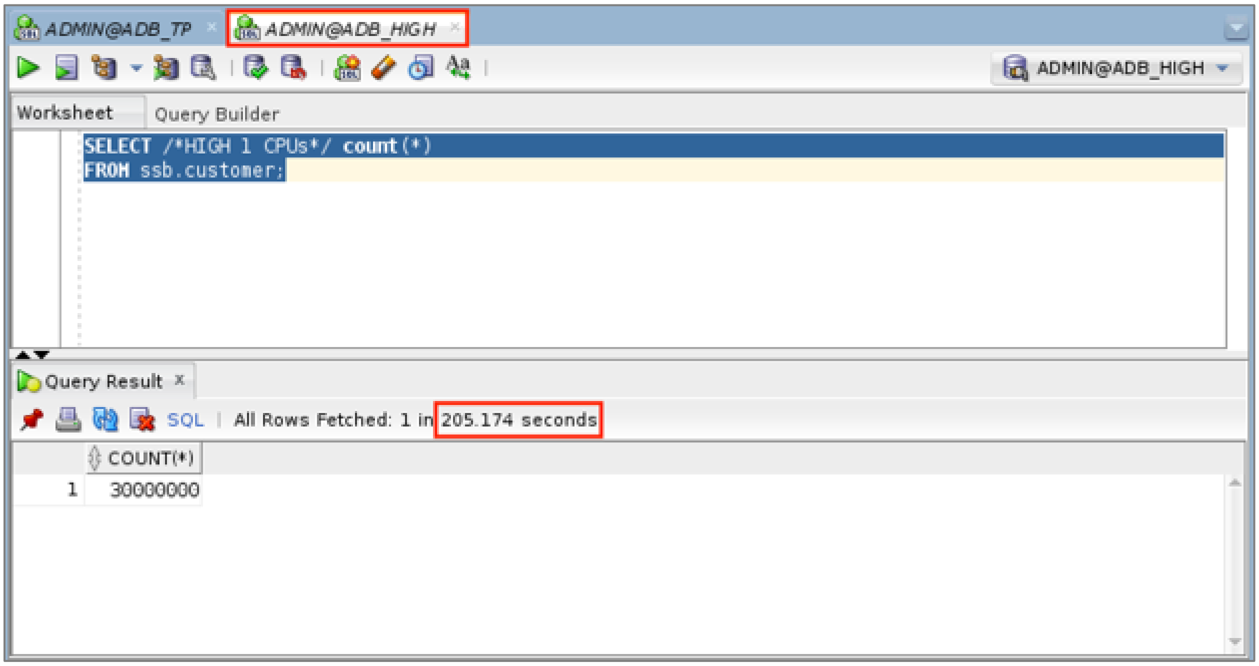
ATPLabUser05

- Wait for the service to start.



- Connect to **HIGH** service connection that you have just created in SQL Developer. Copy and paste the following example query and select **Run Script** (or F5). Note the runtime of the query. (if you connected already before the Stop/Start operation, you would need to reconnect)

```
SELECT /*HIGH 1 CPUs*/ count(*)
FROM ssb.customer;
```



```
SELECT /*HIGH 1 CPUs*/ count(*)
FROM ssb.customer;
```

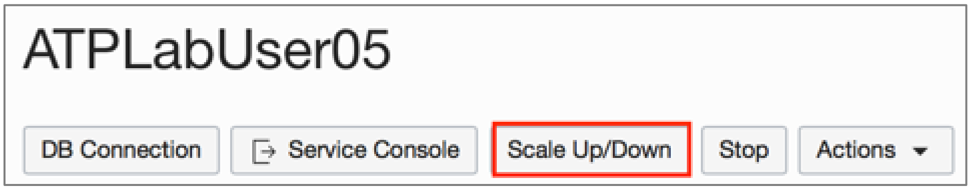
All Rows Fetched: 1 in 205.174 seconds

	COUNT(*)
1	30000000

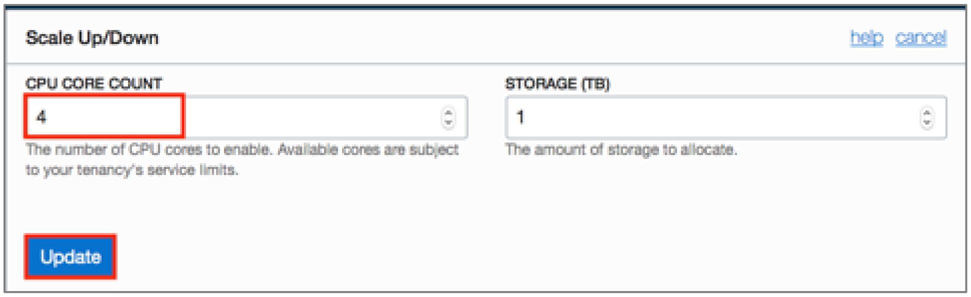
- From the above screenshot, note that the execution time of the sample query using **1 CPUs** is about **200 secs** (YMMV).

STEP 3: Scale Up CPUs and Rerun the Sample Query

- Let's now scale up the **CPUs** to **4.** Click **Scale Up/Down** once again.



- In the **Scale Up/Down** pop up, modify the **CPU Core Count** to **4** and click **Update**.


Scale Up/Down

help cancel

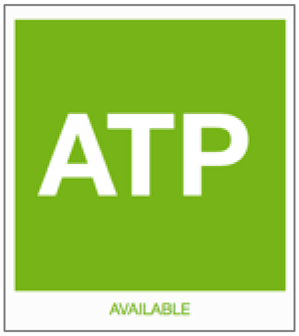
CPU CORE COUNT: 4

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

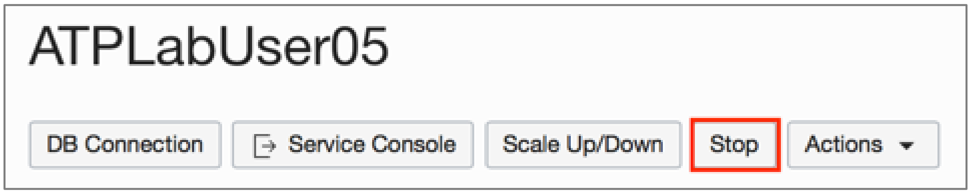
STORAGE (TB): 1

The amount of storage to allocate.

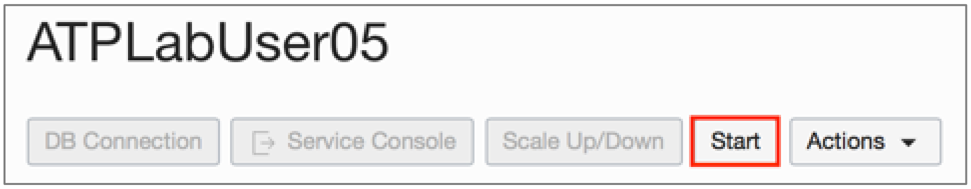
- Again, wait for the scaling task to complete.



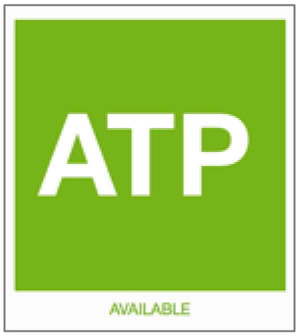
- Once again, to get a good performance comparison, you need to stop and restart the service. Click **Stop** to stop the service.



- Click **Start** to start your service.

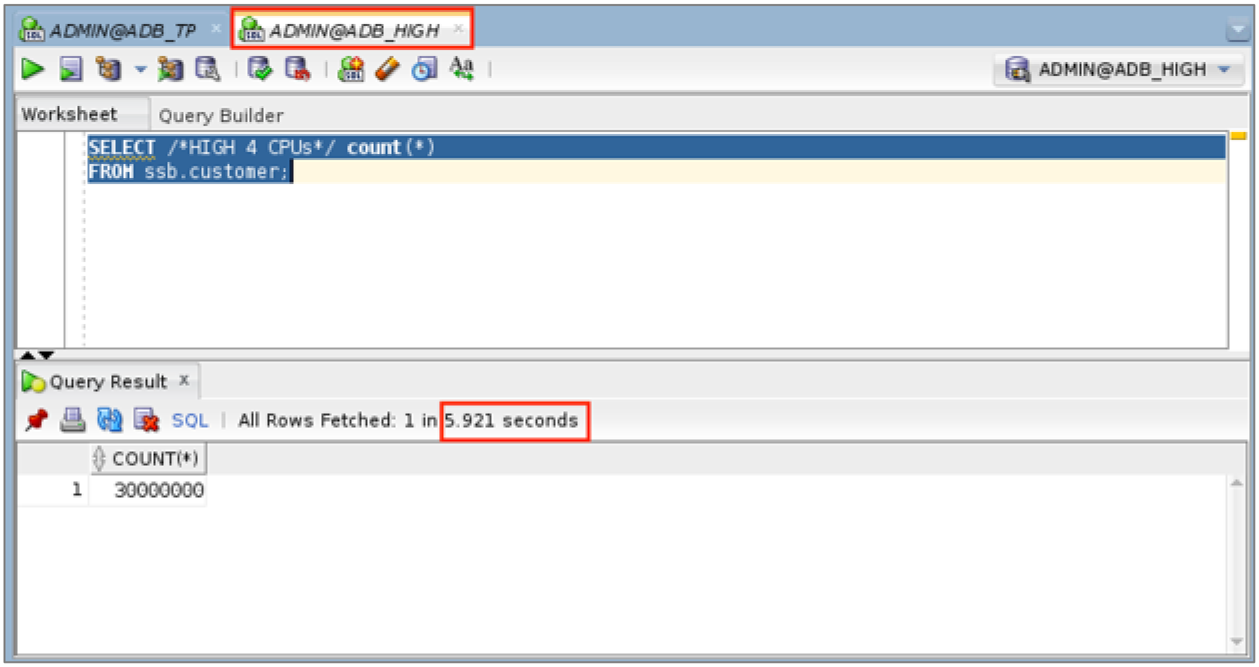


- Wait for the service to start.



- Re-Connect your SQL Developer connection to **HIGH** service. Copy and paste the following example query and select **Run Script** (or F5). Note the runtime of the query. (if you connected already before the Stop/Start operation, you would need to reconnect)

```
SELECT /*HIGH 4 CPUs*/ count(*)
FROM ssb.customer;
```



```
SELECT /*HIGH 4 CPUs*/ count(*)
FROM ssb.customer;
```

All Rows Fetched: 1 in 5.921 seconds

	COUNT(*)
1	30000000

- From the above screenshot, notice that the execution time of the sample query using **4 CPUs** is about **6 secs** (your numbers will vary). The workload runs much faster with **4 CPUs** when compared previously with **1** CPUs.

- Since Oracle ADB customers get billed only for the CPUs allocated, you are in control of allocating the right number of CPUs to achieve the best cost-performance balance for your workload.

- You have now experienced that scaling has an immediate impact on performance. You have successfully completed this lab.

Lab 6-3: Performance Monitoring

The **Overview** and **Activity** tabs in the **Service Console** provide information about the performance of the ADB service and the workload.

In this lab you will utilize the built-in performance monitoring screens of the autonomous database.

Objectives

- Explore performance monitoring capability of ADB using the service console.

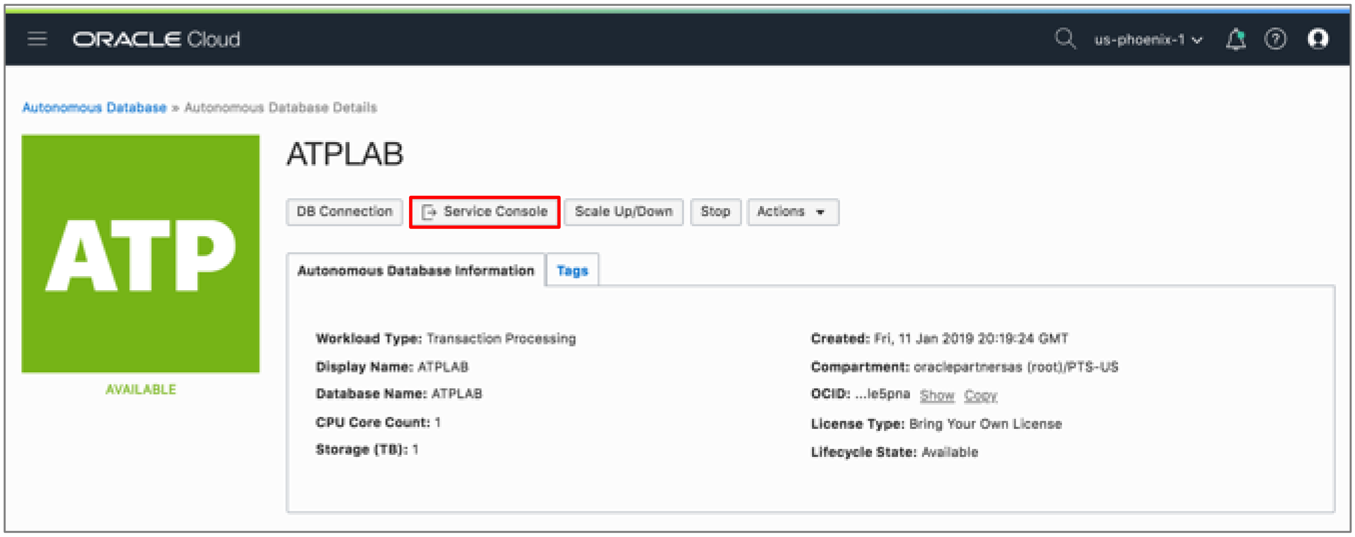
Prerequisite Steps

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** instance.
- Ensure that you have completed the **Scalability and Performance** lab.

Lab Steps

STEP 1: Monitoring Performance Using ADB Service Console

- Sign in to your **ADB Service Console** and browse to **Autonomous Database Details** and click on **Service Console**.



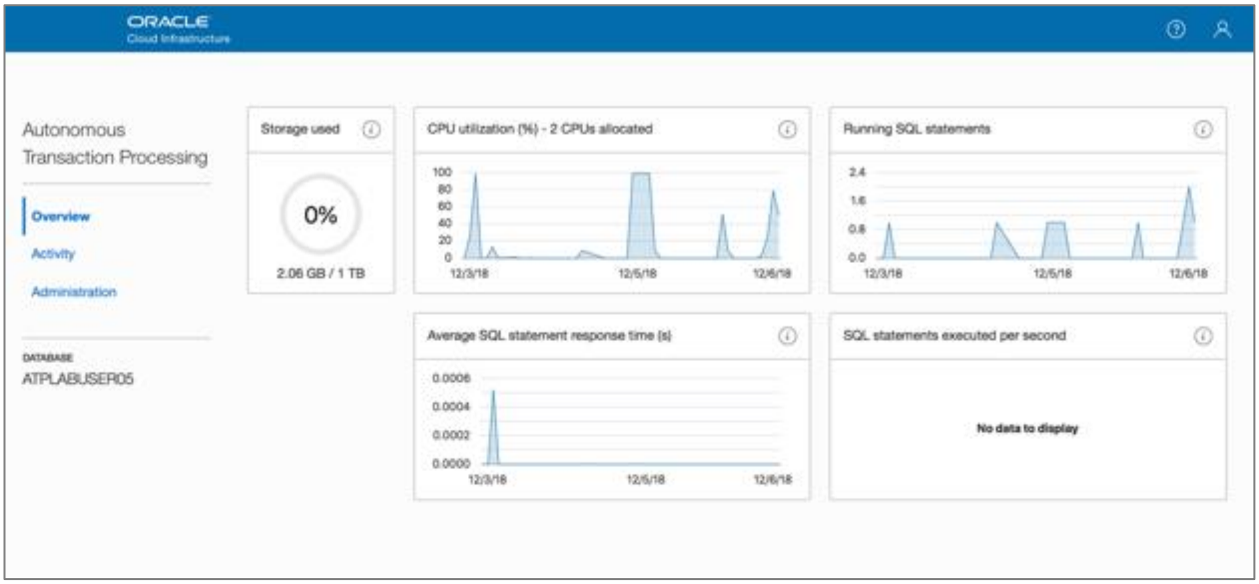
ATPLAB

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

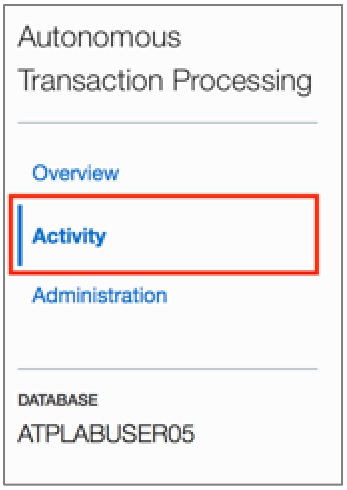
Created: Fri, 11 Jan 2019 20:19:24 GMT
Compartment: oraclepartnersas (root)/PTS-US
OCID: ...le5pna
License Type: Bring Your Own License
Lifecycle State: Available

- You will be presented the **Overview** page. The **Overview** page shows real-time and historical information about the utilization of the service. 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 :** This chart shows the average response time of SQL statements historically.
  - **SQL statements executed per second :** This chart shows the SQL statements executed per second.



- Select **Activity** from the left-menu.

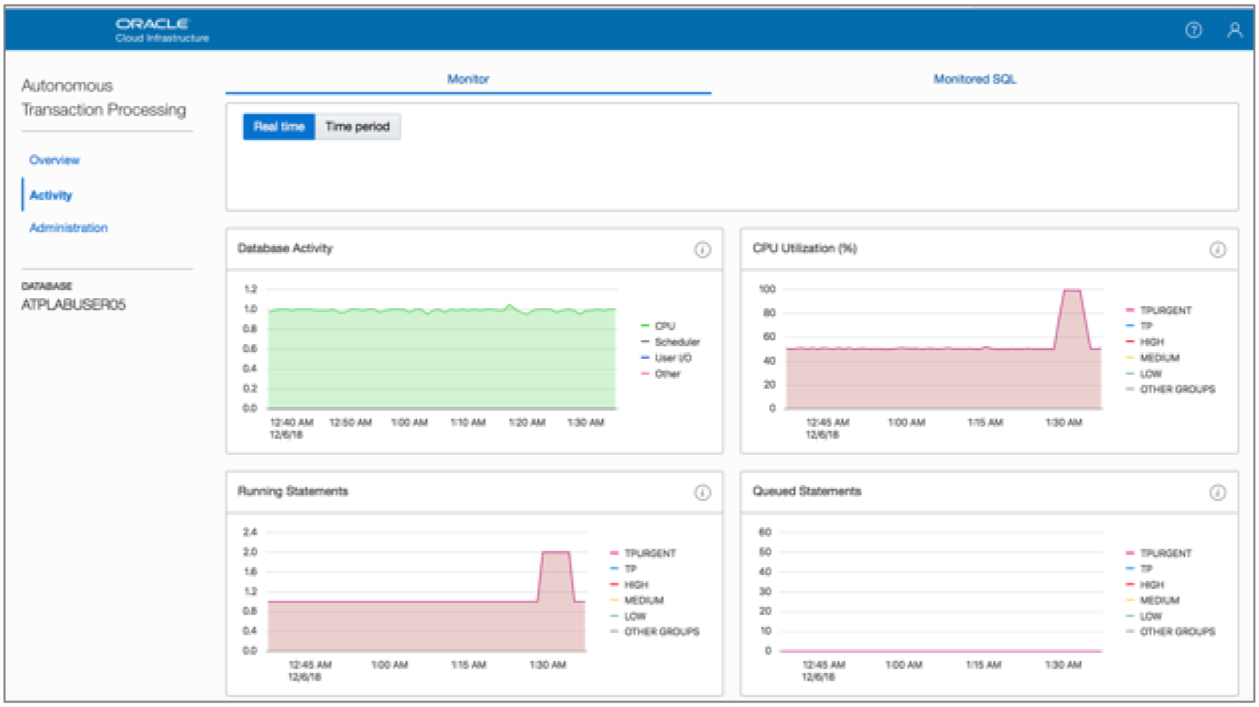


- The **Activity** page has two main tabs - **Monitor** and **Monitored SQL.**
- The **Monitor** tab (also the landing tab for Activity) shows real-time and historical information about the utilization of the service. The components on this page are:
  - **Database Activity:** This chart shows the average number of sessions in the database using CPU or waiting on a wait event.
  - **CPU Utilization:** This chart shows the CPU utilization of each consumer group.

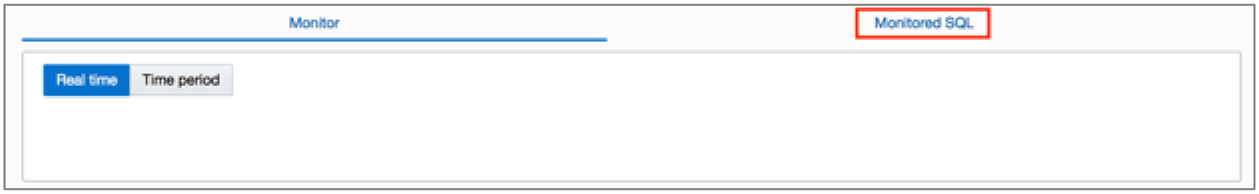
   - **Running Statements:** This chart shows the average number of running SQL statements in each consumer group.

   - **Queued Statements:** This chart shows the average number of queued SQL statements in each consumer group.

- Explore the **Monitor** tab. Notice that as you hover over each chart you will see the different metrics for the different services. Running the cursor over any of the graphs will provide more detailed information. The default is to report **Real Time** activity, but specific time period activity can be examined by selecting the **Time Period** button.

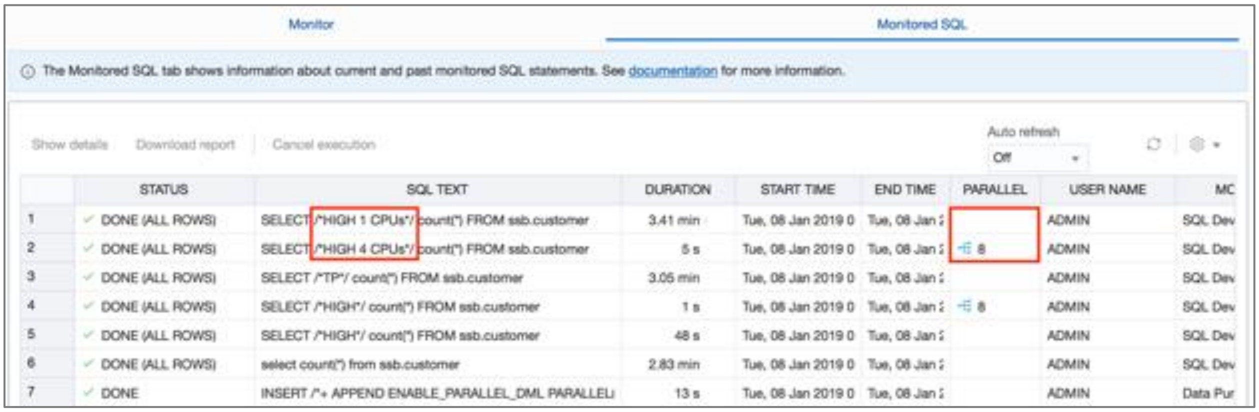


- To analyze specific SQL statements, click on the **Monitored SQL** tab.



- This tab displays the SQLs (completed or running) in chronological order. Note the following SQLs that you recently ran using the **HIGH** service were monitored and show up on this screen.

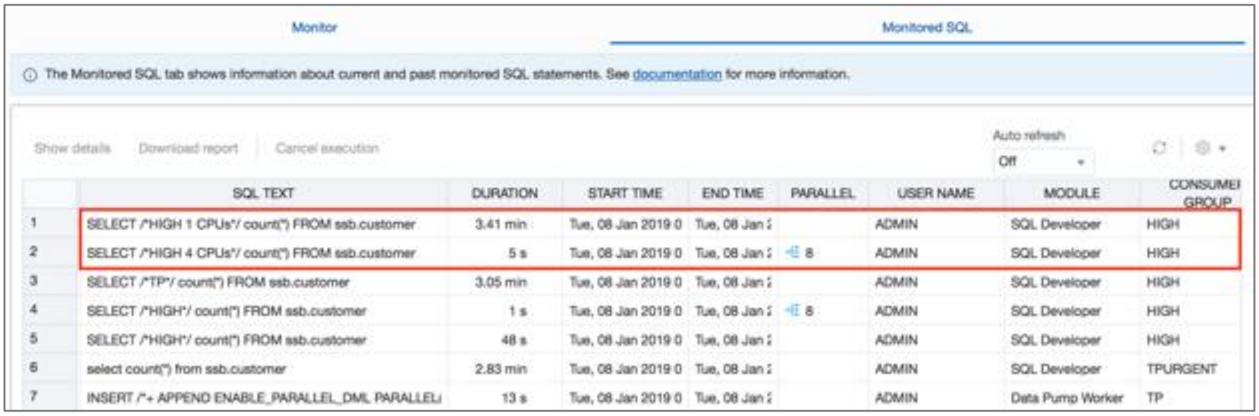
Note: If you want to see additional SQL statements on the Monitored SQL tab, run in SQL Developer Star Schema Benchmark Queries from here.



The Monitored SQL tab shows information about current and past monitored SQL statements. See documentation for more information.

	STATUS	SQL TEXT	DURATION	START TIME	END TIME	PARALLEL	USER NAME	MC
1	✓ DONE (ALL ROWS)	SELECT /*HIGH 1 CPUs*/ count(*) FROM ssb.customer	3.41 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Dev
2	✓ DONE (ALL ROWS)	SELECT /*HIGH 4 CPUs*/ count(*) FROM ssb.customer	5 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	SQL Dev
3	✓ DONE (ALL ROWS)	SELECT /*TP*/ count(*) FROM ssb.customer	3.05 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Dev
4	✓ DONE (ALL ROWS)	SELECT /*HIGH*/ count(*) FROM ssb.customer	1 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	SQL Dev
5	✓ DONE (ALL ROWS)	SELECT /*HIGH*/ count(*) FROM ssb.customer	48 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Dev
6	✓ DONE (ALL ROWS)	select count(*) from ssb.customer	2.83 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Dev
7	✓ DONE	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL(	13 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	Data Pur

- Upon scrolling the table to the right, you can see the **Consumer Group** and the **Parallel** degree that was used by the SQLs. Note that the **HIGH** consumer group with 1 CPU ran the query serially (no parallel).

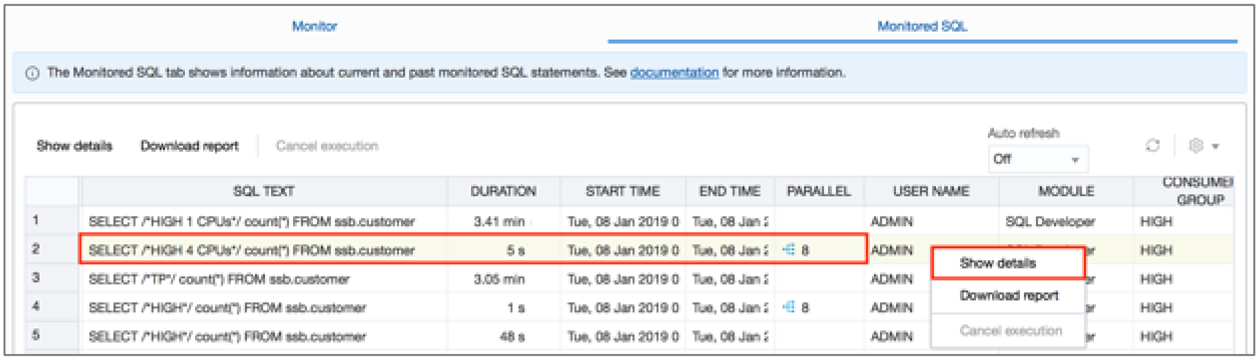


The Monitored SQL tab shows information about current and past monitored SQL statements. See documentation for more information.

	SQL TEXT	DURATION	START TIME	END TIME	PARALLEL	USER NAME	MODULE	CONSUMER GROUP
1	SELECT /*HIGH 1 CPUs*/ count(*) FROM ssb.customer	3.41 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Developer	HIGH
2	SELECT /*HIGH 4 CPUs*/ count(*) FROM ssb.customer	5 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	SQL Developer	HIGH
3	SELECT /*TP*/ count(*) FROM ssb.customer	3.05 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Developer	HIGH
4	SELECT /*HIGH*/ count(*) FROM ssb.customer	1 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	SQL Developer	HIGH
5	SELECT /*HIGH*/ count(*) FROM ssb.customer	48 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Developer	HIGH
6	select count(*) from ssb.customer	2.83 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Developer	TPURGENT
7	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL(	13 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	Data Pump Worker	TP

- Let's further analyze the above SQLs to get additional insight on how they were executed.

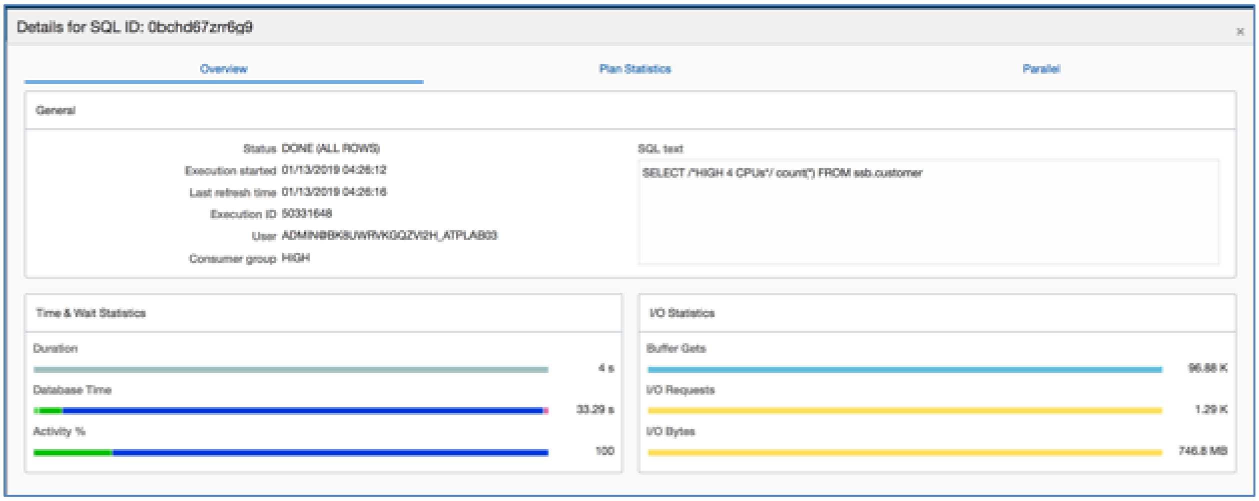
- Right-click the row with the **HIGH** consumer group and parallel of **8** and select **Show Details**.



The Monitored SQL tab shows information about current and past monitored SQL statements. See documentation for more information.

	SQL TEXT	DURATION	START TIME	END TIME	PARALLEL	USER NAME	MODULE	CONSUMER GROUP
1	SELECT /*HIGH 1 CPUs*/ count(*) FROM ssb.customer	3.41 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	SQL Developer	HIGH
2	SELECT /*HIGH 4 CPUs*/ count(*) FROM ssb.customer	5 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	r	HIGH
3	SELECT /*TP*/ count(*) FROM ssb.customer	3.05 min	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	r	HIGH
4	SELECT /*HIGH*/ count(*) FROM ssb.customer	1 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :	8	ADMIN	r	HIGH
5	SELECT /*HIGH*/ count(*) FROM ssb.customer	48 s	Tue, 08 Jan 2019 0	Tue, 08 Jan :		ADMIN	r	HIGH

- The **Show Details -> Overview** tab provides information about the SQL that was executed, user, execution times, and service used (consumer group). Notice that this was the query executed in the **_HIGH** service. Also look at **Duration** in the **Time & Wait Statistics** section on the lower left.



Details for SQL ID: 0bchd67zrr6g9

General

Status DONE (ALL ROWS)
Execution started 01/13/2019 04:26:12
Last refresh time 01/13/2019 04:26:16
Execution ID 50331648
User ADMIN@BK8UWRVKGQZVI2H_ATPLAB03
Consumer group HIGH

SQL text

SELECT /*HIGH 4 CPUs*/ count(*) FROM ssb.customer

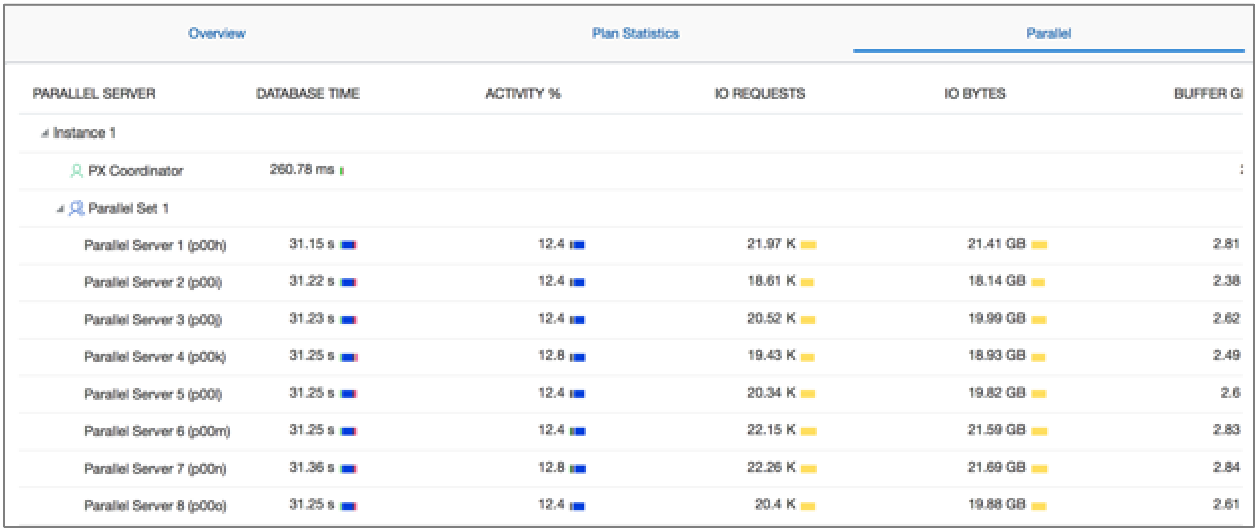
Time & Wait Statistics

Duration 4 s
Database Time 33.29 s
Activity % 100

I/O Statistics

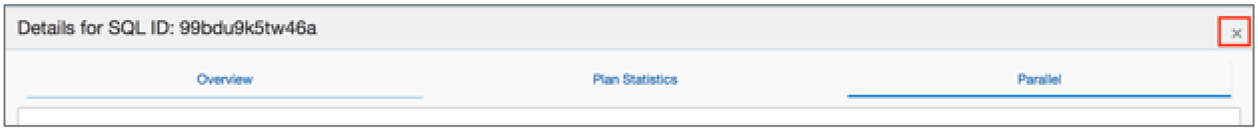
Buffer Gets 96.88 K
I/O Requests 1.29 K
I/O Bytes 746.8 MB

- **HIGH** service uses parallelism and to corroborate that click the **Parallel** tab. You will notice that the query executed in parallel, with **8** parallel threads (indicated by the **Parallel Server** information). That is because the _HIGH service will automatically parallelize transactions depending on the number of CPU's available.



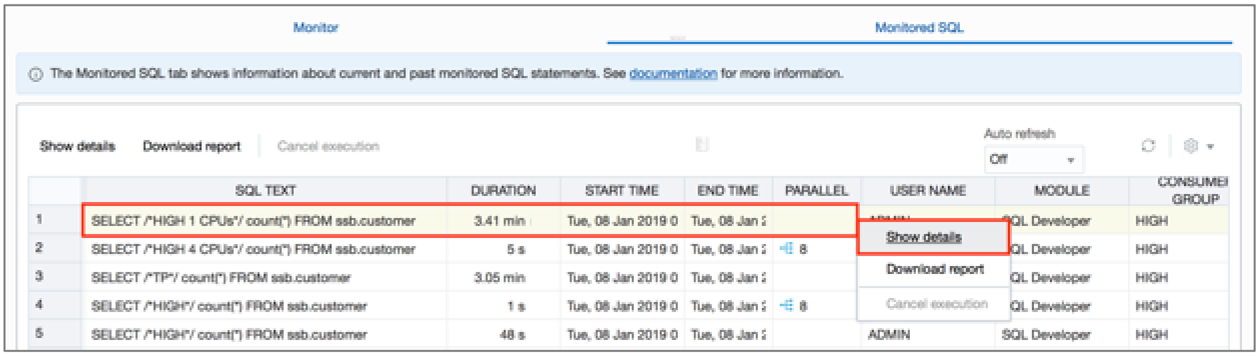
PARALLEL SERVER	DATABASE TIME	ACTIVITY %	IO REQUESTS	IO BYTES	BUFFER G
Instance 1					
PX Coordinator	260.78 ms				:
Parallel Set 1					
Parallel Server 1 (p00h)	31.15 s	12.4	21.97 K	21.41 GB	2.81
Parallel Server 2 (p00i)	31.22 s	12.4	18.61 K	18.14 GB	2.38
Parallel Server 3 (p00j)	31.23 s	12.4	20.52 K	19.99 GB	2.62
Parallel Server 4 (p00k)	31.25 s	12.8	19.43 K	18.93 GB	2.49
Parallel Server 5 (p00l)	31.25 s	12.4	20.34 K	19.82 GB	2.6
Parallel Server 6 (p00m)	31.25 s	12.4	22.15 K	21.59 GB	2.83
Parallel Server 7 (p00n)	31.36 s	12.8	22.26 K	21.69 GB	2.84
Parallel Server 8 (p00o)	31.25 s	12.4	20.4 K	19.88 GB	2.61

- Exit the above screen.



Details for SQL ID: 99bdu9k5tw46a

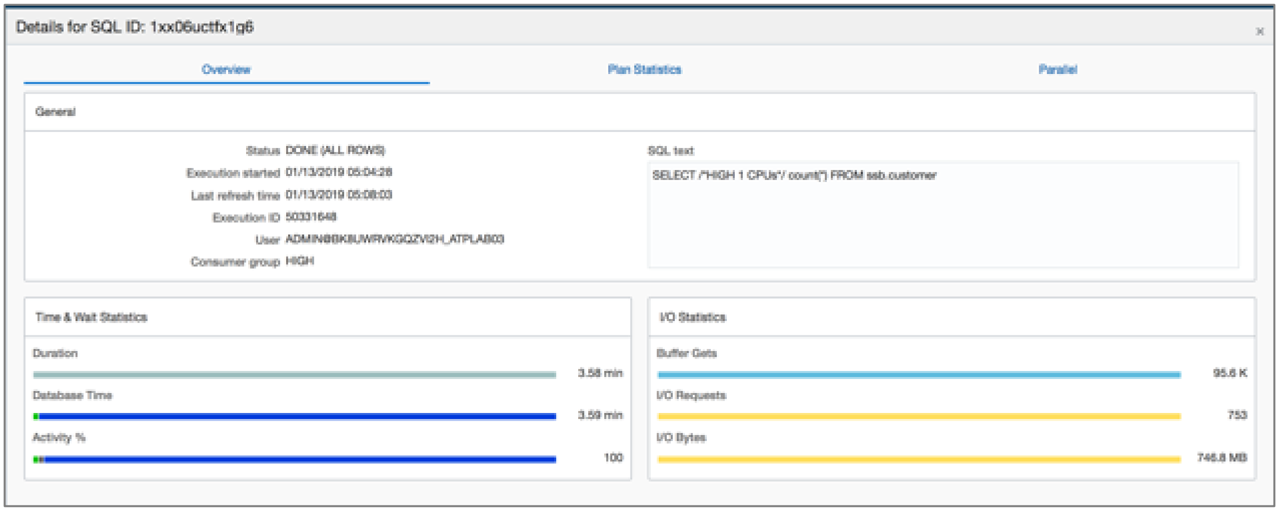
- Next, from the **Activity -> Monitored SQL** page, right-click the row that ran using the **HIGH** consumer group but with **1 CPUs** (i.e. serially) and select **Show Details**.



ⓘ The Monitored SQL tab shows information about current and past monitored SQL statements. See documentation for more information.

	SQL TEXT	DURATION	START TIME	END TIME	PARALLEL	USER NAME	MODULE	CONSUMER GROUP
1	SELECT /*HIGH 1 CPUs*/ count(*) FROM ssb.customer	3.41 min	Tue, 08 Jan 2019 0	Tue, 08 Jan 2			QL Developer	HIGH
2	SELECT /*HIGH 4 CPUs*/ count(*) FROM ssb.customer	5 s	Tue, 08 Jan 2019 0	Tue, 08 Jan 2	8		QL Developer	HIGH
3	SELECT /*TP*/ count(*) FROM ssb.customer	3.05 min	Tue, 08 Jan 2019 0	Tue, 08 Jan 2			QL Developer	HIGH
4	SELECT /*HIGH*/ count(*) FROM ssb.customer	1 s	Tue, 08 Jan 2019 0	Tue, 08 Jan 2	8		QL Developer	HIGH
5	SELECT /*HIGH*/ count(*) FROM ssb.customer	48 s	Tue, 08 Jan 2019 0	Tue, 08 Jan 2		ADMIN	SQL Developer	HIGH

- Which brings up the **Overview** page. Note that this query was run using the **_HIGH** service.



Details for SQL ID: 1xx06uctfx1g6

General

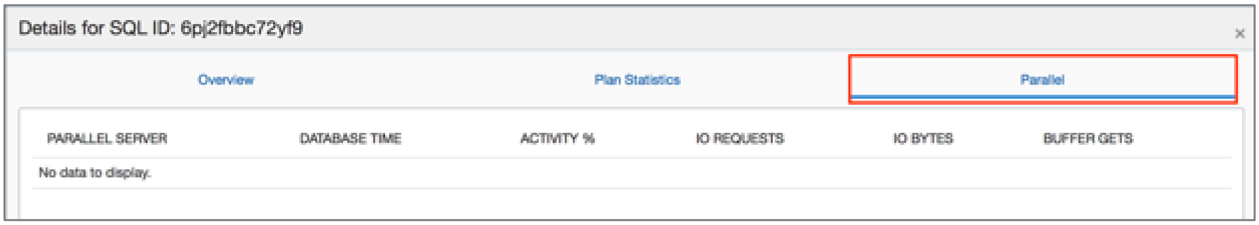
Status DONE (ALL ROWS)
Execution started 01/13/2019 05:04:28
Last refresh time 01/13/2019 05:08:03
Execution ID 50331648
User ADMIN@BK8UWRVKGQZVI2H_ATPLAB03
Consumer group HIGH

SQL text
SELECT /*HIGH 1 CPUs*/ count(*) FROM ssb.customer

Time & Wait Statistics
Duration 3.58 min
Database Time 3.59 min
Activity % 100

I/O Statistics
Buffer Gets 95.6 K
I/O Requests 753
I/O Bytes 746.8 MB

- Click on the **Parallel** and observe that there is no data to display, confirming there was no parallelization for this query (query ran serially as there were only **1 CPUs** allotted).



Details for SQL ID: 6pj2fbbc72yf9

PARALLEL SERVER	DATABASE TIME	ACTIVITY %	IO REQUESTS	IO BYTES	BUFFER GETS
No data to display.					

- You have successfully completed this lab.

7. Optional Labs

Lab 7-1: Exploring Autonomous Database Using SQL Developer

In this lab you will explore the **DBA** view in **Oracle SQL Developer** and browse the ADB configuration and settings.

Please note that most of the database settings and parameters cannot be modified in a fully-managed Oracle Autonomous Database (ATP and ADW) environment and that is the whole point of the ADB service. Nevertheless, this lab will help the DBAs observe that the underlying engine for Oracle ADB is the same Oracle Database they are usually accustomed to managing on-premise or in the Oracle Database Cloud Service.

Objectives

- Explore the DBA view of SQL Developer when connected to the autonomous database.

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** instance.
- **Oracle SQL Developer** installed, or access to a lab VM with Oracle SQL Developer.

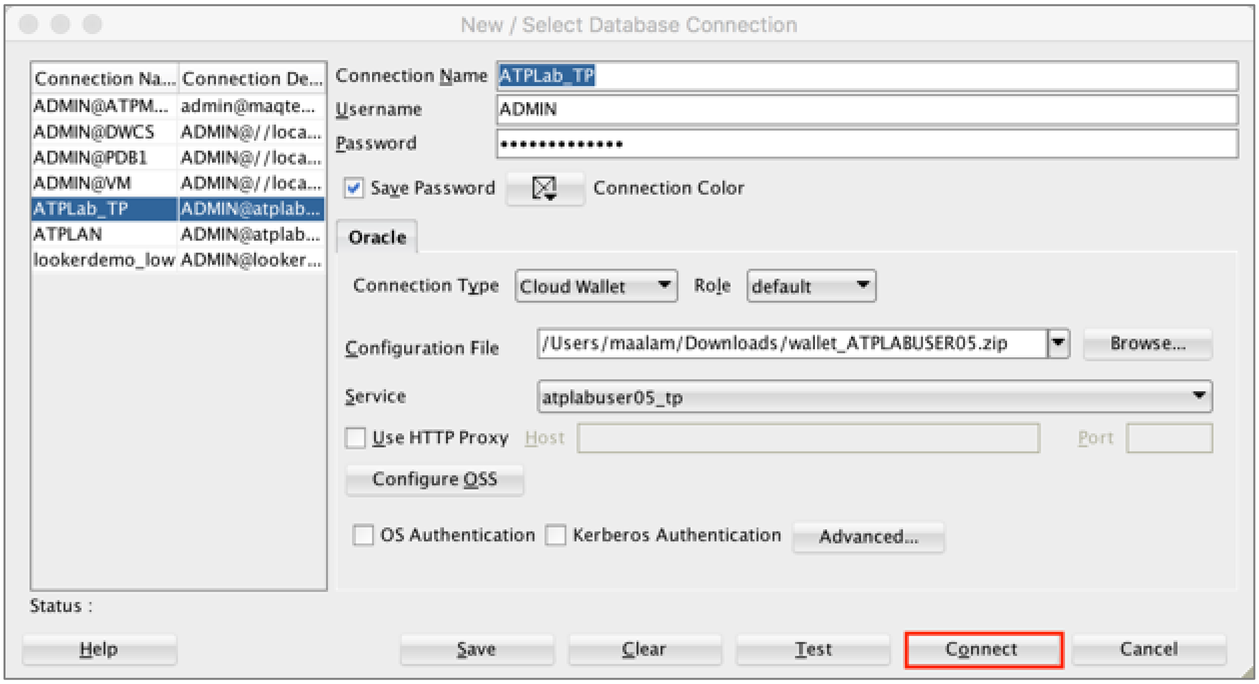
Lab Steps

STEP 1: Exploring ADB Using DBA View of SQL Developer

1) Create the DBA Connection

- Start **SQL Developer** and connect any of the pre-defined services of the autonomous database using **ADMIN** user.

> **Note:** Ensure that you use **ADMIN** user as regular users do not have the privileges to view any database configuration.



New / Select Database Connection

Connection Na...	Connection De...
ADMIN@ATPM...	admin@maqte...
ADMIN@DWCS	ADMIN@//loca...
ADMIN@PDB1	ADMIN@//loca...
ADMIN@VM	ADMIN@//loca...
ATPLab_TP	ADMIN@atplab...
ATPLAN	ADMIN@atplab...
lookerdemo_low	ADMIN@looker...

Connection Name: ATPLab_TP

Username: ADMIN

Password: ••••••••••••••

☑ Save Password

Connection Color

Oracle

Connection Type: Cloud Wallet

Role: default

Configuration File: /Users/maalam/Downloads/wallet_ATPLABUSER05.zip

Service: atplabuser05_tp

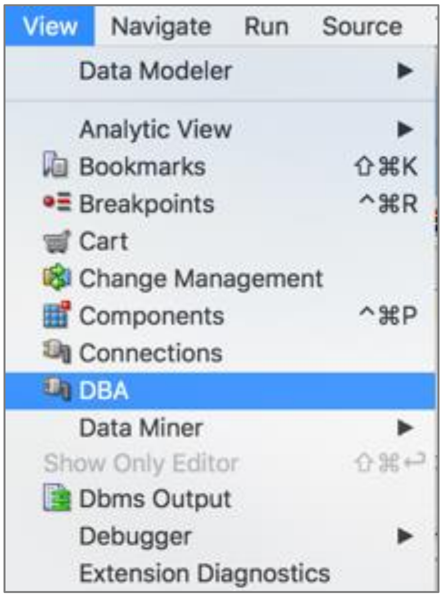
□ Use HTTP Proxy Host Port

□ OS Authentication □ Kerberos Authentication

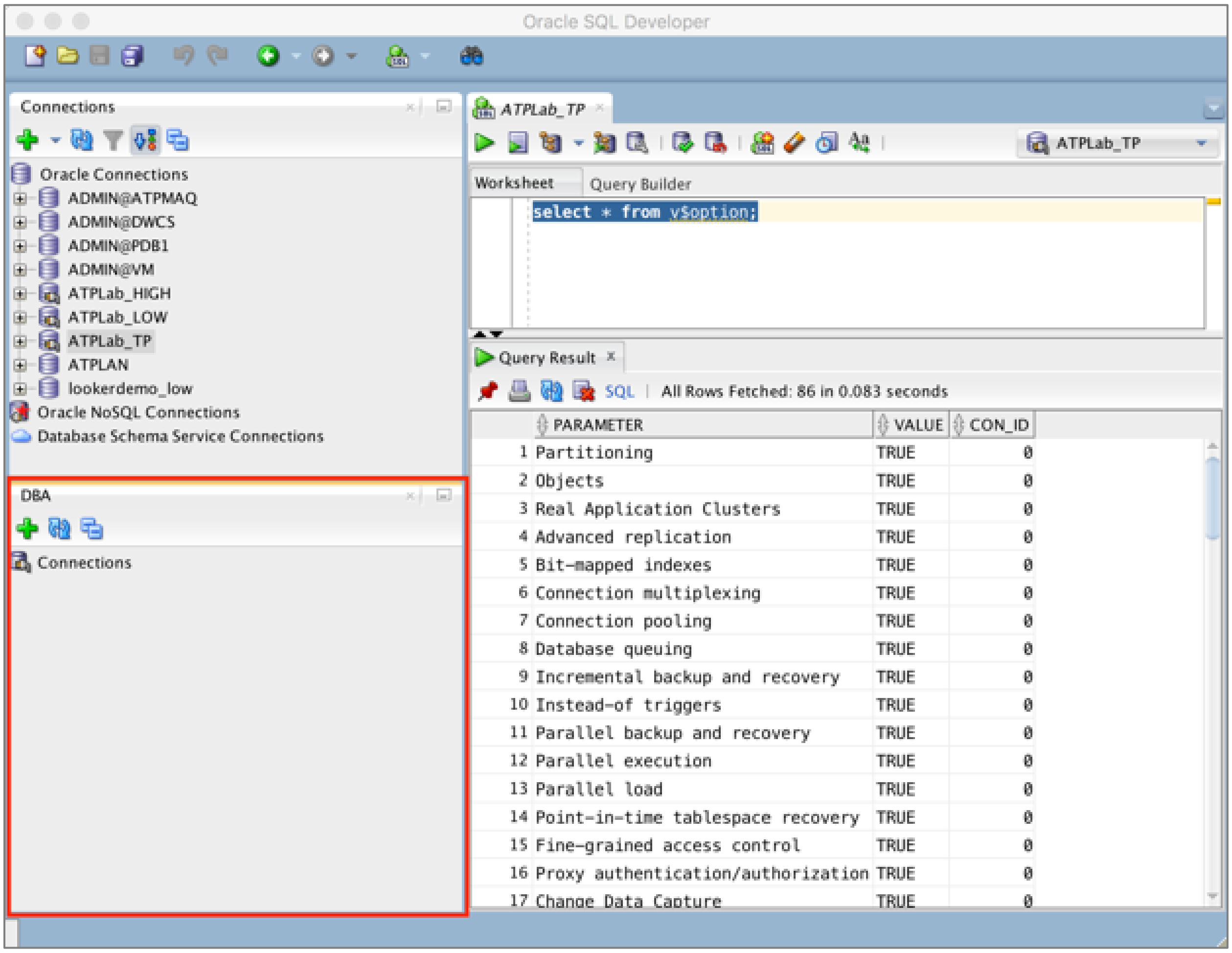
Status :

- Add the **DBA** view in SQL Developer by clicking **View -> DBA**.

> **Note:** If you are not able to see the **DBA** view on the left after doing the above, or unable to locate the **DBA** option in the **View** menu, go to **Window** and select **Reset Window to Factory Setting** and try again.



- The **DBA** view appears on the bottom left.

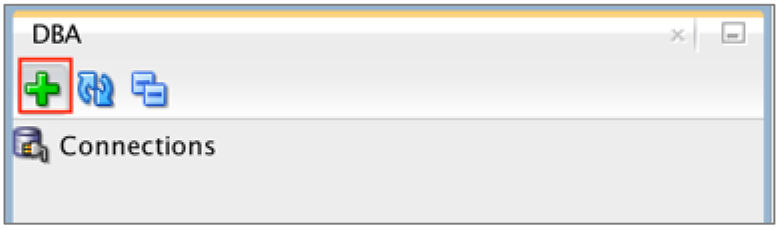


```
select * from v$option;
```

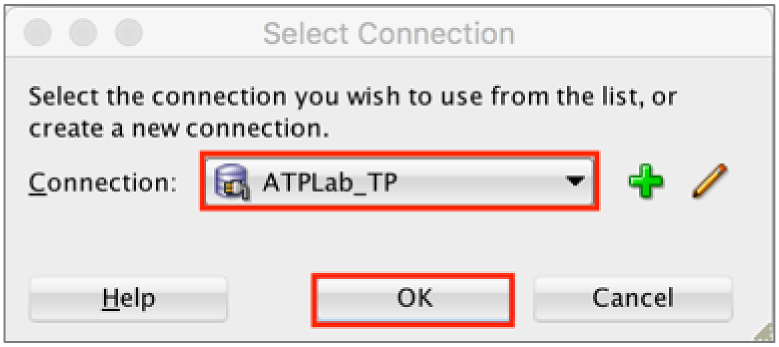
All Rows Fetched: 86 in 0.083 seconds

	PARAMETER	VALUE	CON_ID
1	Partitioning	TRUE	0
2	Objects	TRUE	0
3	Real Application Clusters	TRUE	0
4	Advanced replication	TRUE	0
5	Bit-mapped indexes	TRUE	0
6	Connection multiplexing	TRUE	0
7	Connection pooling	TRUE	0
8	Database queuing	TRUE	0
9	Incremental backup and recovery	TRUE	0
10	Instead-of triggers	TRUE	0
11	Parallel backup and recovery	TRUE	0
12	Parallel execution	TRUE	0
13	Parallel load	TRUE	0
14	Point-in-time tablespace recovery	TRUE	0
15	Fine-grained access control	TRUE	0
16	Proxy authentication/authorization	TRUE	0
17	Change Data Capture	TRUE	0

- Click the green plus sign in the DBA view to add a new connection.



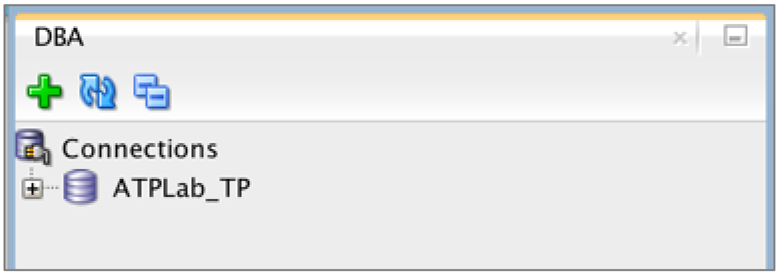
- Select the connection you created earlier that connects to the ADB service using the **ADMIN** user.



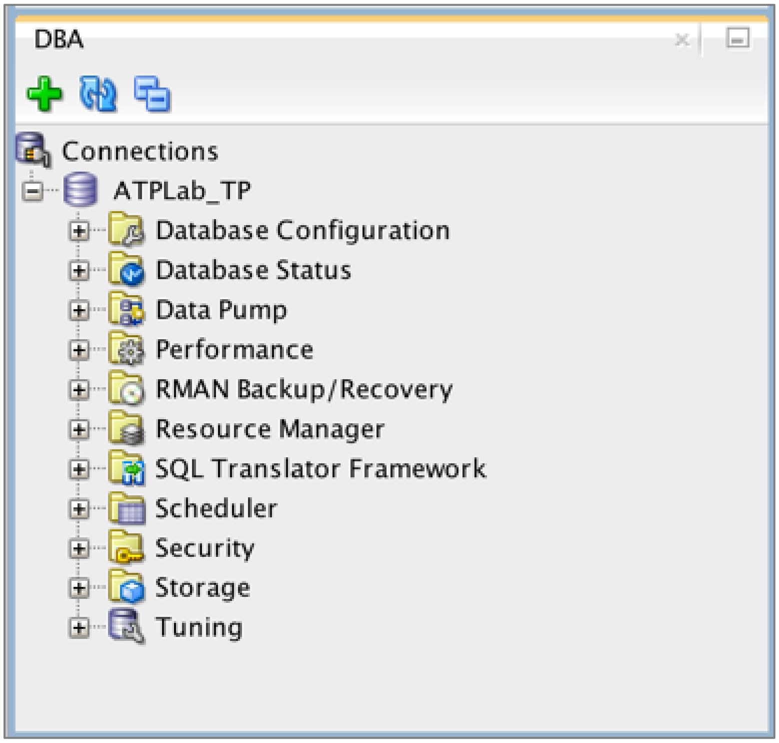
Select the connection you wish to use from the list, or create a new connection.

Connection: ATPLab_TP

- You should see the connection added to the **DBA** view.

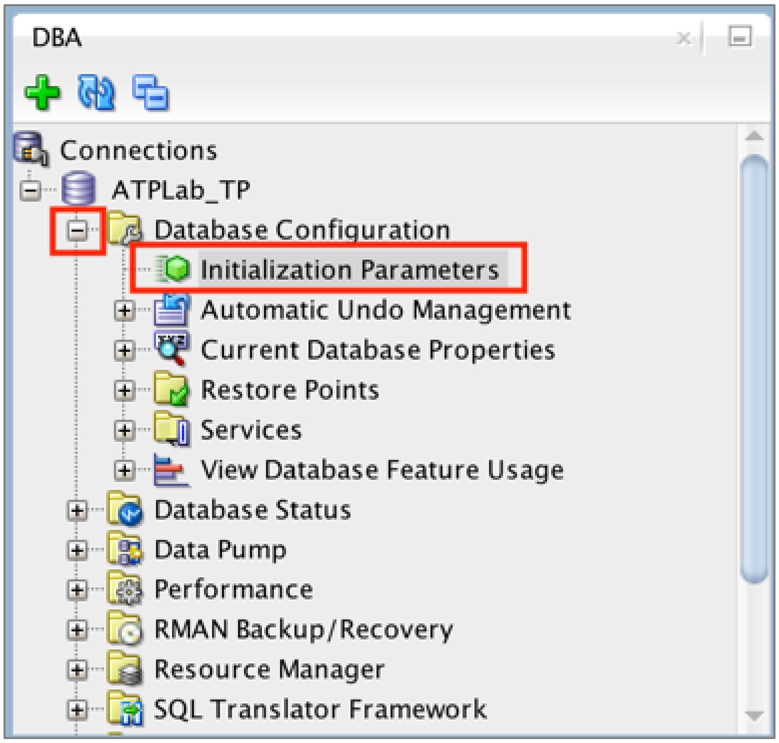


- Expand the connection by clicking on the plus sign. This will open up all the **DBA** views of the database.



2) View the Initialization Parameters

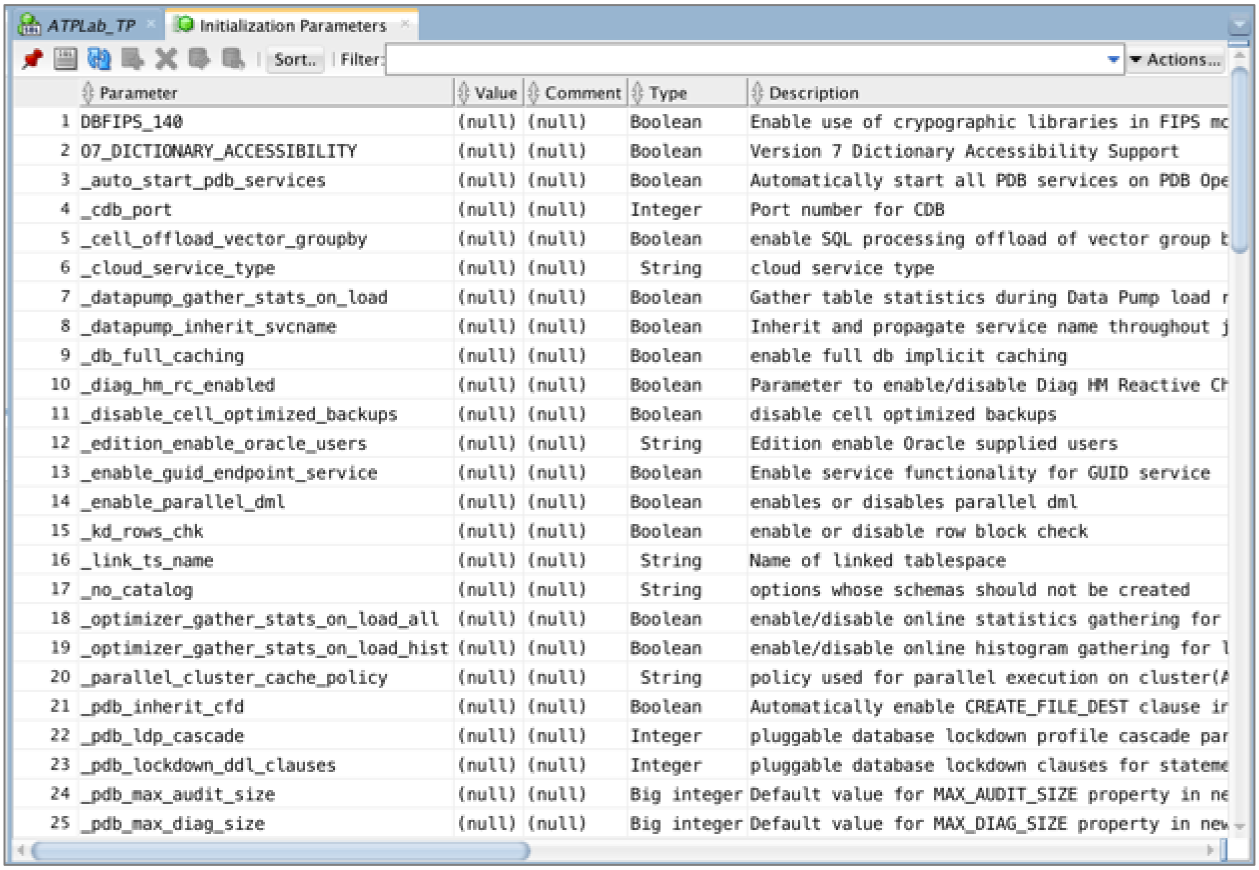
- Expand **Database Configuration** and click on **Initialization Parameters** to view the database initialization parameters and their settings.



DBA

Connections
- ATPLab_TP
  - Database Configuration
    - Initialization Parameters
    - Automatic Undo Management
    - Current Database Properties
    - Restore Points
    - Services
    - View Database Feature Usage
  - Database Status
  - Data Pump
  - Performance
  - RMAN Backup/Recovery
  - Resource Manager
  - SQL Translator Framework

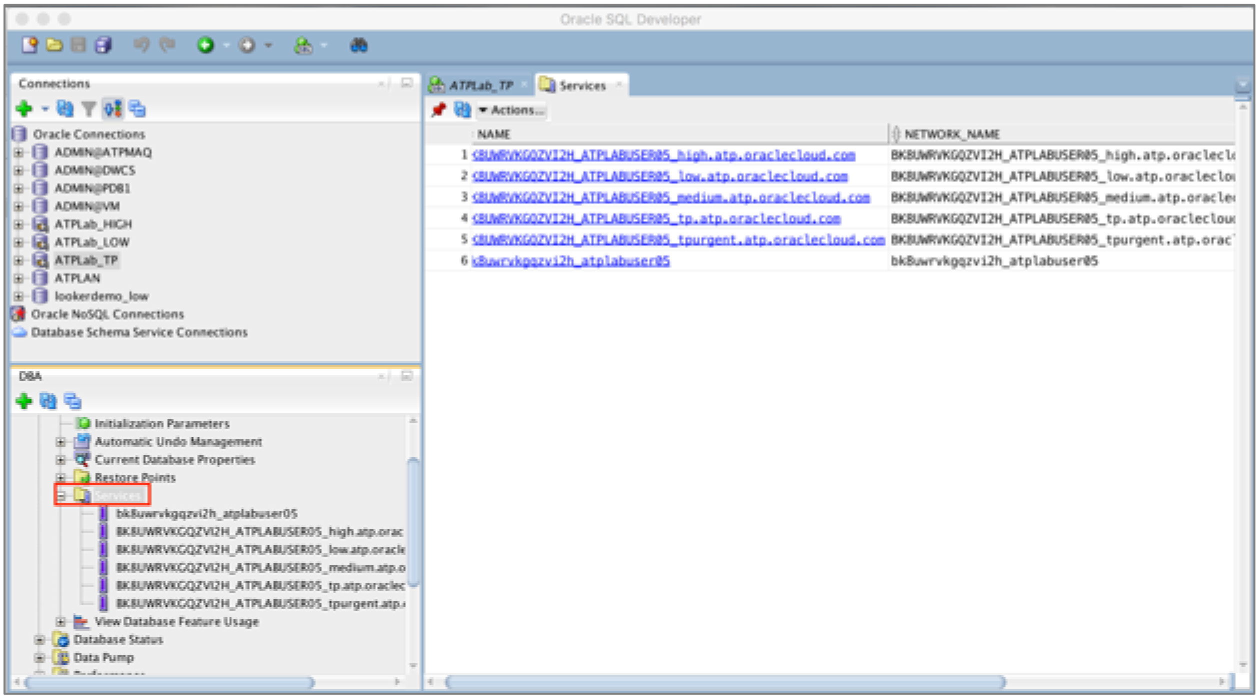
- ADB configures the database initialization parameters based on the compute and storage capacity provisioned. You do not need to set any initialization parameters to start using your ADB Service but you may modify a few parameters if there is a need (the list of modifiable parameters is provided in the documentation).



	Parameter	Value	Comment	Type	Description
1	DBFIPS_140	(null)	(null)	Boolean	Enable use of crypographic libraries in FIPS mo
2	O7_DICTIONARY_ACCESSIBILITY	(null)	(null)	Boolean	Version 7 Dictionary Accessibility Support
3	_auto_start_pdb_services	(null)	(null)	Boolean	Automatically start all PDB services on PDB Ope
4	_cdb_port	(null)	(null)	Integer	Port number for CDB
5	_cell_offload_vector_groupby	(null)	(null)	Boolean	enable SQL processing offload of vector group b
6	_cloud_service_type	(null)	(null)	String	cloud service type
7	_datapump_gather_stats_on_load	(null)	(null)	Boolean	Gather table statistics during Data Pump load r
8	_datapump_inherit_svcname	(null)	(null)	Boolean	Inherit and propagate service name throughout j
9	_db_full_caching	(null)	(null)	Boolean	enable full db implicit caching
10	_diag_hm_rc_enabled	(null)	(null)	Boolean	Parameter to enable/disable Diag HM Reactive Ch
11	_disable_cell_optimized_backups	(null)	(null)	Boolean	disable cell optimized backups
12	_edition_enable_oracle_users	(null)	(null)	String	Edition enable Oracle supplied users
13	_enable_guid_endpoint_service	(null)	(null)	Boolean	Enable service functionality for GUID service
14	_enable_parallel_dml	(null)	(null)	Boolean	enables or disables parallel dml
15	_kd_rows_chk	(null)	(null)	Boolean	enable or disable row block check
16	_link_ts_name	(null)	(null)	String	Name of linked tablespace
17	_no_catalog	(null)	(null)	String	options whose schemas should not be created
18	_optimizer_gather_stats_on_load_all	(null)	(null)	Boolean	enable/disable online statistics gathering for
19	_optimizer_gather_stats_on_load_hist	(null)	(null)	Boolean	enable/disable online histogram gathering for l
20	_parallel_cluster_cache_policy	(null)	(null)	String	policy used for parallel execution on cluster(A
21	_pdb_inherit_cfd	(null)	(null)	Boolean	Automatically enable CREATE_FILE_DEST clause in
22	_pdb_ldp_cascade	(null)	(null)	Integer	pluggable database lockdown profile cascade par
23	_pdb_lockdown_ddl_clauses	(null)	(null)	Integer	pluggable database lockdown clauses for stateme
24	_pdb_max_audit_size	(null)	(null)	Big integer	Default value for MAX_AUDIT_SIZE property in ne
25	_pdb_max_diag_size	(null)	(null)	Big integer	Default value for MAX_DIAG_SIZE property in new

3) View ADB Services

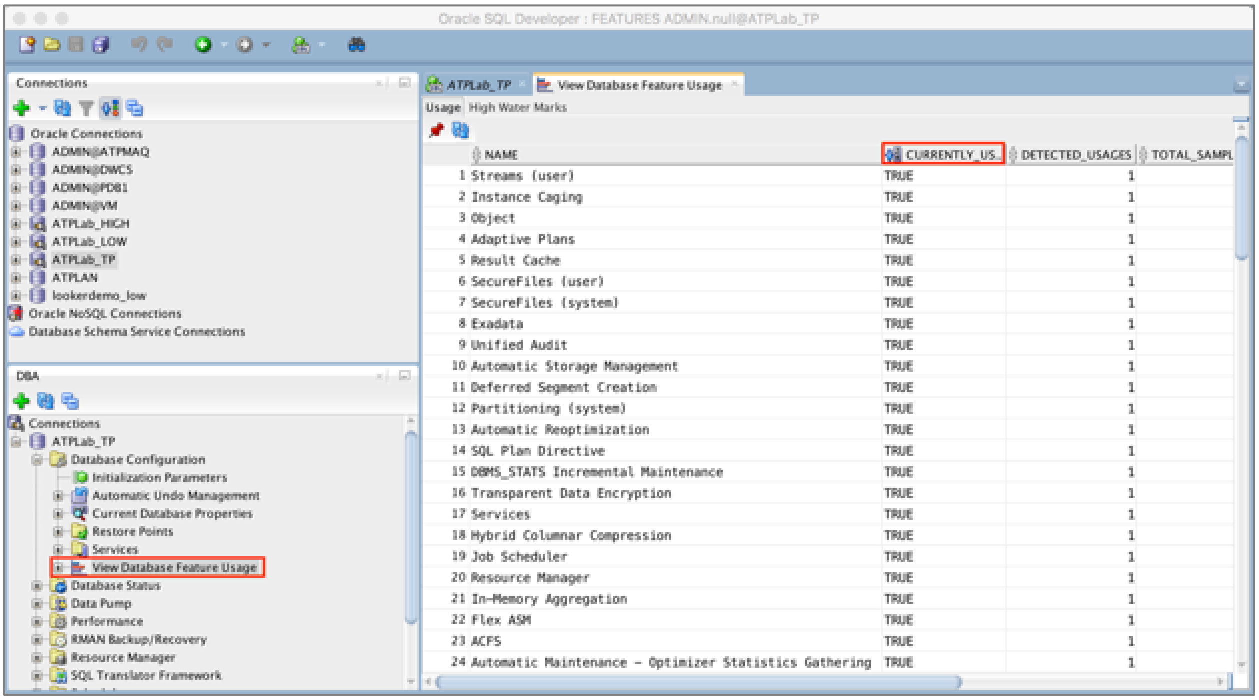
- Click on **Services.** Observe the five services (in case of ATP) configured and discussed in the earlier labs (the sixth one is the default service of the database).


Oracle SQL Developer
Connections
Oracle Connections
ADMIN@ATPMAQ
ADMIN@DWCS
ADMIN@PDB1
ADMIN@VM
ATPLab_HIGH
ATPLab_LOW
ATPLab_TP
ATPLAN
lookerdemo_low
Oracle NoSQL Connections
Database Schema Service Connections
DBA
Initialization Parameters
Automatic Undo Management
Current Database Properties
Restore Points
Services
bk8uwrvkgqzvi2h_atplabuser05
BK8UWRVKGQZVI2H_ATPLABUSER05_high.atp.orac
BK8UWRVKGQZVI2H_ATPLABUSER05_low.atp.oracle
BK8UWRVKGQZVI2H_ATPLABUSER05_medium.atp.o
BK8UWRVKGQZVI2H_ATPLABUSER05_tp.atp.oraclec
BK8UWRVKGQZVI2H_ATPLABUSER05_tpurgent.atp.
View Database Feature Usage
Database Status
Data Pump
ATPLab_TP
Services
Actions...

	NAME	NETWORK_NAME
1	K8UWRVKGQZVI2H_ATPLABUSER05_high.atp.oraclecloud.com	BK8UWRVKGQZVI2H_ATPLABUSER05_high.atp.oraclecl
2	K8UWRVKGQZVI2H_ATPLABUSER05_low.atp.oraclecloud.com	BK8UWRVKGQZVI2H_ATPLABUSER05_low.atp.oraclecl
3	K8UWRVKGQZVI2H_ATPLABUSER05_medium.atp.oraclecloud.com	BK8UWRVKGQZVI2H_ATPLABUSER05_medium.atp.oracle
4	K8UWRVKGQZVI2H_ATPLABUSER05_tp.atp.oraclecloud.com	BK8UWRVKGQZVI2H_ATPLABUSER05_tp.atp.oraclecl
5	K8UWRVKGQZVI2H_ATPLABUSER05_tpurgent.atp.oraclecloud.com	BK8UWRVKGQZVI2H_ATPLABUSER05_tpurgent.atp.orac
6	k8uwrvkgqzvi2h_atplabuser05	bk8uwrvkgqzvi2h_atplabuser05

4) View DB Features In-Use

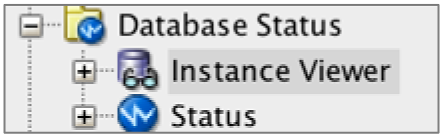
- Click on **Database Features Usage** you can see the features currently in use. As with other views, columns can be sorted, so for example you can sort the **CURRENTLY_USED** column by **TRUE** to see all the features being used.



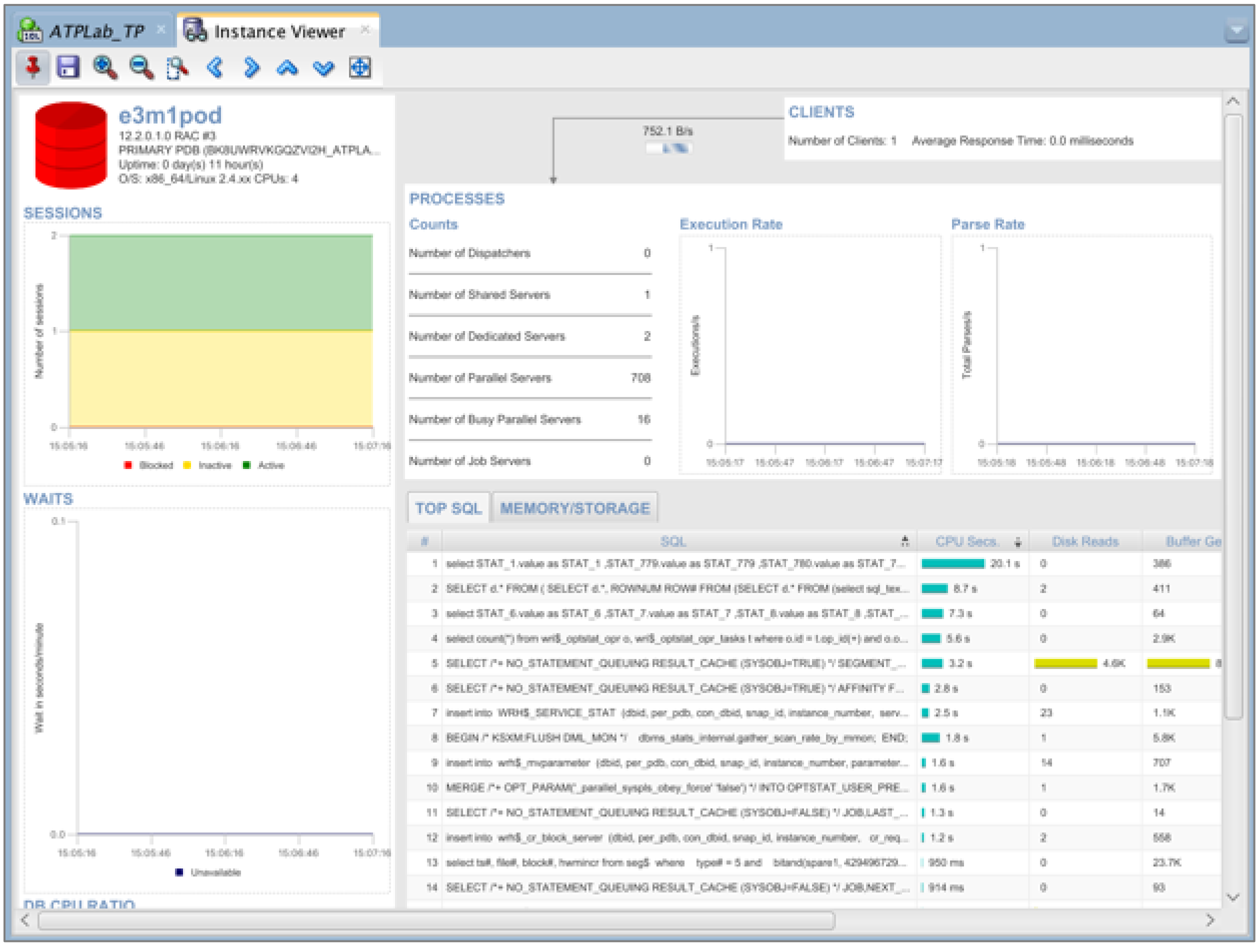
	NAME	CURRENTLY_US...	DETECTED_USAGES	TOTAL_SAMPL
1	Streams (user)	TRUE	1	
2	Instance Caging	TRUE	1	
3	Object	TRUE	1	
4	Adaptive Plans	TRUE	1	
5	Result Cache	TRUE	1	
6	SecureFiles (user)	TRUE	1	
7	SecureFiles (system)	TRUE	1	
8	Exadata	TRUE	1	
9	Unified Audit	TRUE	1	
10	Automatic Storage Management	TRUE	1	
11	Deferred Segment Creation	TRUE	1	
12	Partitioning (system)	TRUE	1	
13	Automatic Reoptimization	TRUE	1	
14	SQL Plan Directive	TRUE	1	
15	DBMS_STATS Incremental Maintenance	TRUE	1	
16	Transparent Data Encryption	TRUE	1	
17	Services	TRUE	1	
18	Hybrid Columnar Compression	TRUE	1	
19	Job Scheduler	TRUE	1	
20	Resource Manager	TRUE	1	
21	In-Memory Aggregation	TRUE	1	
22	Flex ASM	TRUE	1	
23	ACFS	TRUE	1	
24	Automatic Maintenance - Optimizer Statistics Gathering	TRUE	1	

5) View Instance Charts

- Expand the **Database Status** tree and double click on **Instance Viewer**.



- Give it a few seconds to generate the charts. This displays a lot of information about the instance, usage and configuration as well as the TOP SQL running on the instance.



e3m1pod
12.2.0.1.0 RAC #3
PRIMARY PDB (BK8UWRVKGQZVI2H_ATPLA...
Uptime: 0 day(s) 11 hour(s)
O/S: x86_64/Linux 2.4.xx CPUs: 4

CLIENTS

Number of Clients: 1 Average Response Time: 0.0 milliseconds

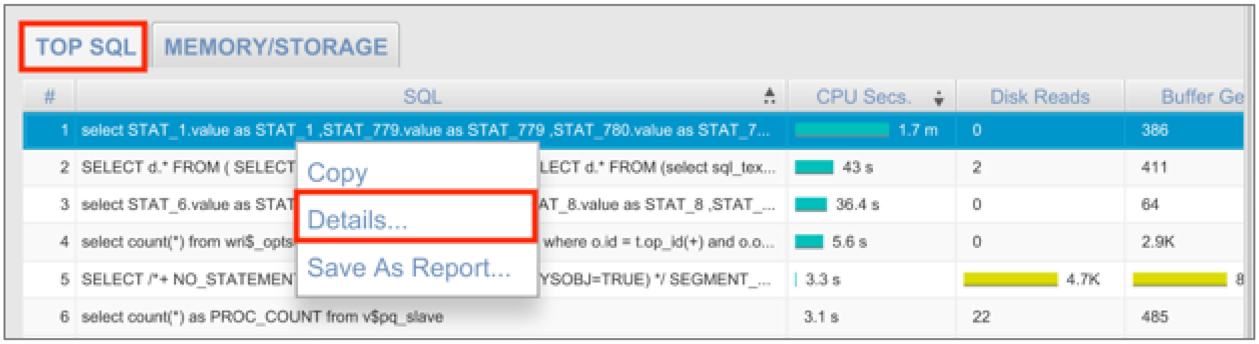
PROCESSES

Counts

Number of Dispatchers	0
Number of Shared Servers	1
Number of Dedicated Servers	2
Number of Parallel Servers	708
Number of Busy Parallel Servers	16
Number of Job Servers	0

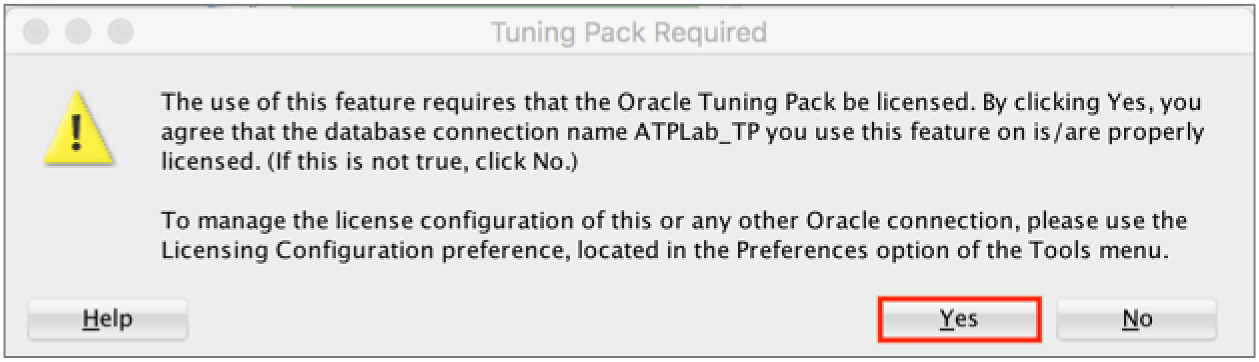
#	SQL	CPU Secs.	Disk Reads	Buffer Ge
1	select STAT_1.value as STAT_1 ,STAT_779.value as STAT_779 ,STAT_780.value as STAT_7...	20.1 s	0	386
2	SELECT d.* FROM (SELECT d.*, ROWNUM ROW# FROM (SELECT d.* FROM (select sql_tex...	8.7 s	2	411
3	select STAT_6.value as STAT_6 ,STAT_7.value as STAT_7 ,STAT_8.value as STAT_8 ,STAT_...	7.3 s	0	64
4	select count(*) from wri$_optstat_opr o, wri$_optstat_opr_tasks t where o.id = t.op_id(+) and o.o...	5.6 s	0	2.9K
5	SELECT /*+ NO_STATEMENT_QUEUING RESULT_CACHE (SYSOBJ=TRUE) */ SEGMENT_...	3.2 s	4.6K	8
6	SELECT /*+ NO_STATEMENT_QUEUING RESULT_CACHE (SYSOBJ=TRUE) */ AFFINITY F...	2.8 s	0	153
7	insert into WRH$_SERVICE_STAT (dbid, per_pdb, con_dbid, snap_id, instance_number, serv...	2.5 s	23	1.1K
8	BEGIN /* KSXM:FLUSH DML_MON */ dbms_stats_internal.gather_scan_rate_by_mmon; END;	1.8 s	1	5.8K
9	insert into wrh$_mvparameter (dbid, per_pdb, con_dbid, snap_id, instance_number, parameter...	1.6 s	14	707
10	MERGE /*+ OPT_PARAM('_parallel_syspls_obey_force' 'false') */ INTO OPTSTAT_USER_PRE...	1.6 s	1	1.7K
11	SELECT /*+ NO_STATEMENT_QUEUING RESULT_CACHE (SYSOBJ=FALSE) */ JOB.LAST_...	1.3 s	0	14
12	insert into wrh$_cr_block_server (dbid, per_pdb, con_dbid, snap_id, instance_number, cr_req...	1.2 s	2	558
13	select ts#, file#, block#, hwmincr from seg$ where type# = 5 and bitand(spare1, 429496729...	950 ms	0	23.7K
14	SELECT /*+ NO_STATEMENT_QUEUING RESULT_CACHE (SYSOBJ=FALSE) */ JOB.NEXT_...	914 ms	0	93

- You can select any SQL from **TOP SQL**, right click and select **Details**.



#	SQL	CPU Secs.	Disk Reads	Buffer Ge
1	select STAT_1.value as STAT_1 ,STAT_779.value as STAT_779 ,STAT_780.value as STAT_7...	1.7 m	0	386
2	SELECT d.* FROM (SELECT LECT d.* FROM (select sql_tex...	43 s	2	411
3	select STAT_6.value as STAT AT_8.value as STAT_8 ,STAT_...	36.4 s	0	64
4	select count(*) from wri$_opts where o.id = t.op_id(+) and o.o...	5.6 s	0	2.9K
5	SELECT /*+ NO_STATEMEN YSOBJ=TRUE) */ SEGMENT_...	3.3 s	4.7K	8
6	select count(*) as PROC_COUNT from v$pq_slave	3.1 s	22	485

- You may be prompted to acknowledge that you have **Oracle Tuning Pack** license. Click **Yes** to continue.

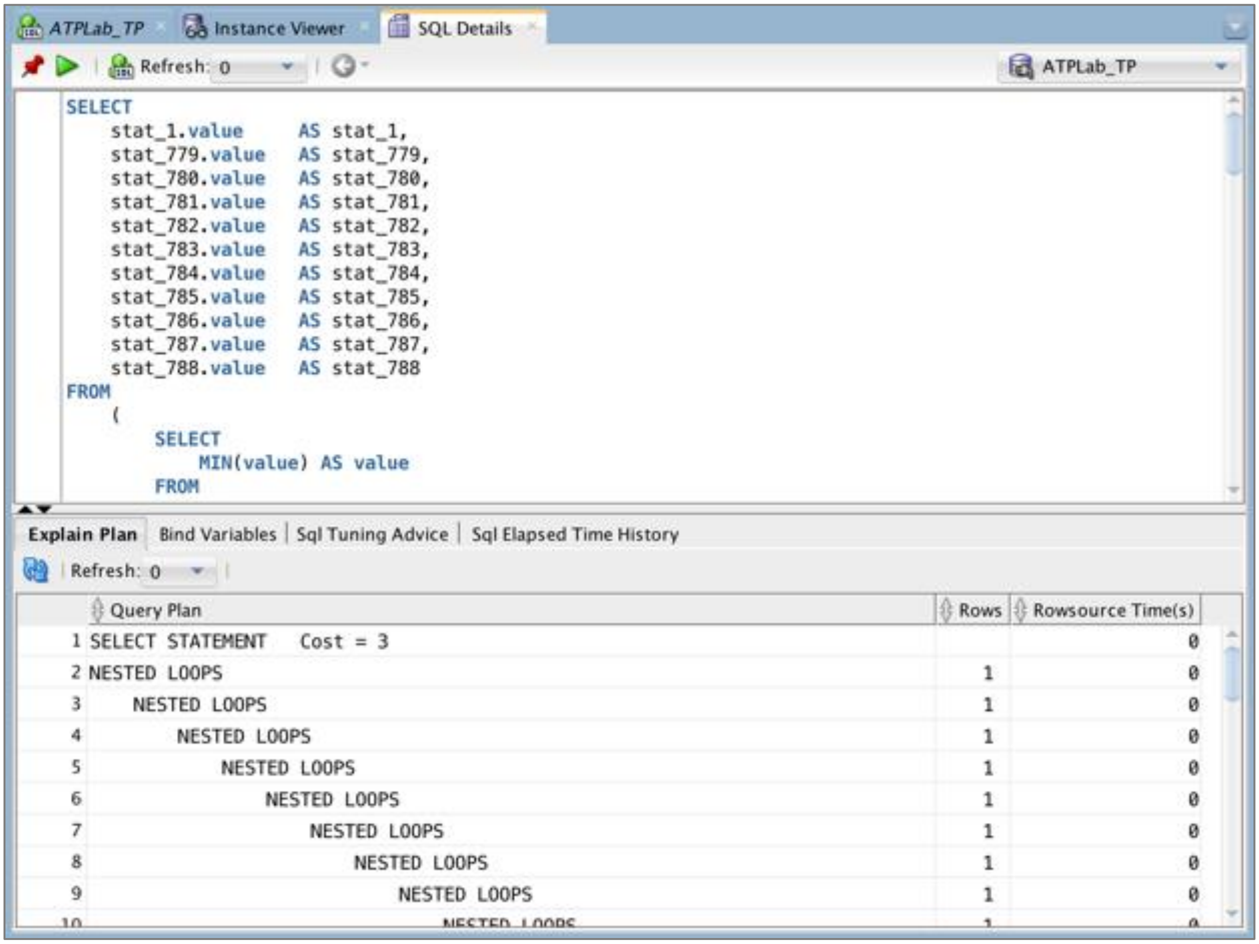


Tuning Pack Required

The use of this feature requires that the Oracle Tuning Pack be licensed. By clicking Yes, you agree that the database connection name ATPLab_TP you use this feature on is/are properly licensed. (If this is not true, click No.)

To manage the license configuration of this or any other Oracle connection, please use the Licensing Configuration preference, located in the Preferences option of the Tools menu.

- The **SQL Details** tab will be opened. You may examine the SQL specifics such as the SQL Text and Explain Plan output further.



```
SELECT
    stat_1.value     AS stat_1,
    stat_779.value   AS stat_779,
    stat_780.value   AS stat_780,
    stat_781.value   AS stat_781,
    stat_782.value   AS stat_782,
    stat_783.value   AS stat_783,
    stat_784.value   AS stat_784,
    stat_785.value   AS stat_785,
    stat_786.value   AS stat_786,
    stat_787.value   AS stat_787,
    stat_788.value   AS stat_788
FROM
    (
        SELECT
            MIN(value) AS value
        FROM
```

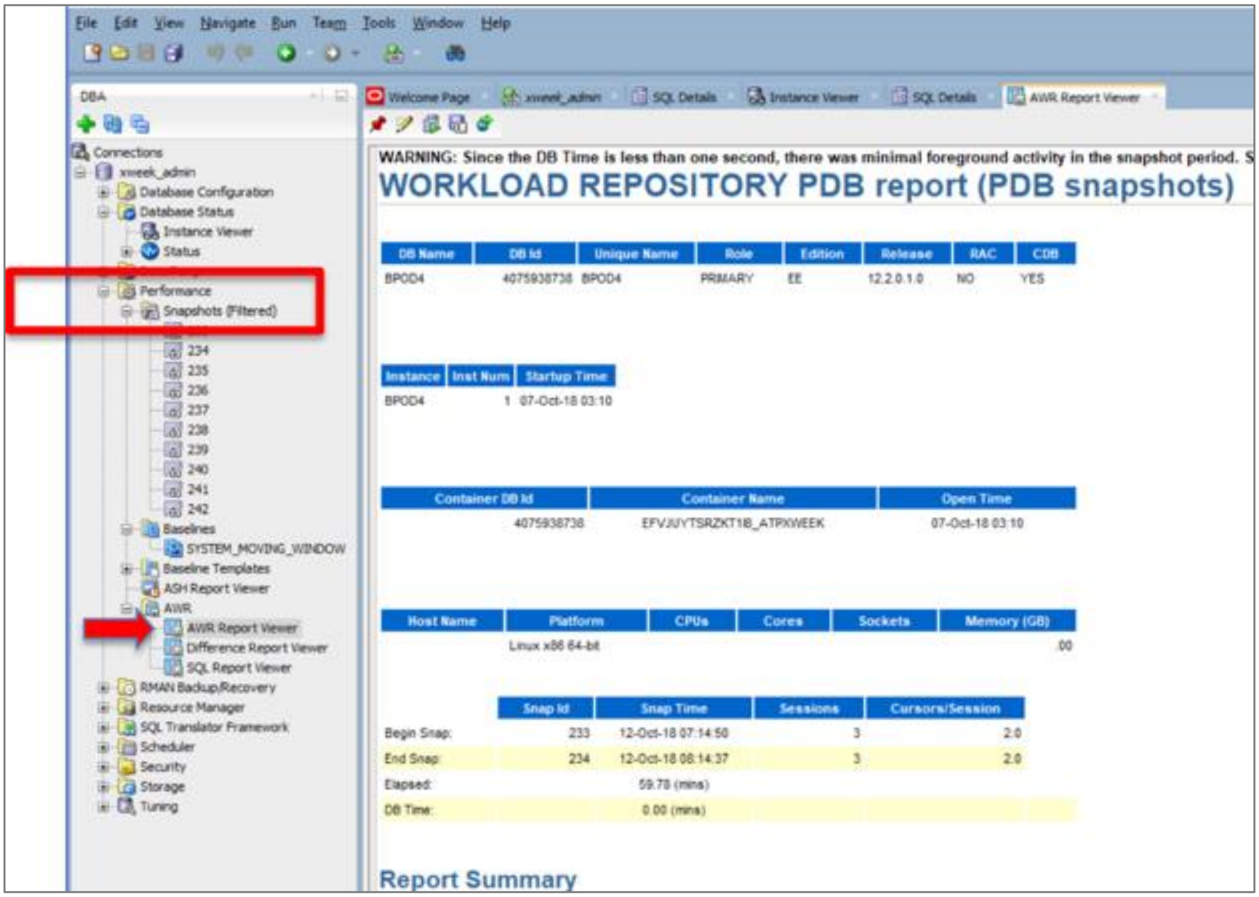
	Query Plan	Rows	Rowsource Time(s)
1	SELECT STATEMENT Cost = 3		0
2	NESTED LOOPS	1	0
3	NESTED LOOPS	1	0
4	NESTED LOOPS	1	0
5	NESTED LOOPS	1	0
6	NESTED LOOPS	1	0
7	NESTED LOOPS	1	0
8	NESTED LOOPS	1	0
9	NESTED LOOPS	1	0
10	NESTED LOOPS	1	0

6) Browse the AWR Reports

- One of the most familiar tools for the DBAs is the AWR Report. Expand the **Performance -> AWR** menu, select **AWR Report Viewer** and select the snapshot range to explore the AWR Report.

> **IMPORTANT:** You may not see any Snapshots as they may not be created (yet). Please proceed to the next step in that case.

Note: Click **Yes** when prompted to acknowledge **Oracle Diagnostic Pack** license.



WARNING: Since the DB Time is less than one second, there was minimal foreground activity in the snapshot period. S

WORKLOAD REPOSITORY PDB report (PDB snapshots)

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
BPOD4	4075938738	BPOD4	PRIMARY	EE	12.2.0.1.0	NO	YES

Instance	Inst Num	Startup Time
BPOD4	1	07-Oct-18 03:10

Container DB Id	Container Name	Open Time
4075938738	EFVJUYTSRZKT1B_ATPXWEEK	07-Oct-18 03:10

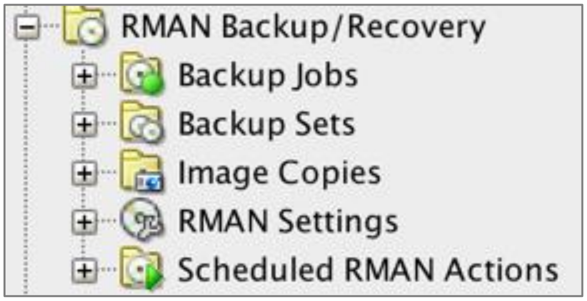
Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
	Linux x86 64-bit				.00

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	233	12-Oct-18 07:14:50	3	2.0
End Snap:	234	12-Oct-18 08:14:37	3	2.0
Elapsed:		59.78 (mins)		
DB Time:		0.00 (mins)		

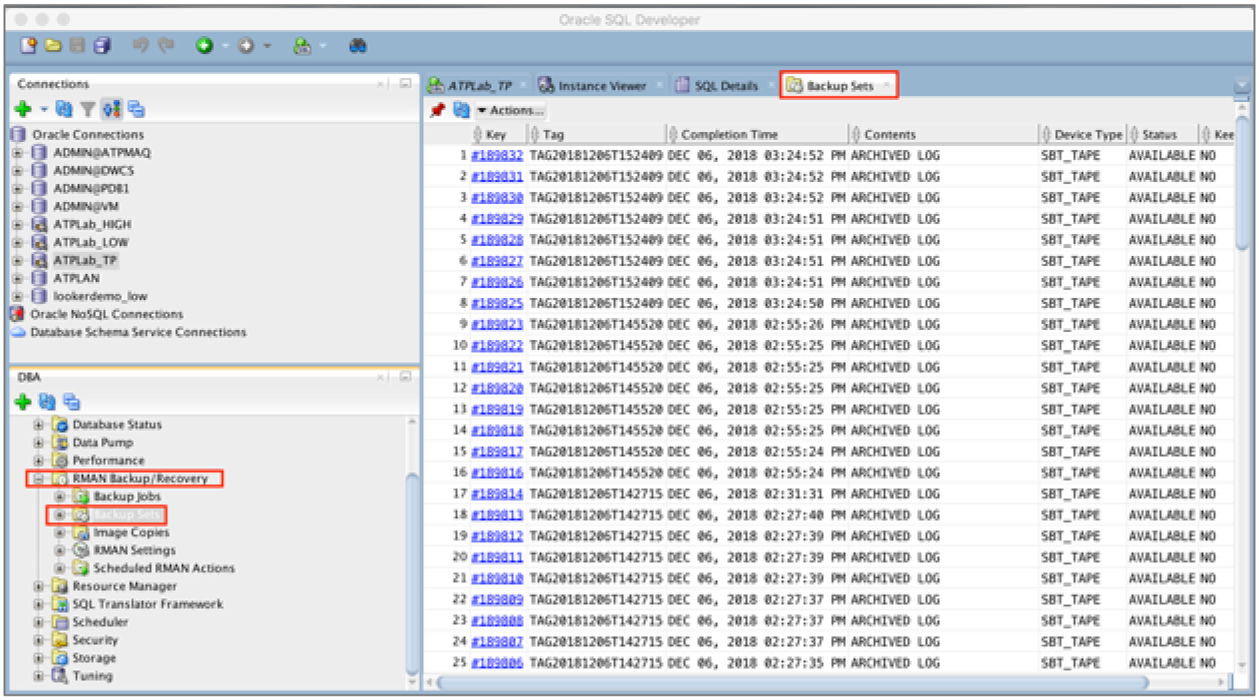
Report Summary

7) Browse Database Backups

- The **RMAN Backup/Recovery** section contains information about scheduled backups, backup sets, RMAN Settings and schedules. Customers and DBAs are often interested about backups on ADB and those can be explored in more detail in this section.



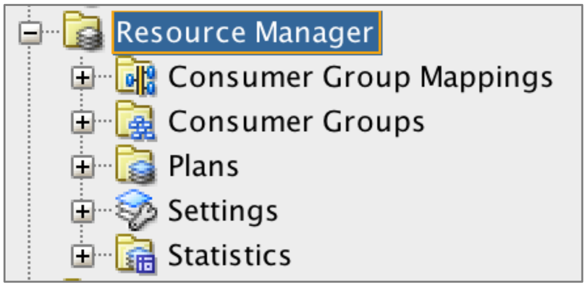
- In the screenshot below the backup sets for this instance are displayed (your instance may not display any backups since it was just created).



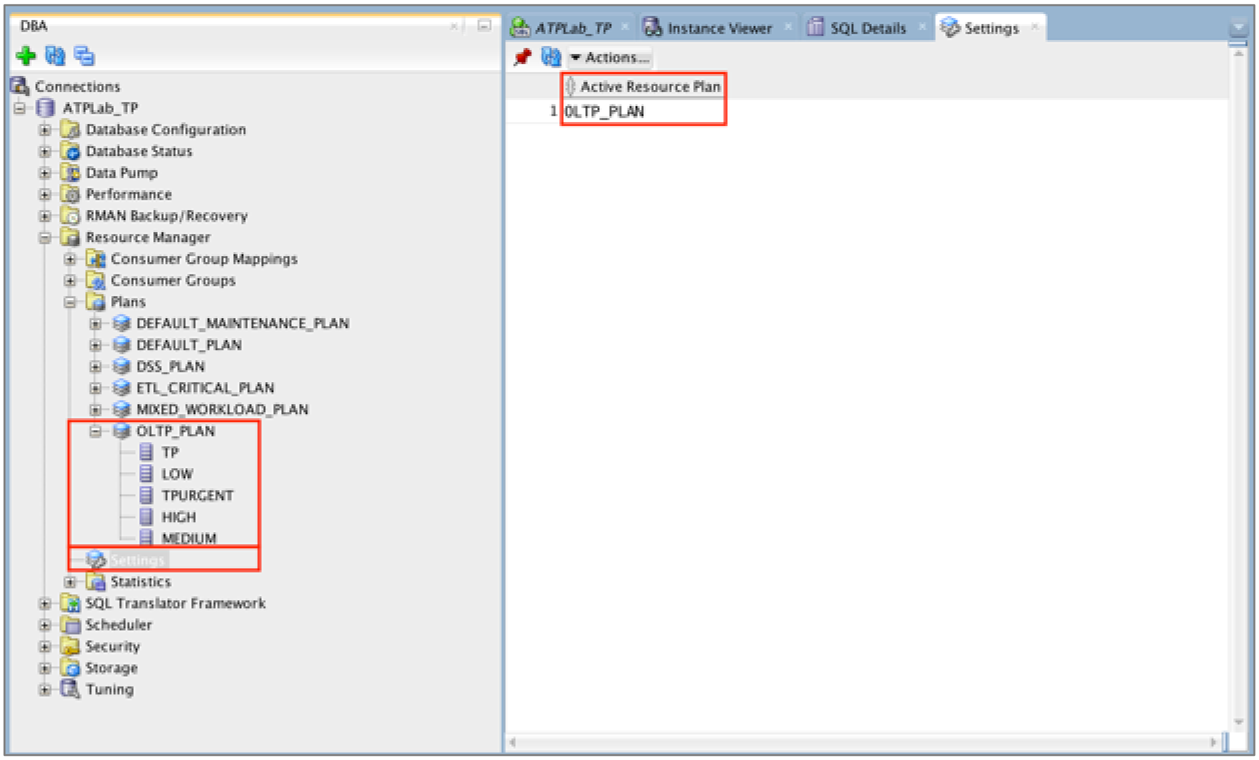
	Key	Tag	Completion Time	Contents	Device Type	Status	Kee
1	#189832	TAG20181206T152409	DEC 06, 2018 03:24:52 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
2	#189831	TAG20181206T152409	DEC 06, 2018 03:24:52 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
3	#189830	TAG20181206T152409	DEC 06, 2018 03:24:52 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
4	#189829	TAG20181206T152409	DEC 06, 2018 03:24:51 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
5	#189828	TAG20181206T152409	DEC 06, 2018 03:24:51 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
6	#189827	TAG20181206T152409	DEC 06, 2018 03:24:51 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
7	#189826	TAG20181206T152409	DEC 06, 2018 03:24:51 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
8	#189825	TAG20181206T152409	DEC 06, 2018 03:24:50 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
9	#189823	TAG20181206T145520	DEC 06, 2018 02:55:26 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
10	#189822	TAG20181206T145520	DEC 06, 2018 02:55:25 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
11	#189821	TAG20181206T145520	DEC 06, 2018 02:55:25 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
12	#189820	TAG20181206T145520	DEC 06, 2018 02:55:25 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
13	#189819	TAG20181206T145520	DEC 06, 2018 02:55:25 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
14	#189818	TAG20181206T145520	DEC 06, 2018 02:55:25 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
15	#189817	TAG20181206T145520	DEC 06, 2018 02:55:24 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
16	#189816	TAG20181206T145520	DEC 06, 2018 02:55:24 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
17	#189814	TAG20181206T142715	DEC 06, 2018 02:31:31 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
18	#189813	TAG20181206T142715	DEC 06, 2018 02:27:40 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
19	#189812	TAG20181206T142715	DEC 06, 2018 02:27:39 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
20	#189811	TAG20181206T142715	DEC 06, 2018 02:27:39 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
21	#189810	TAG20181206T142715	DEC 06, 2018 02:27:39 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
22	#189809	TAG20181206T142715	DEC 06, 2018 02:27:37 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
23	#189808	TAG20181206T142715	DEC 06, 2018 02:27:37 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
24	#189807	TAG20181206T142715	DEC 06, 2018 02:27:37 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO
25	#189806	TAG20181206T142715	DEC 06, 2018 02:27:35 PM	ARCHIVED LOG	SBT_TAPE	AVAILABLE	NO

8) Resource Manager and Plans

- The **Resource Manager** section contains information about different consumer groups and plans that are defined by ADB, 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.

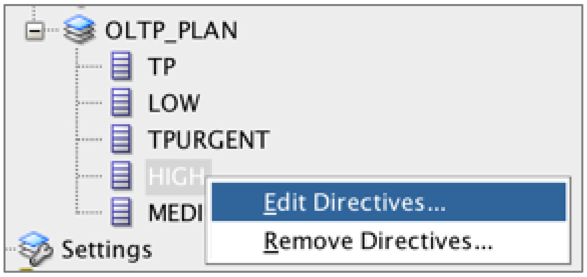


- Under **Plans -> OLTP_PLAN** you will find the current plan being used, which you can verify by double clicking on **Settings**. This will display which plan is being used.

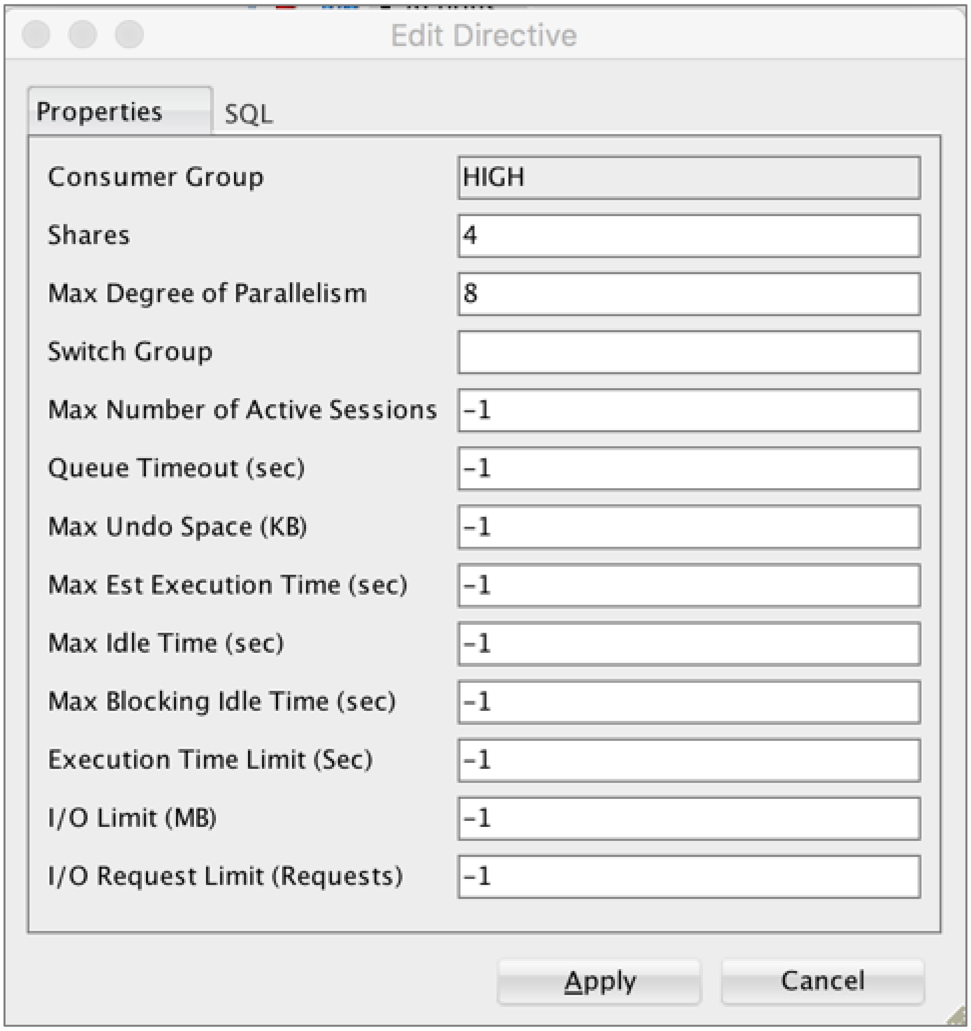


	Active Resource Plan
1	OLTP_PLAN

- Under **OLTP_PLAN** (applicable for ATP) if you select and right click on **HIGH** (or any of the plans), and select **Edit Directives**, the specifics of the plan will be displayed.



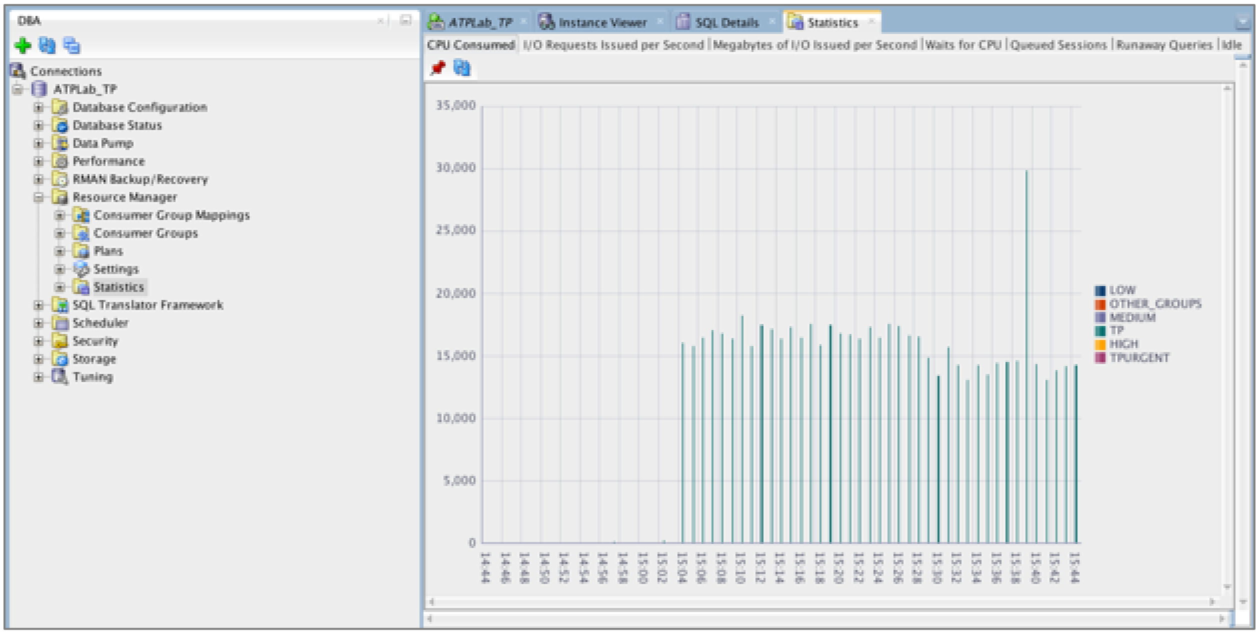
- You will see the directives for the **_HIGH** plan.



Edit Directive

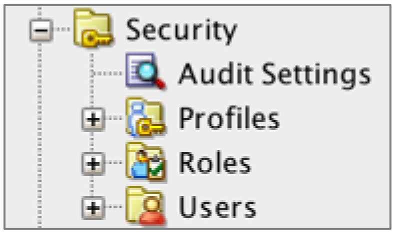
Consumer Group	HIGH
Shares	4
Max Degree of Parallelism	8
Switch Group	
Max Number of Active Sessions	-1
Queue Timeout (sec)	-1
Max Undo Space (KB)	-1
Max Est Execution Time (sec)	-1
Max Idle Time (sec)	-1
Max Blocking Idle Time (sec)	-1
Execution Time Limit (Sec)	-1
I/O Limit (MB)	-1
I/O Request Limit (Requests)	-1

- The last entry under Resource Manager is **Statistics**. This view provides graphical views into system usage by plan user. Double click on **Statistics** and explore the different screens.

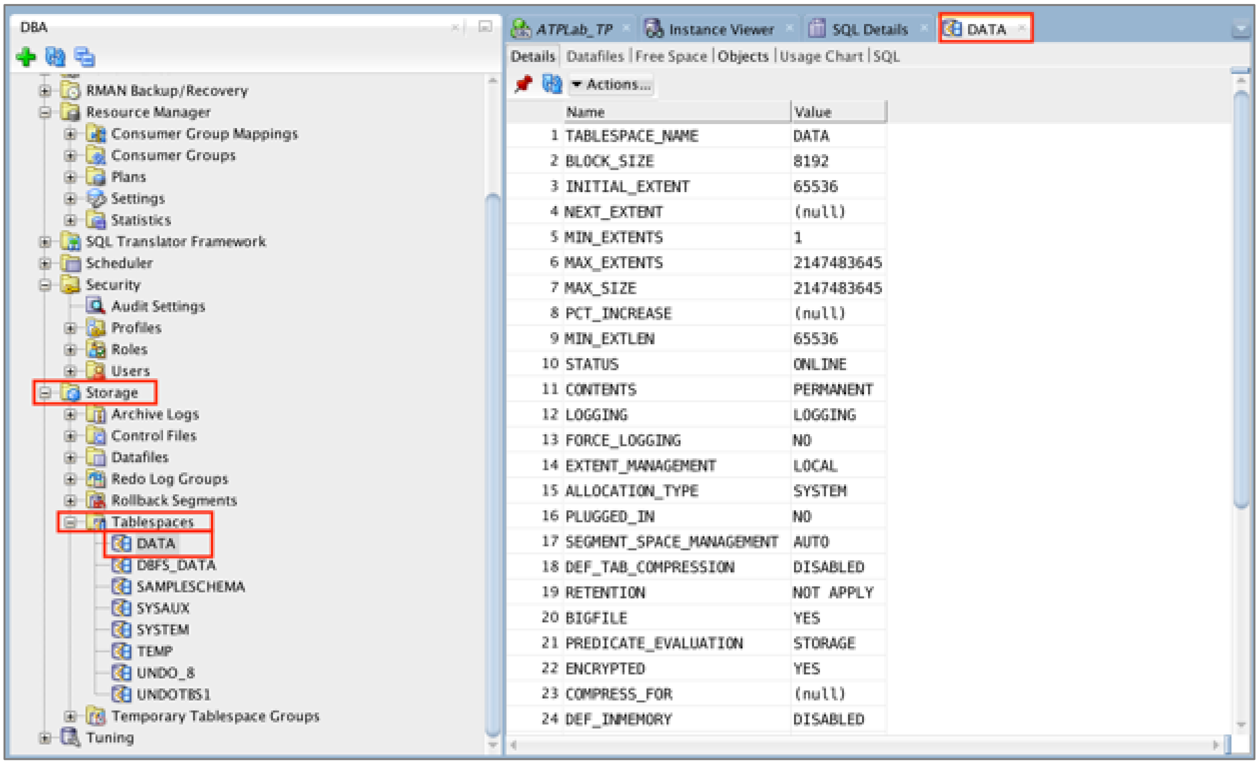


9) Security and Storage

- The **Security** section is useful to DBAs as they can explore all the Profiles, Roles defined in the system, Users, and privileges such as system and role privileges granted to the user.



- The **Storage** section can be used to display all the storage characteristics of the ADB database. Most of these characteristics cannot be changed, as the autonomous database automatically manages and optimizes storage for the service. However, the configurations can be examined. For example, any data in a database will be stored in the DATA Tablespace. Below is a screenshot of the parameters of the DATA Tablespace (expand **Storage -> Tablespaces** and then double click on **DATA**).



	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

- This concludes the lab. SQL Developer is a very comprehensive powerful tool and you should continue to explore the benefits it has to offer.

Lab 7-2: Use Swingbench to Generate Workload and Monitor Performance

Swingbench is an easy to use performance benchmarking tool for the Oracle database. It comes with its own set of benchmark schemas, data generators, workload generators, along with the capability to create your own workloads or customize the out-of-the-box workloads. You can explore more on Swingbench on **http://www.dominicgiles.com/index.html**.

> **Note:** The lab VM has the Swingbench load generator software pre-installed and configured to run the Simple Order Entry (SOE) benchmark. The database for the SOE benchmark was loaded into your autonomous database instance in one of the earlier labs, using **Oracle Data Pump**.

Objectives

- Use Swingbench to generate a workload and observe workload performance with scaling.

Required Artifacts

- Ensure you have provisioned an Oracle **Autonomous Transaction Processing** service.
- Ensure that **Swingbench** is installed, or you have access to a lab VM.
- Ensure that **Migrating to ADB Using Data Pump** lab is completed and Data Pump was used to import the **soe_export.dmp** file.

Lab Steps

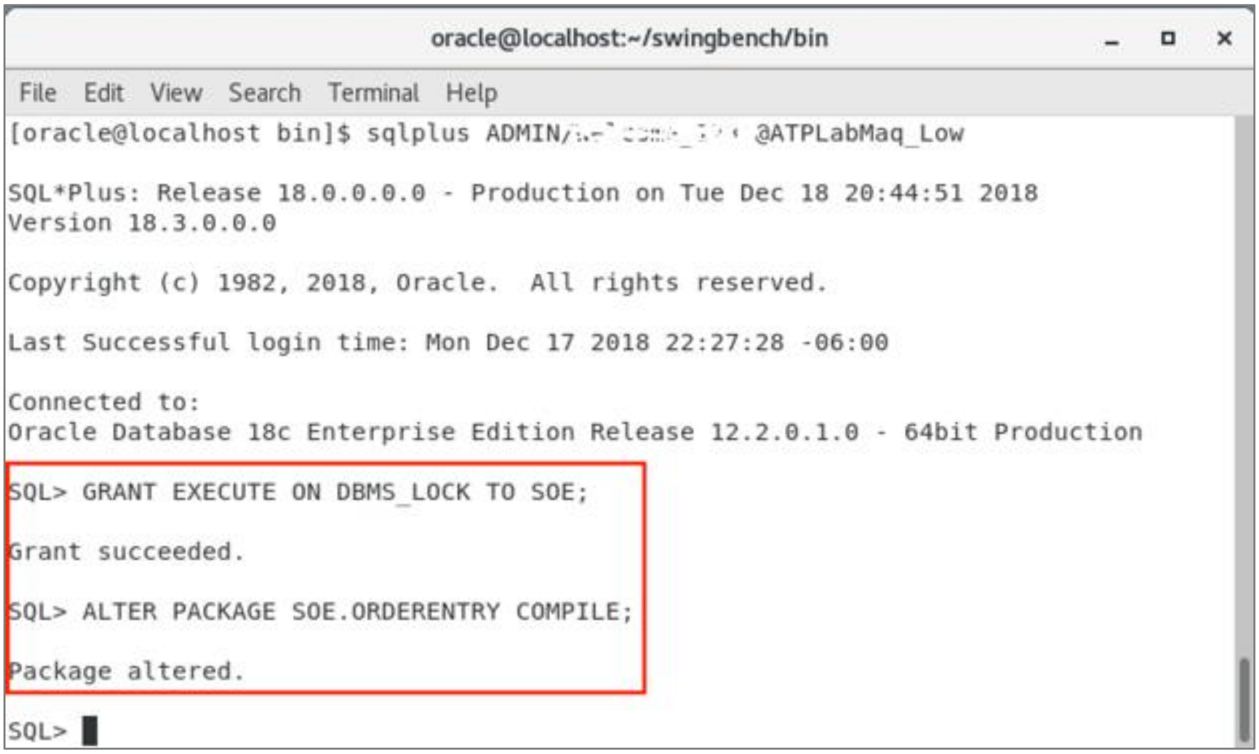
STEP 1: Validate Swingbench Schema Objects

- Let's first validate the Swingbench setup in the lab VM. Open a **Terminal** session and start a **sqlplus** session.

```
$ sqlplus ADMIN/<Admin_Password>@<ATP_Service>
```

- Upon a successful connection, issue following SQLs to grant the missing privilege and to recompile the **ORDERENTRY** package.

```
SQL> GRANT EXECUTE ON DBMS_LOCK TO SOE;
SQL> ALTER PACKAGE SOE.ORDERENTRY COMPILE;
```


oracle@localhost:~/swingbench/bin
File Edit View Search Terminal Help

```
[oracle@localhost bin]$ sqlplus ADMIN/[redacted] @ATPLabMaq_Low

SQL*Plus: Release 18.0.0.0.0 - Production on Tue Dec 18 20:44:51 2018
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle.  All rights reserved.

Last Successful login time: Mon Dec 17 2018 22:27:28 -06:00

Connected to:
Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> GRANT EXECUTE ON DBMS_LOCK TO SOE;

Grant succeeded.

SQL> ALTER PACKAGE SOE.ORDERENTRY COMPILE;

Package altered.

SQL>
```

Note: The above step is needed because Data Pump import did not carry over **EXECUTE ON DBMS_LOCK** privilege after creating the **SOE** thereby making the **ORDERENTRY** package invalid.

- Exit from the **sqlplus** session.

```
SQL> EXIT;
```

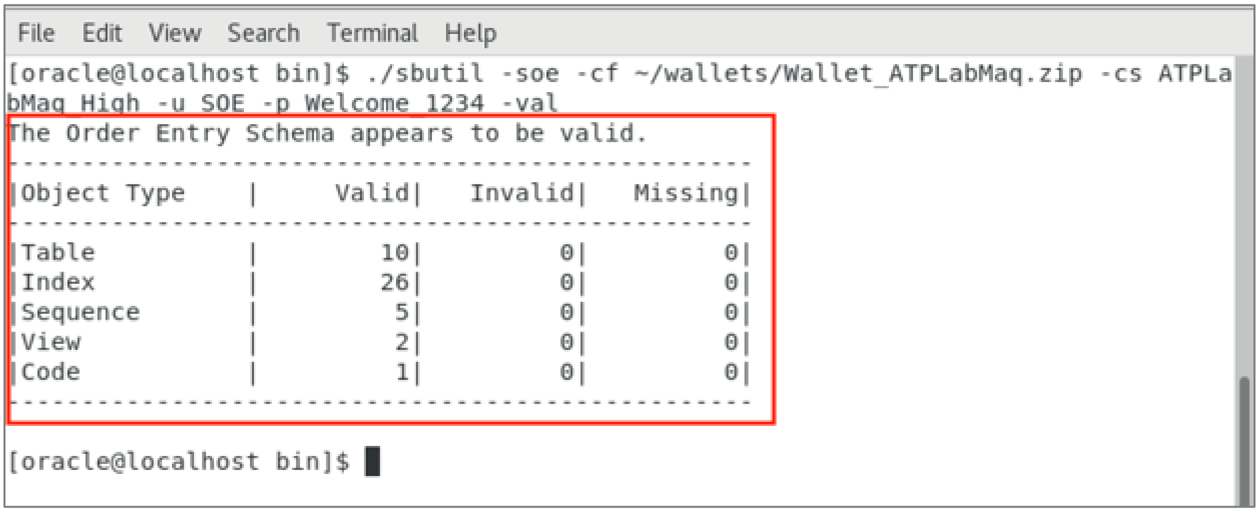
- Change directory to **~/swingbench/bin**.

```
$ cd ~/swingbench/bin
```

- You can verify the Swingbench schema setup by running the following **sbutil** command. Ensure that you specify the full path to your wallet zip file, input the ATP service name and the SOE user's password:

```
$ ./sbutil -soe -cf <Wallet_Zip_File> -cs <ATP_Service> -u SOE -p Welcome_1234 -val
```

- Your results should be similar to the screenshot below:

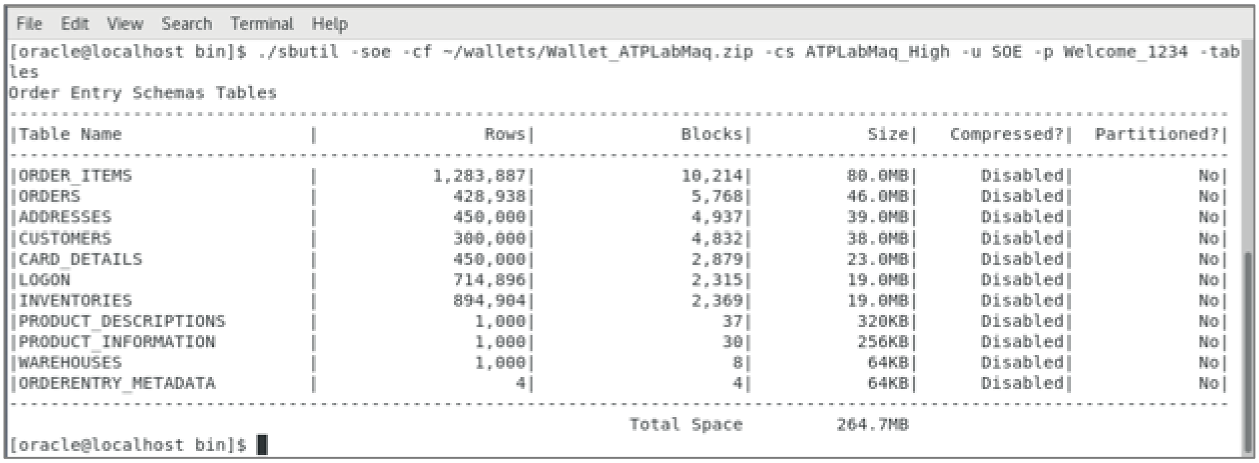


```
[oracle@localhost bin]$ ./sbutil -soe -cf ~/wallets/Wallet_ATPLabMaq.zip -cs ATPLa
bMaq_High -u SOE -p Welcome_1234 -val
The Order Entry Schema appears to be valid.
--------------------------------------------------
|Object Type    |      Valid|   Invalid|   Missing|
--------------------------------------------------
|Table          |         10|         0|         0|
|Index          |         26|         0|         0|
|Sequence       |          5|         0|         0|
|View           |          2|         0|         0|
|Code           |          1|         0|         0|
--------------------------------------------------

[oracle@localhost bin]$ █
```

- To see the number of rows in each table run the following command:

```
$ ./sbutil -soe -cf <Wallet_Zip_File> -cs <ATP_Service> -u SOE -p Welcome_1234 -tables
```

- Your results should be similar to the screenshot below:

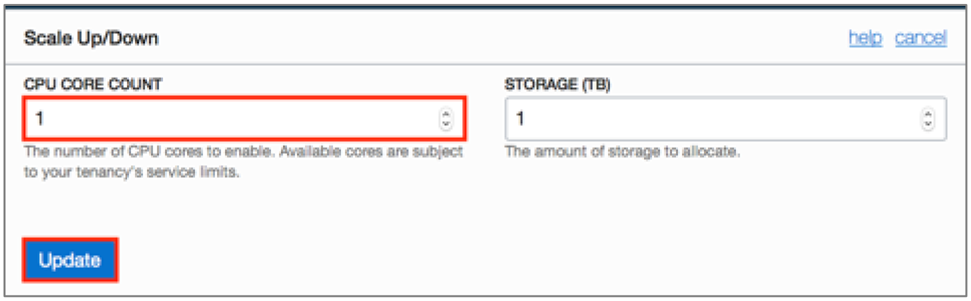


```
[oracle@localhost bin]$ ./sbutil -soe -cf ~/wallets/Wallet_ATPLabMaq.zip -cs ATPLabMaq_High -u SOE -p Welcome_1234 -tab
les
Order Entry Schemas Tables
----------------------------------------------------------------------------------------------------------------------
|Table Name                  |              Rows|                 Blocks|              Size|   Compressed?|  Partitioned?|
----------------------------------------------------------------------------------------------------------------------
|ORDER_ITEMS                 |         1,283,887|                 10,214|            80.0MB|      Disabled|            No|
|ORDERS                      |           428,938|                  5,768|            46.0MB|      Disabled|            No|
|ADDRESSES                   |           450,000|                  4,937|            39.0MB|      Disabled|            No|
|CUSTOMERS                   |           300,000|                  4,832|            38.0MB|      Disabled|            No|
|CARD_DETAILS                |           450,000|                  2,879|            23.0MB|      Disabled|            No|
|LOGON                       |           714,896|                  2,315|            19.0MB|      Disabled|            No|
|INVENTORIES                 |           894,904|                  2,369|            19.0MB|      Disabled|            No|
|PRODUCT_DESCRIPTIONS        |             1,000|                     37|             320KB|      Disabled|            No|
|PRODUCT_INFORMATION         |             1,000|                     30|             256KB|      Disabled|            No|
|WAREHOUSES                  |             1,000|                      8|              64KB|      Disabled|            No|
|ORDERENTRY_METADATA         |                 4|                      4|              64KB|      Disabled|            No|
----------------------------------------------------------------------------------------------------------------------
                                                      Total Space          264.7MB
[oracle@localhost bin]$ █
```

STEP 2: Run the Swingbench SOE Workload

You are now ready to run the Swingbench workload. Workloads are simulated by users submitting transactions to the database.

- Firstly, scale down the CPUs of your service to **1**. We will scale them up later.



Scale Up/Down

CPU CORE COUNT: 1

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

STORAGE (TB): 1

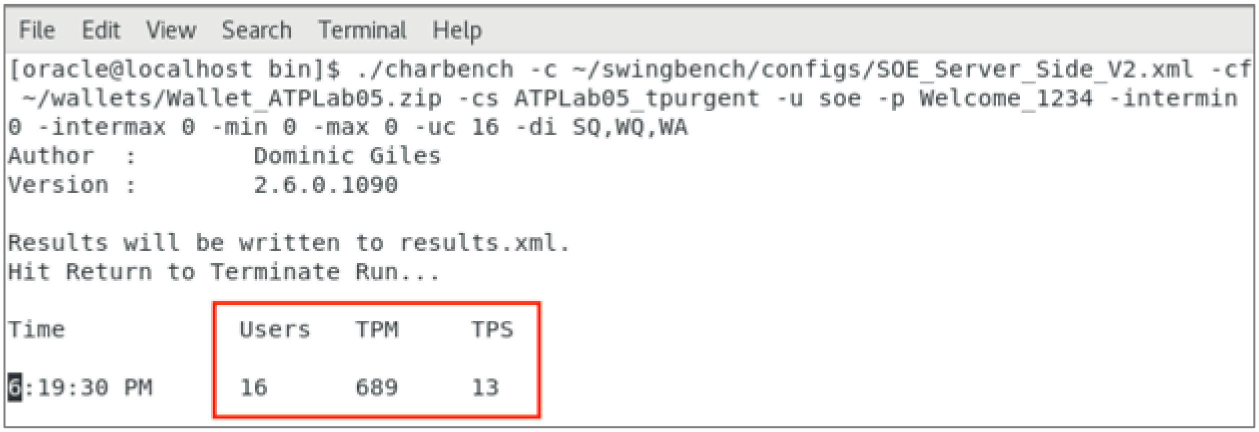
The amount of storage to allocate.

- Execute the **charbench** utility to generate a workload. Here are some commonly used parameters for charbench:
  - **-cs** : Your <ATP_TPURGENT_Service> name (Use the **TPURGENT** service)
  - **-min** and **-max** : The time to sleep between each DML operation in a transaction (intra sleep). A Transaction is made up of many DML operations.
  - **-intermin** and **-intermax** : The time to sleep between each transaction.
  - **-uc** : 16 (the number of users that will be ramped-up)
  - **-di** : SQ,WQ,WA (indicates that transactions SQ,WQ,WA are disabled, as these are reporting queries).

> **Note:** charbench will place a high load on the database, so don't run for long periods of time. You can stop running charbench at any time with CTRL-C.

```
./charbench -c ~/swingbench/configs/SOE_Server_Side_V2.xml \
            -cf <Wallet_Zip_File>  \
            -cs <ATP_TPURGENT_Service> \
            -u soe \
            -p Welcome_1234 \
            -intermin 0 \
            -intermax 0 \
            -min 0 \
            -max 0 \
            -uc 16 \
            -di SQ,WQ,WA
```

- Once your workload gets stabilized, the output may look similar to the screenshot below (your results will vary). The columns indicate the wall clock time, the number of users connected, the transactions per minute (TPM), and the transactions per second (TPS). There are many other parameters that can be included to show more information.



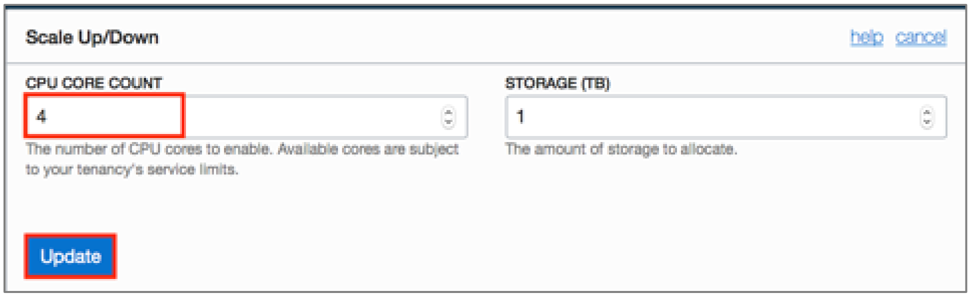
```
[oracle@localhost bin]$ ./charbench -c ~/swingbench/configs/SOE_Server_Side_V2.xml -cf
 ~/wallets/Wallet_ATPLab05.zip -cs ATPLab05_tpurgent -u soe -p Welcome_1234 -intermin
0 -intermax 0 -min 0 -max 0 -uc 16 -di SQ,WQ,WA
Author  :        Dominic Giles
Version :        2.6.0.1090

Results will be written to results.xml.
Hit Return to Terminate Run...

Time                Users    TPM      TPS

6:19:30 PM          16       689      13
```

- The above output was taken with 1 CPUs and connected to **TPURGENT** service. The **TPS** obtained was ~10-15 and **TPM** ~650-700.

> **Note:** Remember that using the **_TPURGENT** service for the workload ensures the highest performance for OLTP.

STEP 3: Scale Up the CPUs and Monitor Performance

- Once the workload has stabilized and you get a consistent **TPM**, scale up the CPUs to **4.**



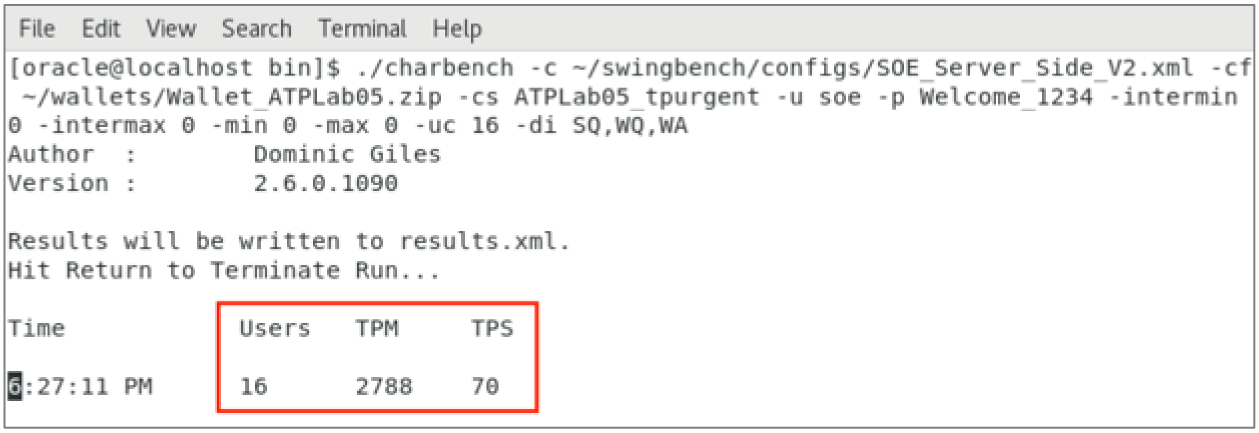
CPU CORE COUNT: 4

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.

- Monitor the TPS column to see variation in the transactions per minute from the previous run with 1 CPUs. Notice the improvement from **~10-15 TPS** to **~50-70 TPS** for the same workload when the CPUs were scaled from **1 to 4**.



```
[oracle@localhost bin]$ ./charbench -c ~/swingbench/configs/SOE_Server_Side_V2.xml -cf
 ~/wallets/Wallet_ATPLab05.zip -cs ATPLab05_tpurgent -u soe -p Welcome_1234 -intermin
0 -intermax 0 -min 0 -max 0 -uc 16 -di SQ,WQ,WA
Author  :        Dominic Giles
Version :        2.6.0.1090

Results will be written to results.xml.
Hit Return to Terminate Run...

Time              Users    TPM      TPS

6:27:11 PM        16       2788     70
```

- **IMPORTANT:** Before you move on, stop the charbench utility by pressing **Enter** in the **Terminal** window.

- You have successfully completed this lab.

Lab 7-3: Backup and Recovery

Oracle Autonomous Database (ADB) is configured to perform automatic backups of your database and retain them for 60 days. The schedule for these built-in backups is to run a full backup every week and incremental backups every day.

You can restore and recover your database to any point-in-time in the 60-day retention period, or to a manually saved backup, in case you have initiated that separately.

Manual Backups

You can perform on-demand manual backups as well, provided you setup a seprate OCI Object Storage bucket to store the manual backups. This is ideal when you want to take a backup before some major application change, or for additional data retention requirements due to regulations which is beyond the 60 days provided by the ADB service.

Recovery

You can initiate recovery for your ADB using the cloud console. ADB automatically restores and recovers your database to the point-in-time you specify. The recovery process internally decides which backup to use to provide you the fastest recovery, either the automatic backups or the manual backups.

Objectives

- Perform a manual backup of the autonomous database
- Perform a point-in-time recovery of the autonomous database

Required Artifacts

- Please ensure you have provisioned an Oracle **Autonomous Transaction Processing** or an **Autonomous Data Warehouse** instance.
- **Oracle SQL Developer** installed, or access to a lab VM with Oracle SQL Developer.
- Database Credential object created from the lab **Loading Data from Object Storage**.

> **IMPORTANT**: It is preferred that you complete the REST API lab and use the instance created using REST APIs for backup and recovery. This will avoid disruption in your lab work when your main lab instance becomes unavailable.

Lab Steps

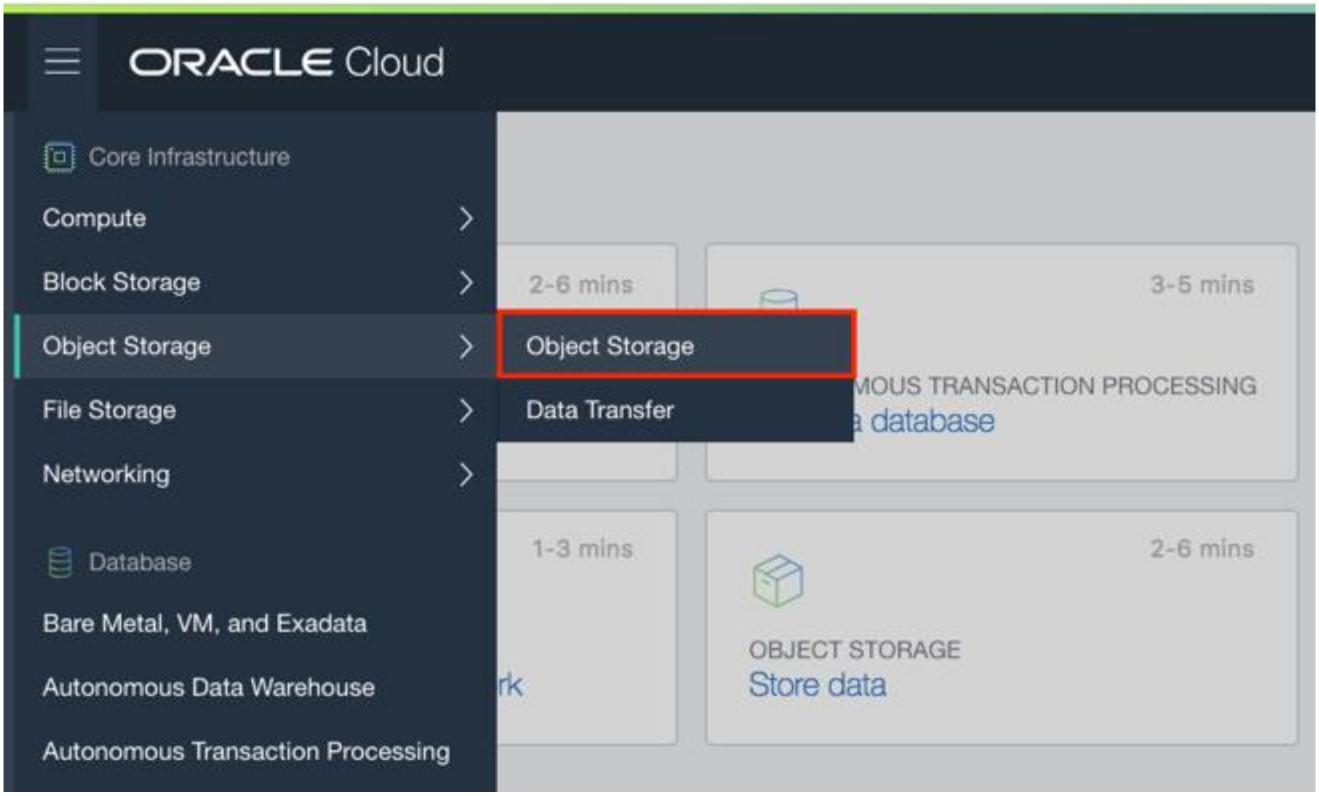
STEP 1: Prerequisite Steps for Manual Backups

Manual backups require that you perform a one-time setup for creating an object store bucket in which the backups will be stored, along with the appropriate credentials to access the bucket. These steps are typically done once but will need to be repeated once the URL, the credentials, or OCI bucket changes.

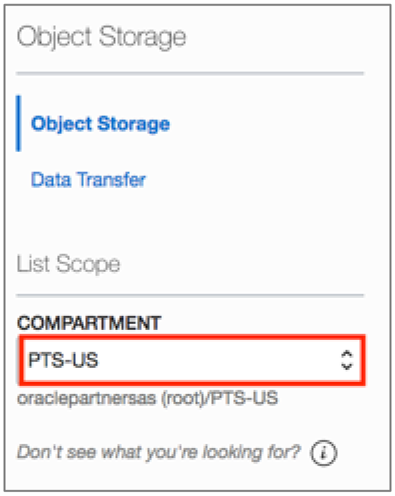
Follow these steps to perform the one-time setup tasks for the manual backups.

1) Create an Object Store Bucket to Store Manual Backups

- Create a bucket to hold the backups. Login to the Oracle Cloud Infrastructure Console and select **Object Storage->Object Storage** from the top left menu.



- Select a **Compartment** from the drop-down on the left. Choose the compartment assigned to you by your instructor.



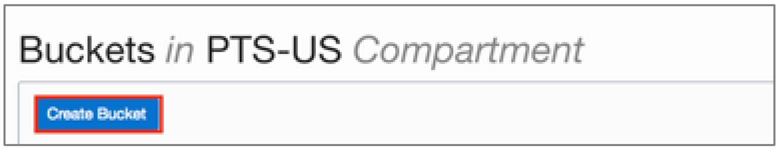
COMPARTMENT

PTS-US

oraclepartnersas (root)/PTS-US

Don't see what you're looking for?

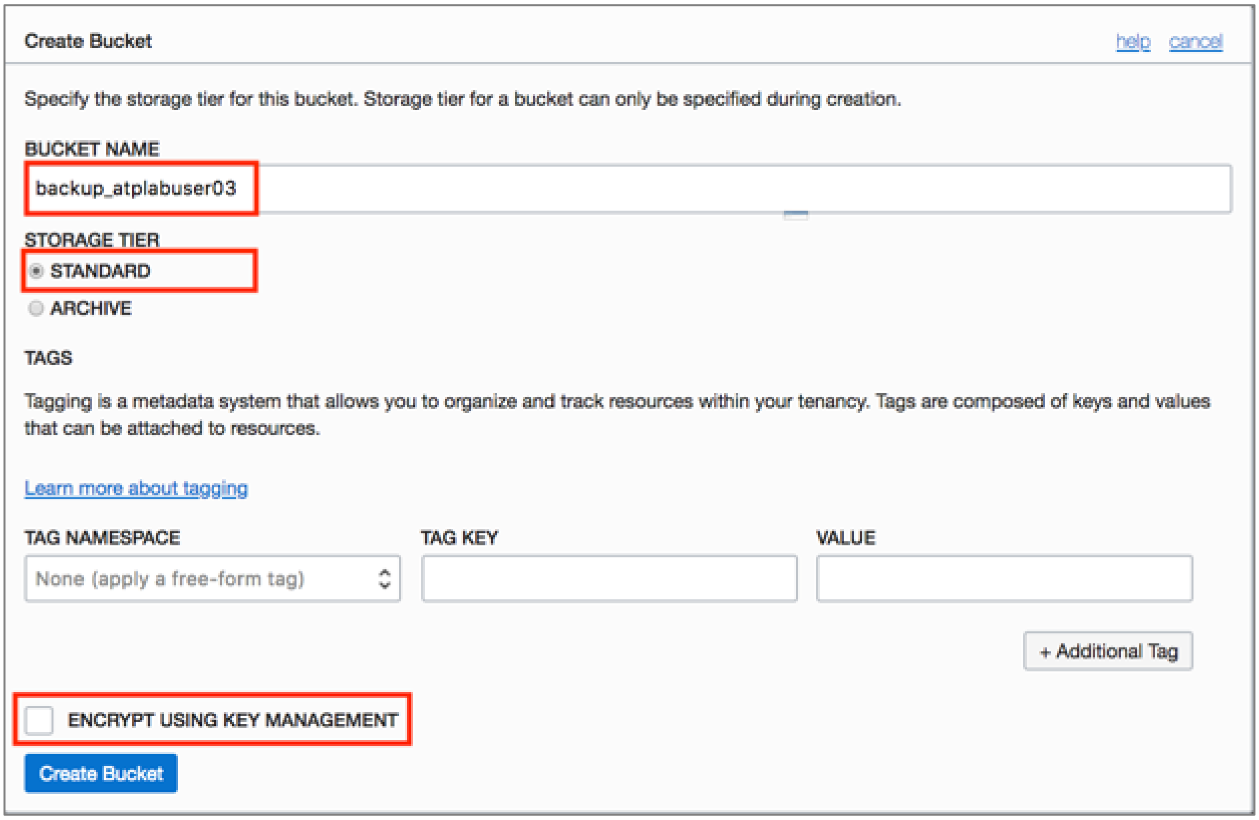
- Click on **Create Bucket** to create the object storage bucket.



- In the **Create Bucket** dialog box, enter the following:
  - **Bucket Name**: The format of the bucket name is backup_<databasename>, where <databasename> is your database name in lowercase. For example, if you provision an ADB instance named **ATPLabUser03**, the bucket name should be **backup_atplabuser03**.
  - **Storage Tier**: Choose **Standard** as manual backups are only supported with buckets created in the standard storage tier.
  - **Encrypt Using Key Management**: Unchecked

Note: It is important that you follow the above naming convention and stick with the above settings as not all options are allowed for a bucket to be used for manual backups.

- Click on **Create Bucket.**



Create Bucket

Specify the storage tier for this bucket. Storage tier for a bucket can only be specified during creation.

BUCKET NAME
backup_atplabuser03

STORAGE TIER
● STANDARD
○ ARCHIVE

TAGS

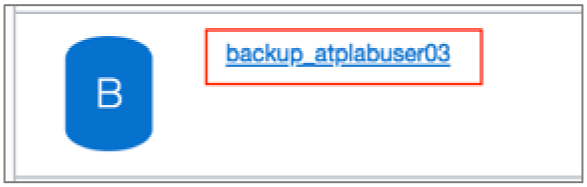
Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

Learn more about tagging

TAG NAMESPACE	TAG KEY	VALUE
None (apply a free-form tag)		

□ ENCRYPT USING KEY MANAGEMENT

- Verify the bucket got successfully created.



backup_atplabuser03

2) Construct the Bucket URL

Construct the URL that points to the location of the bucket. The URL is structured as follows (the values for you to specify are in bold):

```
https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/<tenant_name>
```

- **<region_name>** : The region you have created your Object Storage bucket. Typically, this would be **us-phoenix-1**, **us-ashburn-1**, etc.
- **<tenant_name>** : The OCI tenancy name

In the below example of the URL when the region name is **us-phoenix-1** and the tenancy name is **oraclepartnersas** (**IMPORTANT:** Your URL would be different, so please modify accordingly):

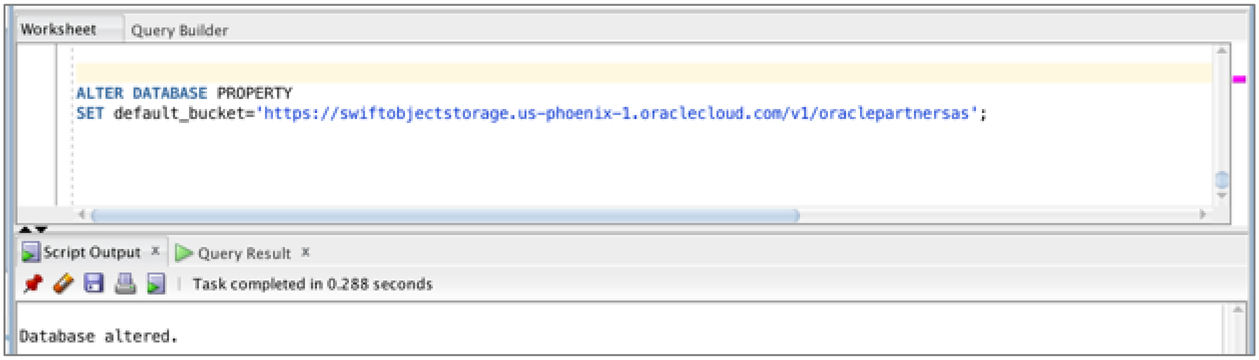
```
https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas
```

3) Set the DEFAULT_BUCKET Property

- Open **SQL Developer** and connect to your ADB as the **ADMIN** user.
- Run the following SQL to set the database **default_bucket** property to point to the URL constructed above, using the following **ALTER DATABASE** command (but using your own URL):

```
ALTER DATABASE PROPERTY SET
default_bucket='https://swiftobjectstorage.<region_name>.oraclecloud.com/v1/
<tenant_name>';
```

- Here is a screenshot of a successful execution of the command:



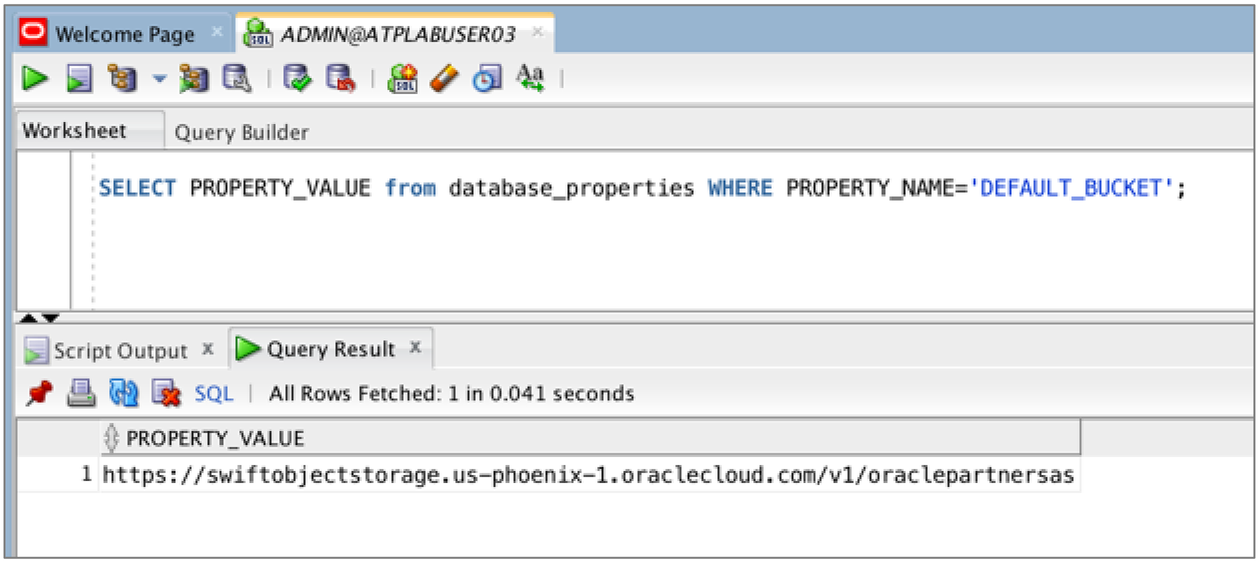
```
ALTER DATABASE PROPERTY
SET default_bucket='https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas';
```

Task completed in 0.288 seconds

```
Database altered.
```

- Verify the **default_bucket** was altered using the following SQL statement:

```
SELECT PROPERTY_VALUE from database_properties WHERE
PROPERTY_NAME='DEFAULT_BUCKET';
```



```
SELECT PROPERTY_VALUE from database_properties WHERE PROPERTY_NAME='DEFAULT_BUCKET';
```

All Rows Fetched: 1 in 0.041 seconds

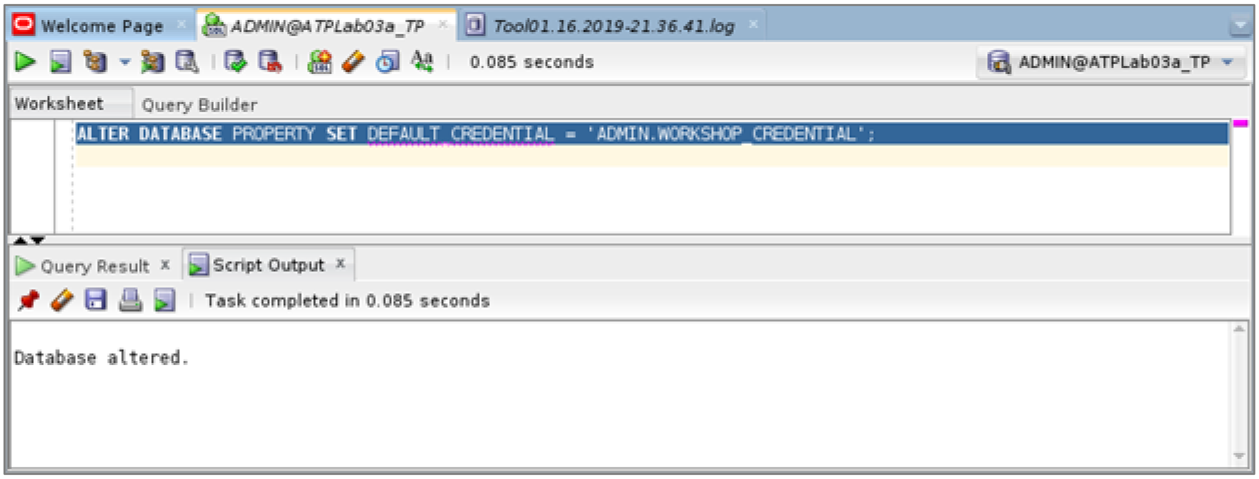
```
  PROPERTY_VALUE
1 https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/oraclepartnersas
```

4) Set the DEFAULT_CREDENTIAL Database Property

In the lab titled **Loading Data from Object Storage**, you should have created a database credential object named **WORKSHOP_CREDENTIAL.** The backup process requires a database credential to access the object storage and it does that by relying on a database property that you set; named **DEFAULT_CREDENTIAL.**

- Set the **default_credential** database property to the credential just created.

```
ALTER DATABASE PROPERTY SET default_credential = 'ADMIN.WORKSHOP_CREDENTIAL';
```



```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.WORKSHOP_CREDENTIAL';
```

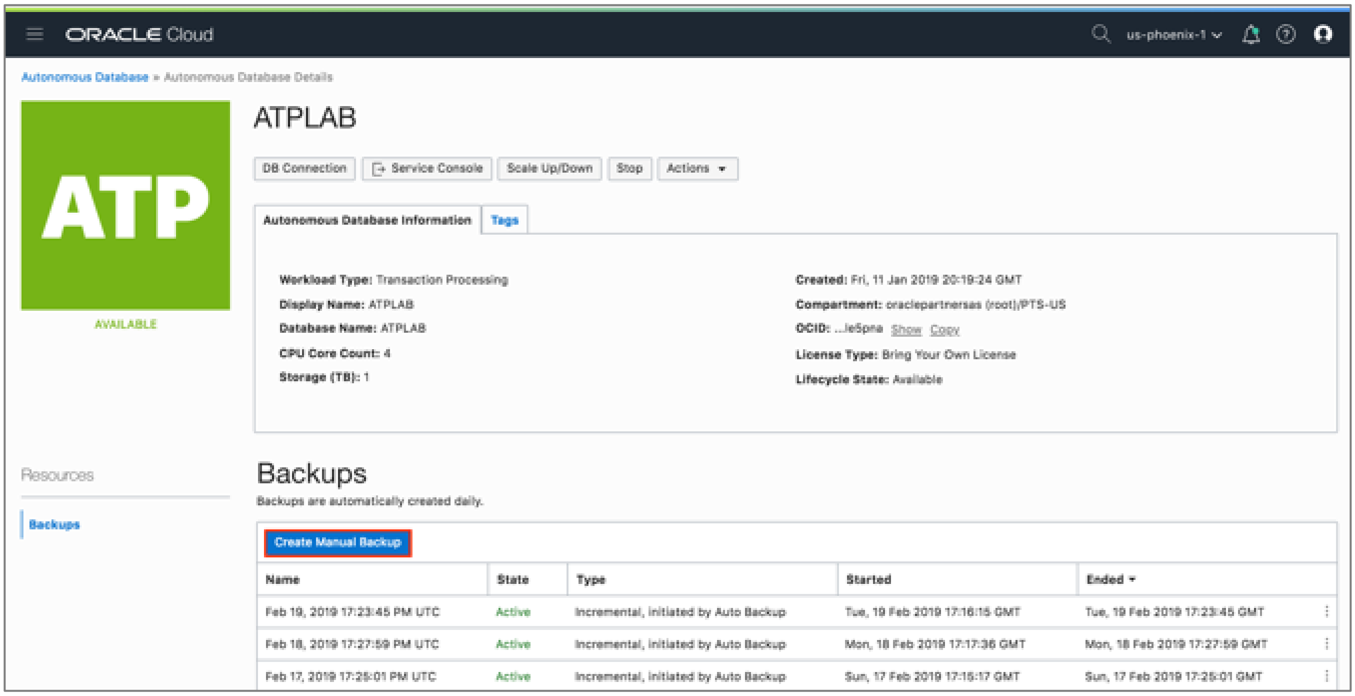
Task completed in 0.085 seconds

```
Database altered.
```

STEP 2: Perform a Manual Backup

Each manual backup creates a full backup on your Oracle Cloud Infrastructure Object Storage bucket and the backup can only be used by the ADB instance when you initiate a point-in-time-recovery.

- From **Oracle Autonomous Transaction Processing** or **Oracle Autonomous Data Warehouse** page, select your ADB instance.
- On the details page, under **Backups**, click **Create Manual Backup**.



ATPLAB

Workload Type: Transaction Processing
Display Name: ATPLAB
Database Name: ATPLAB
CPU Core Count: 4
Storage (TB): 1

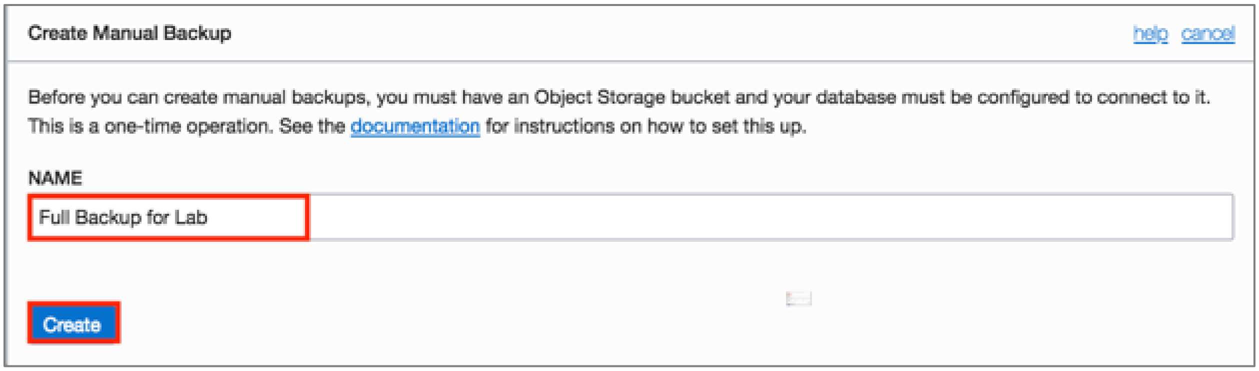
Created: Fri, 11 Jan 2019 20:19:24 GMT
Compartment: oraclepartnersas (root)/PTS-US
OCID: ...le5pna Show Copy
License Type: Bring Your Own License
Lifecycle State: Available

Backups

Backups are automatically created daily.

Name	State	Type	Started	Ended
Feb 19, 2019 17:23:45 PM UTC	Active	Incremental, initiated by Auto Backup	Tue, 19 Feb 2019 17:16:15 GMT	Tue, 19 Feb 2019 17:23:45 GMT
Feb 18, 2019 17:27:59 PM UTC	Active	Incremental, initiated by Auto Backup	Mon, 18 Feb 2019 17:17:36 GMT	Mon, 18 Feb 2019 17:27:59 GMT
Feb 17, 2019 17:25:01 PM UTC	Active	Incremental, initiated by Auto Backup	Sun, 17 Feb 2019 17:15:17 GMT	Sun, 17 Feb 2019 17:25:01 GMT

- In the **Create Manual Backup** dialog enter a name in the **Name** field and click **Create.**

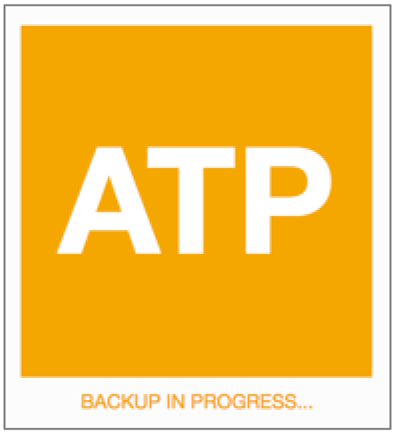


Before you can create manual backups, you must have an Object Storage bucket and your database must be configured to connect to it. This is a one-time operation. See the documentation for instructions on how to set this up.

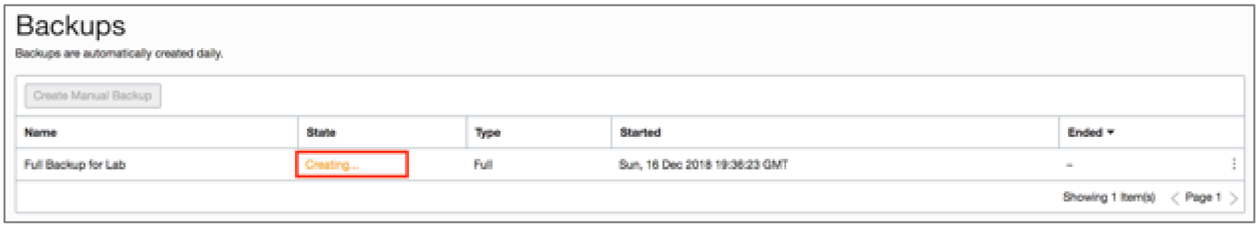
NAME

Full Backup for Lab

- Notice that the status of **Backup in Progress**:



- And the state of the backups will display as **Creating** in the **Backups** section:



Backups

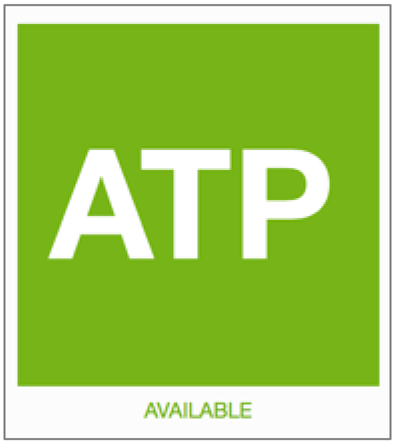
Backups are automatically created daily.

Name	State	Type	Started	Ended ▾
Full Backup for Lab	Creating...	Full	Sun, 16 Dec 2018 19:36:23 GMT	–

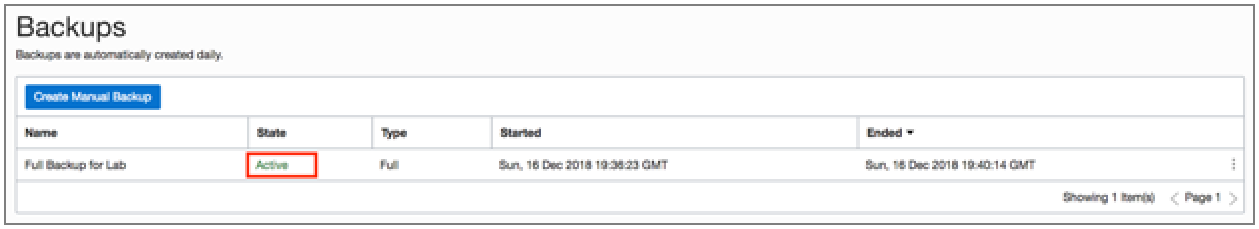
Showing 1 item(s) < Page 1 >

Note: While backing up a database, the database is fully functional; however, during the backup lifecycle management operations are not allowed. For example, stopping the database is not allowed during the backup.

- The backup will take a few minutes to complete and the status of your service will change back to **Available**:



- And the state of the backups will display as **Active** in the **Backups** section:



Backups

Backups are automatically created daily.

Name	State	Type	Started	Ended ▾
Full Backup for Lab	Active	Full	Sun, 16 Dec 2018 19:36:23 GMT	Sun, 16 Dec 2018 19:40:14 GMT

Showing 1 item(s) < Page 1 >

STEP 3: Restore the ADB Database to a Point-in-time

You can initiate recovery for your Autonomous Transaction Processing database using a push button approach from the cloud console.

1) Generate Transactions

Let's first generate some transactions to validate the restore process.

- Open a SQL Developer worksheet connected to any service that you may have defined earlier (**TP**, **HIGH**, etc.).

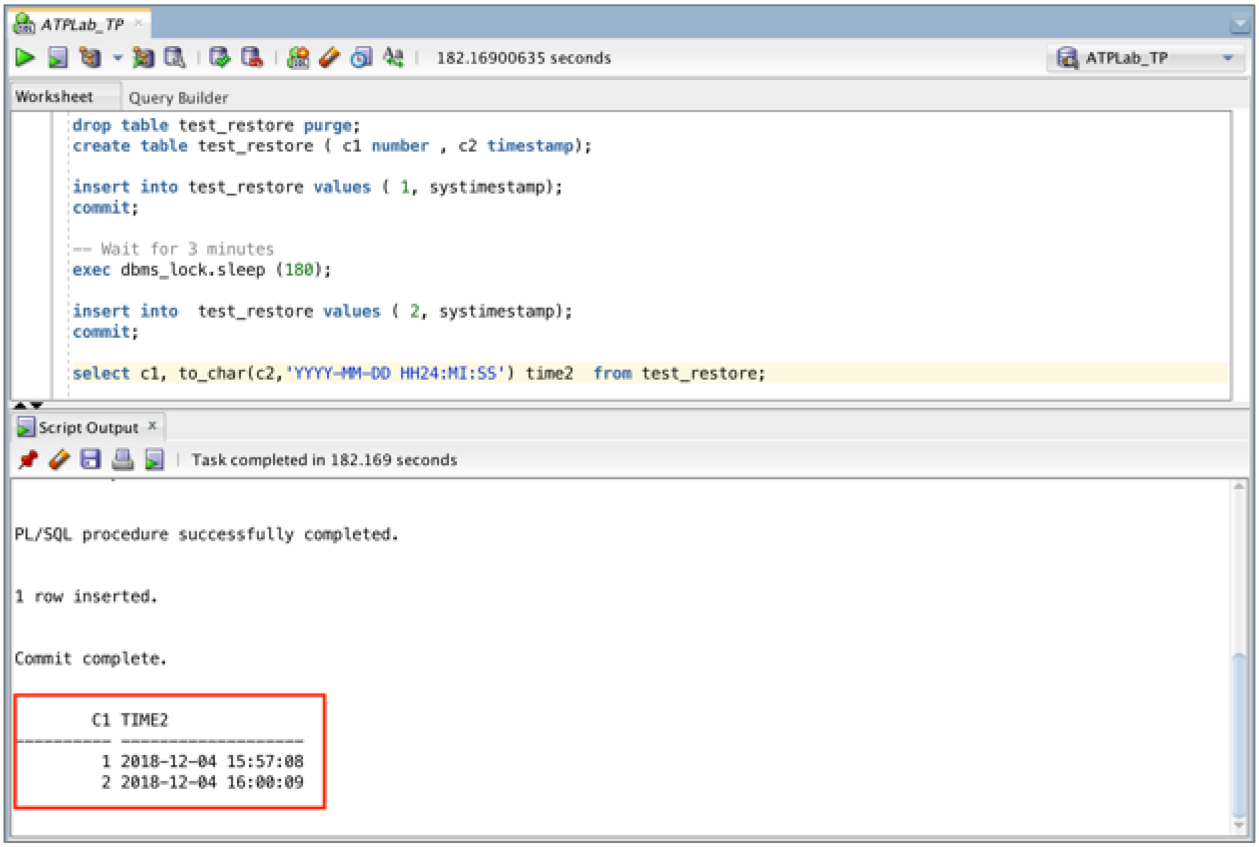
- Copy and paste the following SQL script to the worksheet. This script will record some transactions. This is to test the restore process and ensure that we have recovered to the correct point-in-time.

Note: The test script creates a table named **test_restore**. It then inserts one row and commits. After waiting for 3 minutes, it inserts a second row and commits.

```
drop table test_restore purge;
create table test_restore (c1 number , c2 timestamp);

insert into test_restore values (1, systimestamp);
commit;

-- Wait for 3 minutes
exec dbms_lock.sleep (180);

insert into  test_restore values (2, systimestamp);
commit;

select c1, to_char(c2,'YYYY-MM-DD HH24:MI:SS') time2  from test_restore;
```

- Run the script by selecting all statements and clicking **Run Script** button or pressing **F5**. The script will take about 3 minutes to run as it is designed to wait for 180 seconds between the two insert statements.

Note: Ignore any **DROP** errors reported.



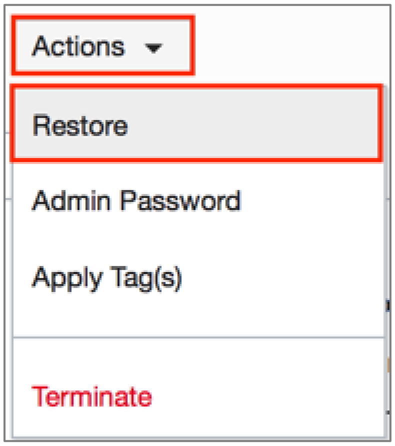
```
drop table test_restore purge;
create table test_restore ( c1 number , c2 timestamp);

insert into test_restore values ( 1, systimestamp);
commit;

-- Wait for 3 minutes
exec dbms_lock.sleep (180);

insert into  test_restore values ( 2, systimestamp);
commit;

select c1, to_char(c2,'YYYY-MM-DD HH24:MI:SS') time2  from test_restore;
```

```
PL/SQL procedure successfully completed.


1 row inserted.


Commit complete.

        C1 TIME2
---------- -------------------
         1 2018-12-04 15:57:08
         2 2018-12-04 16:00:09
```

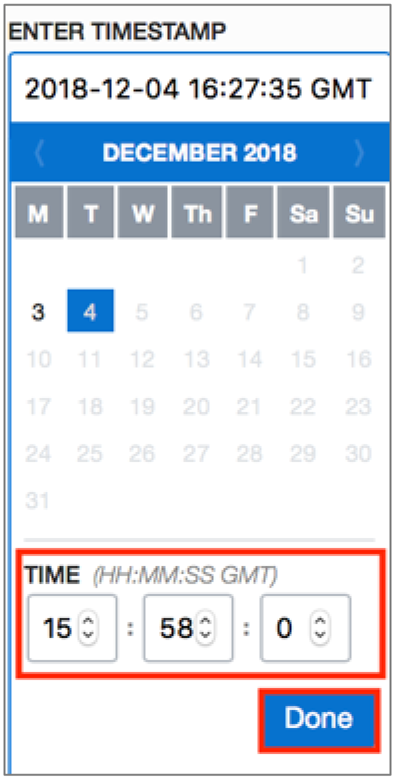
- Please note the timestamps on the two rows inserted as you will use this info in the restore step. From the above SQL Developer output, the first row was inserted at **2018-12-04 15:57:08** and the second row at **2018-12-04 16:00:09**.

2) Restore Database

- Now you are ready to restore your ADB database. From the **Autonomous Transaction Processing Database Details** page, click **Actions -> Retore**.

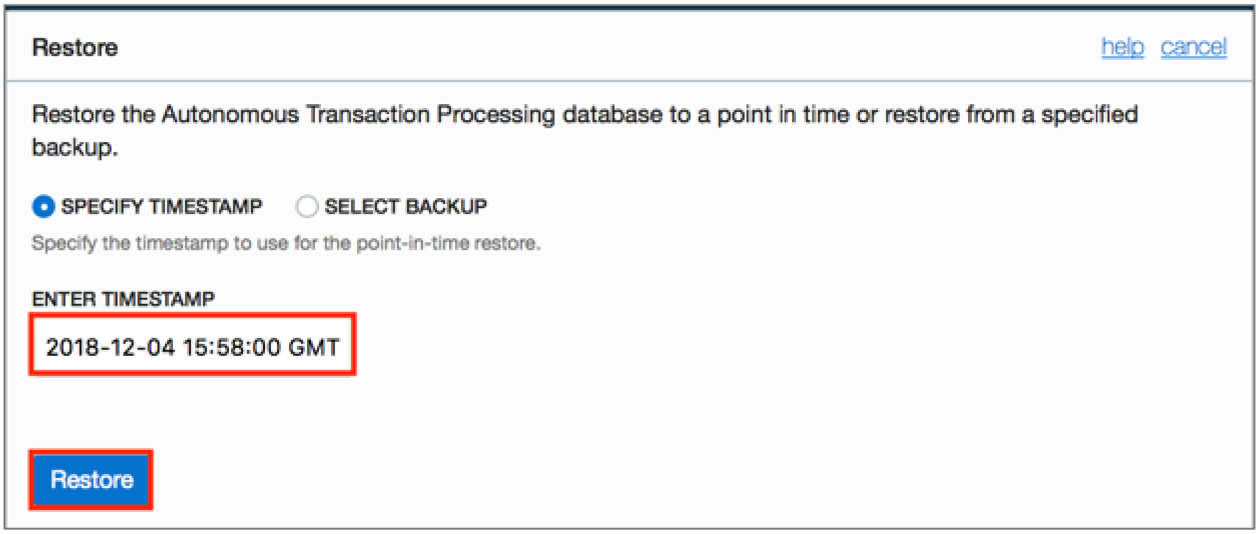


- In the **Restore** dialog, select **SPECIFY TIMESTAMP** and enter the timestamp such that it is in between the insert of first row and that of the second row. In the above example, the time to choose will be between **2018-12-04 15:57:08** and **2018-12-04 16:00:09** so we will choose **2018-12-04 15:58:00**. Click **Done**.


ENTER TIMESTAMP
2018-12-04 16:27:35 GMT

TIME *(HH:MM:SS GMT)*
15 : 58 : 0

- Verify the timestamp one more time and click **Restore**.



Restore

Restore the Autonomous Transaction Processing database to a point in time or restore from a specified backup.

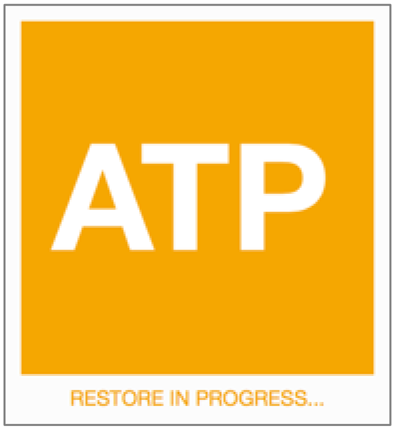
● SPECIFY TIMESTAMP ○ SELECT BACKUP

Specify the timestamp to use for the point-in-time restore.

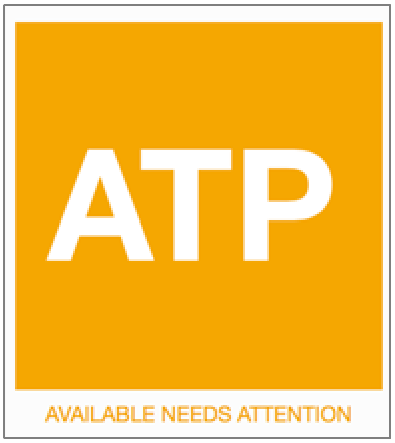
ENTER TIMESTAMP

2018-12-04 15:58:00 GMT

- The restore process will be started. Notice the service status changes to **RESTORE IN PROGRESS**.



- The restore process takes about 10 minutes in the lab. In real world, the restore time depends on the size of the database and how far back in time you need to go and the amount of redo that needs to be applied.
- After the restore operation is completed, the database will be opened in **READ ONLY** mode. This is useful to verify that the point-in-time recovery is successful. The **READ ONLY** mode is indicated by the **AVAILABLE NEEDS ATTENTION** status:



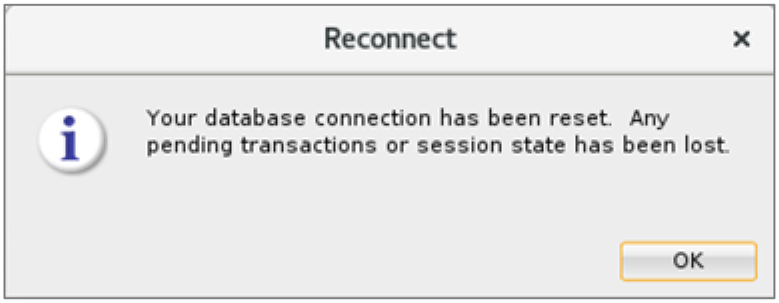
Note: When you restore your database all backups after the restore timestamp are invalidated.

3) Test the Restore

- Test the restore process before opening the database in **READ WRITE** mode.
- In the previous SQL Developer session, run the following query to check the rows in the **test_restore** table.

```
select c1, to_char(c2,'YYYY-MM-DD HH24:MI:SS') time2 from test_restore;
```

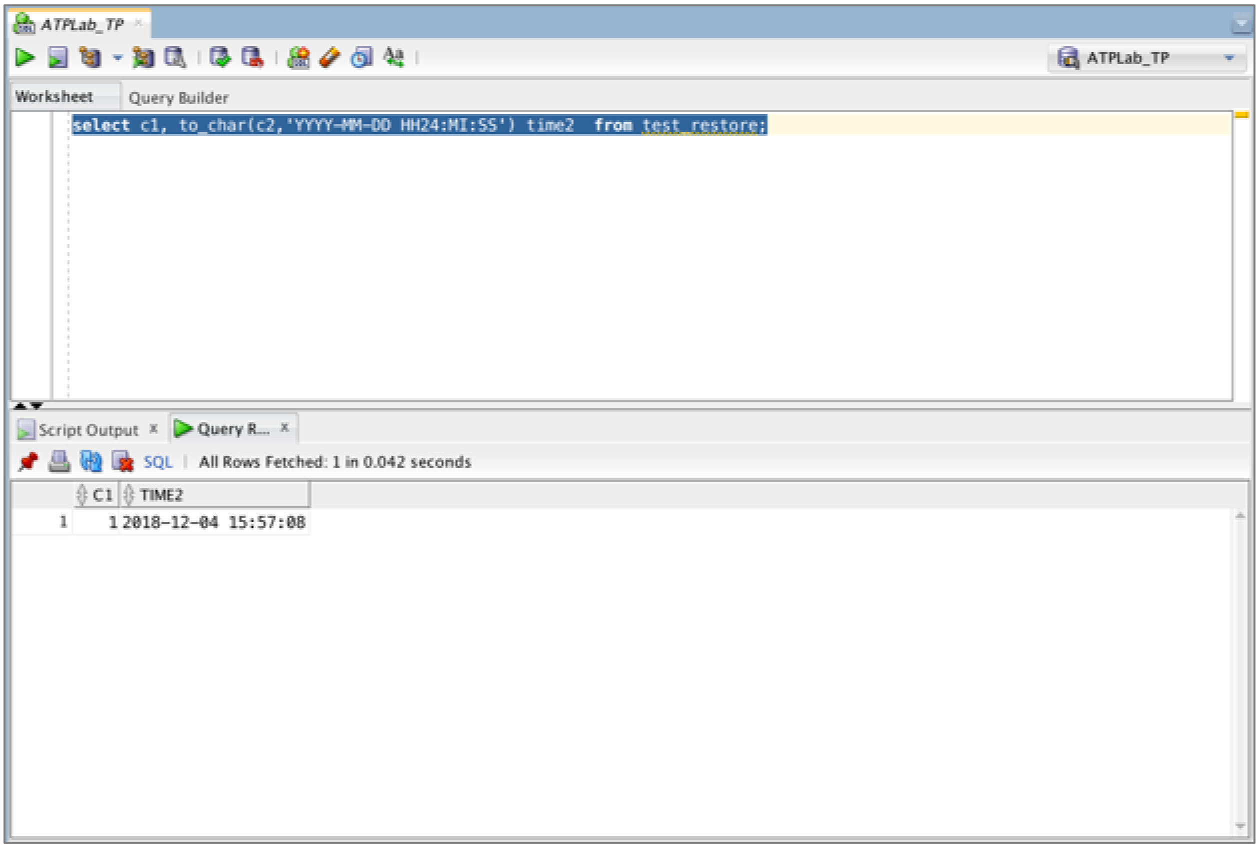
- When you run the above query, you would see the connection **Reconnect** message. Click **OK** to proceed.



Reconnect

Your database connection has been reset. Any pending transactions or session state has been lost.

- You should only see one row returned from the SQL, which is the row with the **2018-12-04 15:57:08** timestamp in the lab example.



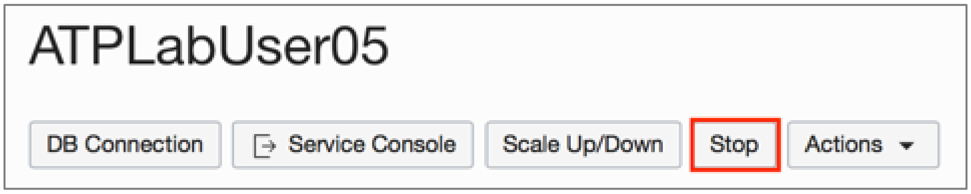
```
select c1, to_char(c2,'YYYY-MM-DD HH24:MI:SS') time2  from test_restore;
```

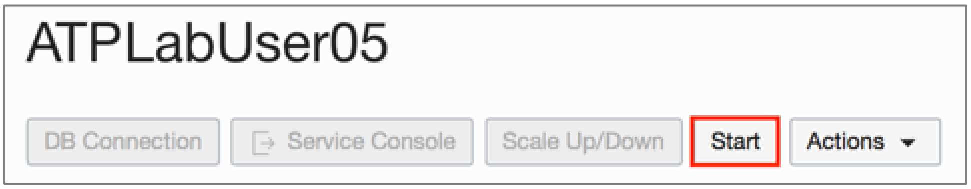
All Rows Fetched: 1 in 0.042 seconds

	C1	TIME2
1	1	2018-12-04 15:57:08

4) Restart the Database Service

- Once the restore is successful, **Stop** and **Start** the service to open the database in **READ WRITE** mode.





- Once the restart of the service is complete the service status will change to **AVAILABLE**.
- You have successfully completed the objectives of this lab.