ORACLE
Sales and Partner Academy

Oracle Autonomous Transaction Processing (ATP)

HANDS-ON LAB GUIDE

JANUARY 2019(V1.1)

TABLE OF CONTENTS

eugenio.galiano@oracle.com

Lab Introduction

The objective of the Autonomous Transaction Processing (ATP) Hands-on Lab is to provide you with an end-to end training on how to set up and use the ATP service with several tools and development environments so that you can obtain hands on experience and also have a basis to create end to end demo's you can share with customers. The objective is not to become an expert in any of these tools or services but to be proficient when having a discussion or demo with a customer. The labs are divided into Level 1, 2, and 3 labs. Level 1 labs introduce the basics of ATP and are included with step by step explanation and screenshots and intended for those with no familiarity with ATP. Level 2 labs focus on monitoring and performance of ATP and focus on DBA activities. Level 3 labs focus on developers and current development environments and require extensive software installation on your computer and DBA knowledge. They also provide the framework for extending the basic lab into customer specific demos.

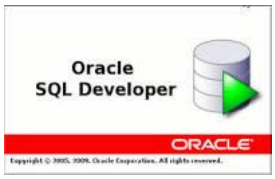
Please note – all these labs are designed at a point in time. There are a lot of software components. Both the Oracle Autonomous Database Service and the software in these labs change quickly so you may observe differences between versions, steps and screenshots in this document and what you are experiencing as you go through these labs.

ATP Hands-on Lab software prerequisites

All the labs in this document are performed in a Windows 10 environment, and installation instructions are for Windows 10. Most download locations also have Mac and Linux downloads but that process is not covered in this lab. Different labs will require that you install different tools. The first set of labs (provisioning, scaling, administration) build on one another and should be run in sequence. The development labs (node.js, Python, Swingbench) don't build on one another so they can be run independently but they all depend on the ATP database created in Lab 1. In order to run all the labs you will need to install the following:

- Oracle SQL Developer
- Oracle Thin Client
- Node.js
- Python
- Swingbench
- Windows Docker

The only tool you will need to install get through Level 1 labs is Oracle SQL Developer. The steps for installation follow.



Oracle SQL Developer

Oracle SQL Developer is a free, integrated development environment that simplifies the development and management of Oracle Database in both traditional and Cloud deployments. SQL Developer offers a worksheet for running queries and scripts, a DBA console for managing the database, a reports interface, a complete data modelling solution, and a migration platform for moving your 3rd party databases to Oracle.

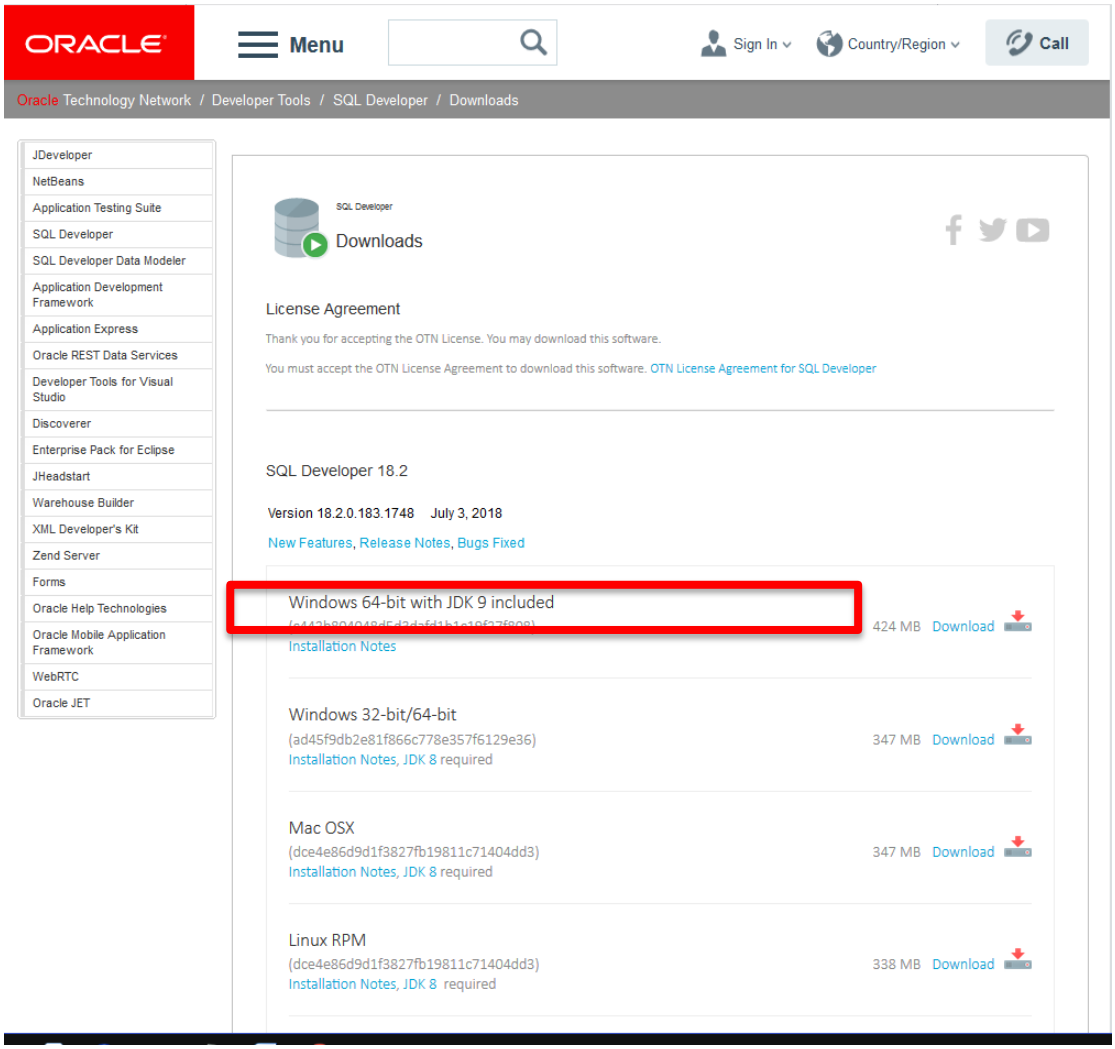
Installing Oracle SQL Developer

You can download SQL Developer Version 18.3 from OTN (make sure you download the correct version for your type of computer and operating system, you can also search for Oracle SQL Developer download if you have problems with the link):

https://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html

Make sure to accept the License Agreement or the download option won't appear. Then select your appropriate architecture and download.

eugenio.galiano@oracle.com



PLEASE NOTE Mac OSX and Linux:

Java SE Development Kit 8u181 is required and can be installed from here (please read SQL Developer installation notes for Max OSX and Linux RPM: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

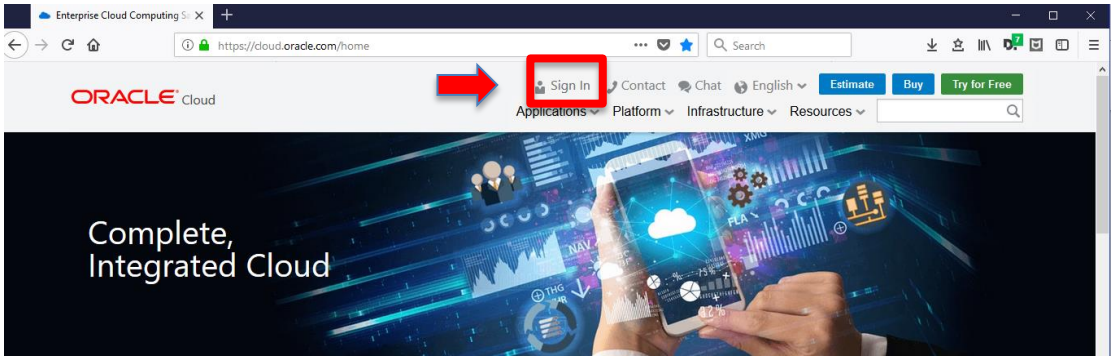
Lab 1. Provisioning an ATP Instance

Objectives

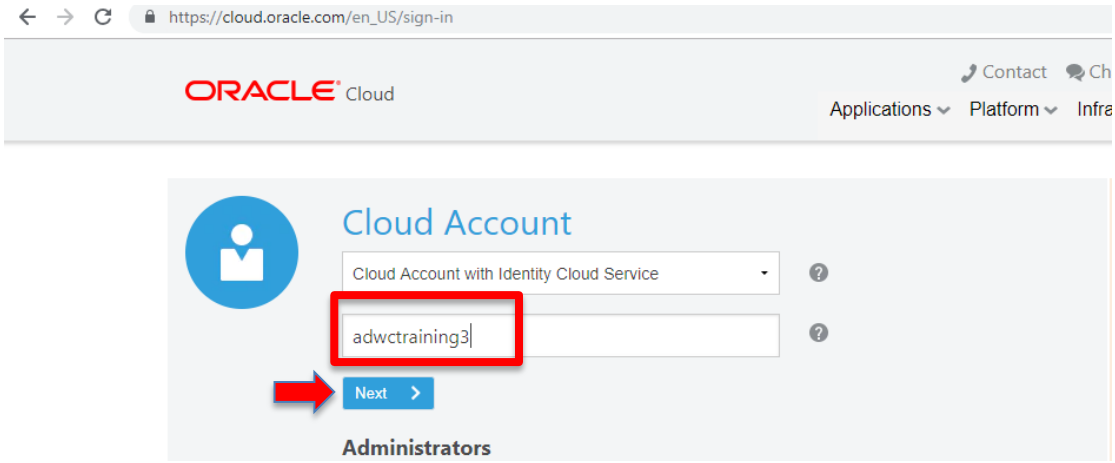
- Learn how to login to the Oracle Cloud Console
- Learn how to provision a new ATP instance

In this section you will be provisioning an ATP database using the cloud console.

Go to cloud.oracle.com, click Sign In to sign in with your Oracle Cloud account.

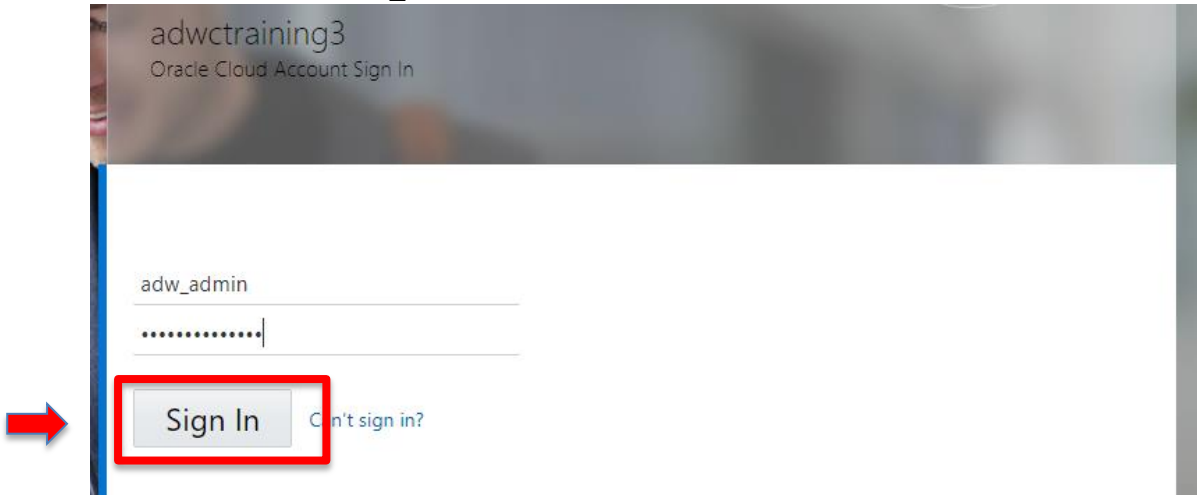


Enter your Cloud Account Name and click **Next**. In this example the Cloud Account Name is adwctraining3.

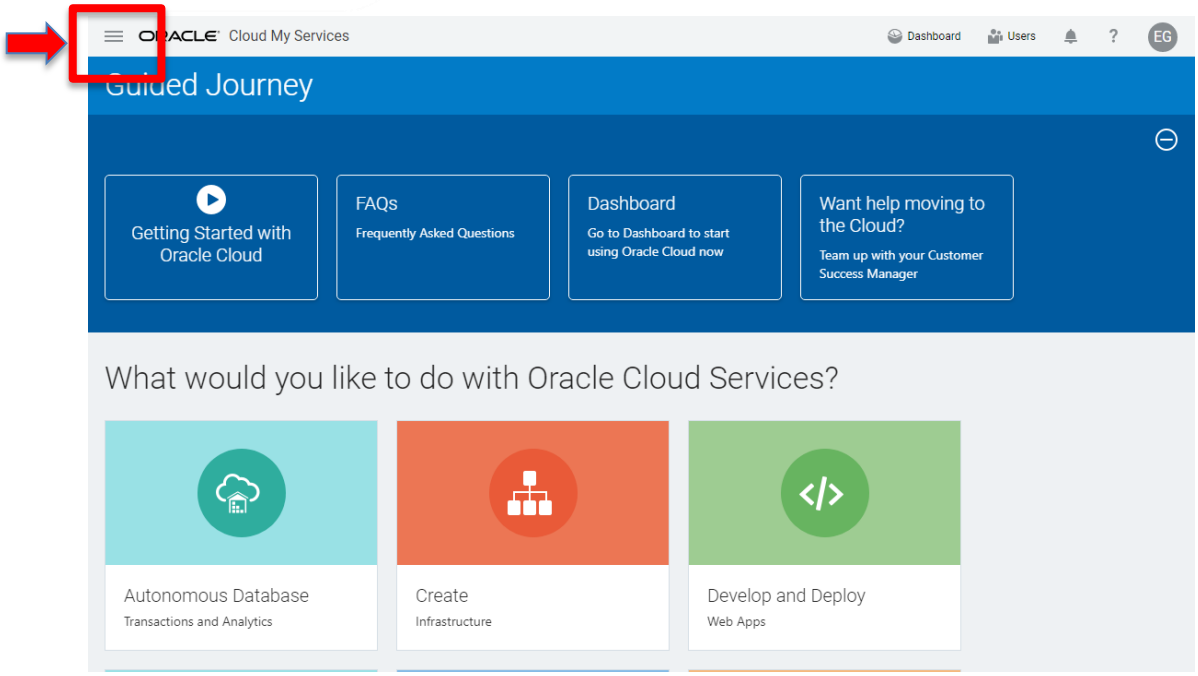


eugenio.galiano@oracle.com

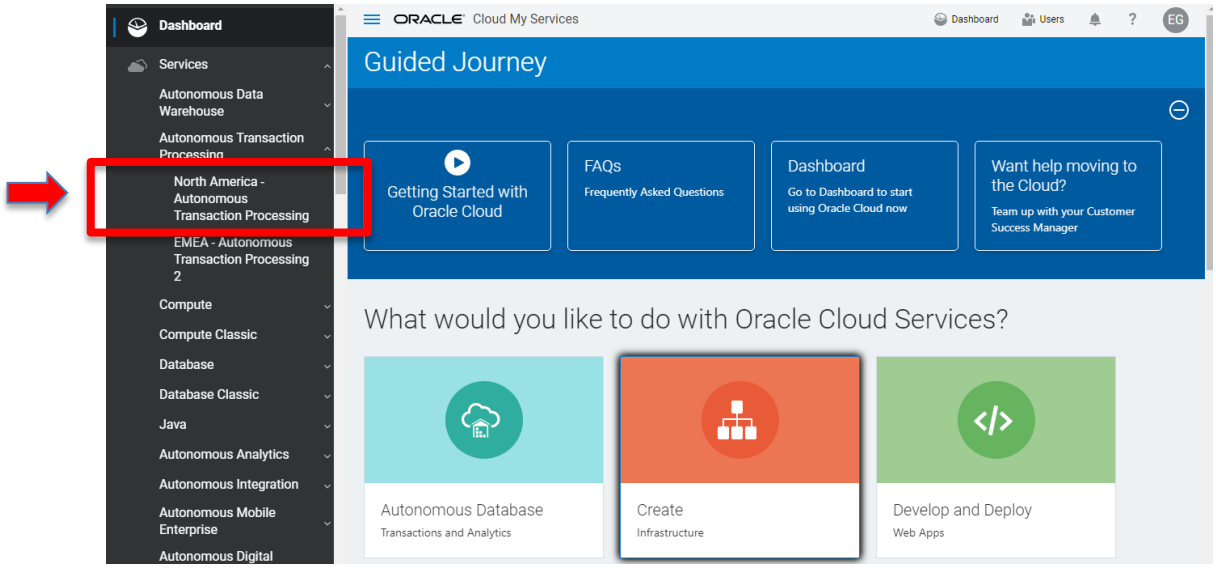
Enter your Cloud username and password, and click Sign In. In this example, the Cloud username is *adw_admin*.



This will bring you to the main cloud page which may differ depending on whether this cloud account has been used before. Below you will see the main screen for a new account, that shows available Guided Journey's. From here you can launch different Oracle cloud services. From the top left select the drop down menu highlighted in red:

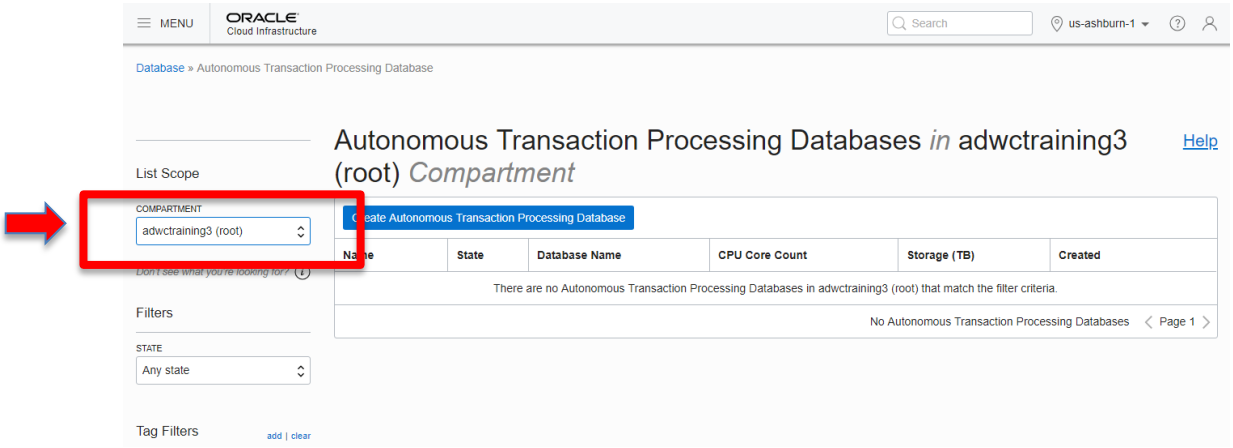


If a datacenter option appears, select **North America – Autonomous Transaction Processing**, under the Autonomous Transaction Processing Menu Item , under the Services Menu Item, highlighted in red (if no datacenter option appears simply choose Autonomous Transaction Processing):

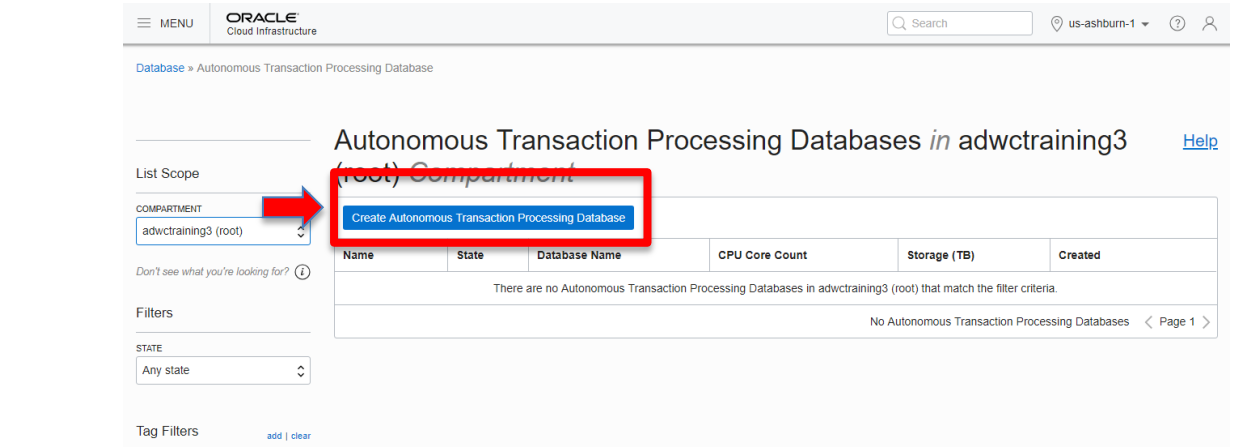


This will put you in the main Autonomous Transaction Processing Service Console (see below). Any ATP Databases created will be listed here. You can also create and access databases from this page.

It is possible to use different compartments to separate databases into different associated groups, we will use the default compartment, so no change just fyi:



Click on the "**Create Autonomous Transaction Processing Database**" button, as shown below:



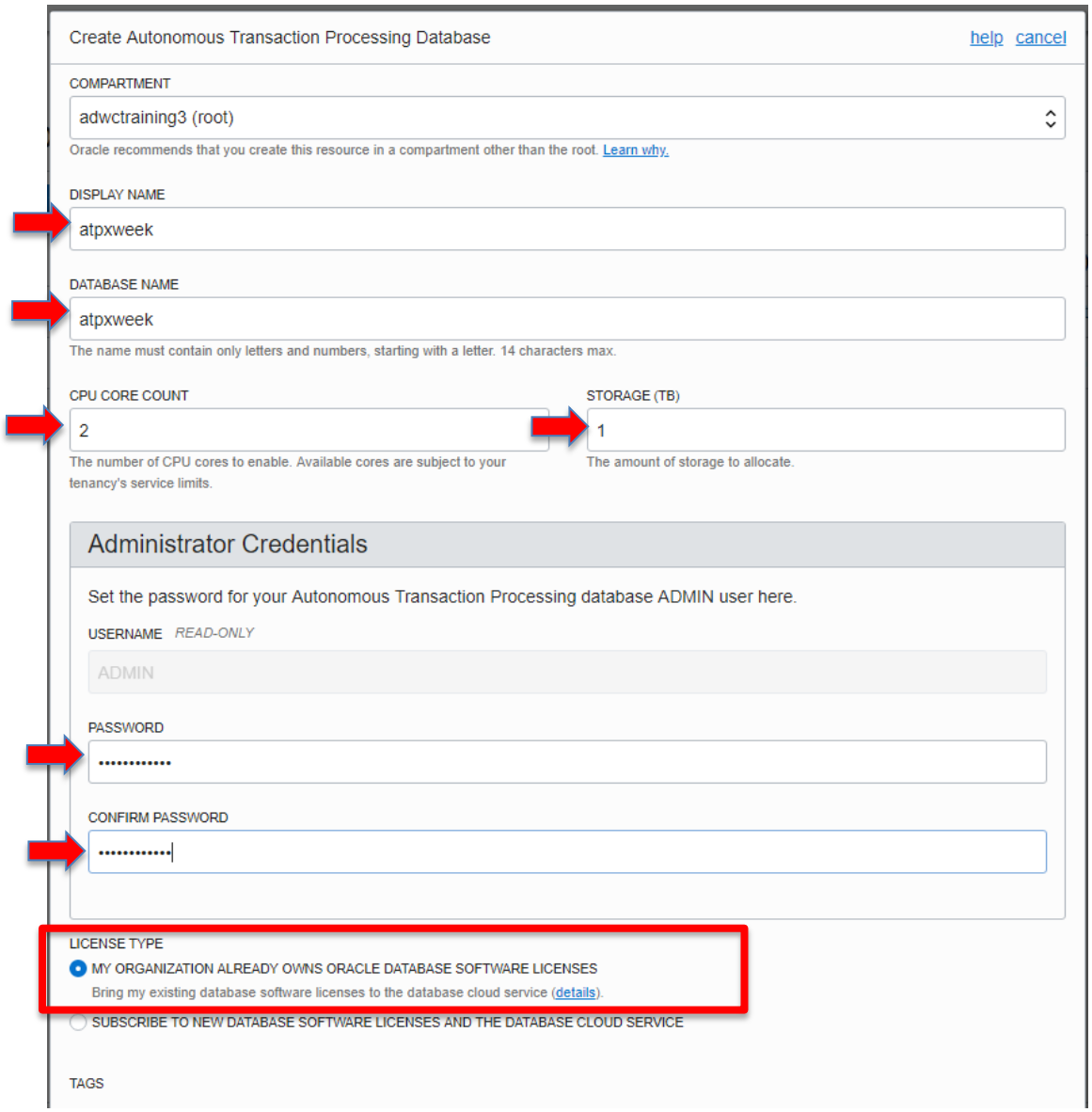
The following information must be filled in this page:

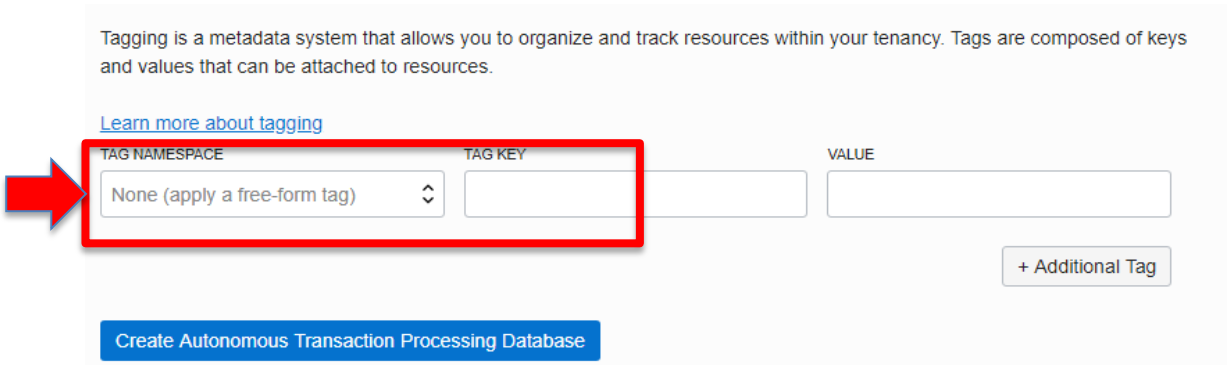
Compartment – This can be changed, to organize and isolate databases
Display Name – The name of the service displayed
Database Name – The name of the actual database
CPU Core Count – Number of CPU's allocated to the database (min 1)
Storage – Storage allocated to database in Terabytes (min 1)
Password – Database "Admin" user password

eugenio.galiano@oracle.com

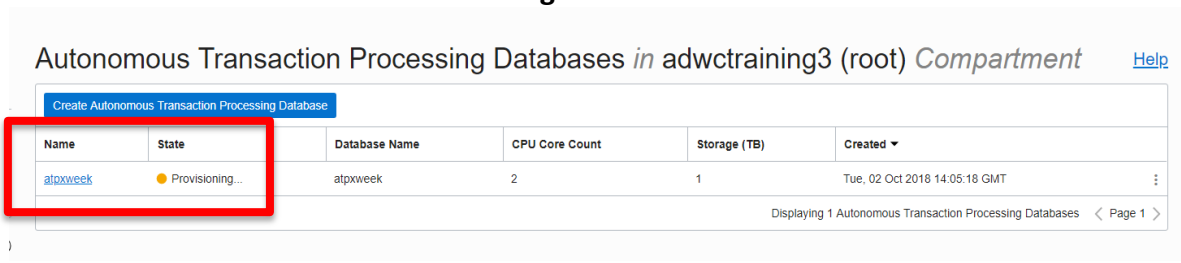
License Type – Select whether customer is using existing on-premises database licenses (BYOL) or requires new licences. Customer charge will be based on selected option
TAGS – can be used for additional security and segmentation

After filling fields, click **Create Autonomous Transaction Processing Database** which will open up the screen to complete you database information:

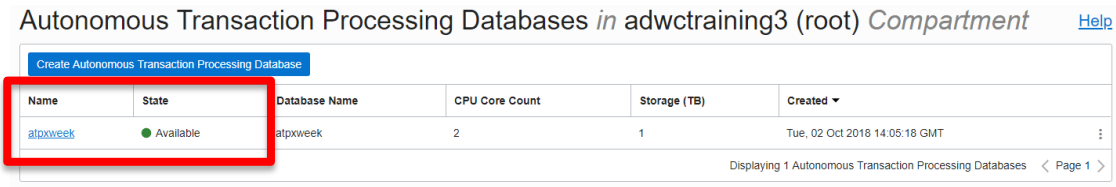




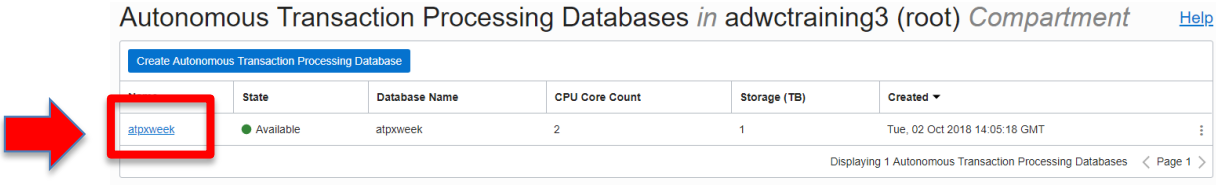
You will see the database in "**Provisioning**" status:



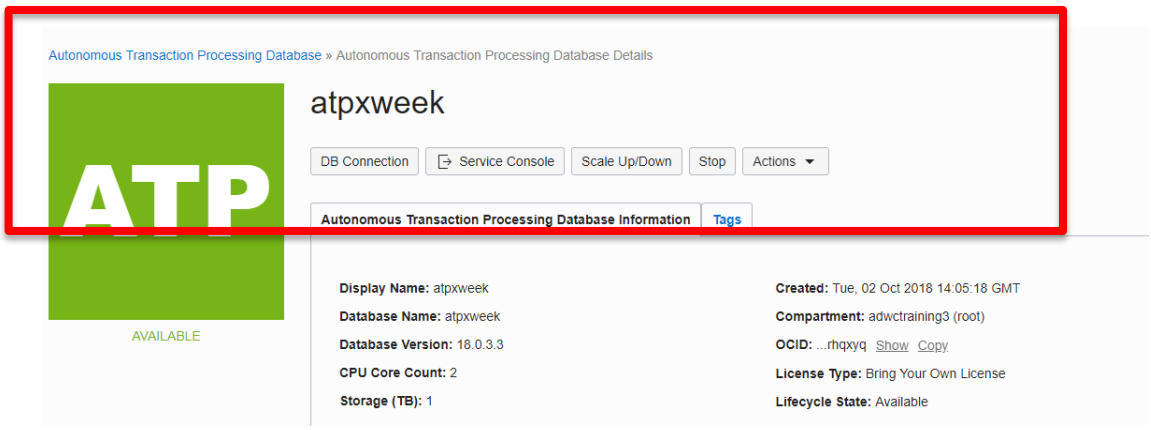
The status will automatically change to "**Available**" when the database is ready in a few minutes):



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

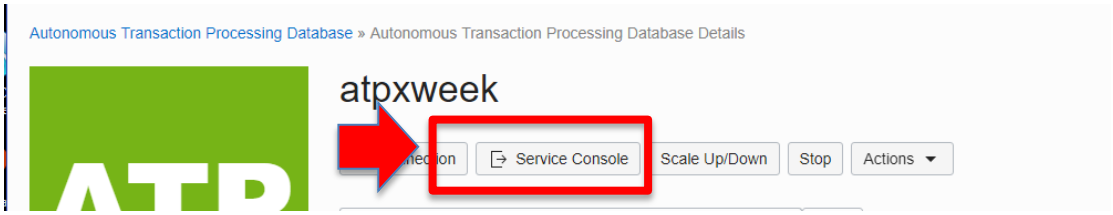


This will display more information about your instance, notice the various menu buttons that help you manage your new instance. Take notice of the green color of the ATP logo indicating the service is available and commands to start, stop, terminate, and scale the service.



Your Autonomous Transaction Processing Database is running!

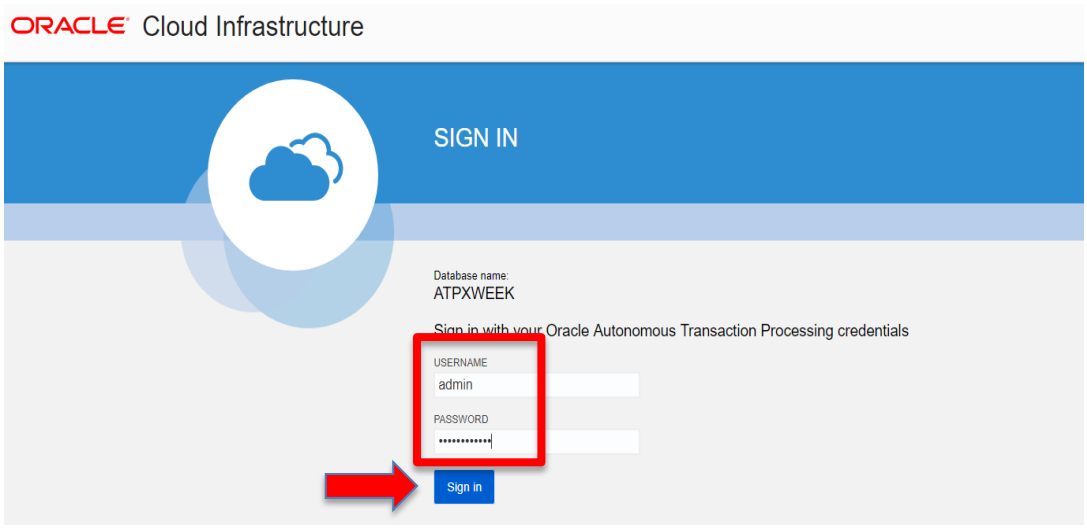
Now connect to your database, click the Service Console button:



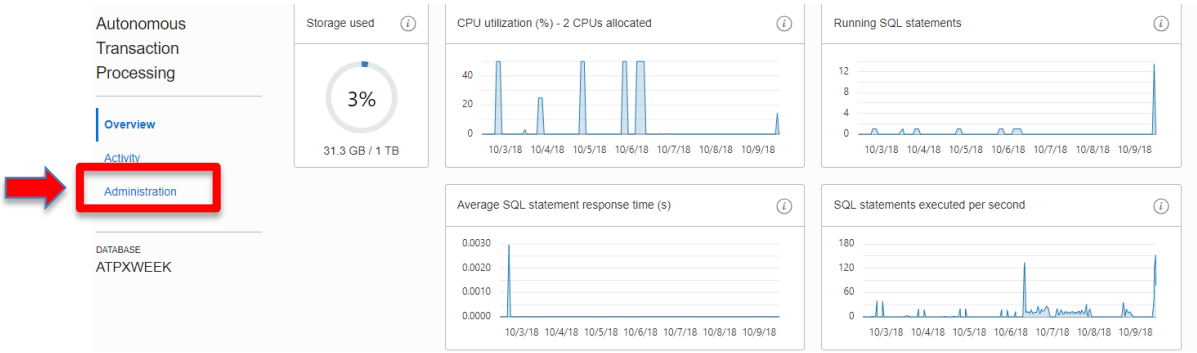
The database is initially created with one user, admin. Fill in the password and select **Sign In**:

Username: **Admin**
Password: **the password you created during the Administration Credentials step**

eugenio.galiano@oracle.com

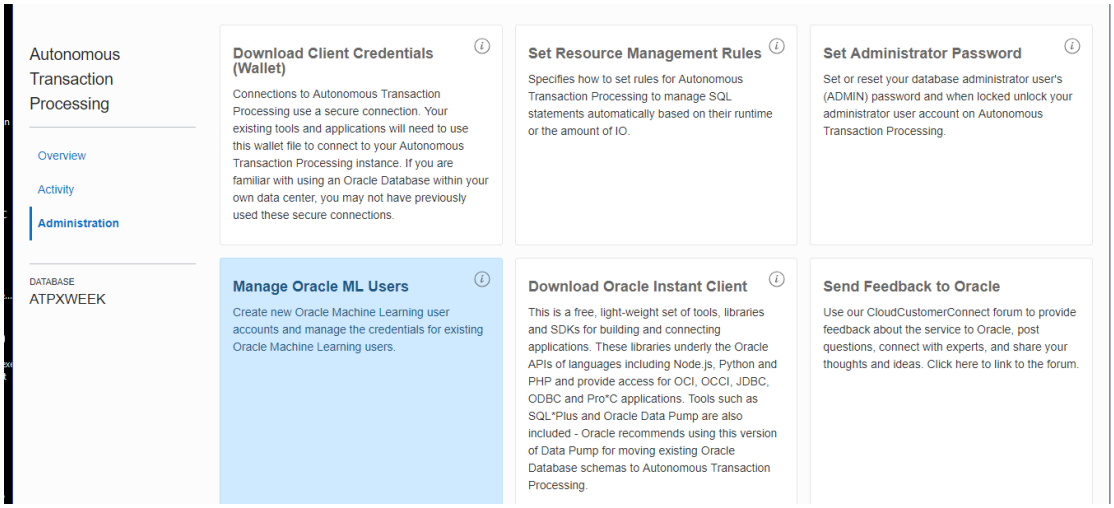


You will be placed in the overview page, you will notice there is no activity displayed because this is a new instance. Select the **Administration** option from the left:



On the administration page there are six options:

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 – ATP has pre-created user resource groups, those can be managed here
Manage Oracle ML Users – Notebook development environment that can be used with ATP
Send Feedback to Oracle – email feedback on ATP



Proceed to the next lab to explore how to connect tools/apps to ATP.

Lab 2. Connecting Tools to ATP

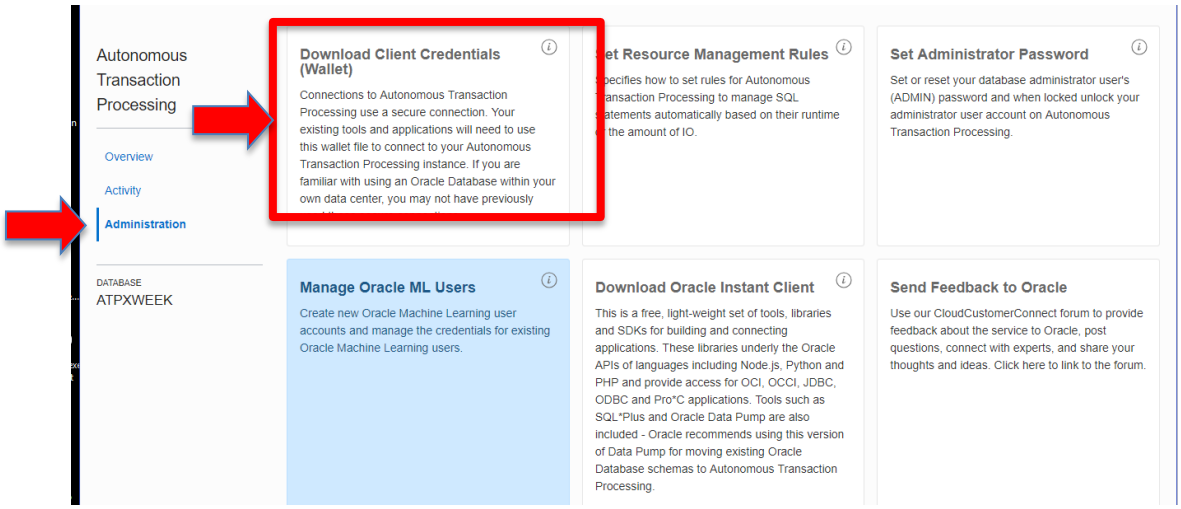
Objectives

- Learn how to download client credentials for your ATP instance
- Explore credentials wallet file information
- Connect to your ATP instance with SQL Developer
- Connect to your ATP instance with Oracle ML Notebooks

2.1 Downloading the Client Credentials (connection wallet)

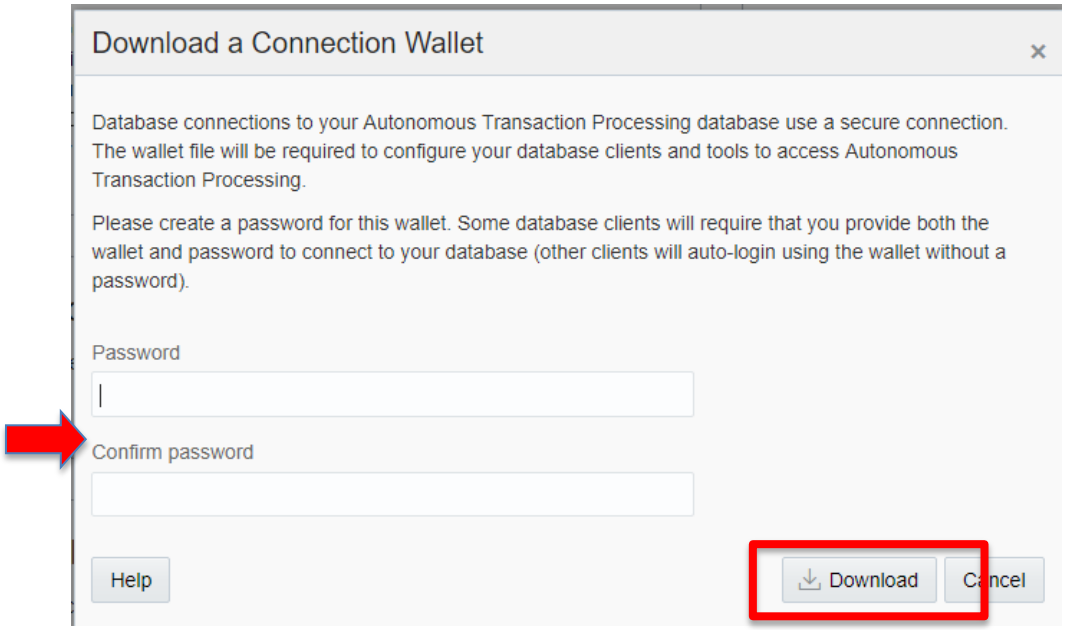
The connection wallet provides the only authentication information that can be used to connect to your ATP database. This wallet must be downloaded to the client that will be connecting to the database.

The wallet is downloaded from the **Administration** page in the ATP service console. Continuing from where we left off in the previous lab, select **Download Client Credentials:**

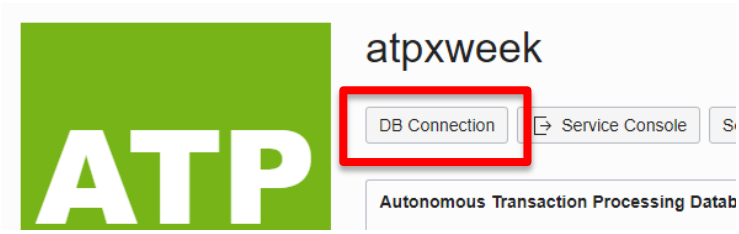


Specify a password of your choice for the wallet. Note that this password is separate from the **Admin** user password created earlier (but the same password can be used). Make sure you know where the file gets downloaded to so you can find in on your system.

eugenio.galiano@oracle.com

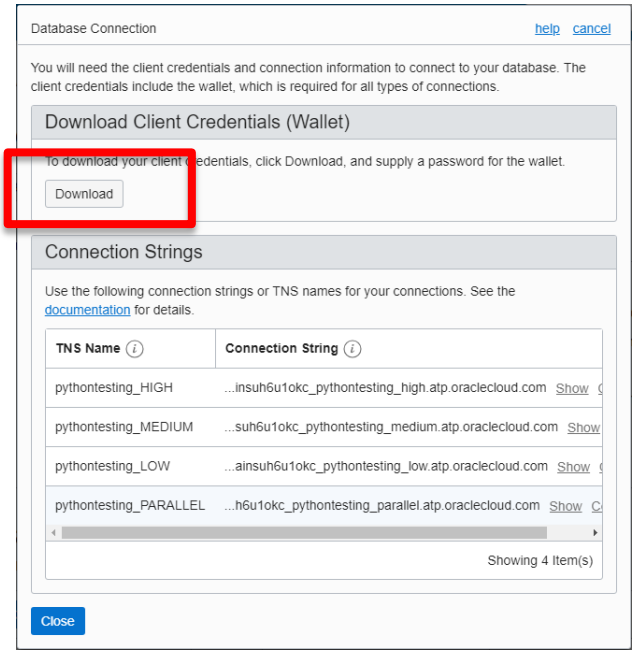


Alternatively use the DB Connection button in the main ATP Service screen to download the credentials:



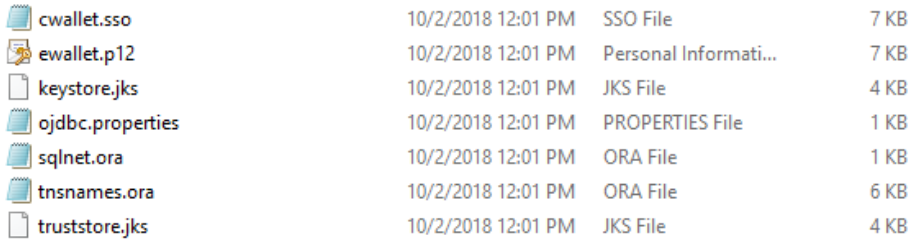
Which opens up a screen where the Wallet can be downloaded as well as connection strings be copied from (more on this in the next lab)

eugenio.galiano@oracle.com



Examining the Wallet File

Navigate to the location in your system where the file was downloaded (typically your DOWNLOADS directory). In this example the file will be called wallet_ATPXWEEK.zip, 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:



There are 4 files you will be working with during the different labs. Some tools use the wallet file (.zip) and some use specific files contained in the wallet. The files used in the labs are:

1. **wallet_ATPXWEEK.zip** – the wallet itself
2. **sqlnet.ora** – points to the location of the wallet for sqlnet connections
3. **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)

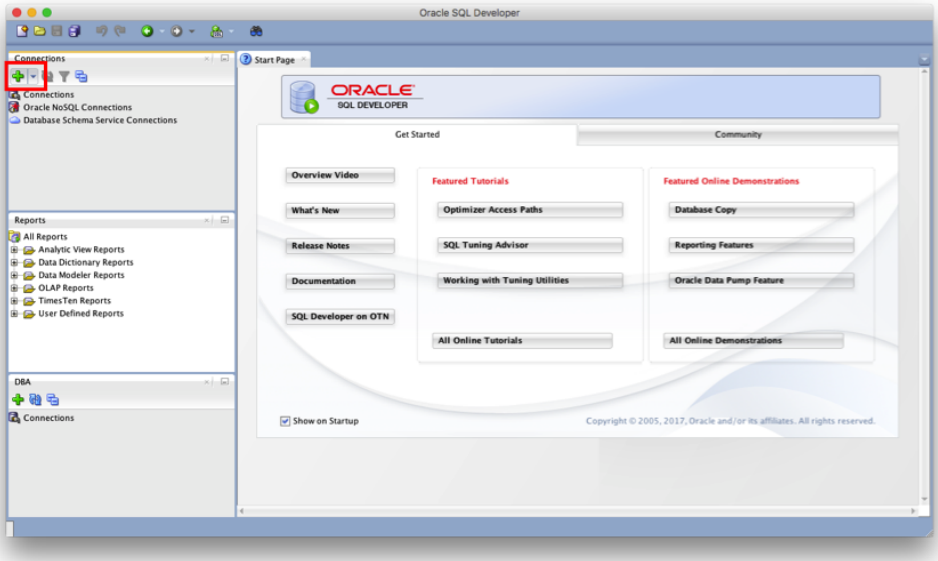
4. **ojdbc.properties** – points to the location of the wallet for jdbc connections

2.2 Connecting to the database using SQL Developer

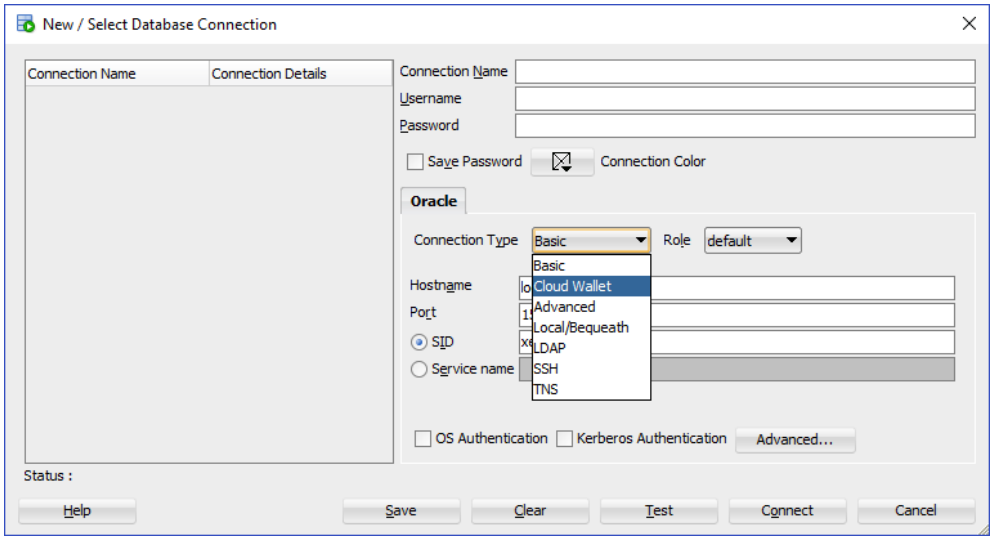
(Make sure you are running the latest version of SQL Developer 18.3 or newer, older versions will not work with ATP, see lab introductions on how to install)

Start SQL Developer (by clicking the icon on your desktop or selecting from the Windows Start menu) and create a connection for your database using the default administrator account, ADMIN, by following these steps.

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

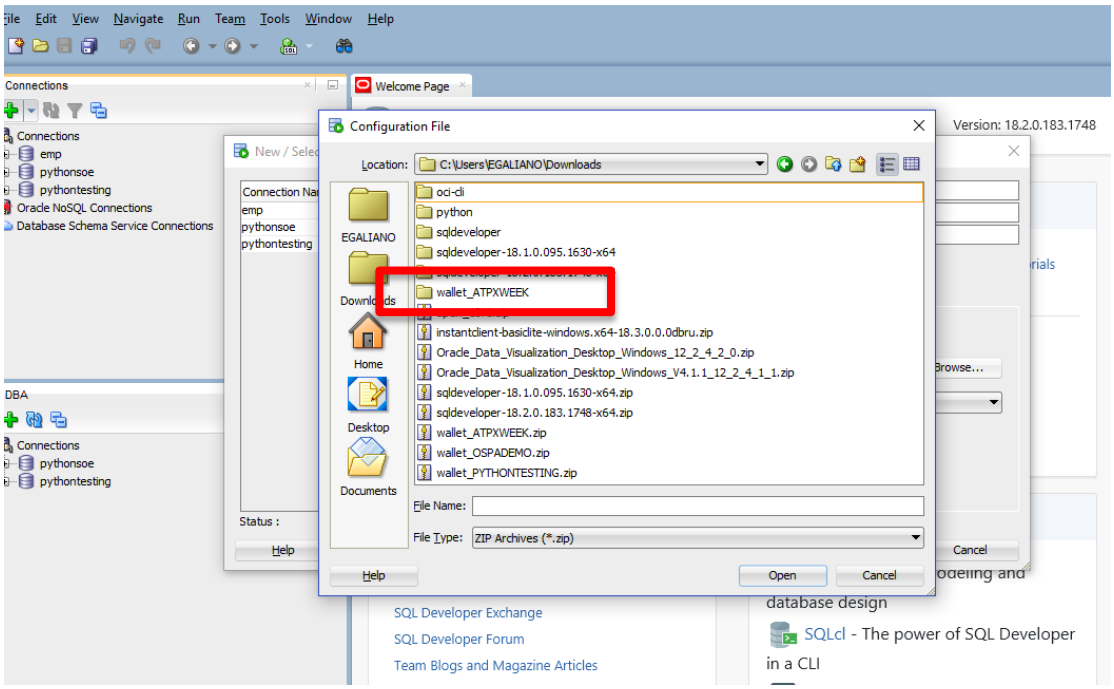


The new Database Connection screen will appear:



Fill in the connection details as below:

Connection Name: **xweek_admin**
Username: **admin**
Password: **The password you specified during provisioning**
Connection Type: **Cloud Wallet**
Configuration File: **Enter the full path for the wallet file you downloaded before (in my example wallet_ATPXWEEK.zip), or click the Browse button to point to the location of the file.**



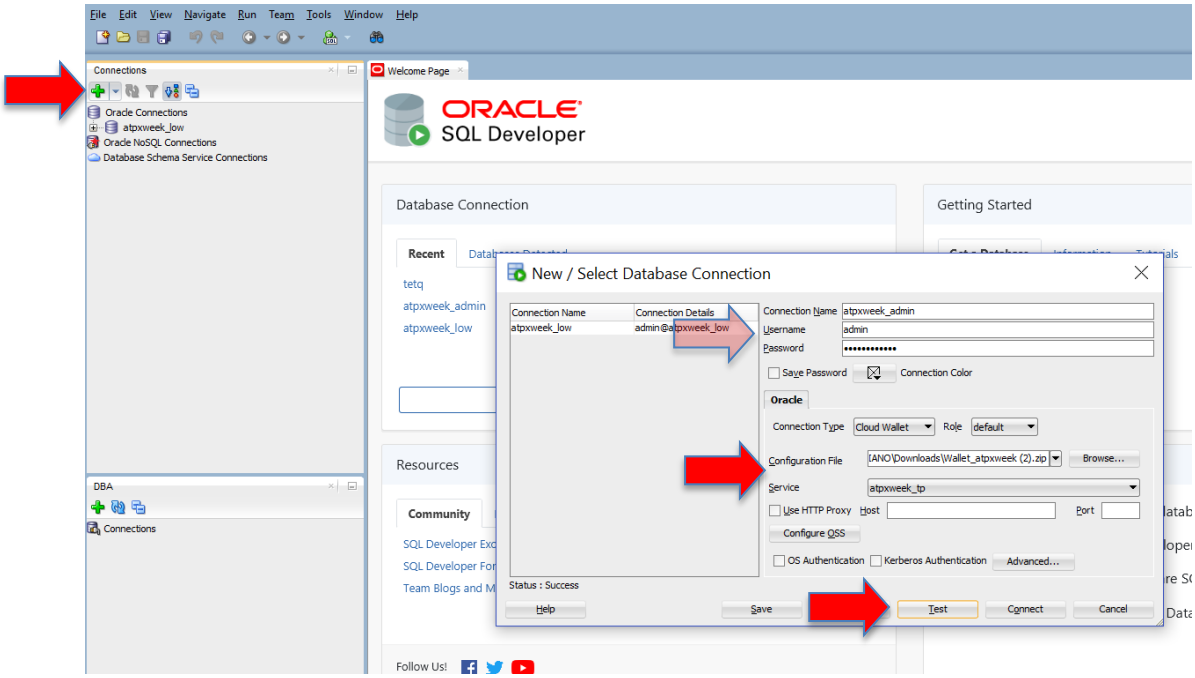
Service: select the service configured specifically for ATP services, the **$dbname_TP** service, for you database. Many services may be listed but make sure you pick the one for with the database name you created. In this example its **atpxweek_TP.**

Test your connection by clicking the **Test** button, if it succeeds

Save your connection information by clicking **Save**

Connect to your database by clicking the **Connect** button.

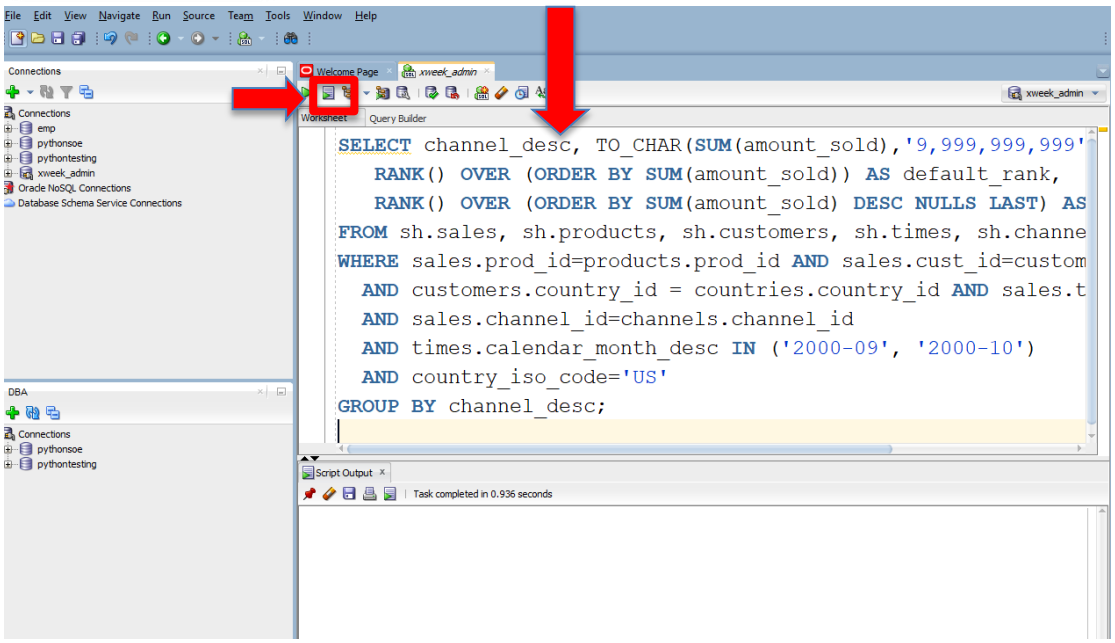
See below for completed input and test. Notice also that after you save your connection it will appear on the list of connections on the top left corner of the main dashboard, under connections.



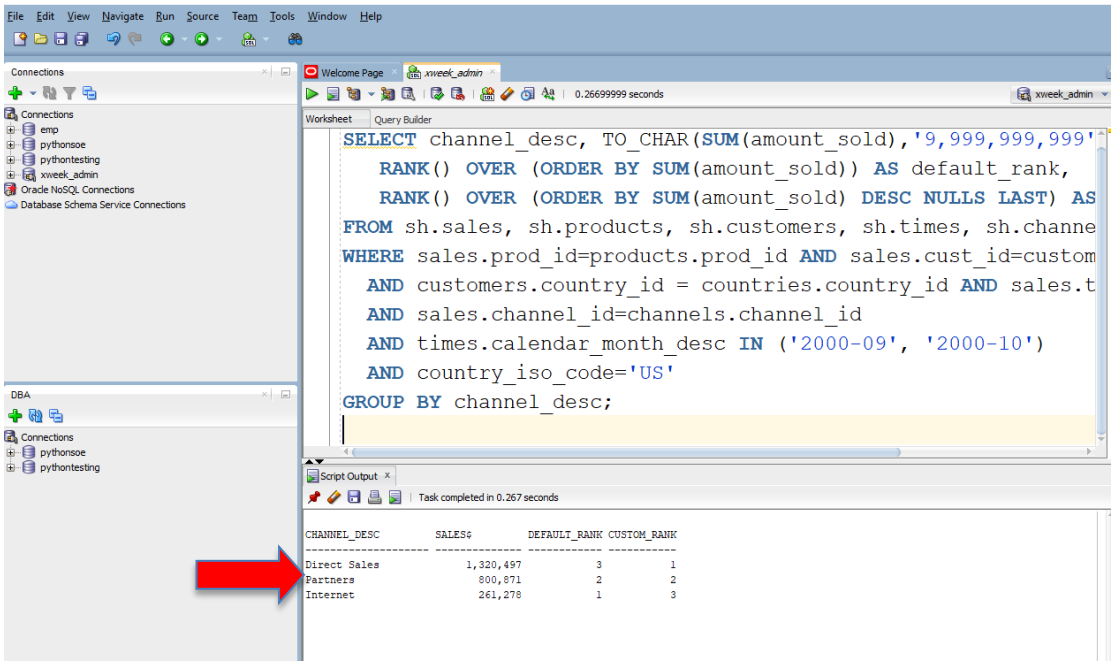
Now that you are connected run test a query. The ATP database you created contains the sample Oracle Sales History (SH) schema, we will use this schema to run a test query to make sure everything is working correctly. Copy the SQL below and paste it on the query builder screen in SQL Developer (with standard Windows copy and paste), then click F5 or the "Run Script" button as indicated below. Make sure you are connected to your database, per the last step on the previous process.

```
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;
```


eugenio.galiano@oracle.com



And you will see the result of your query on the bottom **Script Output** section



You have successfully connected and run an operation against ATP with SQL Developer. We will use SQL Developer throughout other labs.

eugenio.galiano@oracle.com

Now we will a different utility that is included with the ATP service.

2.3 Connecting to the database using OML

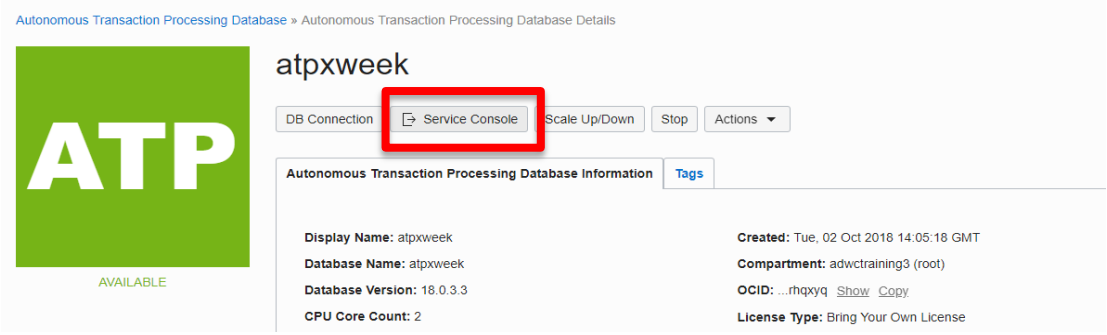
Another tool that can be used to connect and develop in ATP is the included Oracle Machine Learning OML Notebook based environment. Because OML is easy to access from anywhere and included with the ATP service it provides an easy and fast environment to work with ATP. This browser-based application provides a web interface to run SQL queries and scripts, which can be grouped together within a notebook. Notebooks can be used to build single reports, collections of reports and dashboards. OML provides a simple way to share workbooks with other OML users.

OML Key Features

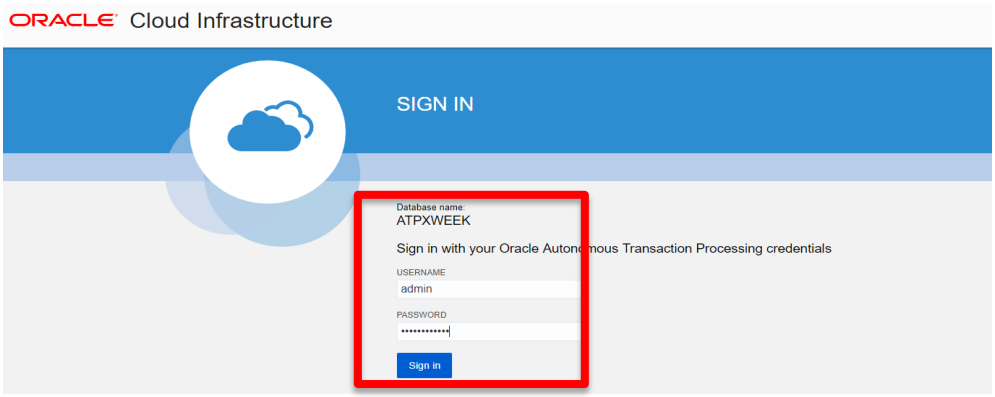
- Collaborative SQL notebook UI for data scientists
- Packaged with Oracle Autonomous Transaction Processing Cloud Service
- Easy access to shared notebooks, templates, permissions, scheduler, etc.
- Access to 30+ parallel, scalable in-database implementations of machine learning algorithms
- SQL and PL/SQL scripting language support
- Enables and Supports Deployments of Enterprise Machine Learning Methodologies.

Once you have a database created in ATP, we need to create an OML user, which is equivalent to creating a database user.

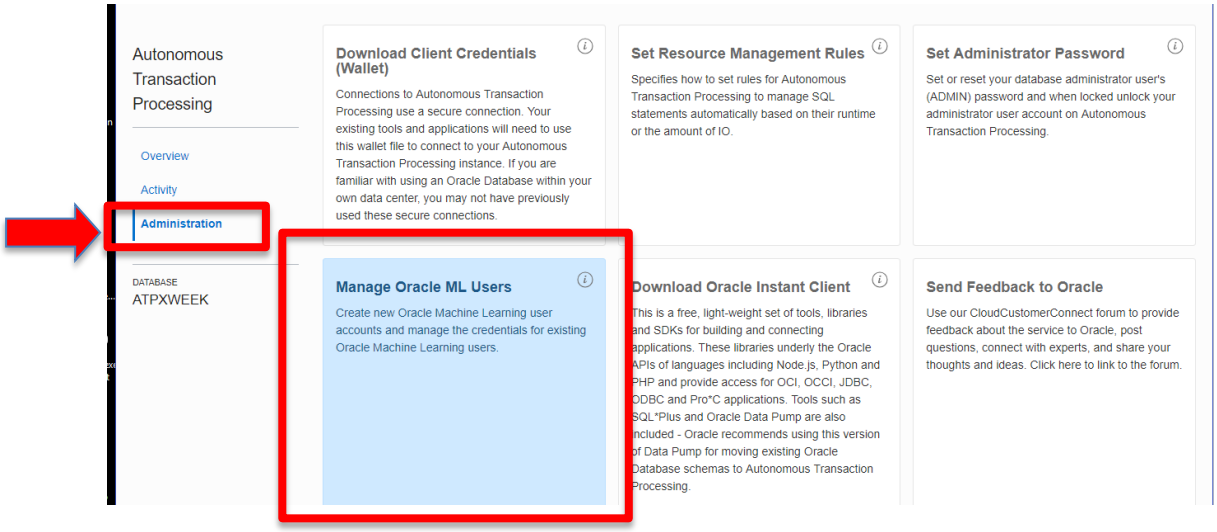
If you are not already logged into the ATP Service Console, in the main ATP service page select Service Console:



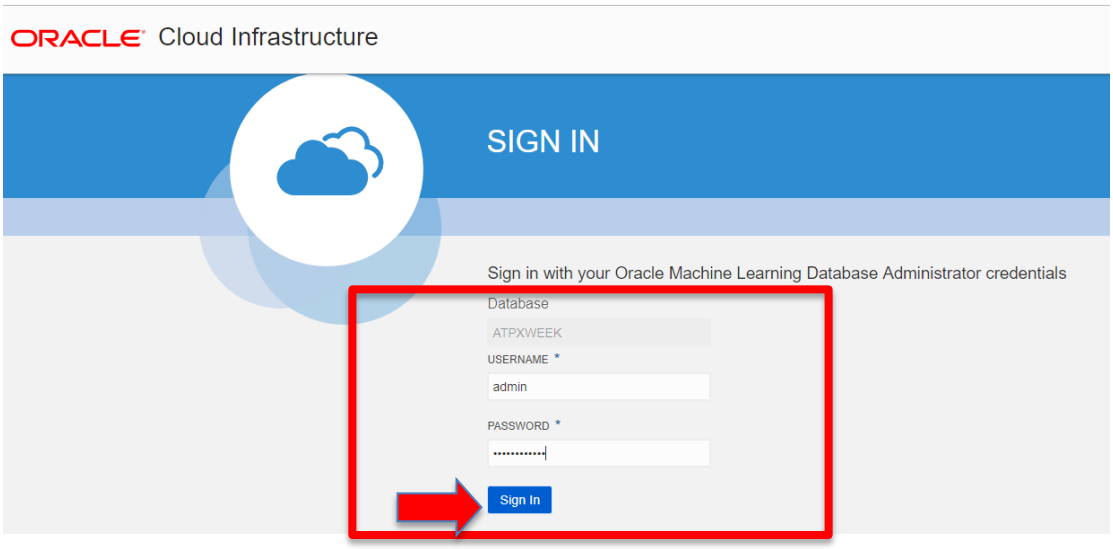
On the next page log in with your ADMIN ATP user name/password and click **Sign in:**



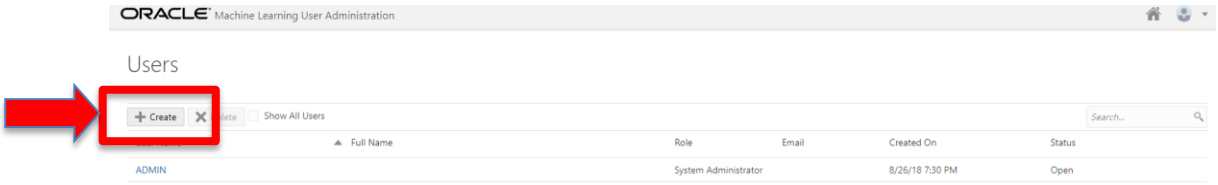
Select Administration from the top left and once on the Administration page select **Manage Oracle ML Users**:



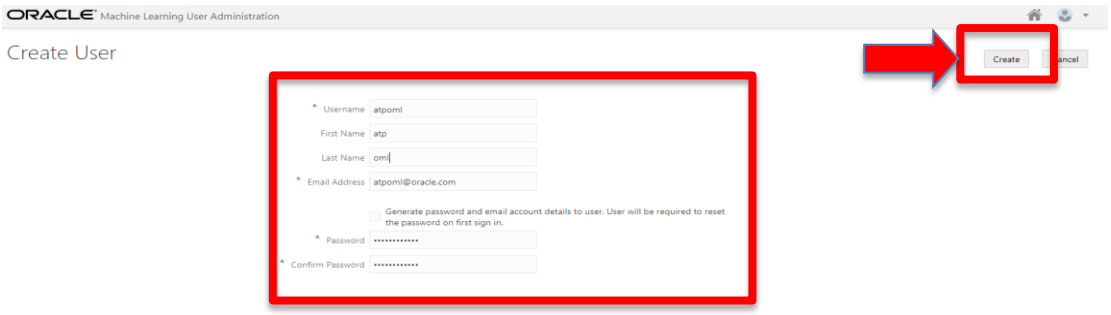
Log into the OML Administration console which is different than the database administration console but uses the same ADMIN account created when the database was created. Fill in the **ADMIN password** and click **Sign In**



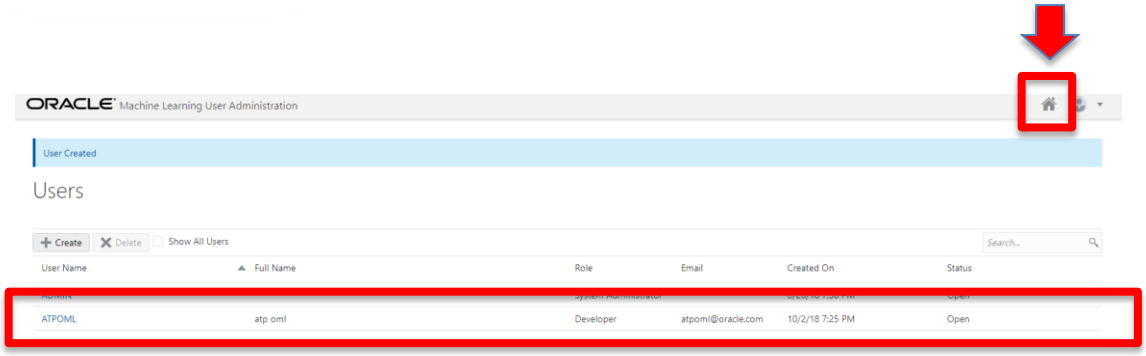
Next create the actual OML user. Click the **Create** button:

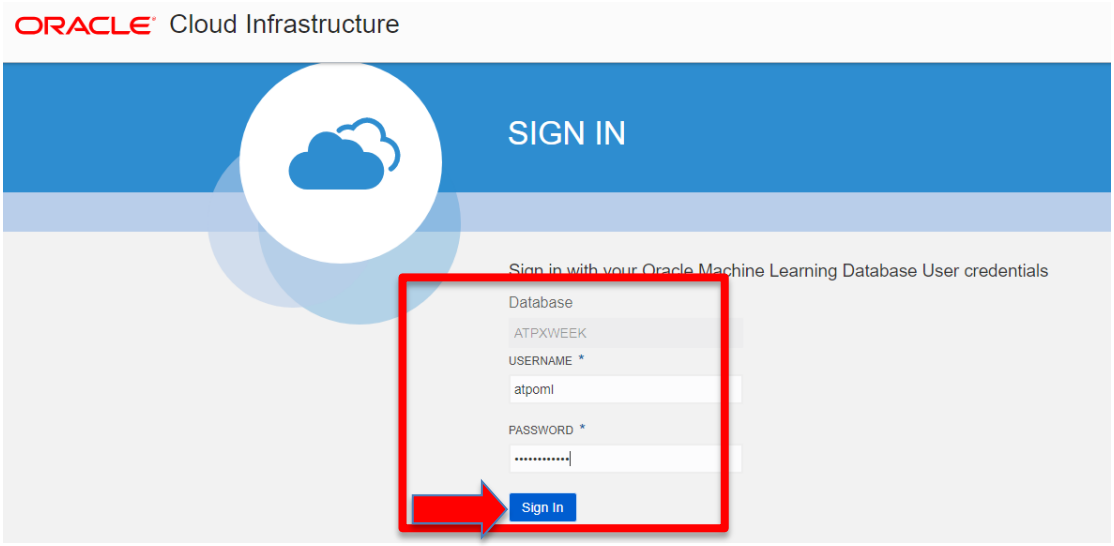


This will open up the user creation page, fill in the information for your new OML user and click **Create**. This is a completely new user account that will be used anytime you want to access OML. Make sure you keep this information. Notice that you can specify an email address where your user information and a direct link to the OML login will be emailed to you. This will help you later when you need to reconnect to OML.

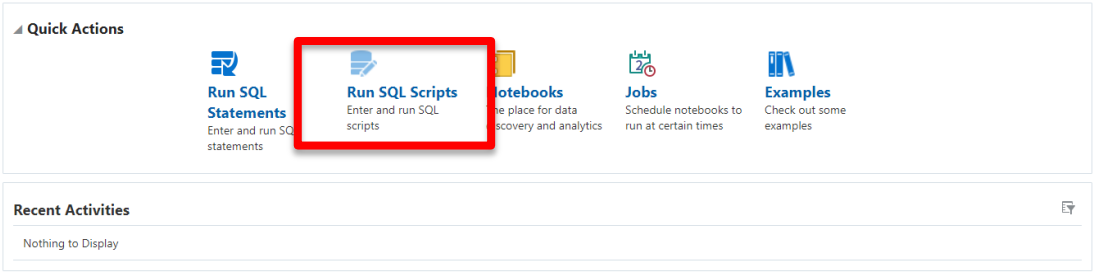


You now have a new OML user! To connect to OML as your new user, click on the Home Icon on the top right, pointed at by the arrow (or the link you received by email). This will open up a new tab with the OML home page. This time log in with the user you just created.





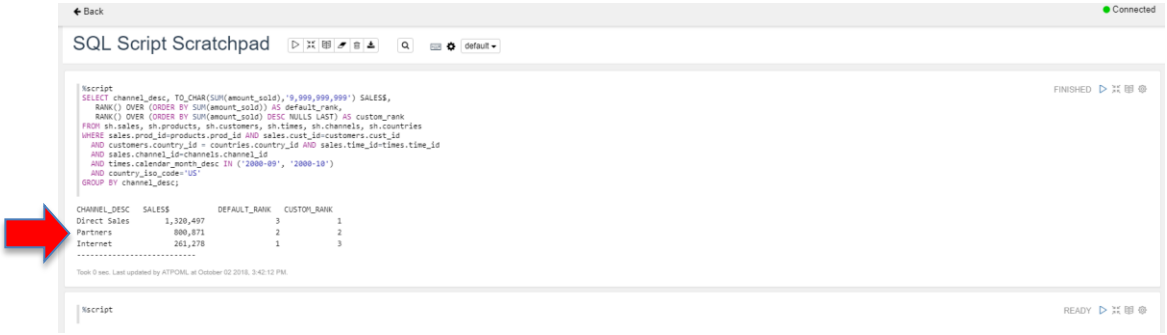
You are connected as an OML Notebook user. Run the same query we ran in SQL Developer now in OML. Select **Run SQL Scripts** from Quick Actions:



Copy the same SQL statement you ran in SQL Developer above and paste it right under the **%script** statement then select the **Run all Paragraphs** icon, as shown below:

eugenio.galiano@oracle.com

The results are shown below (and same as on SQL Developer):



You have successfully connected and run an operation against ATP with Oracle OML. We will use OML in other labs.

eugenio.galiano@oracle.com

Lab 3. ATP Consumer Groups and Workloads

Objectives

- Learn about consumer groups in ATP
- Using different consumer groups to test resource utilization
- Scaling the system up and down

3.1 Managing Priorities on Autonomous Transaction Processing

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_tpurgent
- database_name_tp
- database_name_low
- database_name_medium
- database_name_high

These services map to the LOW, MEDIUM, HIGH, TP and TPURGENT consumer groups. In our example, the defined services are:

- atpxweek _tpurgent
- atpxweek _tp
- atpxweek_low
- atpxweek _medium
- atpxweek _high

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

- **tpurgent**: The highest priority application connection service for time critical transaction processing operations. This connection service supports manual parallelism.
- **tp**: A typical application connection service for transaction processing operations. This connection service does not run with parallelism.

eugenio.galiano@oracle.com

- **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.

As a user you need to pick the database service based on your performance and concurrency requirements.

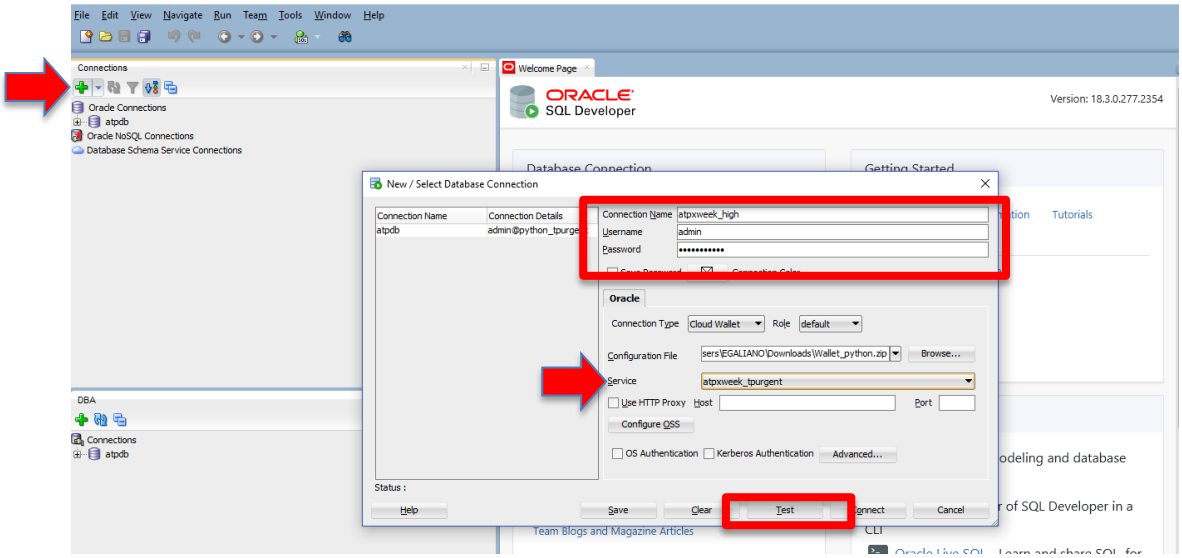
In this section you will use the TP and TPURGENT database services to understand the performance differences between them. For this exercise you will run queries on the SSB schema sample data set provided in your ATP database. You will run the queries first on the TP service and then on the HIGH service. Since we established connections with both SQL Developer and Oracle ML you will test both (please note this is a long running query when only 1 or 2 cpu's are allocated to the environment, it may take up to 20 minutes to run).

```
Select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25;
```

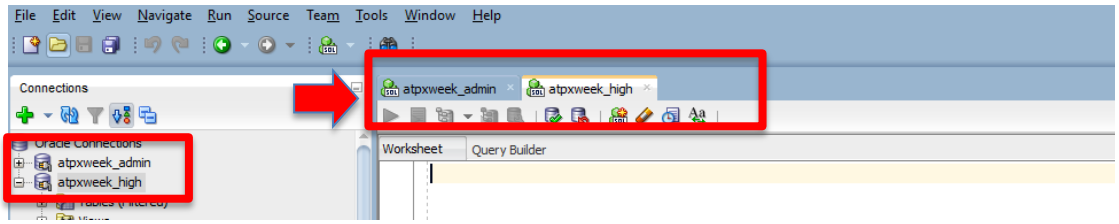
The expectation is that you will see parallelization and therefore faster execution with the HIGH service vs. the non-parallelized execution in the TP service. This is dependent on the number of CPU's allocated to the database and other activity that may be running concurrently.

3.2 Running the Queries in SQL Developer

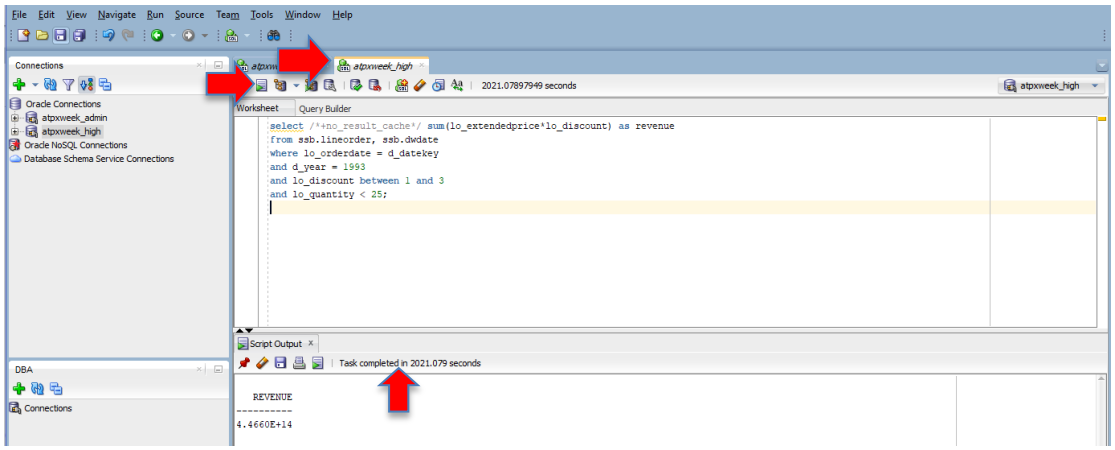
In the previous lab we already created a connection to the **atpxweek_tp** (xweek_admin connection) service. We need to create a new connection to the **atpxweek_high** service to be able to run our test using both services. Please revisit steps above on how to create a connection in SQL Developer. Below you will see the completed form. Name your service **xweek_high** (or your database name_high) and **for Service select atpxweek_tpurgent** (or the _tpurgent service for your database name, DO NOT select the _high service)**.** Remember to use your original admin account and password. **Test, save, and connect** to new service.



You will end up with two connected connections in the Worksheet the original and LOW connections:

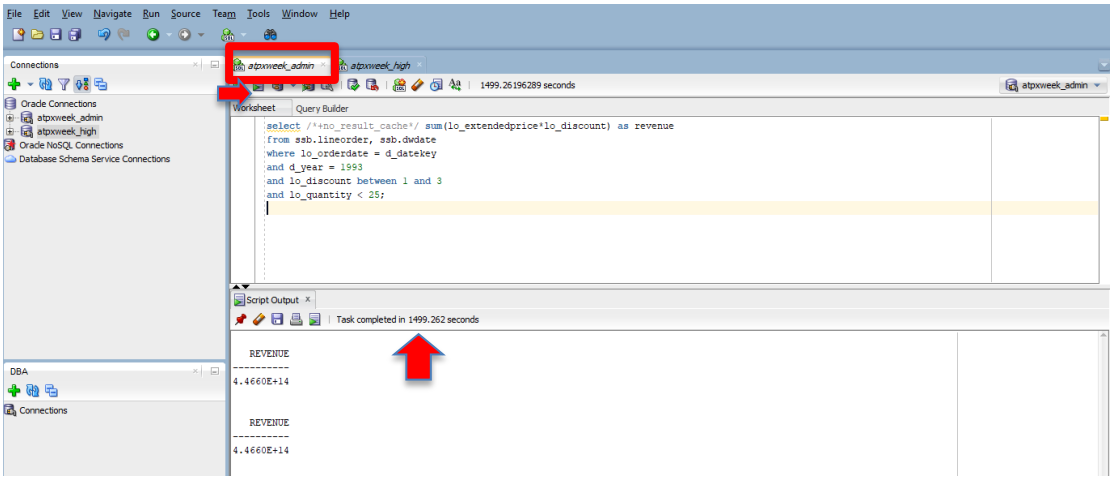


You should be connected to the just created **xweek_high** connection, if not click on the tab labelled xweek_high in the worksheet to connect (see above). Copy and paste the query above into the Query Builder and select **Run Script (or F5):**



You will see the result in the **Script Output** (bottom right square) including the query execution time. In this case the query executed in 492.531 seconds.

Now run the query in the original service (_TP). Do that by clicking on the tab with the original connection (in this case xweek_admin) which will cause SQL Developer to connect to your ATP database with that service. Then paste the same query on the Query Builder and run it. If you already have a query in the Query Builder from a previous lab you can you can **erase it by hitting Ctrl-D or the eraser icon** in the same bar as the Run Query icon

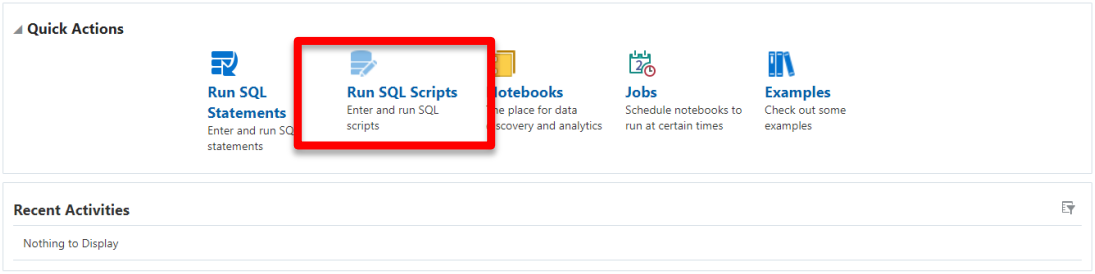


Notice the difference in timings between your runs. One service will parallelize operations and the other one will run it with no parallelization. Depending on how many cpu's you have allocated to your environment and other workloads running on the system you will see different results. In another lab you will learn how to inspect explain plans to determine if operations were parallelized or ran sequentially.

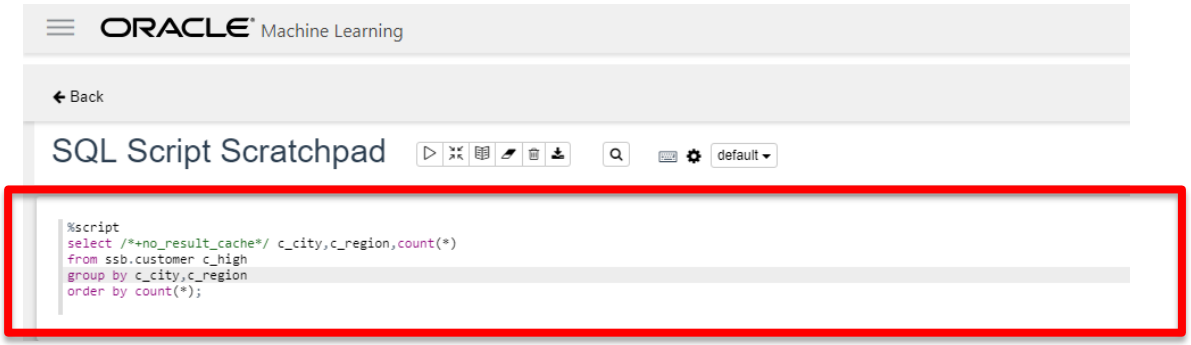
3.3 Running the Queries in OML

Now run the same queries in OML experimenting with different services. OML makes it easier to do this because Notebooks automatically "see" all services and it's easy to prioritize which to use. Notebooks also allow re-running the same query multiple times without having to make any changes or re-posting the query.
Your OML session should still be open from the previous lab, if not go back through the previous OML lab and skip the part where you create a user and simply log in as your OML user (you can also use the link in the email you received when creating your OML user).

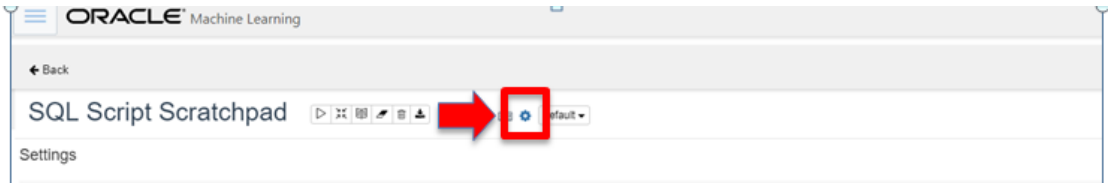
If not already in **Run SQL Scripts,** select it from Quick Actions:



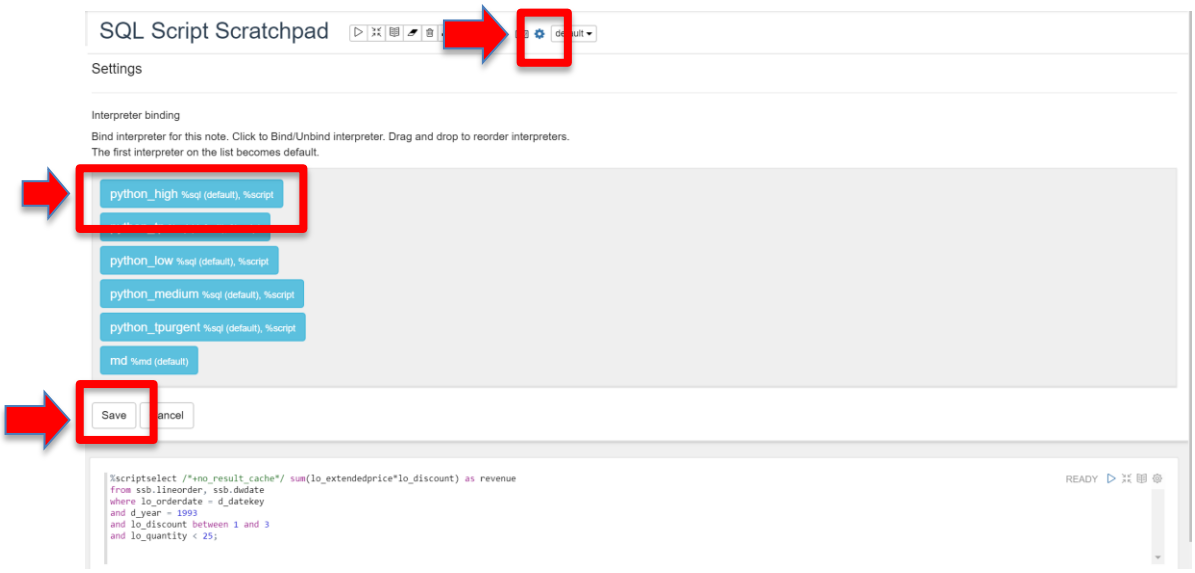
After selecting paste the same SQL you just ran on SQL Developer in a new %script line in the SQL Script Scratchpad:



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



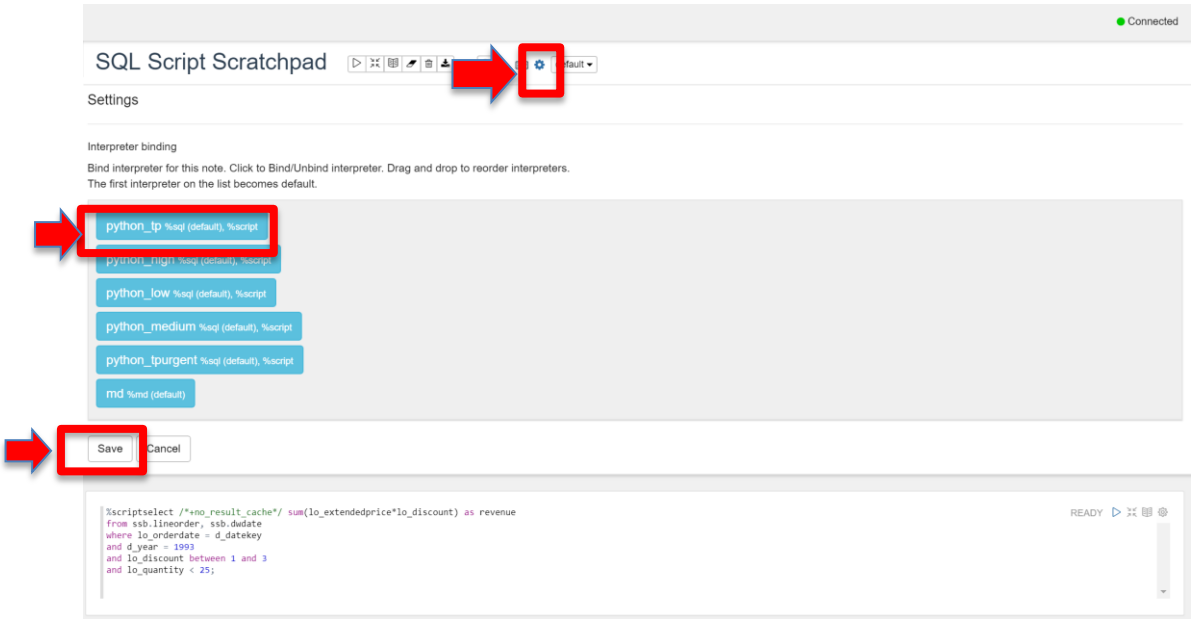
Please notice that a new window opens listing all the services available to run queries. You will see the five services already discussed previously and a new one specific to OML (md). This is a list of services that will be used in order of appearance. So the first service OML can connect to going down the list will be used. A blue color indicates the service is selected, if you click on any of the services it will turn white, which de-selects it. You can drag and drop any service up or down the list to specify the order in which they will be attempted.
For the first test we will use the **_TPURGENT** service so if it is not on the top of the list drag and drop it on top of the list. Select **Save** when done.



Now run the query by selecting the **Run This Paragraph** button (highlighted below) and notice the execution time at the bottom of the results set. Run the same query several times by selecting the **Run This Paragraph** button again. The first time running the query OML establishes a connection that will be reflected in the total execution time, which could be several seconds. With a short query this will substantially add to the total execution time. With our long query the connection time is a small portion of the overall execution time.

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

eugenio.galiano@oracle.com



Now run the exact same query as before by selecting the **Run This Paragraph** button.

Notice the difference in timings between your runs. One service will parallelize operations and the other one will run it with no parallelization. Depending on how many cpu's you have allocated to your environment and other workloads running on the system you will see different results. In another lab you will learn how to inspect explain plans to determine if operations were parallelized or ran sequentially.

Congratulations! You have experimented with the basics of services and resources in ATP and observed the impact they have in execution and scaling of transactions.

Lab 4. Scaling, Performance, and Monitoring with the ATP Console

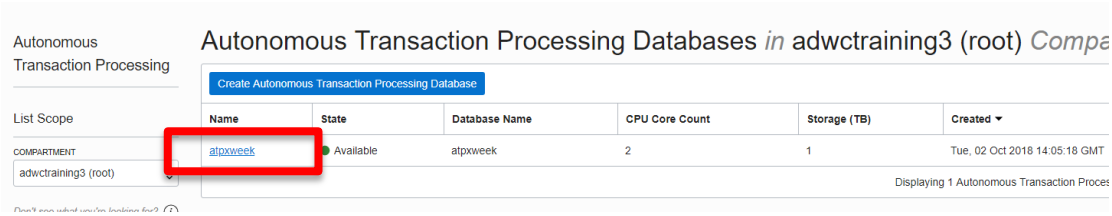
Objectives

- Learn how to scale your ADWC instance
- Learn how to use Cloud Management Console to monitor activity

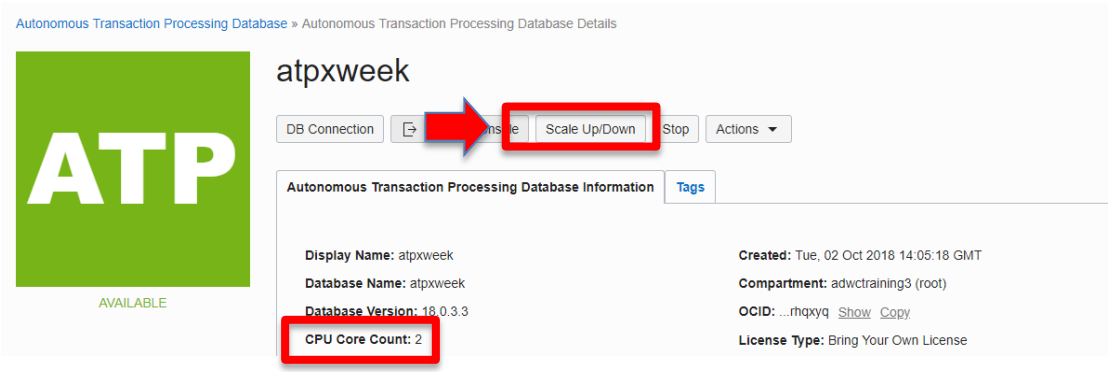
In this section you will scale your ATP instance by increasing (and decreasing) the number of CPU's allocated to the instance. Using the same queries used in the last lab you will explore the impact of allocating and removing CPU's from the instance.

4.1 Scaling the ATP instance

From the Cloud Console you used during the provisioning exercise select your ATP instance:

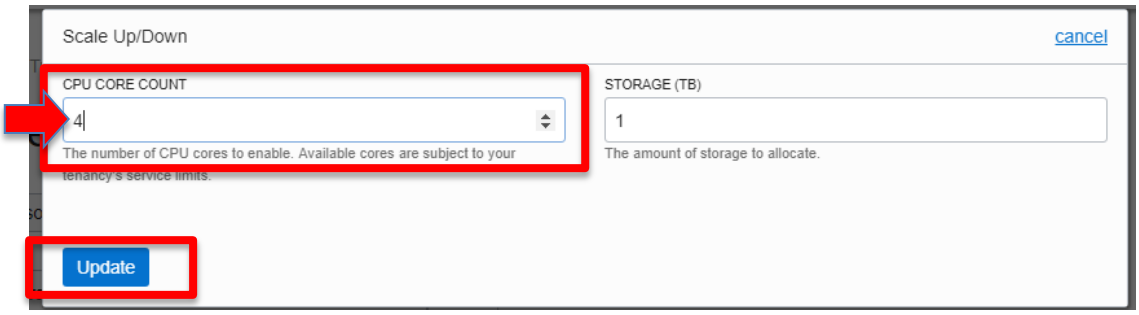


Notice the example currently has 2 CPU's allocated to it. Grow the system to a total of 4 CPU's by selecting the **Scale Up/Down** button.

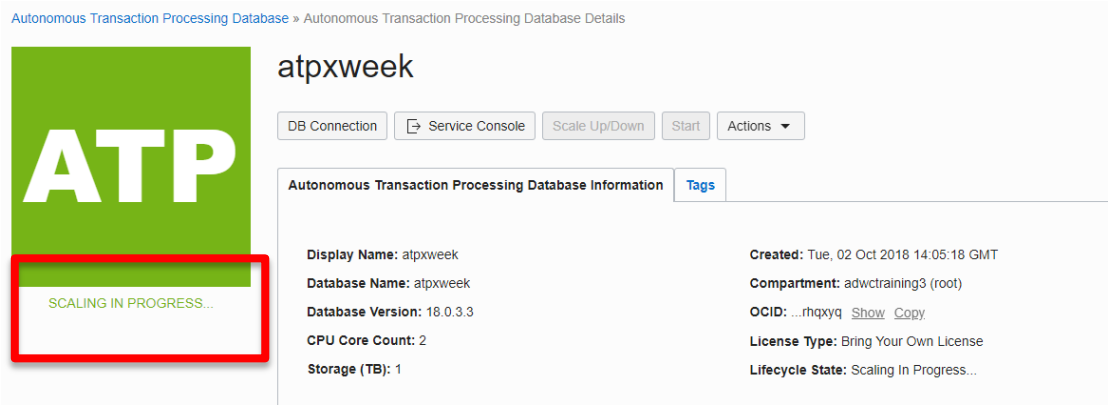


eugenio.galiano@oracle.com

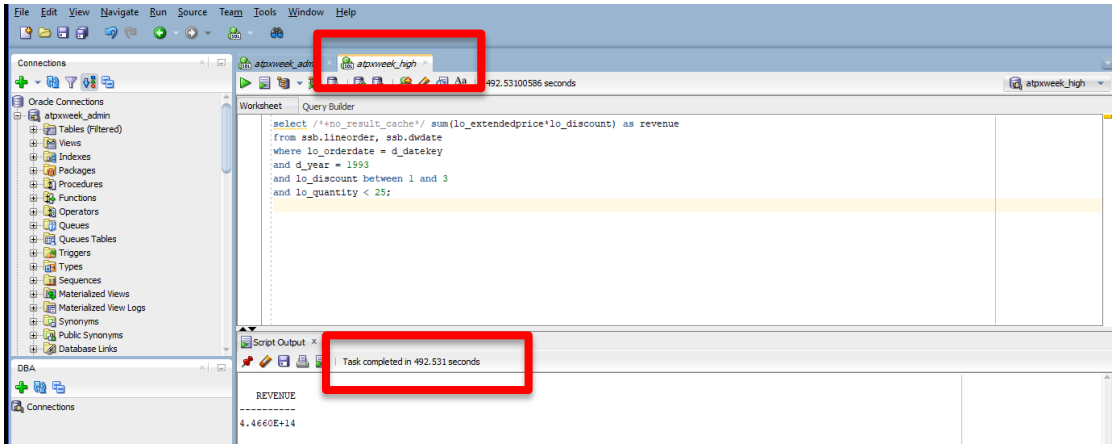
In the **Scale Up/Down screen**, enter **4** in the CPU CORE COUNT, leave STORAGE unchanged and select **Update**



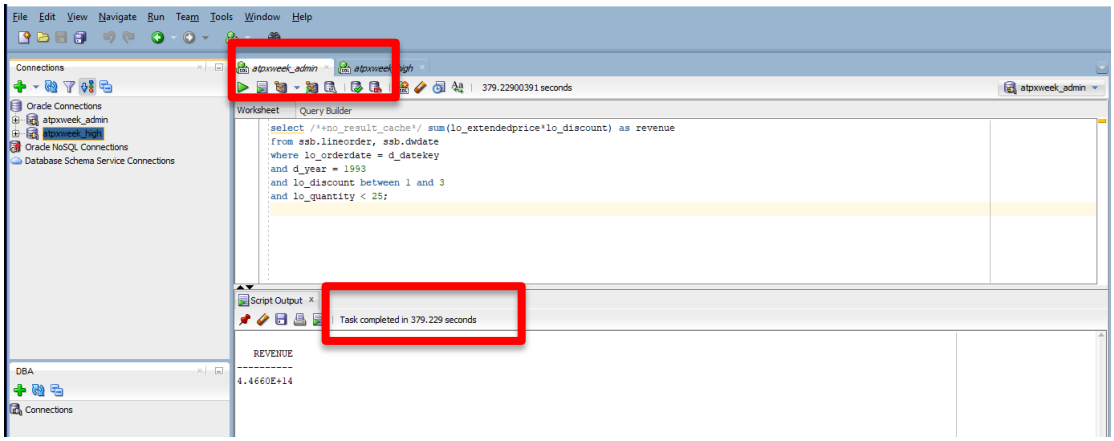
This will return to the database details page and the status will now show "**Scaling in progress**"



The page will automatically refresh once the scale operation has completed and the CPU Core Count will update to "4".
Note that the "ATP" square remains green during the scaling process. No interruption to the service occurs during scaling operations.
To test the impact of the additional CPU's, re-run the queries in SQL Developer that you ran in Lab 3, both in the **_high and _tp**. Please refer back to Lab 3 for steps on running the queries.

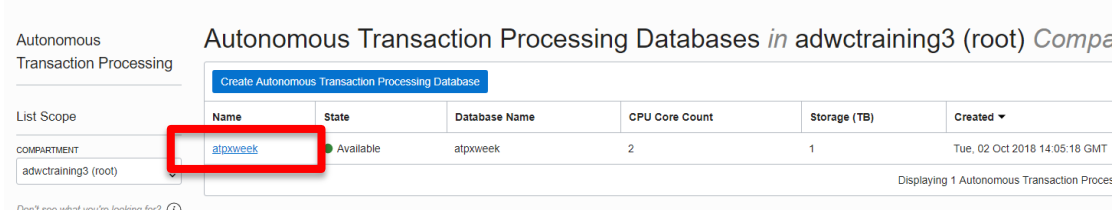


Above are the results for the _HIGH service and below are the results for the _TP (admin) service. The queries should run substantially faster after the system is scaled up. Please note that timings may be different than these due to caching or activity on the system you are using.

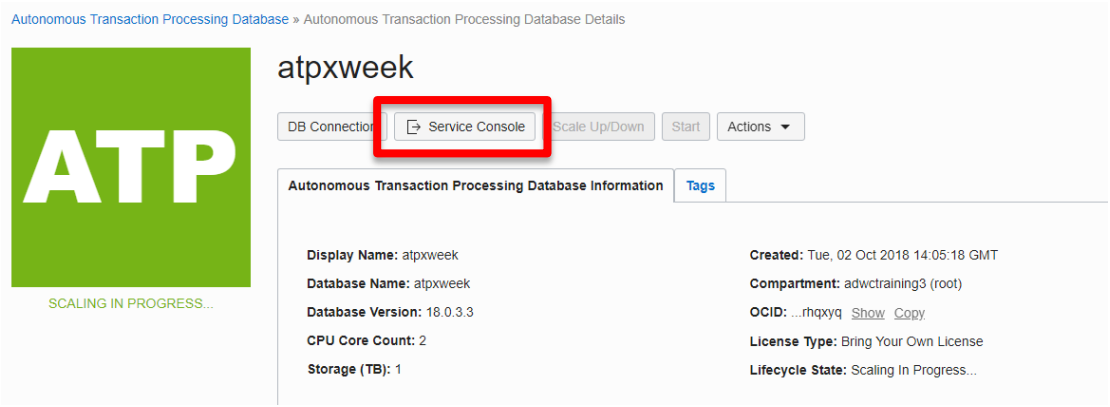


4.2 Monitoring Activity from the Console

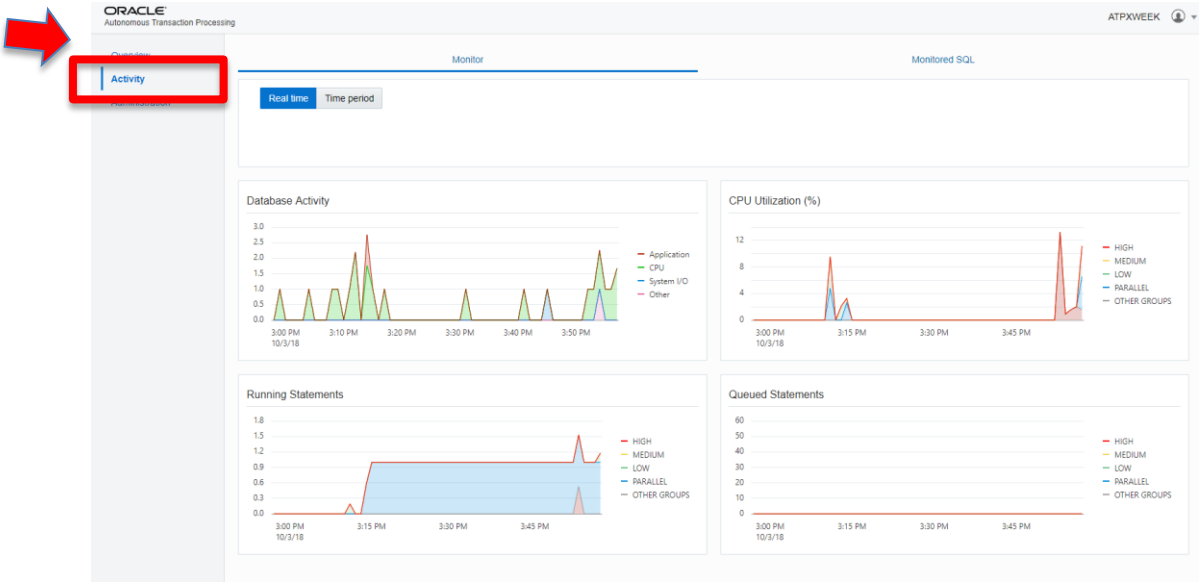
From the Cloud Console used during the provisioning exercise click on your ATP instance:



Select the **Service Console**:



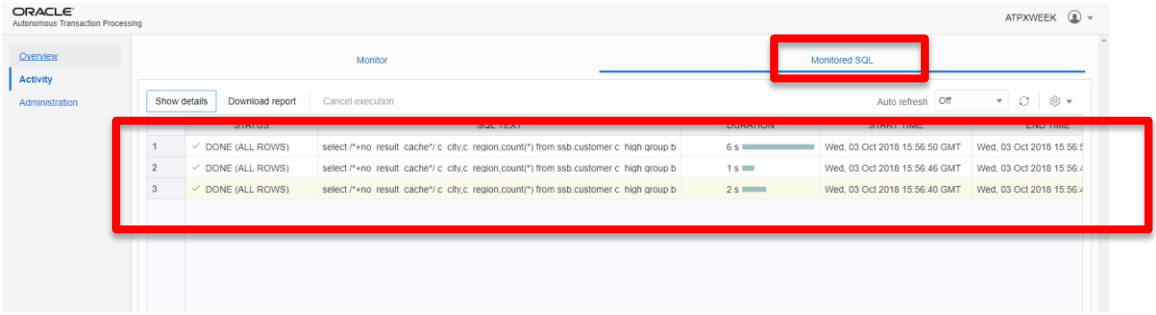
Once on the console select **Activity**. A dashboard of activity on your instance is displayed in four tiles, **Database Activity, CPU Utilization, Running Statements, Queued Statements.** This information is displayed by defined service, as you hoover over each chart you will see the different metrics 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.



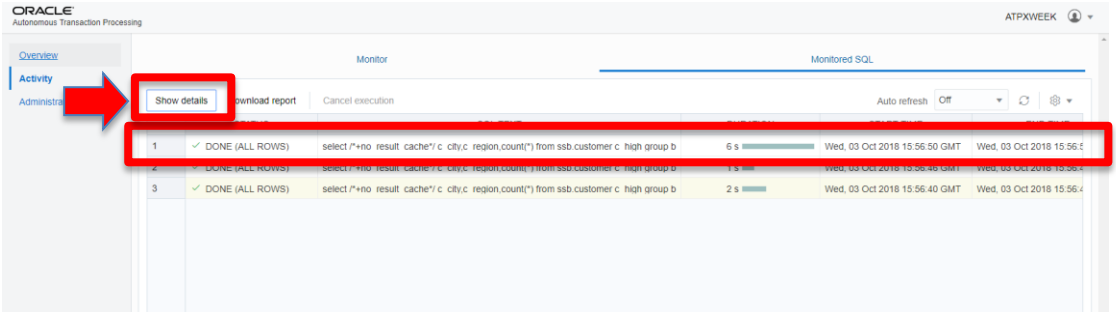
To analyse specific SQL statements, click on the **Monitored SQL** button. This displays SQL that ran or is running in chronological order. The view below displays the two

eugenio.galiano@oracle.com

queries I ran in SQL Developer earlier. Line 1 contains the query run in the **_TP** service and line 3 contains the query run in the **_HIGH** service

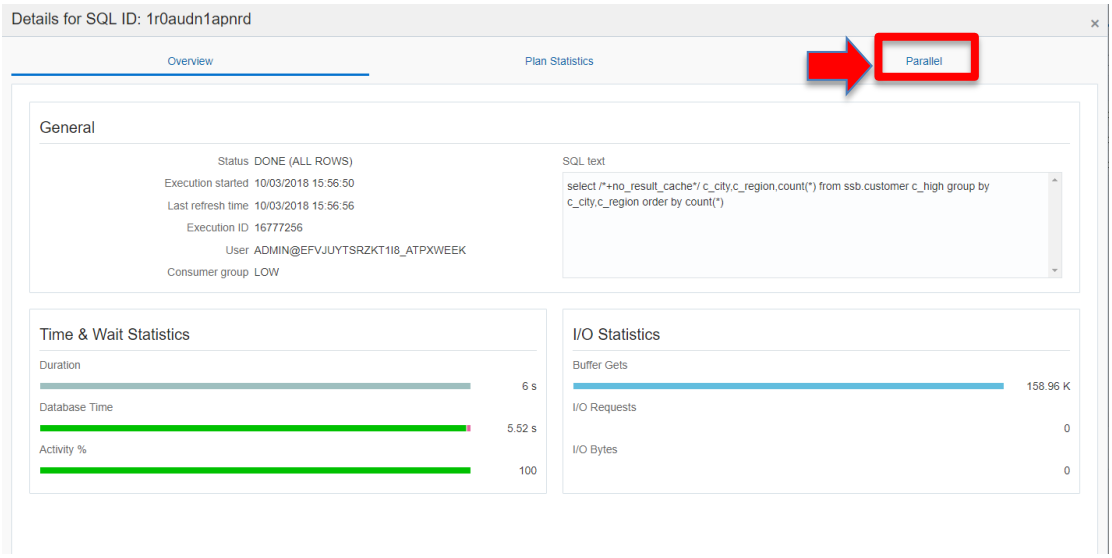


Analyse both statements you ran to get additional insight into how they executed and why they executed differently. In this example I select row 1 by clicking anywhere in the row and then **Show Details:**

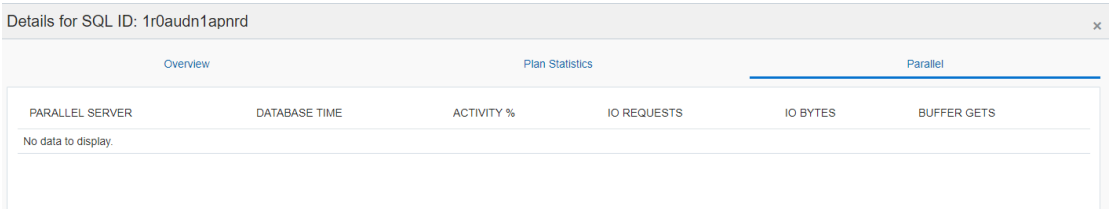


The overview tab provides information about the SQL that was executed, user, times, and service used (consumer group). Notice that this was the query executed in the **_TP** service and it executed in 6 seconds (look at Duration in the Time & Wait Statistics block, lower left). Low service does not use parallelization and to corroborate that click the **Parallel** button:

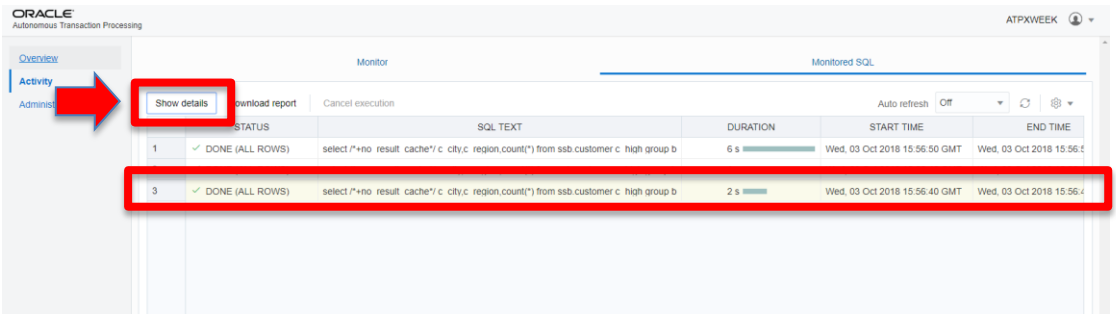
eugenio.galiano@oracle.com



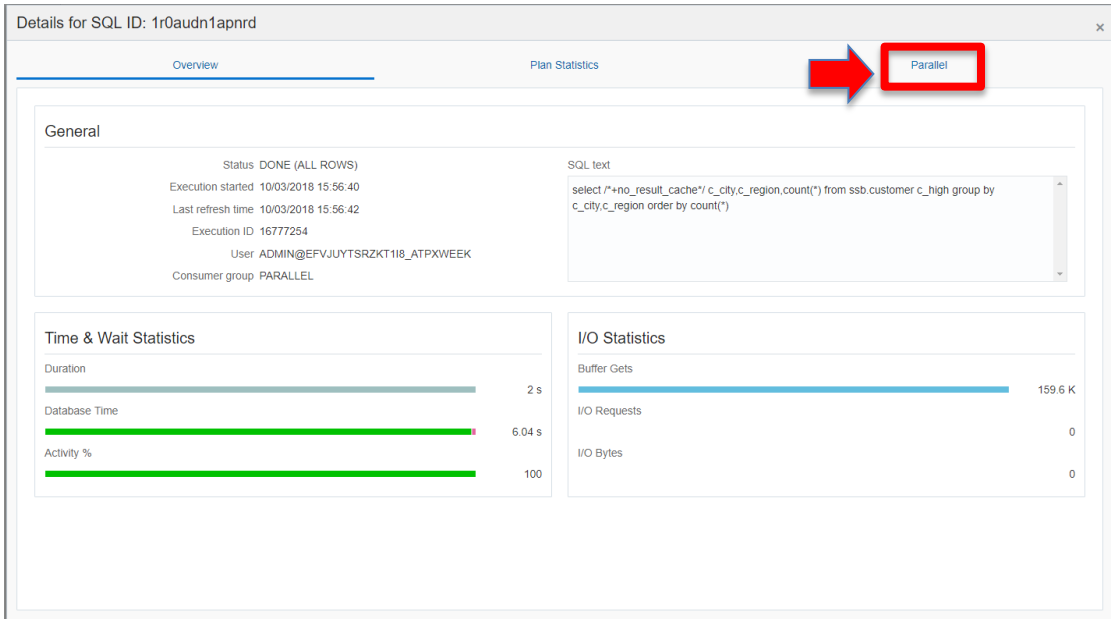
Which brings up an empty page, indicating there was no parallelization for this query.



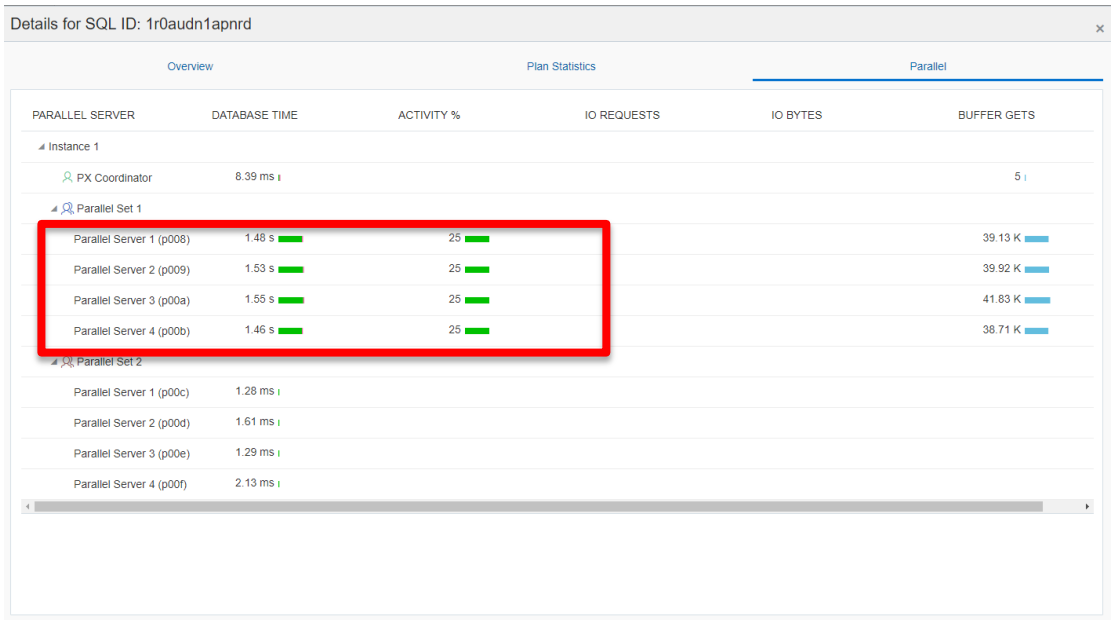
Close the parallel window and select the other statement, in this case 3 and hit **Show details** (this will be different in your environment, you can analyse any statement in the list):



This query executed in the **_HIGH** service (Consumer Group) and it executed in 2 seconds but used 6 seconds of Database time (look at Duration in the Time & Wait Statistics block, lower left). This indicates that the query ran in multiple CPU's and was parallelized. Click on the **Parallel** tab:

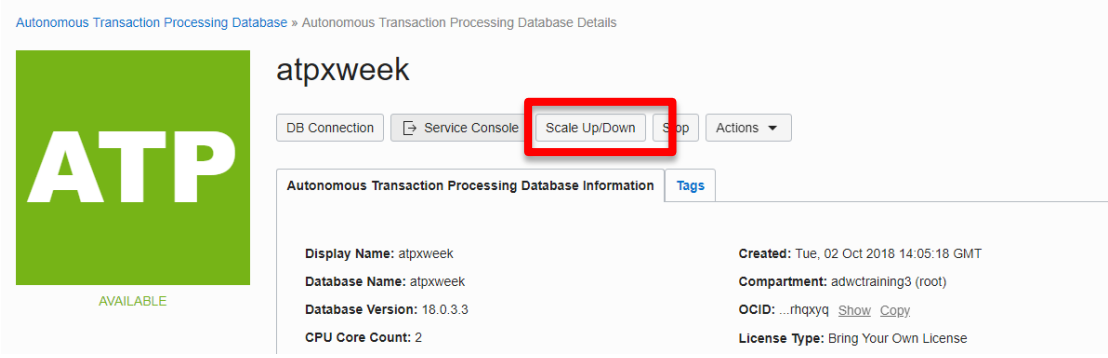


In this case you will notice that the query executed in parallel, with 4 parallel threads (indicated by the Parallel Server information) and each used about 1.5 seconds for the total of about 6 seconds. That is because the **_HIGH** service will automatically parallelize transactions depending on the number of CPU's available (4 in this case).

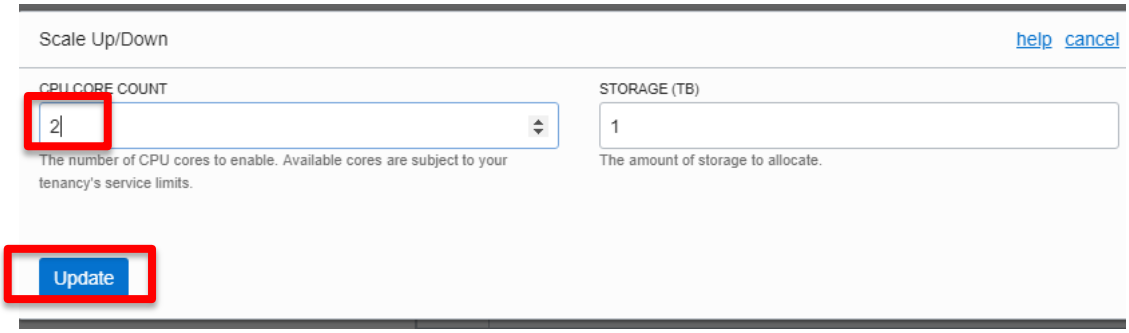


eugenio.galiano@oracle.com

When you are done analysing and testing, scale down your database back to 2 CPU's. From your ATP database main page select Scale Up/Down:



For CPU CORE COUNT enter 2 and then select Update, your instance will return to 2 CPU's



Now you experienced how easy and available scaling is with ATP, and how an immediate impact can be achieved against an application that needs additional resources. And one of the main features of ATP is that customers only get charged for the resource they use, when they are using them!

Lab 5. DBA Exploration of ATP with SQL Developer

Objectives

- Explore the ADMIN view for ATP in SQL Developer

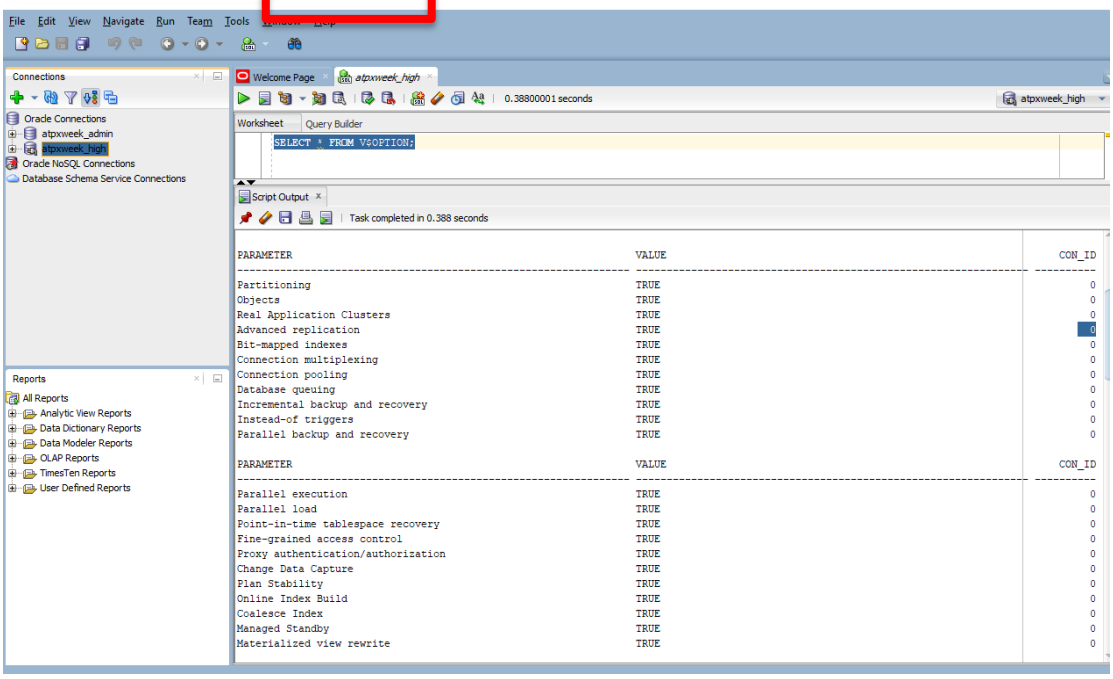
In this lab you will get an overview of the DBA view in SQL Developer and how to explore configuration and settings in your ATP environment. The objective is to be able to dialogue with DBA's that have concerns about the "Autonomous" nature of the service. Being able to see configurations will bring familiarity to DBA's. PLEASE NOTE – Most of these parameters cannot be changed and that is the whole point of the Autonomous Database Service. However this will help DBA's determine compatibility with existing databases they may want to port to ATP.

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

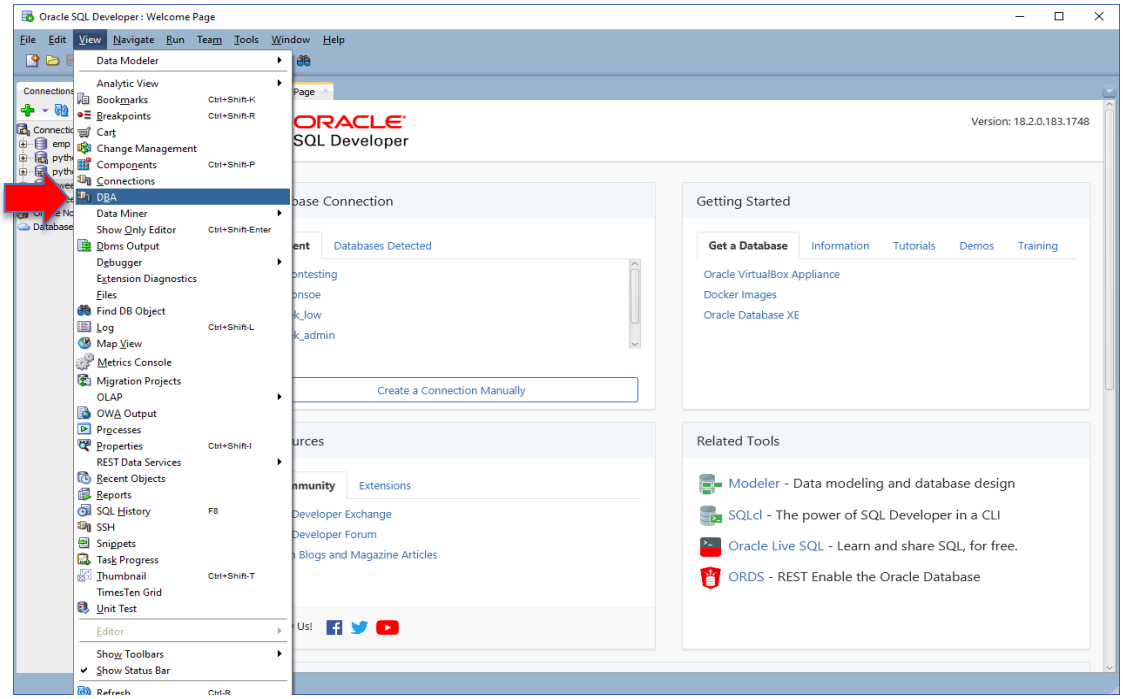
Start SQL Developer. Connect to your database like you did before (as admin). One of the first questions DBA's always ask is what options are available in ATP. They can run the simple select from v$option available in any Oracle implementation to determine the options. In your SQL screen run the following command to get a list of options in ATP.

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


eugenio.galiano@oracle.com

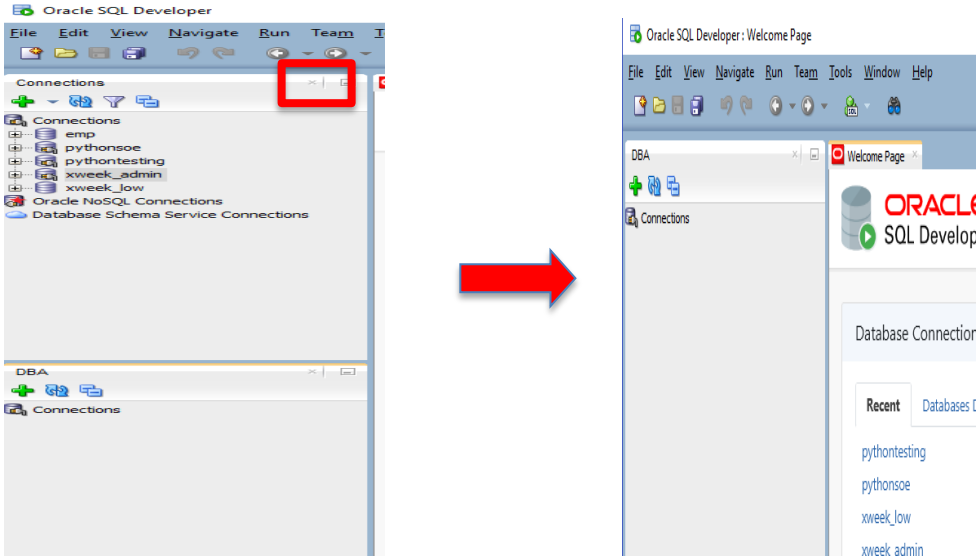


Next you will examine the DBA view in SQL Developer. In the main screen select **View->DBA. If you cannot see the DBA option under View, go to "Window" and select "Reset Window to Factory Setting",** this should let you see the DBA option under View.

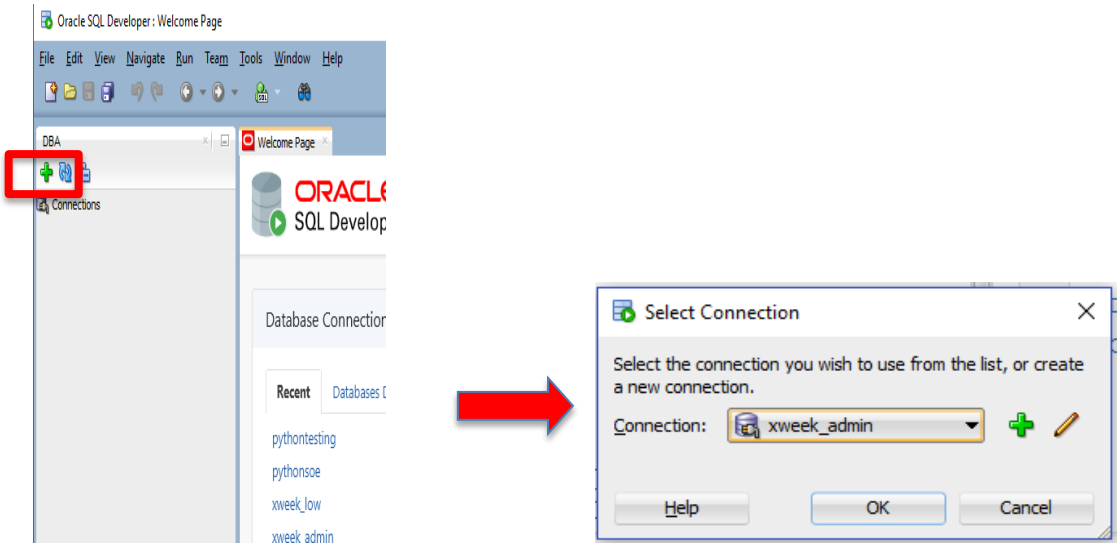


eugenio.galiano@oracle.com

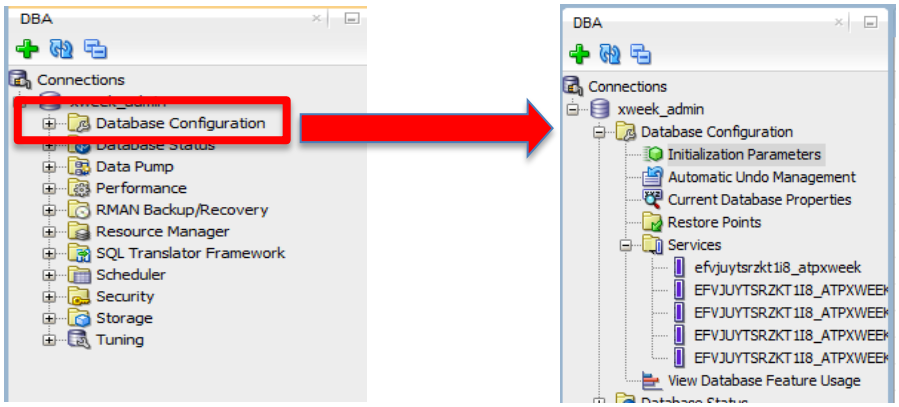
The DBA view appears on the bottom left. IF you have a connection box in the top left (the box used for previous labs), close the connections box by clicking the X on top right of the box. You should only see the DBA view. Below are screenshots of both steps:



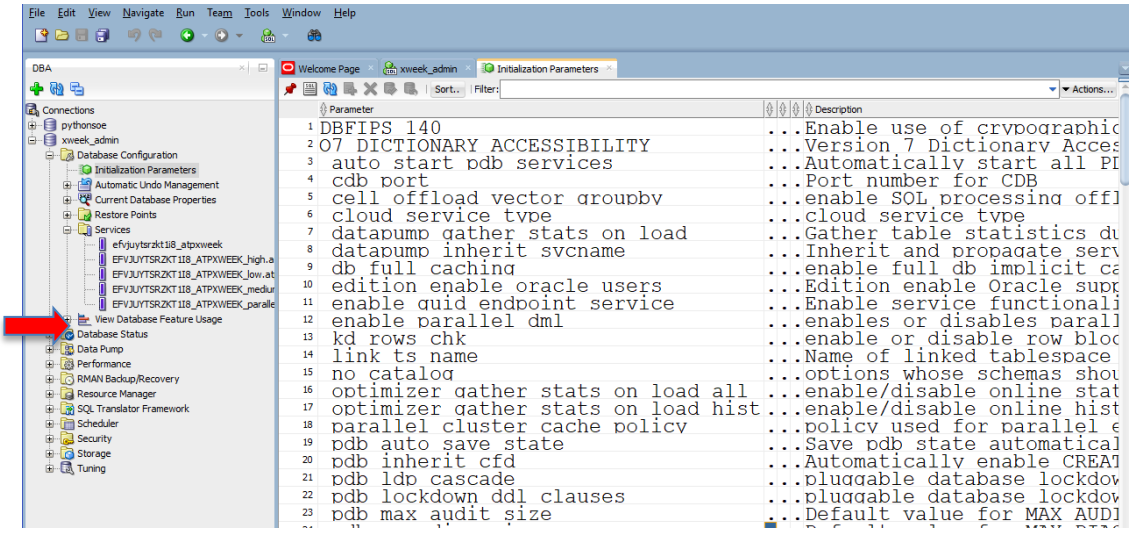
Click the connections green plus sign in the DBA view, it will pop up the connections menu, select the connection you created for your admin database connection (any service _high or _admin) and OK:



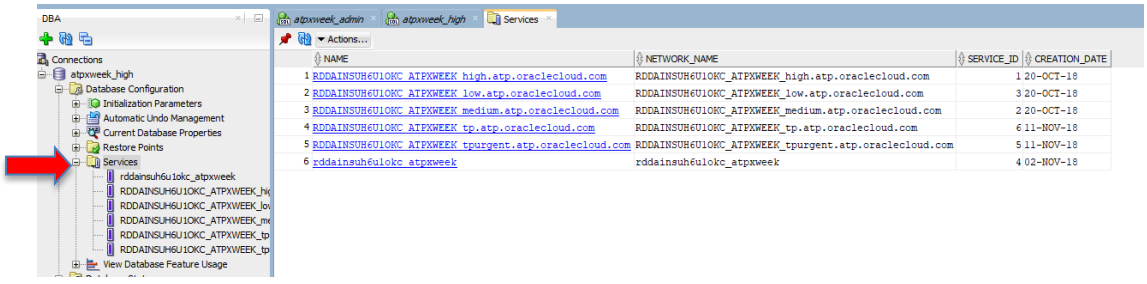
This will open up all the "DBA" views of the database, open Database Configuration:



In **Initialization Parameters** (click it) you can view the parameters used for this database.

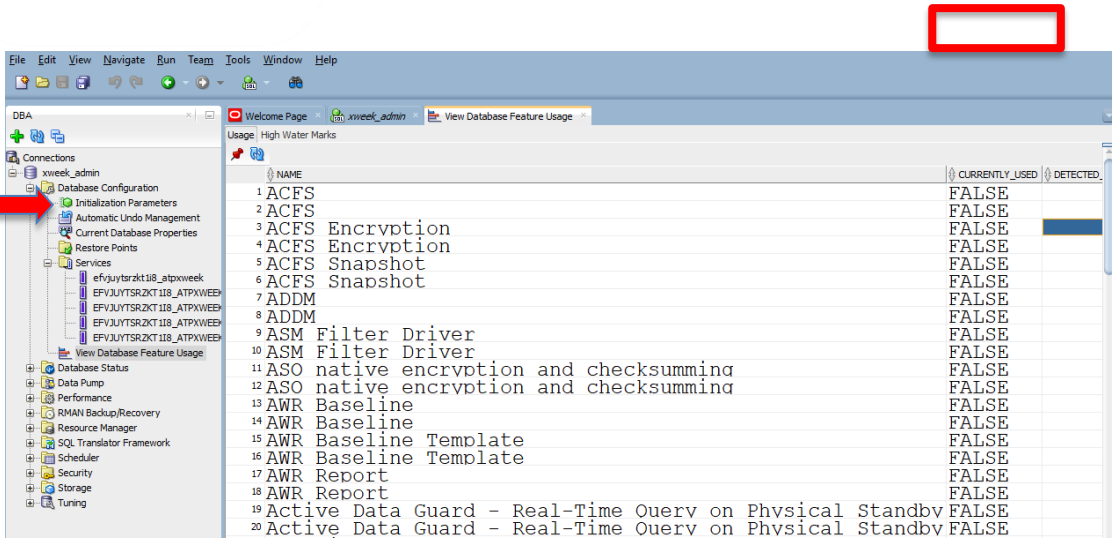


In **Services** you can see five services configured and discussed in the services lab.

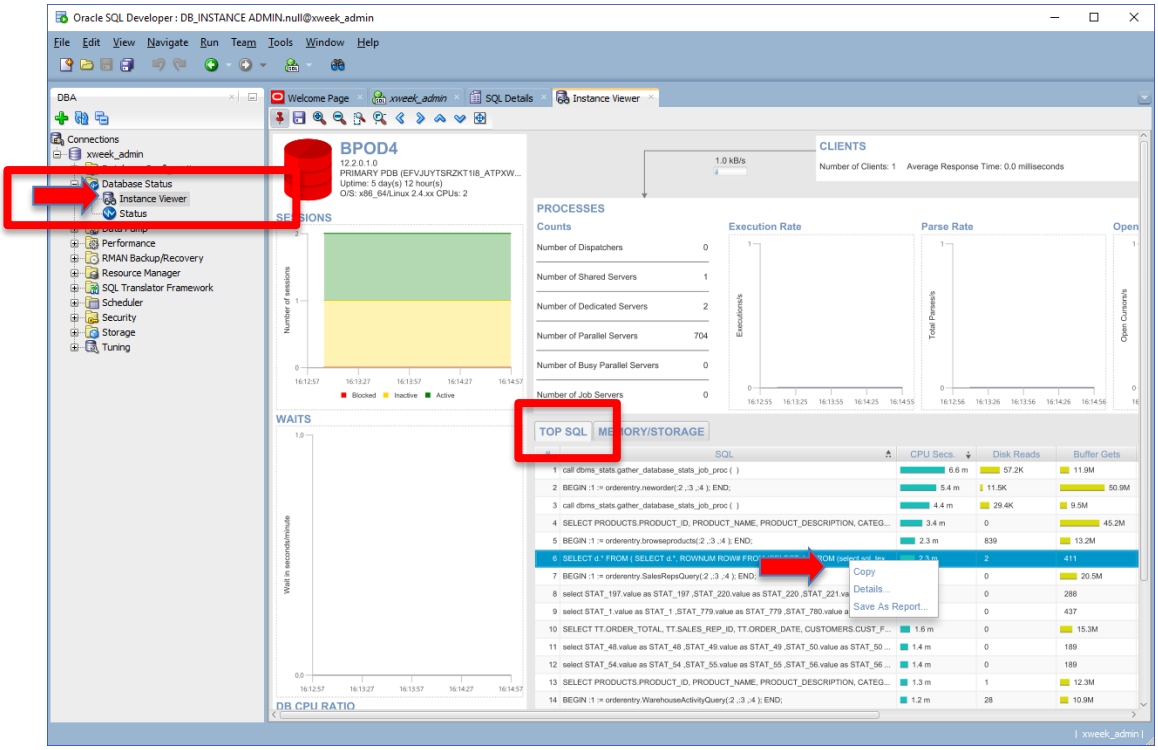


In **View Database Features Usage** you can see the features currently in use. As with other views, columns can be sorted, so for example you can sort the CURRENTLY_USED column by TRUE to see all the features being used:

eugenio.galiano@oracle.com

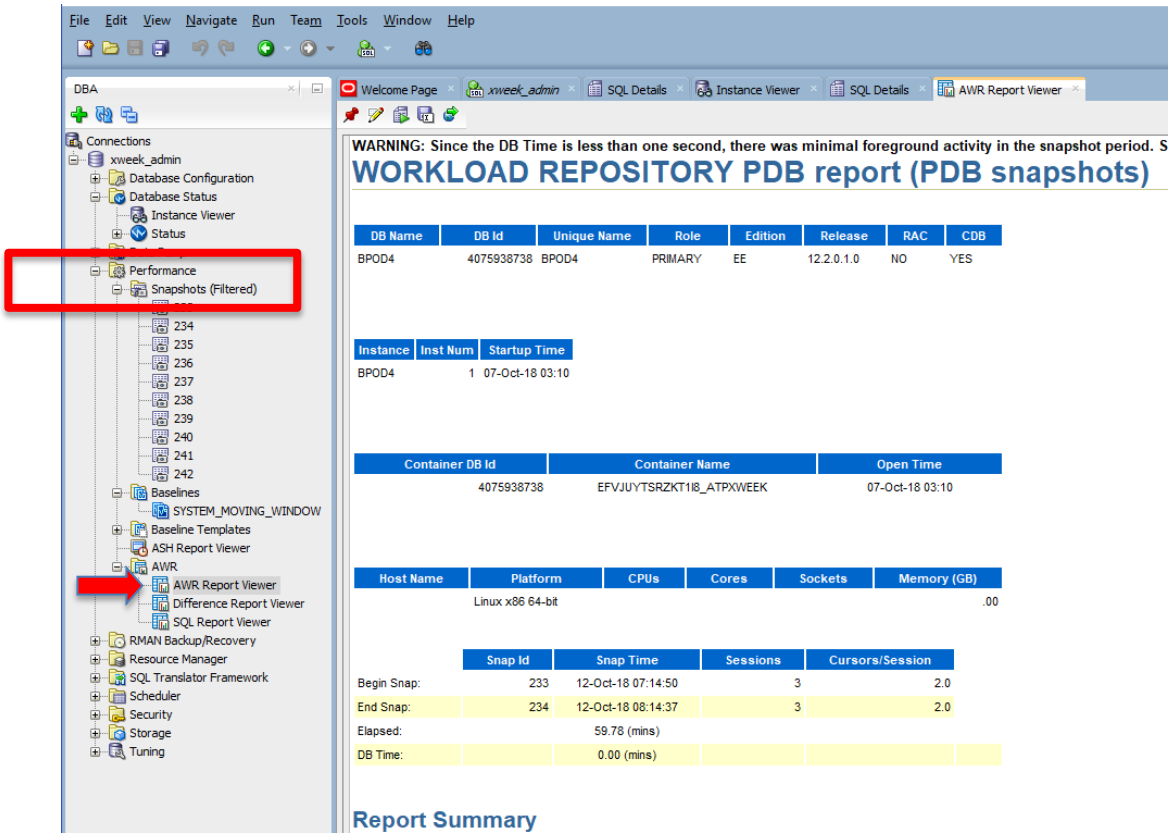


Open up **Database Status** and double click on **Instance Viewer.** This displays a lot of information about the instance, usage and configuration as well as the TOP SQL running on the instance. You can **select any SQL from TOP SQL, left click and select Details**. This will create a new tab on the top called SQL Details and put you in that tab. You can examine the SQL specifics there.

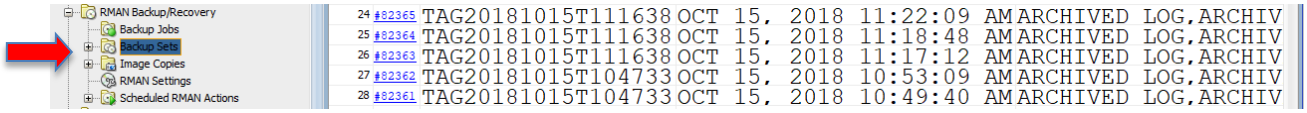


The **Performance** section will contain information relating to how the database is performing. The most familiar tool for DBA's will be the AWR reports, found under **AWR->AWR Report Viewer**

eugenio.galiano@oracle.com

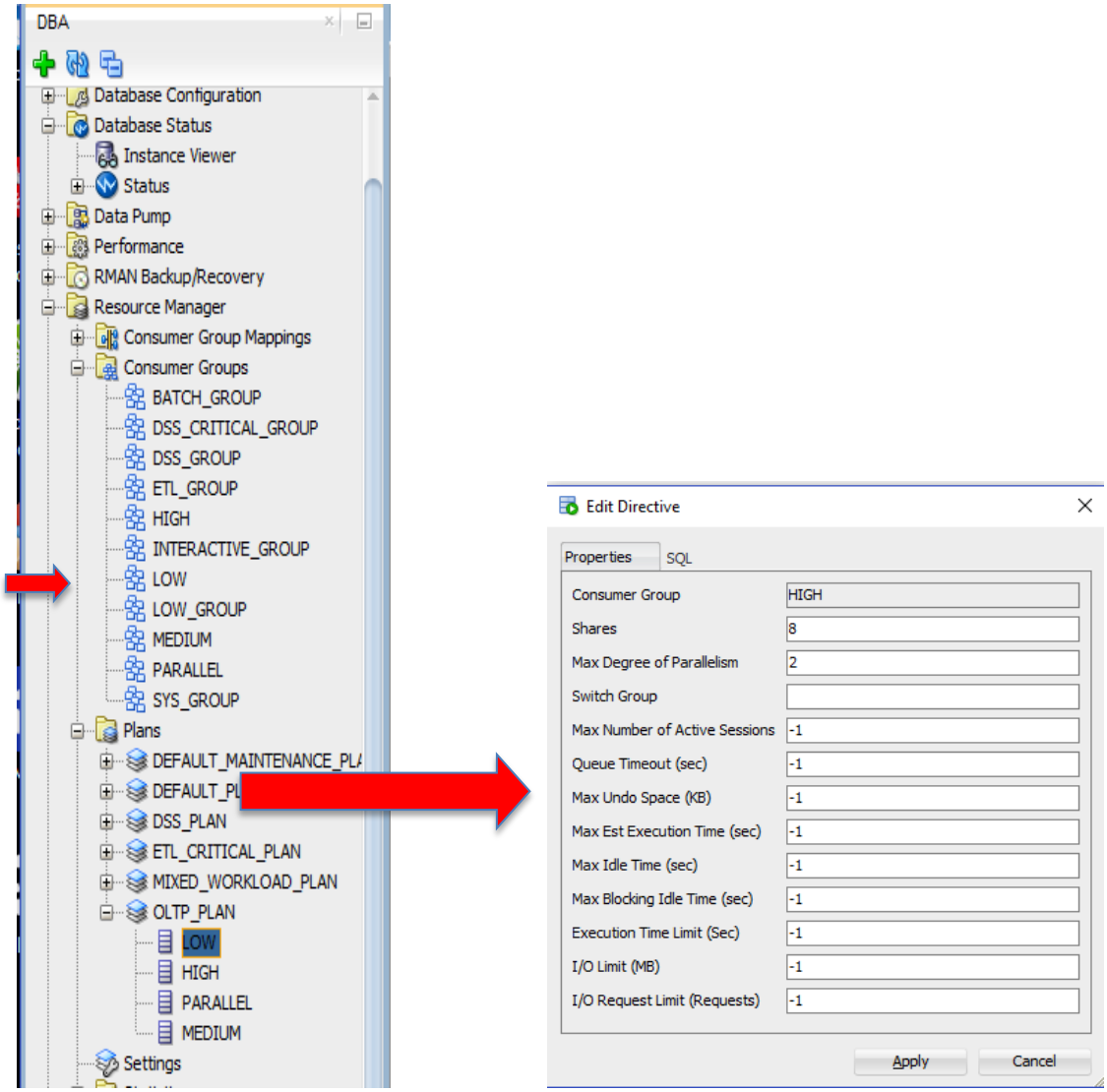


The **RMAN Backup/Recovery** contains information about scheduled backups, backup sets, RMAN Settings and schedules. Customers and DBA's often ask about backups on ATP and those can be explored in more detail in this section. In the screenshoot below the backup sets for this instance are displayed (your instance may not display any backups since it was just created)



The **Resource Manager** contains information about different consumer groups and plans defined, and the current plan in effect. Explore the entries here, most of them will have contextual activities if you select and right click on them. Under **Plans->OLTP_PLAN** you will find the current plan being used, which you can verify by double clicking on **Settings**, the second to last option under **Resource Manager.** This will display which plan is being used. Under **OLTP_PLAN** if you select and right click on any of the plans, and select **Edit Directives**, the specifics of the plan will be displayed. Below you will see the screenshot for the **_HIGH** plan.

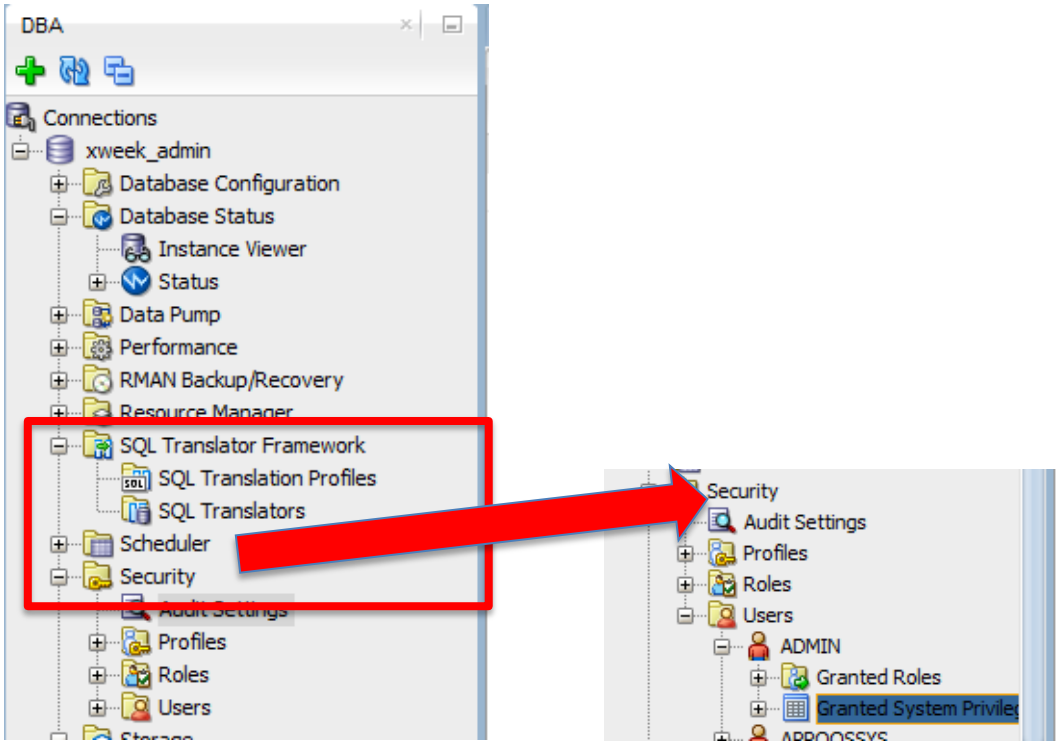
eugenio.galiano@oracle.com



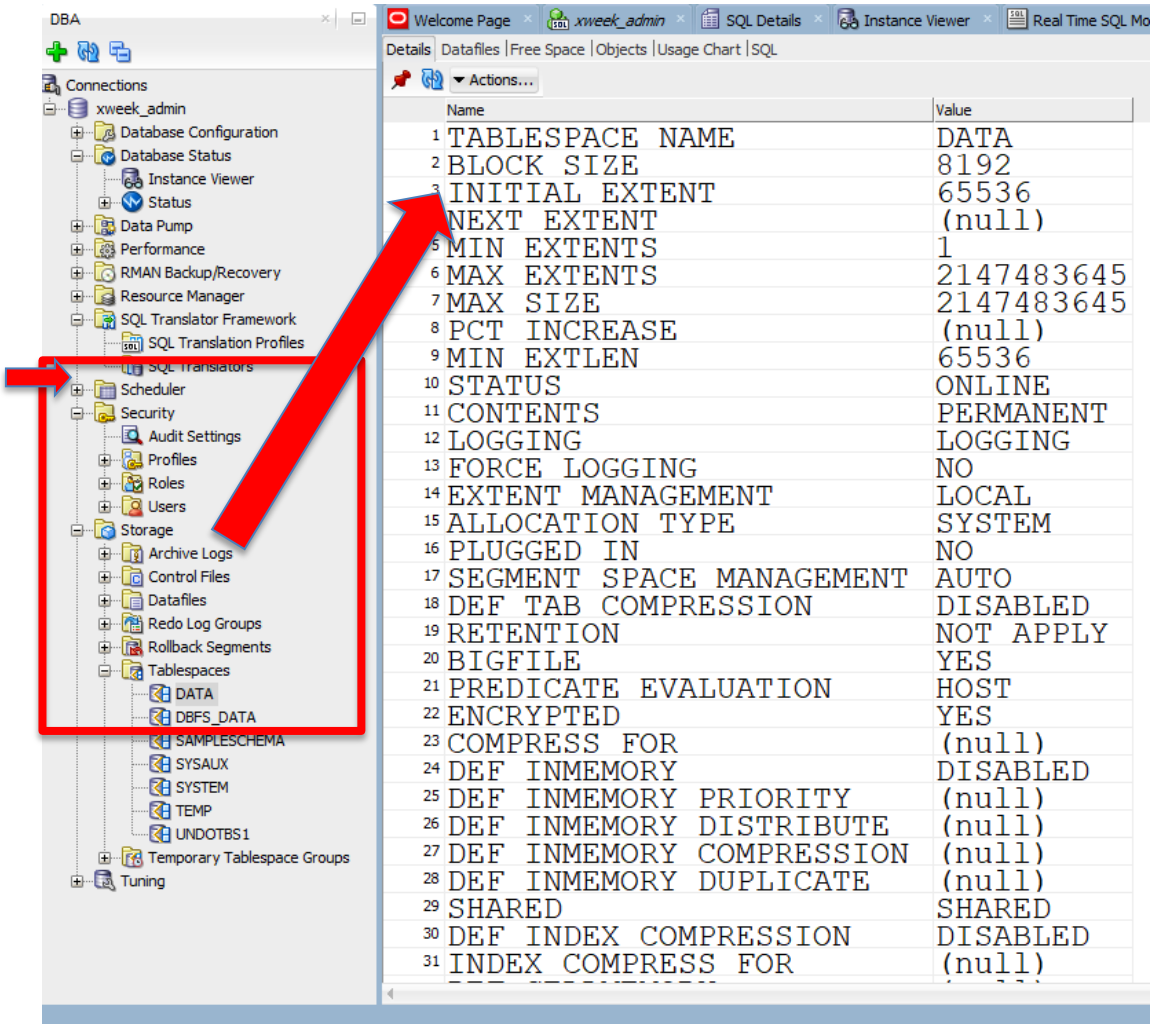
The last entry under **Resource Manager->Statistics** provides graphical views into system usage by plan user. Double click on Statistics and explore the different screens.

The **Security** section will be very useful to DBA's as they can explore all the Profiles, and Roles defined in the system, and under **Users,** roles and system priviledges granted to the user. Spend some time examining this area.

eugenio.galiano@oracle.com



The **Storage** section can be used to display all the storage characteristics of the ATP database. Most of these characteristics cannot be changed, as the Autonomous Database automatically manages and optimizes storage for the service. 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 Tablespaces and then double click on DATA to see). Examine the different entries in this area.



This concludes the SQL Developer Lab. SQL Developer is a very comprehensive powerful tool and you should continue to explore the different benefits it offers. Continue to look at all the attributes in the DBA view to see what else you can find out about your ATP instance.

Lab 6. Installing the Oracle Instant Client

Objectives

- Installing Oracle client libraries
- Install Microsoft Visual Studio Redistributable (Windows install only)

In order to connect and run applications from your PC to remote Oracle databases, Oracle client libraries must be installed on your computer. For the next three labs (node.js, Python, and Swingbench), these libraries need to installed. Please note we will be doing a simple installation. You may already have the libraries installed and they may conflict with directories and libraries being installed in this lab. In a lot of cases conflicts may be resolved with correcting PATH's and configuring the correct TNS_ADMIN location.

6.1 Installing the Oracle Instant Client:

Information on installing the pre-requisites for several operating systems can be found here:

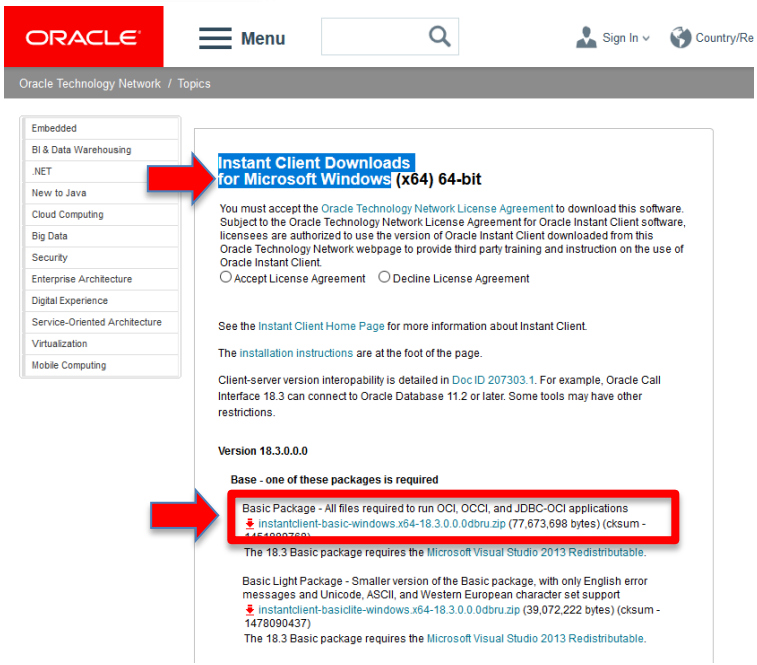
https://oracle.github.io/odpi/doc/installation.html

The basic steps to follow are:

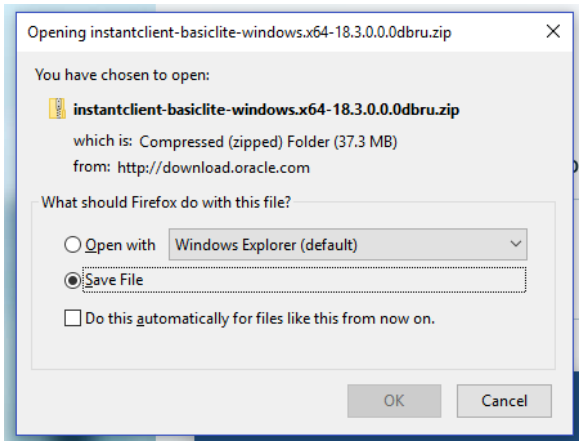
1. Download Oracle 18 "Basic Light" zip file: 64-bit or 32-bit, matching your application architecture.
2. Unzip the package into a single directory that is accessible to your application, for example `C:\oracle\instantclient_18_3`.
3. Set the environment variable `PATH` to include the path that you created in step 2.

For 64-bit Windows 10 Installations go to:

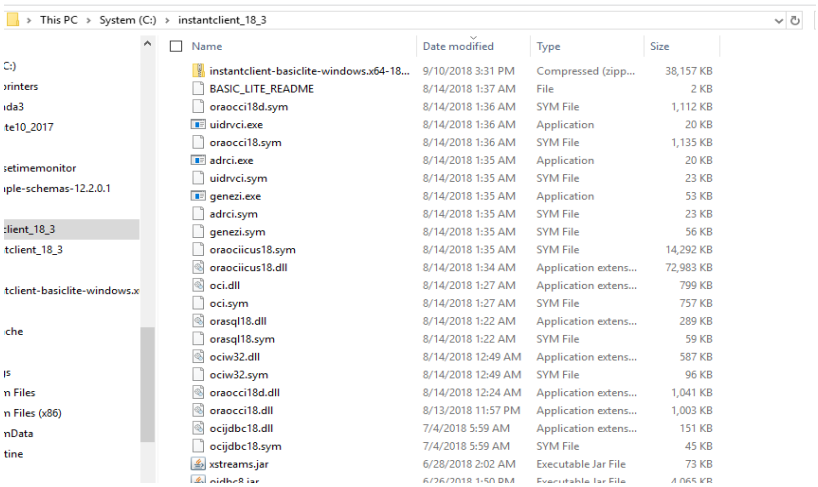
https://www.oracle.com/technetwork/topics/winx64soft-089540.html



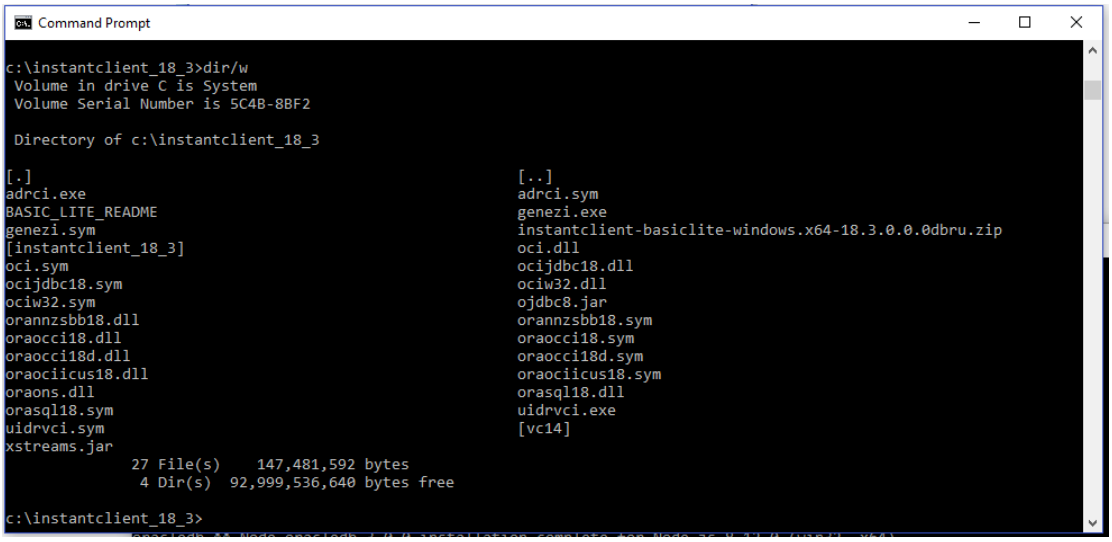
Select the Basic Light Package instantclient-basiclite-windows.x64-18.3.0.0.0dbru.zip. This will require you signing into OTN with your SSO account. If you do not have an account you need to create one. Once logged in, the download process will launch:



Download the file and then proceed to the directory where the file was downloaded (Downloads in Windows). Unzip the file into a directory the **c:\instantclient_18_3**. Below is the directory (folder) where the file was unzipped.



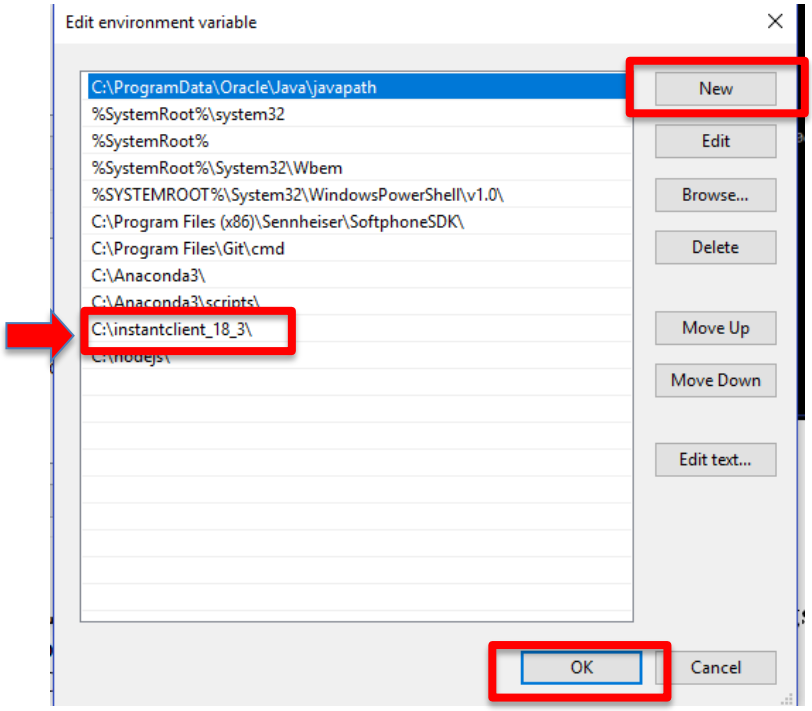
You can open a Command Prompt and navigate to that directory to examine as well. Below you can see the directory in the Command Prompt:



THIS DIRECTORY MUST BE ADDED TO YOUR PATH

In Windows 10 (you can easily google how to update PATH in Windows if this does not work for you, or you have a different operating system):

1. In Search, search for and then select: **Advanced System Settings** (control panel)
2. Click **Environment Variables** at bottom of screen
3. In the **System variables** double click **Path**
4. In the screen that opens up select **NEW**
5. Add full path to the instant client directory (**c:\instantclient_18_3)**



The command window must be restarted before continuing to reflect the new path.

In Windows 7:

Follow similar steps, the PATH environment can be found at:

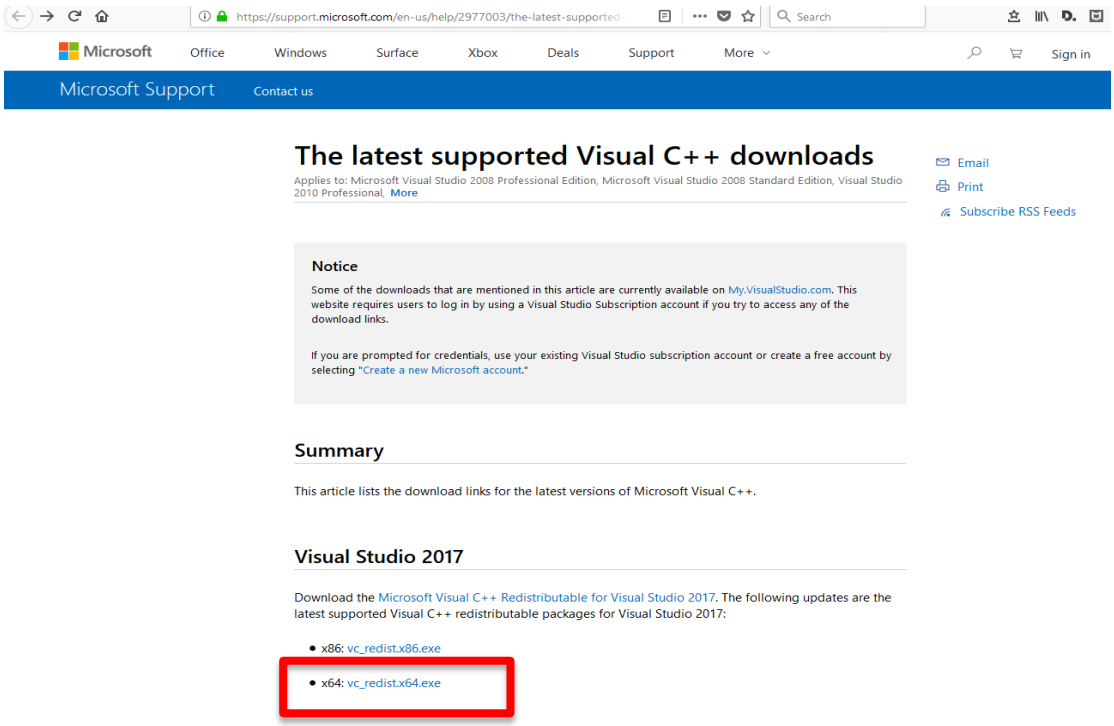
Control Panel -> System -> Advanced System Settings -> Advanced -> Environment Variables -> System Variables -> PATH

6.2 Installing the Microsoft Visual Studio Redistributable (Windows install only):

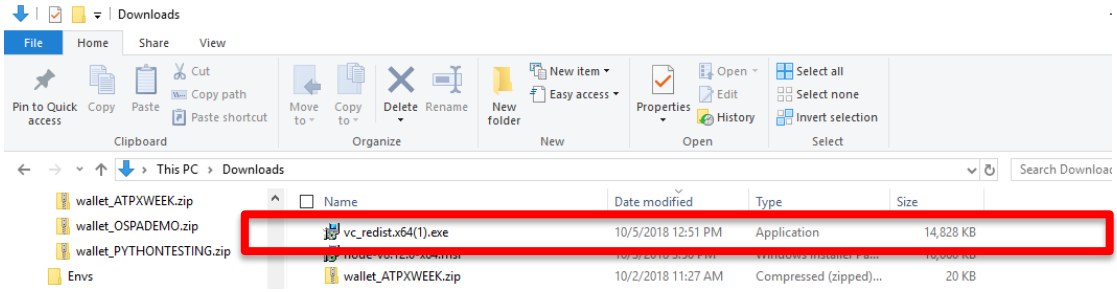
Oracle Client libraries for Windows require the presence of the correct Visual Studio redistributable. For Oracle 18c (12.2) which runs on ATP the version required is VS2013. Follow the link below to install

https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads#bookmark-vs2013

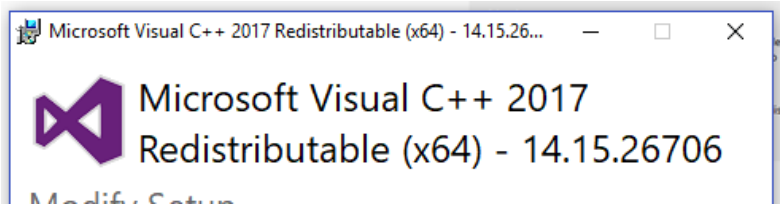
Select the correct architecture - 32 (x86) bit or 64 (x64) bit:



Select and download the file. Navigate to the directory were the file was downloaded.



Double click on the file and proceed with the installation, take all defaults.



This completes the installation of the pre-requisites.

Lab 7. Workload testing and analysis using Swingbench with ATP

Objectives

- Learn how to install Swingbench
- Using Swingbench to demo and analyse workloads with ATP

In this section you will use Swingbench, a suite of utilities developed by Dominic Giles that creates Oracle Database, objects, data, and creates workloads that can then be monitored or used for testing an ATP environment. Among the many uses for Swingbench is the ability to provided and comprehensive demo of the database and associated tools and functionality.

The objective of this lab it to learn how to install Swingbench and connect and use it with ATP, not a comprehensive Swingbench tutorial. At the end of this lab you can find links to much more information on Swingbench and its use.

7.1 Installing Swingbench pre-reqs:

To run Swingbench requires 3 components:

1- Unix BASH shell
2- Java JRE 1.8xx
3- Swingbench (and optional modules)

Installing Unix BASH shell:

To run Swingbench on Windows you need to simulate running a Unix shell and that can be done in many ways. This lab uses the Git BASH shell. The latest version of Windows 10 includes this functionality natively but you many need to enable/install it. The steps for getting BASH running will not be covered in this lab but for Windows 10 you can find detailed instructions here:

https://www.windowscentral.com/how-install-bash-shell-command-line-windows-10

For older versions of Windows you can download and install BASH

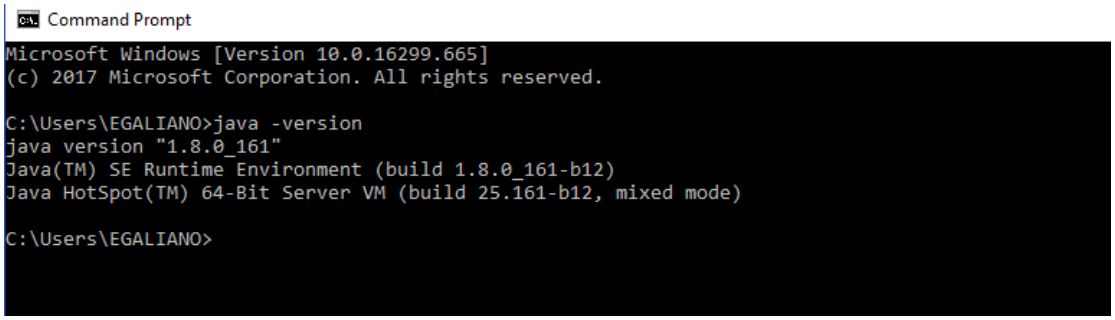
https://gitforwindows.org/

Installing Java JRE 1.8xx:

Swingbench requires Java 8 to run. You must install it if you don't have it installed. Please notice if you already have SQL Developer running then you have the correct Java installed already. You can test to see if you have it installed by running the following command in a Command Prompt window:

```
java –version
```

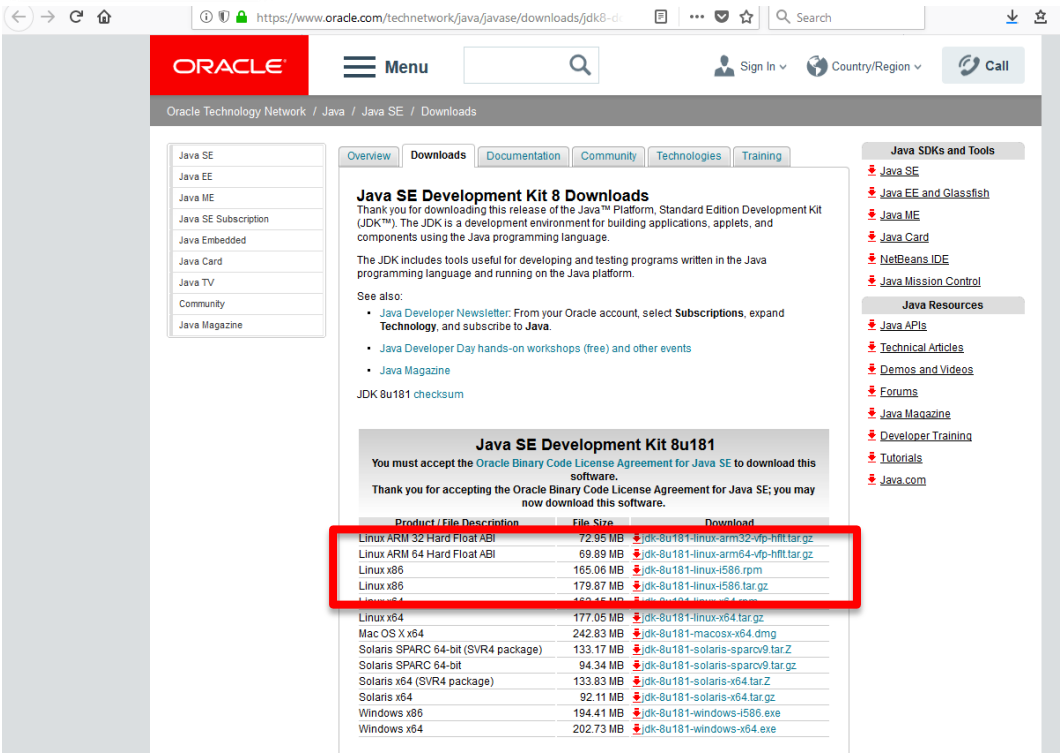
and the version should be 1.8.0_xx



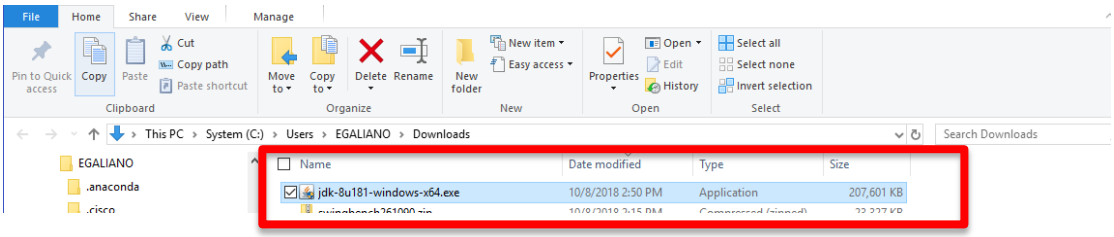
Java can be downloaded and installed from (pick the appropriate operating system and architecture):

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

eugenio.galiano@oracle.com

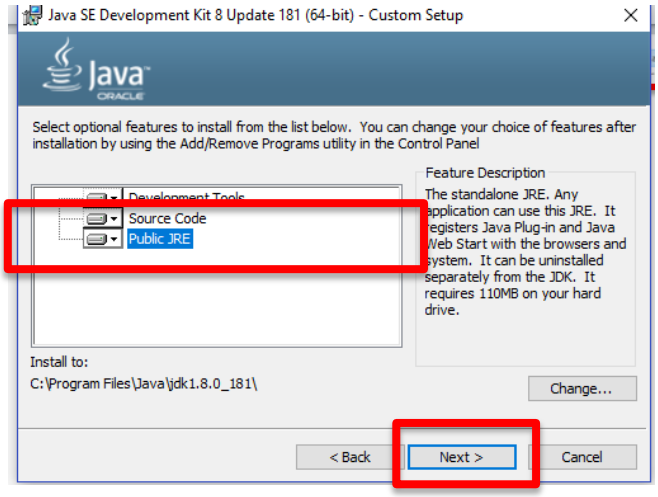


After downloading the file, proceed to the download folder and double click to install:



Click Next

Select Public JRE

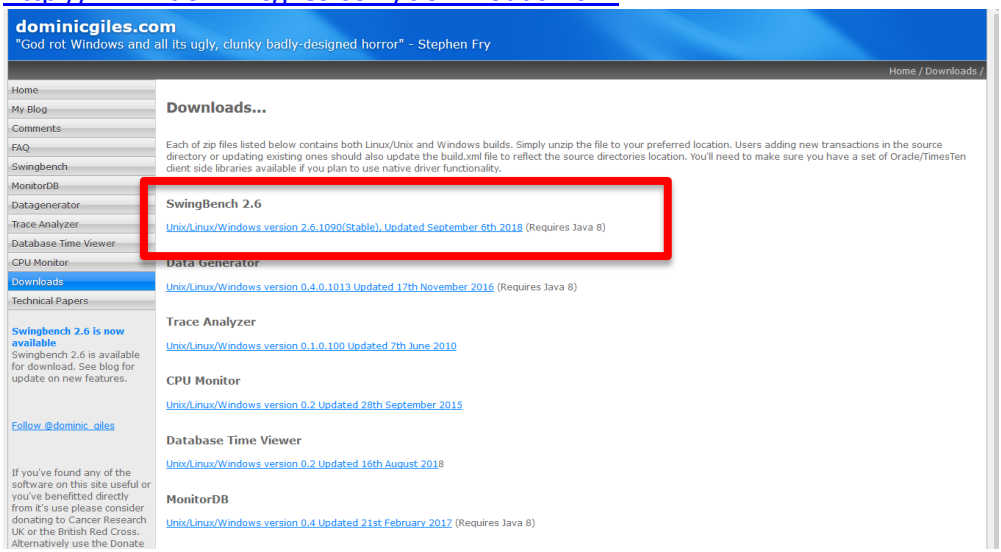


Finish the Java install process.

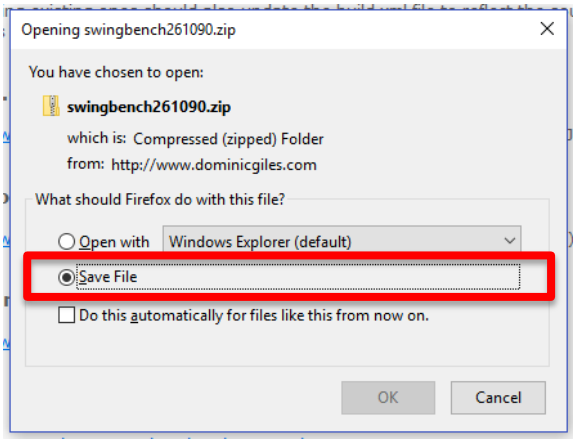
7.2 Installing and running Swingbench:

Download Swingbench from:

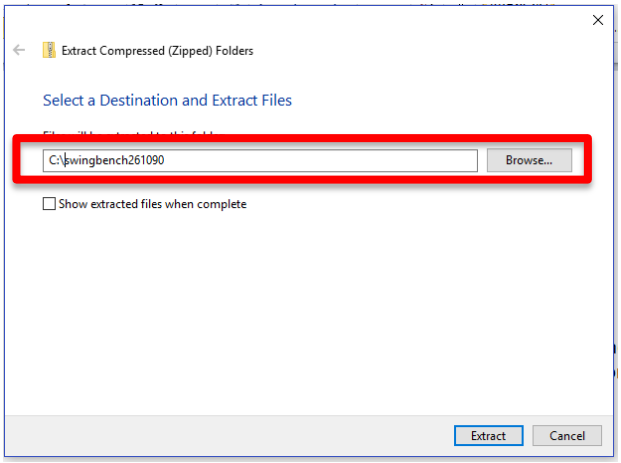
http://www.dominicgiles.com/downloads.html



Save the file:



After the file downloads, go to the folder where it downloaded, and right click on the file, select Extract all (uncompress) to the following directory: **C:\swingbench26,** select **Extract**



Open up a BASH window (in Search Windows type BASH and select)

Change to the “bin” subdirectory where swingbench is installed. In this example: its

```
/c/swingbench261090/swingbench/bin
```

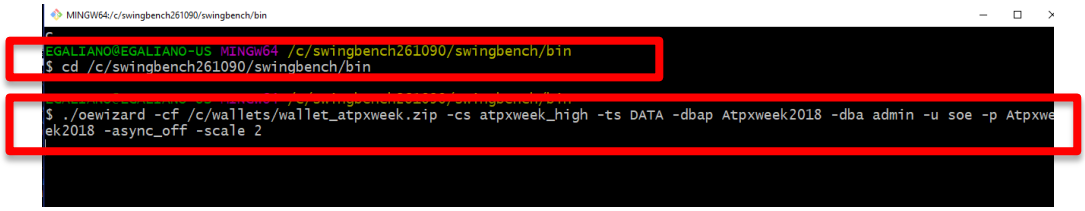
And start swingbench by issuing the command below (see screenshot and explanation of parameters below) (make sure to use your wallet and ATP database user/password information, fields that need to be modified to your values are in black). Notice the command below will create a database user name "soe" (`-u soe` with a password "Atpxweek2018" (`-p Atpxweek2018)` which will be used by Swingbench to load data and run workloads.

```
./oewizard -cf /c/wallets/wallet_atpxweek.zip \
           -cs atpxweek_tpurgent \
           -ts DATA \
           -dbap Atpxweek2018 \
           -dba admin \
           -u soe \
           -p Atpxweek2018 \
           -async_off \
           -scale .5 \
           -hashpart \
           -create \
           -cl \
           -v  \
-debug
```

Understanding the parameters above

- **-cf** speficies the location of the wallet file
- **-cs** specifies the ATP service to connect to

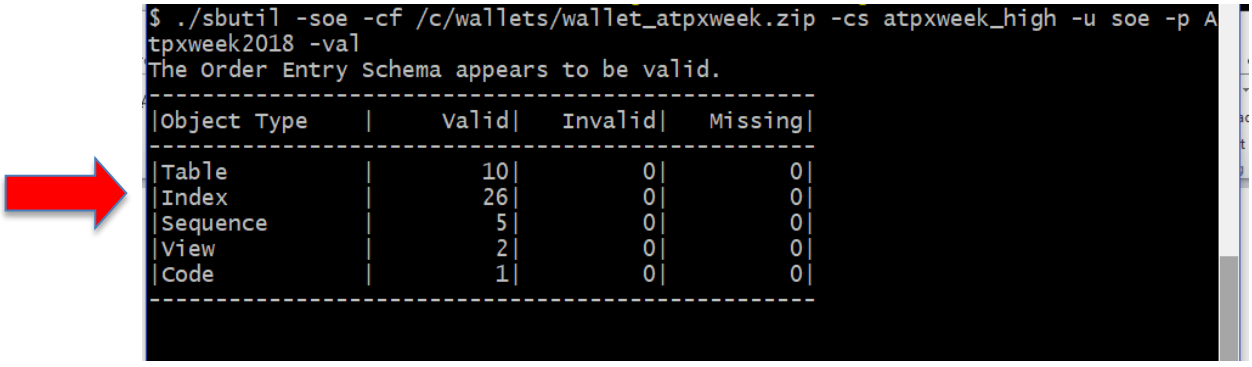
- **-ts** name of the table space to install swingbench into. For ATP it is always DATA
- **-dba** user **admin** created during ATP instance creation
- **-dbap admin** user password
- **-u** new database user created for swingbench data
- **-p** new user (above) password
- **-async_off** async commits are not supported in ATP
- **-scale** in GB's, data size, plus additional 50% of data for indexes
- **-debug** will display back each step of the process – leave this out if too much data is on your screen.



Creation time will vary depending on where you are running it from and may take 20 minutes or more. Once complete verify the tables created correctly by running the following command (from the swingbench bin directory, make sure to use your wallet, change values in black):

```
./sbutil -soe \
     -cf /c/wallets/wallet_atpxweek.zip \
     -cs atpxweek_tpurgent \
     -u soe \
     -p Atpxweek2018 \
-val
```

Results should be similar to the screenshot below:



eugenio.galiano@oracle.com

To see how many rows were inserted on each table run the following command (make sure to use your wallet information):

```
./sbutil -soe \
     -cf /c/wallets/wallet_atpxweek.zip \
     -cs atpxweek_tpurgent \
     -u soe \
     -p Atpxweek2018 \
-tables
```

Results should be similar to the screenshot below:

```
$ ./sbutil -soe -cf /c/wallets/wallet_atpxweek.zip -cs atpxweek_high -u soe -p Atpxweek2018 -tables
Order Entry Schemas Tables
-------------------------------------------------------------------------------------------------------------
|Table Name                |              Rows|            Blocks|          Size|   Compressed?|  Partitioned?|
-------------------------------------------------------------------------------------------------------------
|ORDER_ITEMS               |         2,129,873|            17,237|       136.0MB|      Disabled|            No|
|ORDERS                    |           714,895|            10,097|        80.0MB|      Disabled|            No|
|ADDRESSES                 |           750,000|             8,057|        64.0MB|      Disabled|            No|
|CUSTOMERS                 |           500,000|             7,930|        63.0MB|      Disabled|            No|
|CARD_DETAILS              |           750,000|             4,780|        38.0MB|      Disabled|            No|
|LOGON                     |         1,191,492|             3,898|        31.0MB|      Disabled|            No|
|INVENTORIES               |           894,632|             2,386|        19.0MB|      Disabled|            No|
|PRODUCT_DESCRIPTIONS      |             1,000|                35|         320KB|      Disabled|            No|
|PRODUCT_INFORMATION       |             1,000|                28|         256KB|      Disabled|            No|
|WAREHOUSES                |             1,000|                 5|          64KB|      Disabled|            No|
|ORDERENTRY_METADATA       |                 4|                 5|          64KB|      Disabled|            No|
-------------------------------------------------------------------------------------------------------------
                                                  Total Space        431.7MB
```

To collect statistics on the tables that were created run the command below. If you will be conducting performance testing with Swingbench it is recommended that you collect statistics (make sure to use your wallet information):

```
./sbutil -soe \
     -cf /c/wallets/wallet_atpxweek.zip \
     -cs atpxweek_tpurgent \
     -u soe \
     -p Atpxweek2018 \
-stats
```

```
$ ./sbutil -soe -cf /c/wallets/wallet_atpxweek.zip -cs atpxweek_high -u soe -p Atpxweek2018 -stats
Collecting statistics for the schema
Collected statistics in : 0:00:14.603
```

You are ready to run Swingbench workloads on ATP. Workloads are simulated by users submitting transactions to the database. To do this, the user process must be configured. Run the following command unchanged from the same bin directory you have been running the other commands:

eugenio.galiano@oracle.com

```
sed -i -e
's/<LogonGroupCount>1<\/LogonGroupCount>/<LogonGroupCount
>5<\/LogonGroupCount>/' \
       -e
's/<LogonDelay>0<\/LogonDelay>/<LogonDelay>300<\/LogonDel
ay>/' \
       -e
's/<WaitTillAllLogon>true<\/WaitTillAllLogon>/<WaitTillAl
lLogon>false<\/WaitTillAllLogon>/' \
       ../configs/SOE_Server_Side_V2.xml
```

Generate load on your database by running the charbench utility. (make sure to use your wallet information). Use the command below. There are 2 parameters you can change to modify the amount of load and users being generated. The `-uc` specifies the number of users that will be ramped up, in the case below 64. The `-rt` specifiies the total running time, in the example below 30 seconds. To run continuously remove that parameter, or change the time to specify a maximum run time. Please 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.

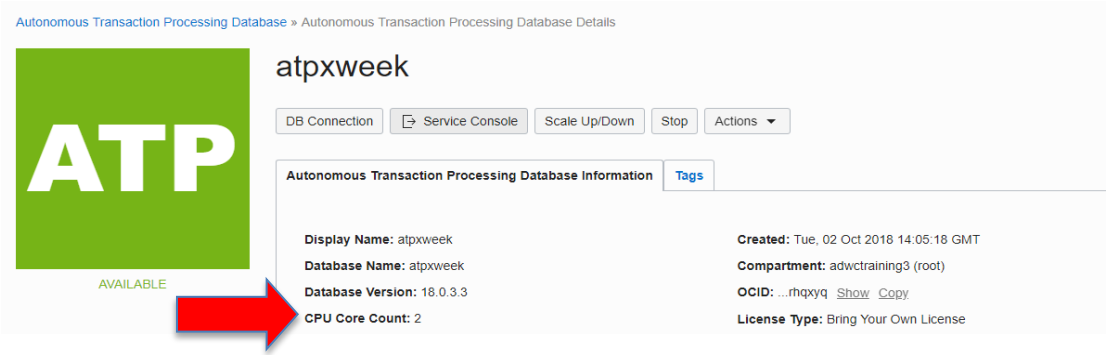
```
./charbench -c ../configs/SOE_Server_Side_V2.xml \
            -cf /c/wallets/wallet_atpxweek.zip \
            -cs atpxweek_tpurgent \
            -u soe \
            -p Atpxweek2018 \
            -v users,tps,vresp \
            -intermin 0 \
            -intermax 0 \
            -min 0 \
            -max 0 \
            -uc 64 \
```

Your output will look similar to the screenshot below. The columns indicate the wall clock time, the number of users connected, the transactions per second (TPS), and number of transactions per type on the following columns. There are many other parameters that can be included to show more information.

```
$ ./charbench -c ../configs/SOE_Server_Side_V2.xml \
>             -cf /c/wallets/wallet_atpxweek.zip \
>             -cs atpxweek_high \
>             -u soe \
>             -p Atpxweek2018 \
>             -v users,tps,vresp \
>             -uc 64 \
>             -rt 0:0.30
Author  :        Dominic Giles
Version :        2.6.0.1090

Results will be written to results.xml.
Hit Return to Terminate Run...

Time     Users       TPS     NCR   UCD   BP    OP    PO    BO    SQ    WQ    WA
12:23:44 [0/64]      0       0     0     0     0     0     0     0     0     0
```

To show how scaling ATP is an online operation and how workloads are immediately affected by scaling the environment run the workload generator in continuous mode and scale the system up and down while it is running. We will monitor the TPS column to see how the transactions per second varies depending on the number of CPU's allocated to the environment. Remember that in this case all workload is being generated through the **_tpurgent** service which will use all resources available. Other services may display different results. Experiment with different services by changing the **–cs parameter** when launching the load generator. In this example we start with 2 CPU's, will scale up to 4 CPU's and then scale down to 1 CPU. For a refresher on scaling refer to Lab 4.

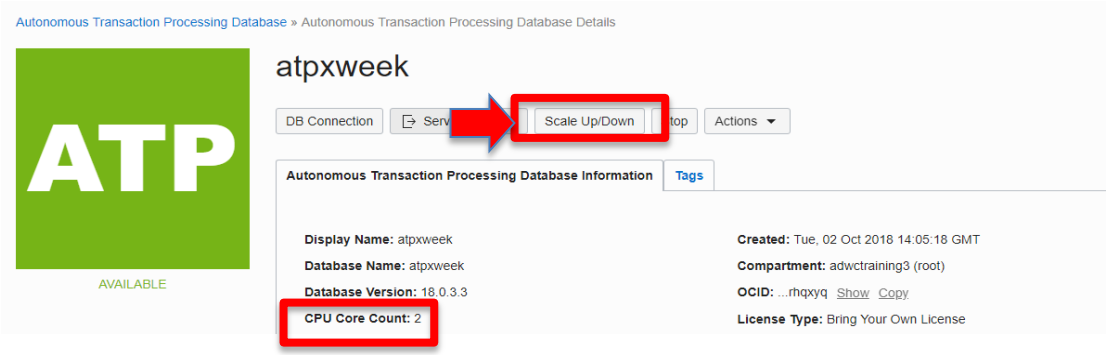
In your service console verify that you are starting with 2 CPU's



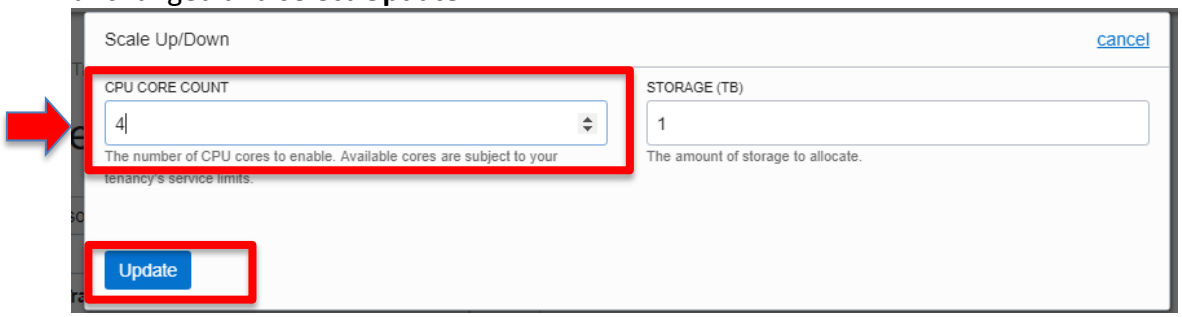
Launch the workload generator with the following command on your BASH window (make sure to use your wallet information):

```
./charbench -c ../configs/SOE_Server_Side_V2.xml \
            -cf /c/wallets/wallet_atpxweek.zip \
            -cs atpxweek_tpurgent \
            -u soe \
            -p Atpxweek2018 \
            -v users,tps,vresp \
            -uc 64
```

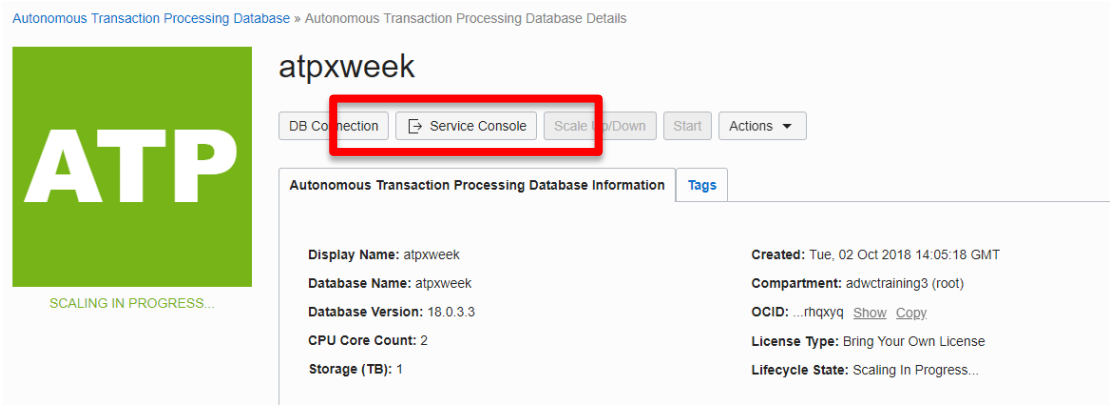
The output will start being generated in the BASH window. While the workload generator is running go to the service console and scale your environment from 2 to 4 CPU's



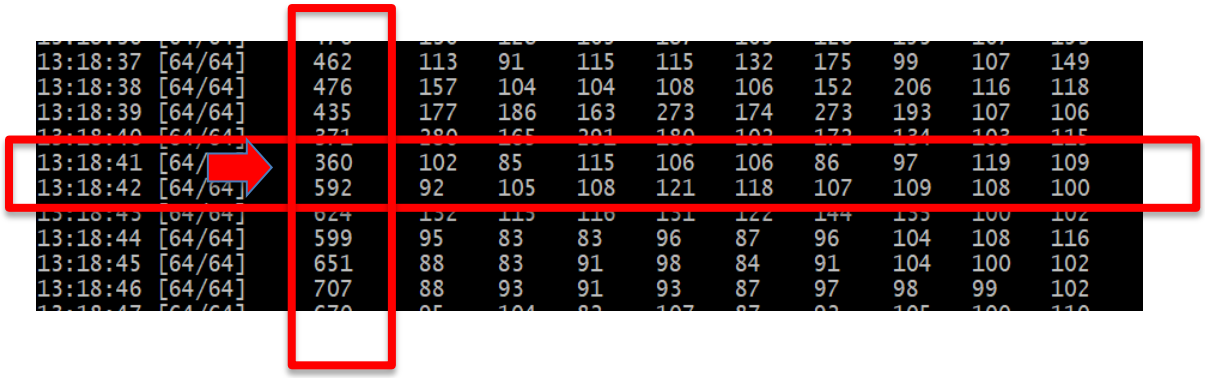
In the Scale Up/Down screen, enter 4 in the CPU CORE COUNT, leave STORAGE unchanged and select **Update**



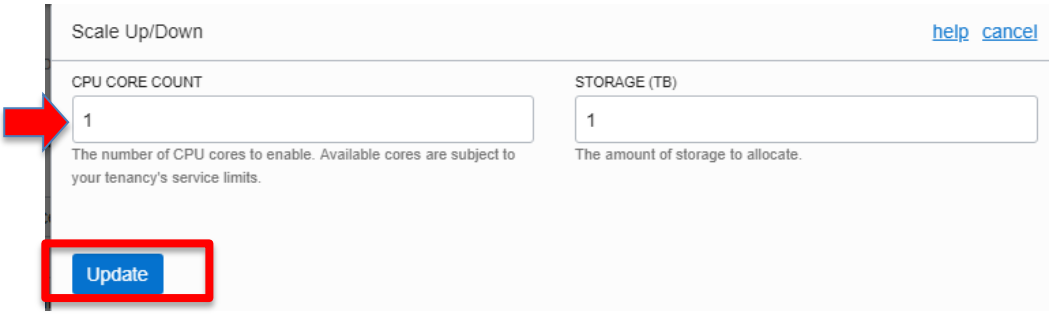
The Service Console will show scaling:



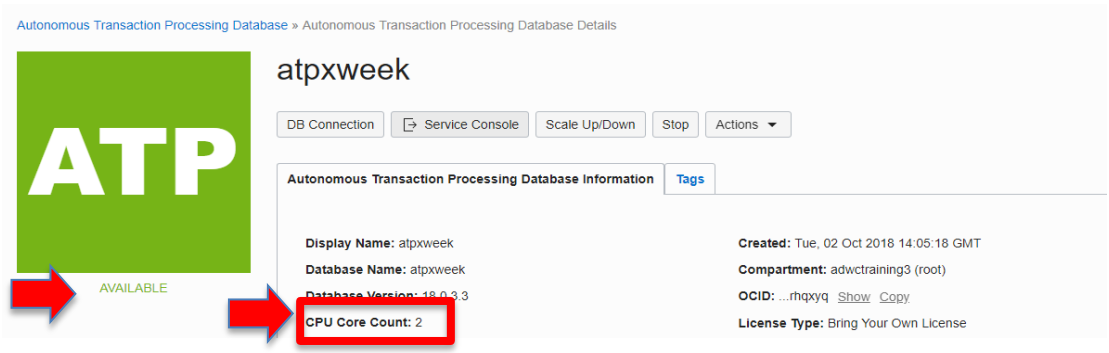
While the scaling is occurring go back to your BASH window and look at the TPS (transactions per second) during the scaling process that number will **increase** when the new CPU's are allocated automatically. Notice the processing never ends. It will be very evident when the additional CPU's are enabled.



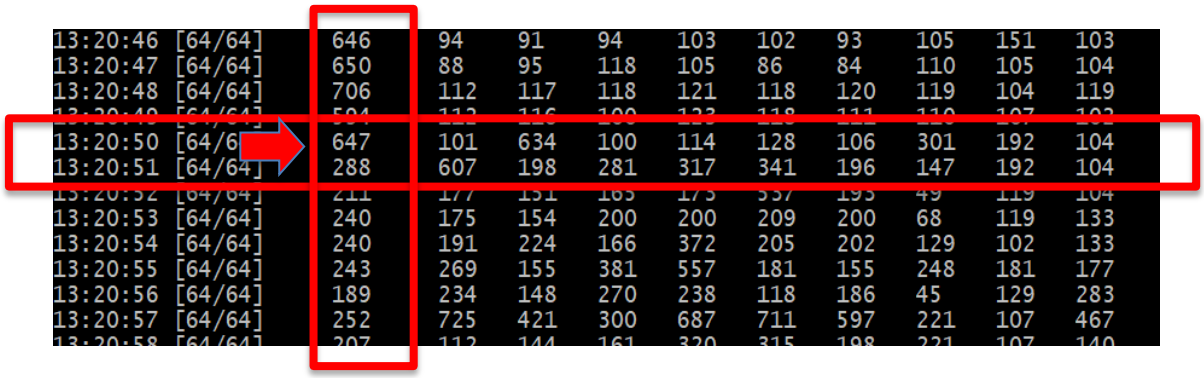
Now repeat the process and scale the environment down to 1 CPU:



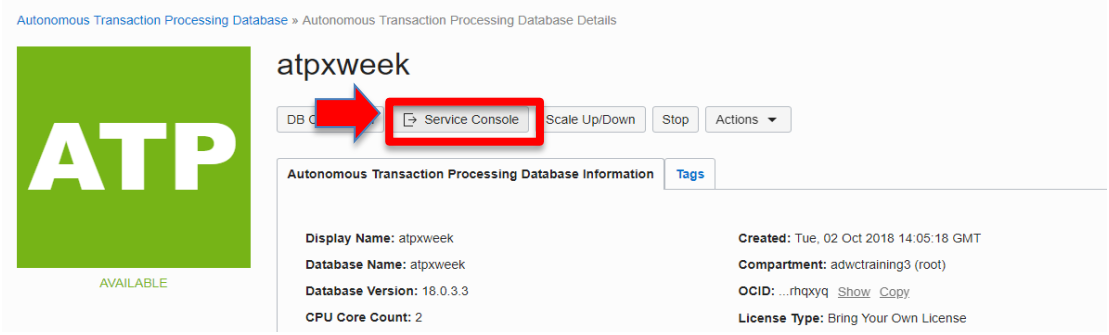
eugenio.galiano@oracle.com

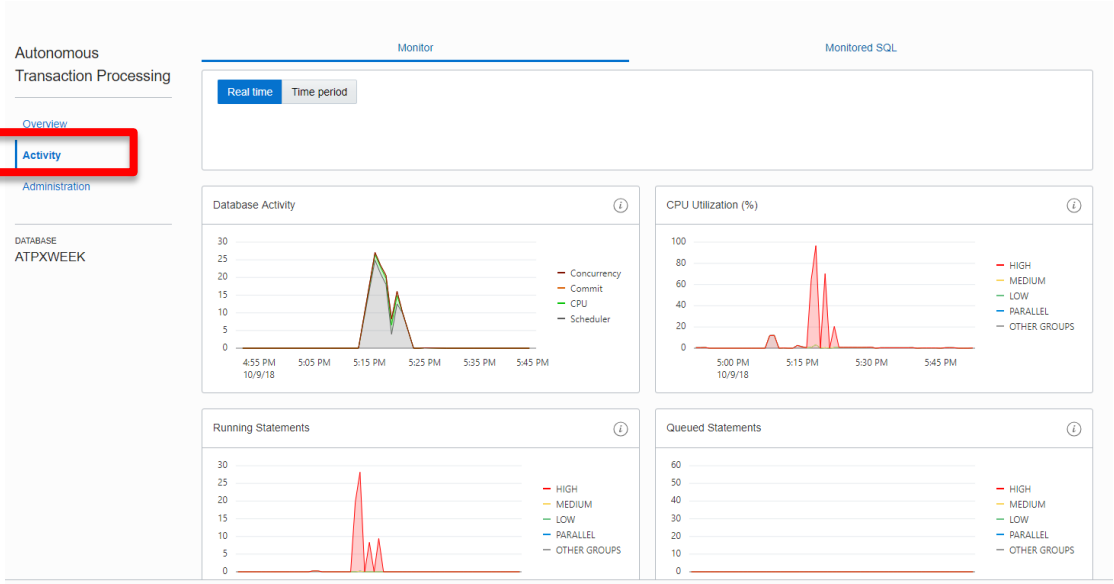


Again, while the scaling is occurring go back to your BASH window and look at the TPS (transactions per second) during the scaling process that number will **decrease** when the extra CPU's are deallocated. Notice the processing never ends. It will be very evident when the additional CPU's are disabled.



Also examine the database workload through the Service Console (refer to previous lab for more details how to do this)





When you are done with this lab, it is very important that you stop the workload generator by hitting Ctrl-C on your BASH Window. Also stop your ATP instance or scale down to 1 CPU.

eugenio.galiano@oracle.com

Lab 8. Using Node.js with ATP to run Javascript

Objectives

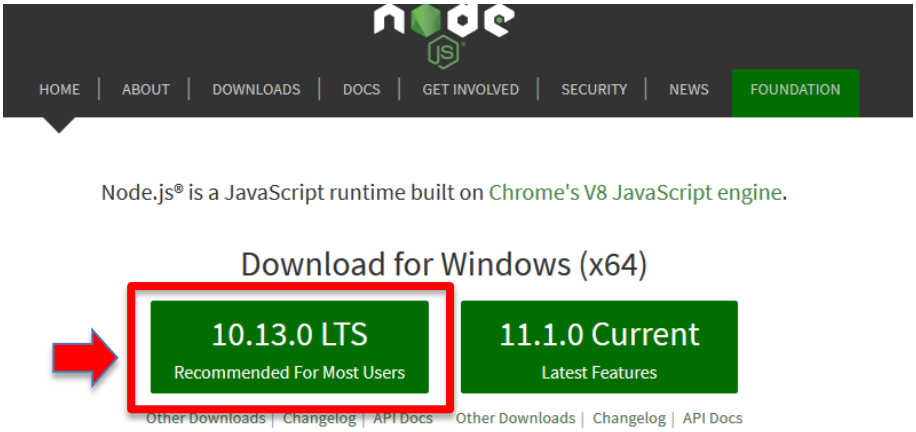
- Install Node.js on your computer
- Run basic node.js code that interacts with an Oracle ATP Database

The objective of this lab is to get a node.js environment running on your Computer that is capable of connecting and running database operations against an Oracle Autonomous Transaction Processing Database. It is not an objective to teach you to code in Javascript. Javascript is a very popular language with developers and its important to communicate to customers that ATP supports working with Javascript.

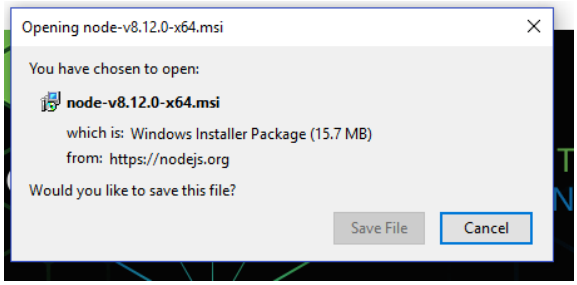
For our Lab we will be using the node.js environment, one of the most common ones for building Javascript applications. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications.

8.1 Installing Node.js

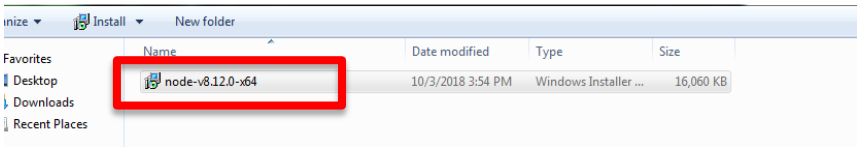
The first step is to install Node.js on your computer. To download the software go to nodejs.org Once there select "**10.13.0 LTS Recommended For Most Users**", see below (please note there may be a newer version, always select the "**Recommended For Most Users"** version:



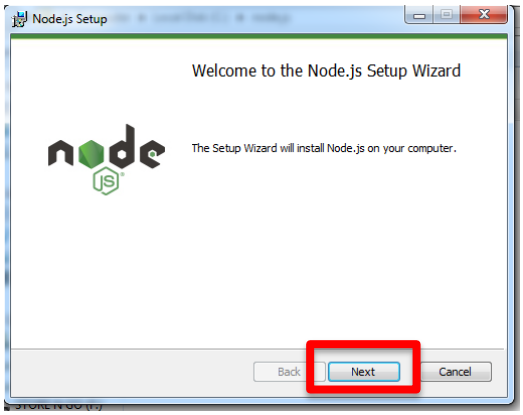
You will download the **.msi** file. Make sure know the directory of the file download



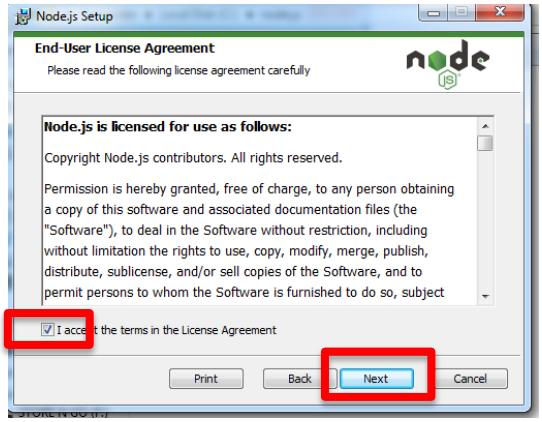
Go to the download directory and double click on the file



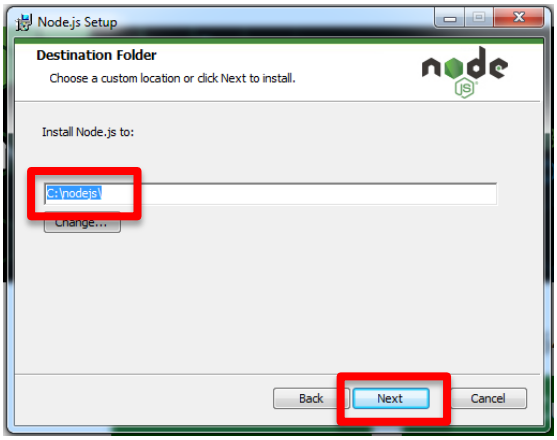
This will bring up the installation screen, click **Next**



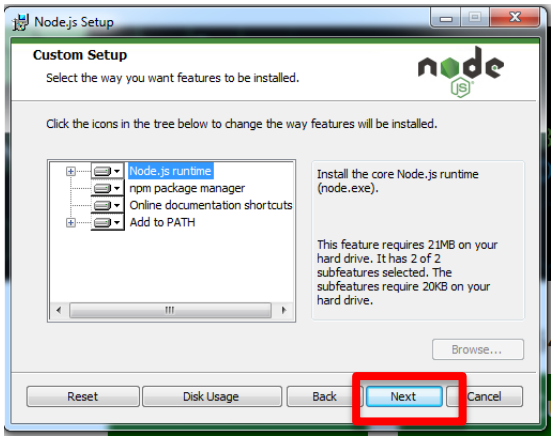
Accept the terms and click **Next**



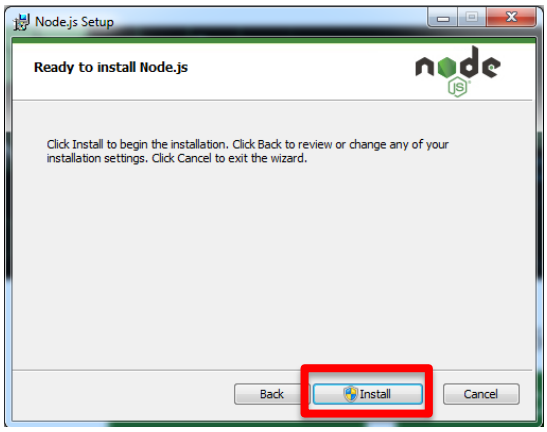
Select the installation directory, for simplicity install on **c:\nodejs** and click **Next**



No change, click **Next**

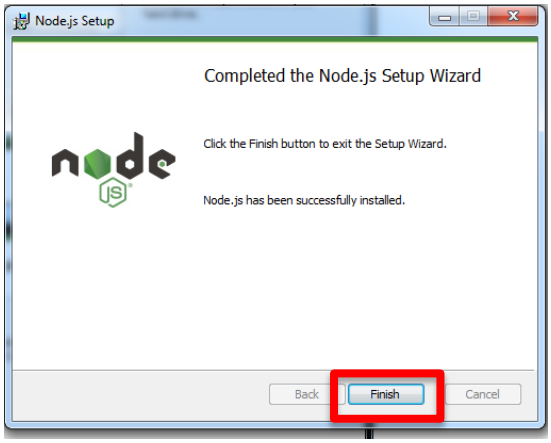


Click **Install**



eugenio.galiano@oracle.com

Click **Finish**



Open up a Command Prompt in Windows and navigate to the directory on which you installed nodejs (in this example **c:\nodejs**) and run the following commands:

node –v
npm –v

These commands will print the version of node and npm installed and they should match the versions below

Copy the following code into a text file name "**test.js**" and save the file into the same directory you installed nodejs in (use notepad or other editor):

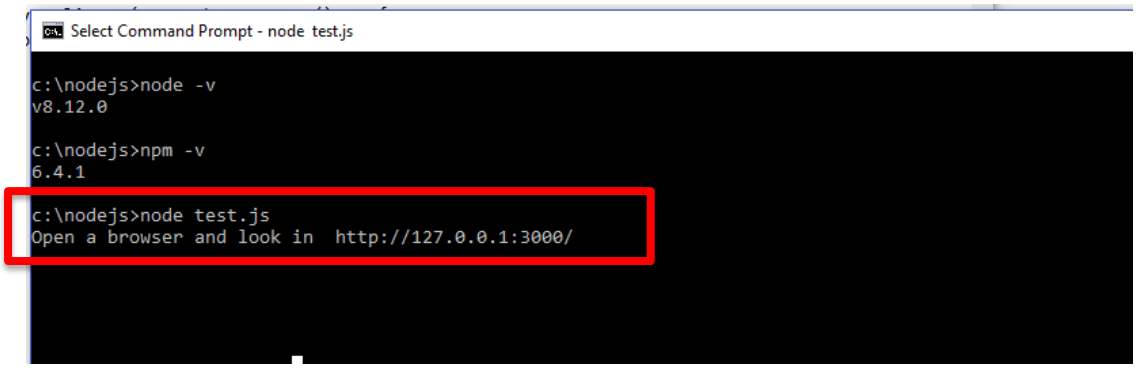
```
const http = require('http');
```


eugenio.galiano@oracle.com

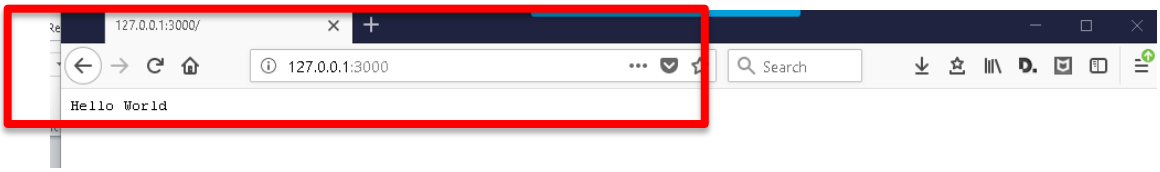
```
const hostname = '127.0.0.1';
const port = 3000;

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

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

Now run your first node.js application. This application will simply print a message on a web browser. So you will need to open a separate tab or web browser and go to the address specified below.
In your command prompt window, in the directory where you installed node.js and saved the test.js file (same one you ran the commands above), run the following command:

```
node test.js
```



Open up a browser and in the address bar enter: http://127.0.0.1:3000/
You will see your first node.js application running!



To terminate your node.js program, **hit Ctrl-C** in your command prompt window.

8.2 Configuring Node.js for to use with ATP

Now we need to install several node.js libraries that will be used to run the lab node.js code. The libraries are installed with the **npm** utility that is part of the node.js environment. Run the following commands in your command prompt window from the directory where you installed nodejs (c:\nodejs):

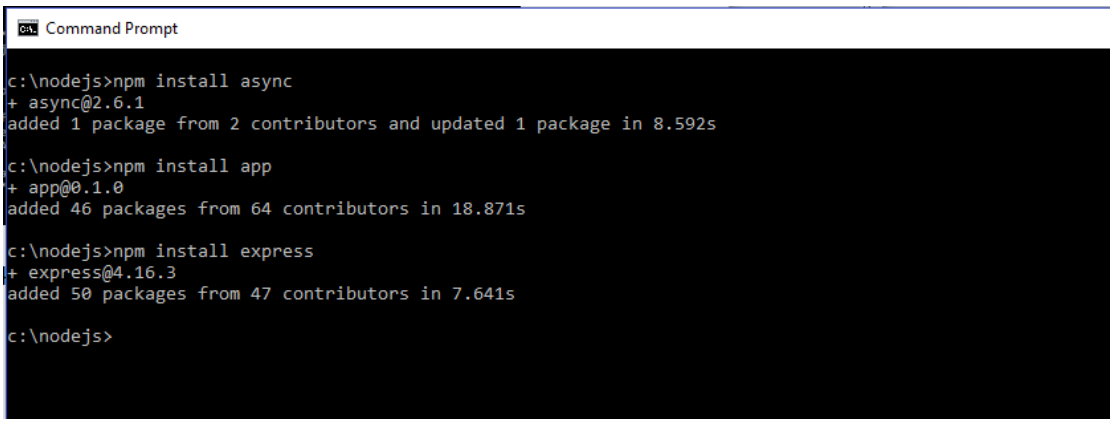
- Install the Oracle Database libraries – run:
npm install oracledb

Install async libraries functions for working with asynchronous JavaScript – run:
npm install async

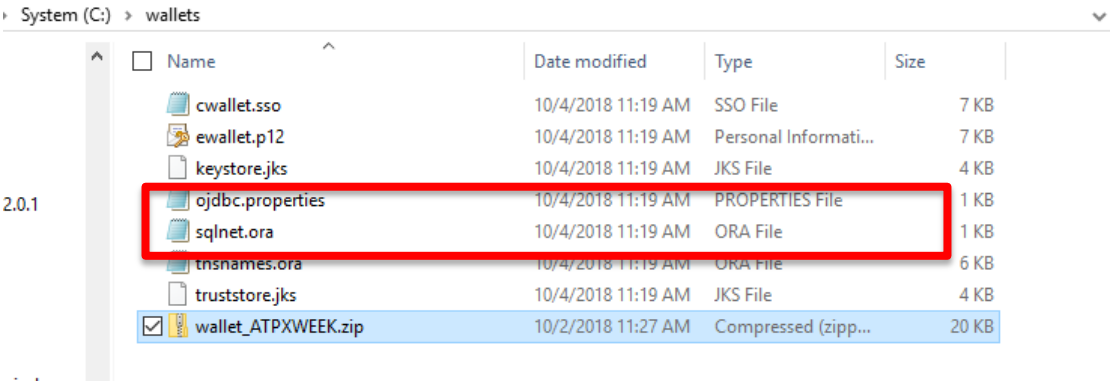
Install app libraries for web application development – run:
npm install app

Install express a set of features for web and mobile applications – run:
npm install express

Below is a screenshot of the library installations



To connect a node.js application to a database the wallet files that were downloaded in Lab 2 will be used. Move the wallet to the **c:\wallets** directory (create the directory if it does not exist). Unzip the zip wallet file in that directory.



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

Edit the **ojdbc.properties** (open with notepad or your favorite editor) to reflect the location of the wallet file. The line entry in the file should be as below, where c:\wallets should be the directory where your wallet files are located:

```
oracle.net.wallet_location=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY="c:\wallets")))
```

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

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="c:\wallets")))
SSL_SERVER_DN_MATCH=yes
```

You must set your TNS_ADMIN environment variable every time you open a Command Prompt. The TNS_ADMIN variable needs to point to the directory where the extracted wallet is located. In this case its "c:\wallets"

set TNS_ADMIN=c:\wallets

Create a directory where all your node.js scripts will be stored. For example scripts under the previously created c:\nodejs directory. So the directory would be: C:\nodejs\scripts

In node scripts you can specify database credentials within the script, but it's easier to create one file that contains the credentials and then call this file in your node script. That way you can independently change the credentials without having to update your program.
Create a common file containing database credentials that will be called from all node.js scripts we run. Call the file dbconfig.js Copy and paste the lines below into the file (replace *italics* with your specific information, do not change anything else):

```
module.exports= {

dbuser: '$databaseuser',
dbpassword: '$databasepassord',
connectString: 'atpxweek_TP'
}
```

dbuser = admin (or another user account if you created another use)
dbpassword = admin's (or other users) password
connectString = your database **_TP** service name

8.3 Connect to an ATP Database with Node.js

The first node.js script we create will connect to the database and if successful will display the username used to connect on a browser at address http://localhost:3050/ Copy the code below and paste it into a file called connectatp.js This file needs to be in the same directory created above (C:\nodejs\scripts).

Please note the first few lines calls node libraries including the oracledb library. The third line calls the dbconfig.js created above with the database credentials. It is important to make sure the file is in the same directory or the app won't find database credentials.

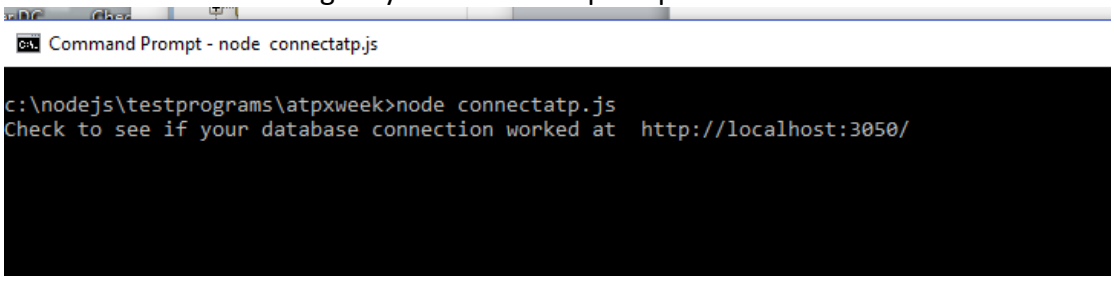
```
var http = require('http');
var oracledb = require('oracledb');
var dbConfig = require('./dbconfig.js');
let error;
let user;

    oracledb.getConnection({
        user: dbConfig.dbuser,
        password: dbConfig.dbpassword,
        connectString: dbConfig.connectString
    },
    function(err, connection) {
        if (err) {
            error = err;
            return;
        }
        connection.execute('select user from dual', [],
function(err, result) {
            if (err) {
                error = err;
                return;
            }
            user = result.rows[0][0];
         console.log(`Check to see if your database
connection worked at  http://localhost:3050/`);
            error = null;
            connection.close(function(err) {
                if (err) {
                    console.log(err);
                }
            });
        })
    }
);


http.createServer(function(request, response) {
    response.writeHead(200, {
        'Content-Type': 'text/plain'
    });
    if (error === null) {
        response.end('Connection test succeeded. You
connected to ATP as ' + user + '!');
    } else if (error instanceof Error) {
```

```
        response.write('Connection test failed. Check the
settings and redeploy app!\n');
        response.end(error.message);
    } else {
        response.end('Connection test pending. Refresh
after a few seconds...');
    }
}).listen(3050);
```

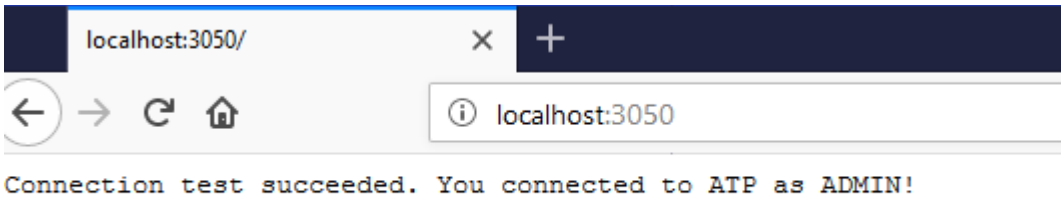
To run the code, in the command prompt, go to the directory where you saved the file and run the following command (the name after node is the name of the file you just created with the code):

node connectatp.js

You will see the following on your command prompt:



Go to your browser and the address http://localhost:3050/ and you will see:



8.4 Node.js script to select from an ATP database

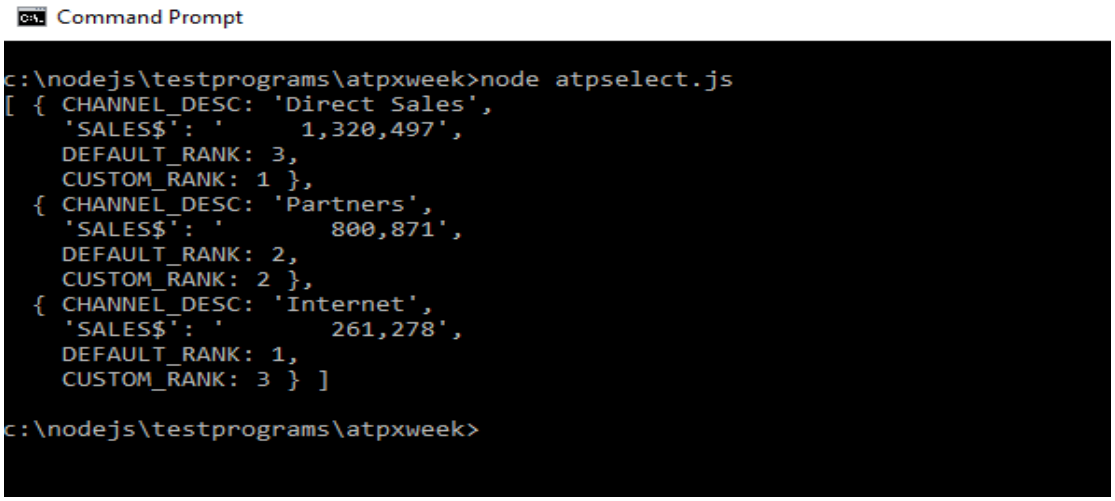
In this node.js script you will run the same query ran in Lab 2 on the SH schema to test SQL through node.js. Copy the code below and place it in a file called atpselect.js Notice again the library calls at the beginning and some Oracle specific parameters such as limiting maximum rows returned, etc.

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

// THIS uses Oracle friendly formatting for results
oracledb.outFormat = oracledb.OBJECT;

// THIS limits the number of rows returned
oracledb.maxRows = 50;

    oracledb.getConnection({
        user: dbConfig.dbuser,
        password: dbConfig.dbpassword,
        connectString: dbConfig.connectString
    },
  function(err, connection)
  {
    if (err) { console.error(err); return; }
    connection.execute(
      `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`,
      function(err, result)
      {
        if (err) { console.error(err); return; }
        console.log(result.rows);
      });
  });
```

To run the code, in the command prompt, go to the directory where you saved the file and run the following command (the name after node is the name of the file you just created with the code):

node atpselect.js

You will see the following on your command prompt:



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

Bind variables are commonly used in node.js scripts and supported with the Oracle database. The following example shows how to bind a variable within the node.js code. Copy the code below and place it in a file called atpselectwithbindvariable.js Notice again the section on binding the variable and also on releasing connections at the end of the script so resources are not wasted.

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

    oracledb.getConnection({
        user: dbConfig.dbuser,
        password: dbConfig.dbpassword,
        connectString: dbConfig.connectString
    },
  function(err, connection) {
    if (err) {
      console.error(err.message);
      return;
    }
    connection.execute(
      // The statement to execute
```

```
      `SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME
       FROM sh.customers
       WHERE CUST_ID = :id`,

      // The "bind value" 5992 for the bind variable
":id", so this customer# will be returned
      [5992],

      { outFormat: oracledb.OBJECT  // query result
format

      },

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

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

To run the code, in the command prompt, go to the directory where you saved the file and run the following command (the name after node is the name of the file you just created with the code):

node atpselectwithbindvariable.js

Check the results on your command prompt.

8.6 Using JSON data in an ATP database

In this app you will drop and create the table `j_purchaseorder`. You can create a table that has JSON columns. You use SQL condition `is json` as a check constraint to ensure that data inserted into a column is (well-formed) JSON data. Oracle recommends that you always use an `is_json` check constraint when you create a column intended for JSON data. In this example you will use the constraint ensure_json and is_json check. Copy the following code into a file called atpcreatejsontable.js

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

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

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

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

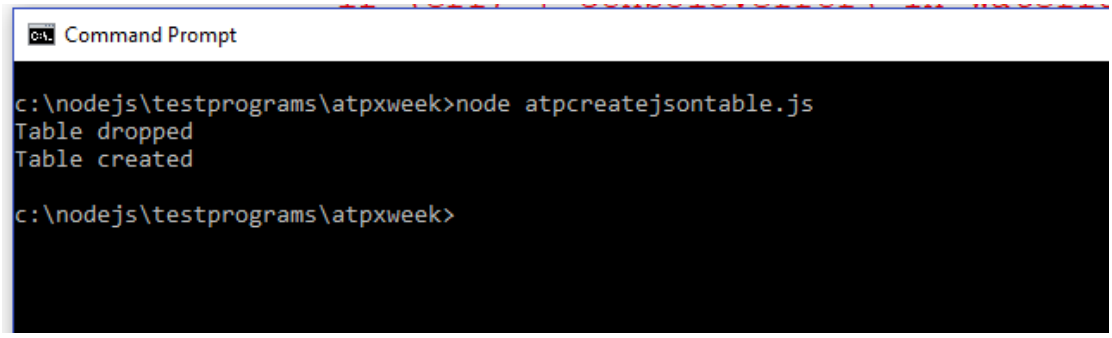

eugenio.galiano@oracle.com

```
        console.log("Table dropped");
        return cb(null, conn);
      }
    });
};

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

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

To run the code, in the command prompt, go to the directory where you saved the file and run the following command (the name after node is the name of the file you just created with the code):

```
node atpcreatejsontable.js
```

The output will simply be two lines in the command prompt indicating the table got dropped and created. But examine the syntax around table manipulation and JSON data.



In this app you will build on the previous app and manipulate JSON data into Oracle Objects for updating, analysis, inserts and deletes from the database. Copy the following code into a file called atpselectjson.js

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

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

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

var dodrop = function (conn, cb) {
  conn.execute(
    `BEGIN
       EXECUTE IMMEDIATE 'DROP TABLE j_purchaseorder';
       EXCEPTION WHEN OTHERS THEN
       IF SQLCODE <> -942 THEN
         RAISE;
       END IF;
     END;`,
    function(err) {
      if (err) {
```

```
        return cb(err, conn);
      } else {
        console.log("Table dropped");
        return cb(null, conn);
      }
    });
};

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

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

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

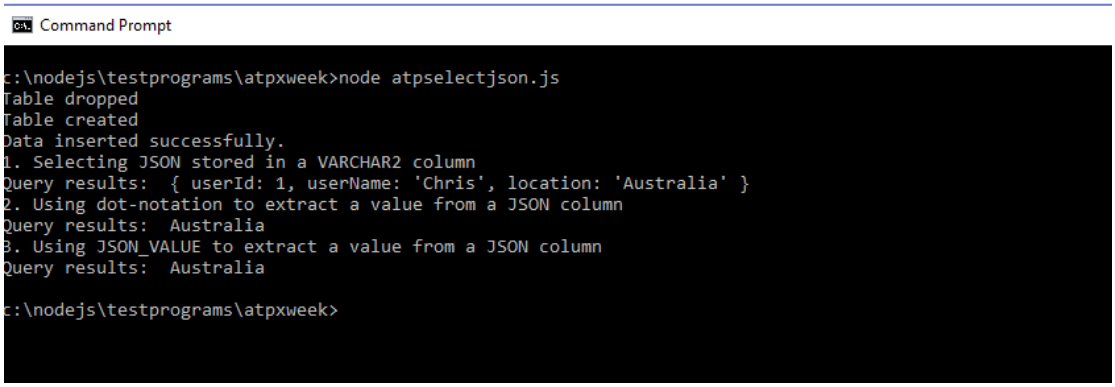
```
        return cb(null, conn);
      }
    });
};

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

// 2. Extract a value from a JSON column.  This syntax
requires Oracle Database 12.2
var dorelationalquerydot = function (conn, cb) {
  console.log('2. Using dot-notation to extract a value
from a JSON column');
  conn.execute(
    "SELECT po.po_document.location FROM j_purchaseorder
po",
    function(err, result) {
      if (err) {
        return cb(err, conn);
      } else {
        console.log('Query results: ', result.rows[0][0]);
// just show first record
        return cb(null, conn);
      }
    });
};

// 3. Using JSON_VALUE to extract a value from a JSON
column
var dorelationalquery = function (conn, cb) {
```

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

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

To run the code, in the command prompt, go to the directory where you saved the file and run the following command (the name after node is the name of the file you just created with the code):

```
node atpselectjson.js
```

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.



This concludes the Node.js Lab. Again, the objective of this lab is to set up a working node.js environment with sample code that can be deployed against an ATP database and demonstrate how that is done.

Lab 9. Using Python with ATP

Objectives

- Learn how to install Python on your computer
- Run basic Python code that interacts with an Oracle ATP Database

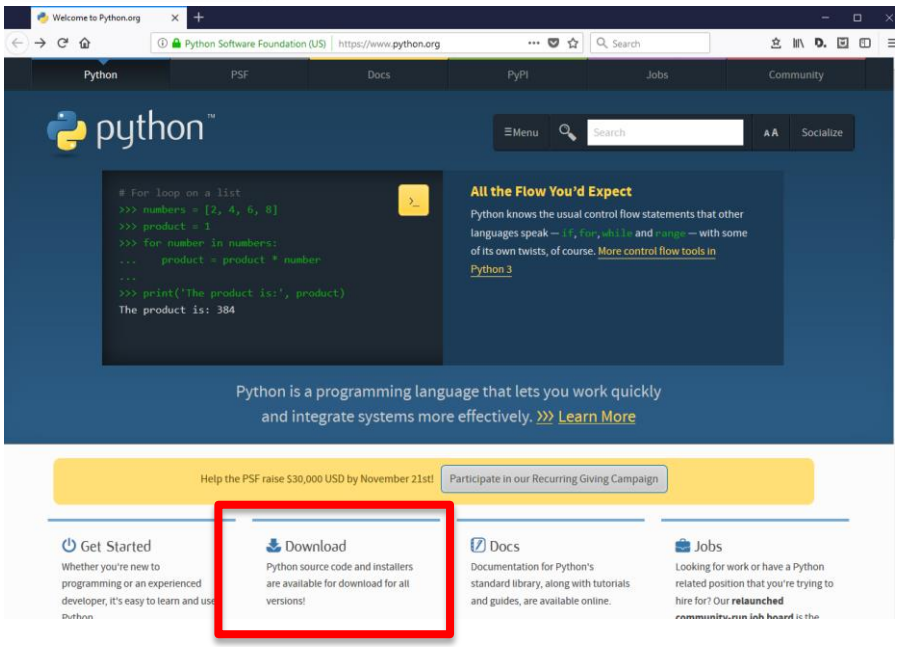
The objective of this lab is to get Python running on your Computer and connecting and running database operations against an Oracle Autonomous Transaction Processing Database in Python. It is not an objective to teach you to code in Python. Python is a very popular language with developers and it's important to communicate to customers that ATP supports working with Python like any other Oracle Database

9.1 Installing Python

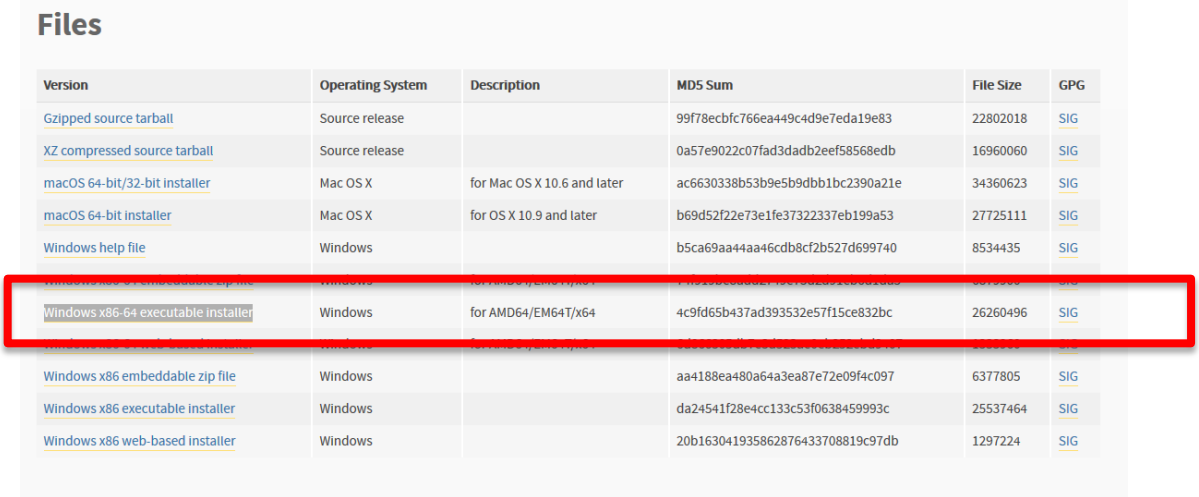
The first step is to install Python. Please note that Python is very sensitive to other installed versions and PATH's associated with previous installations on your computer. If you have other versions of Python installed, remove them and any path's and projects associated with them or this installation may not work. Alternatively if you have a working Python environment you can attempt to follow the lab and skip this installation section.

Download the software from https://www.python.org/

Click on Download (Current version as of this lab is 3.7.1)



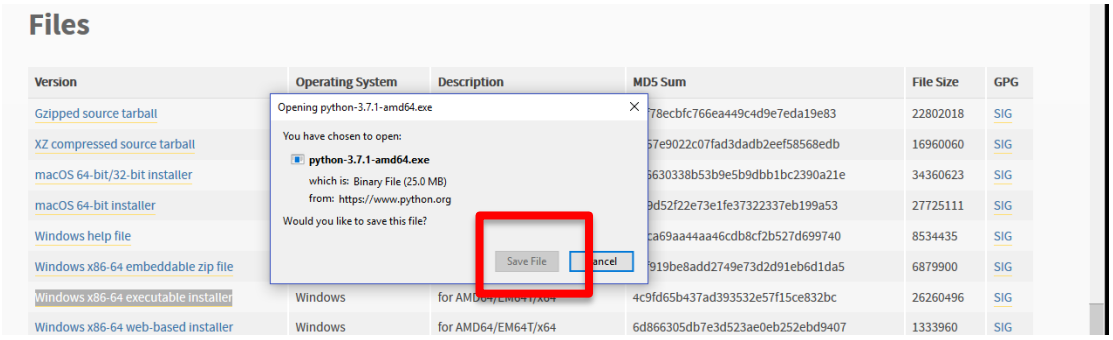
Scroll down and select the distribution for your computer, in this case [Windows x86-64 executable installer]:



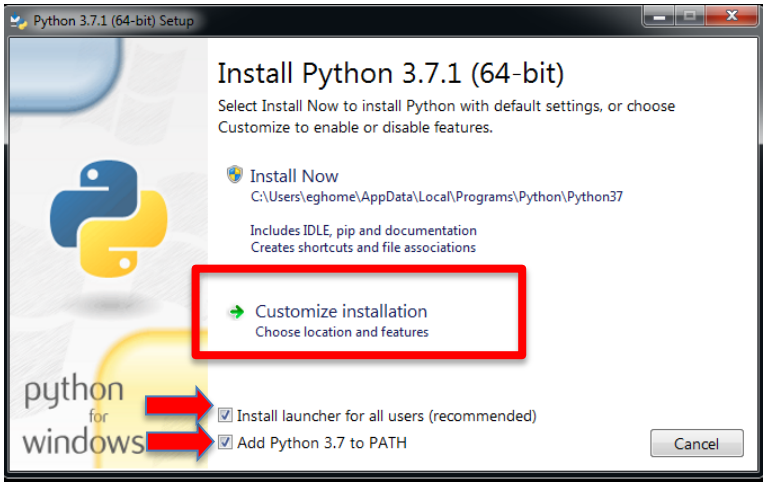
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		99f78ecbfc766ea449c4d9e7eda19e83	22802018	SIG
XZ compressed source tarball	Source release		0a57e9022c07fad3dadb2eef58568edb	16960060	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	ac6630338b53b9e5b9dbb1bc2390a21e	34360623	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	b69d52f22e73e1fe37322337eb199a53	27725111	SIG
Windows help file	Windows		b5ca69aa44aa46cdb8cf2b527d699740	8534435	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	4c9fd65b437ad393532e57f15ce832bc	26260496	SIG
Windows x86 embeddable zip file	Windows		aa4188ea480a64a3ea87e72e09f4c097	6377805	SIG
Windows x86 executable installer	Windows		da24541f28e4cc133c53f0638459993c	25537464	SIG
Windows x86 web-based installer	Windows		20b16304193586287643370881 9c97db	1297224	SIG

Save the file to your downloads directory and then execute it:

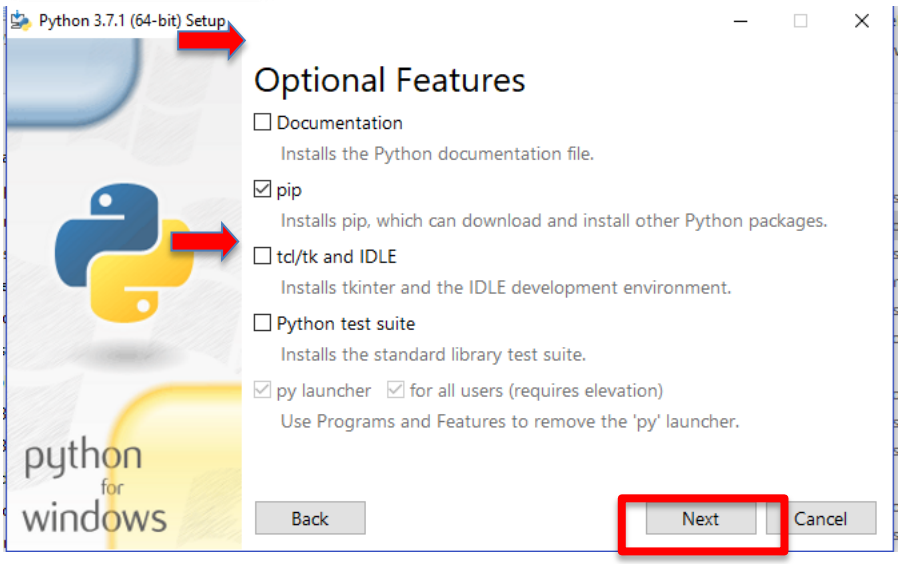


Click **Install launcher for all users**, and **Add Python 3.7 to PATH**. Select **Customize Installation:**

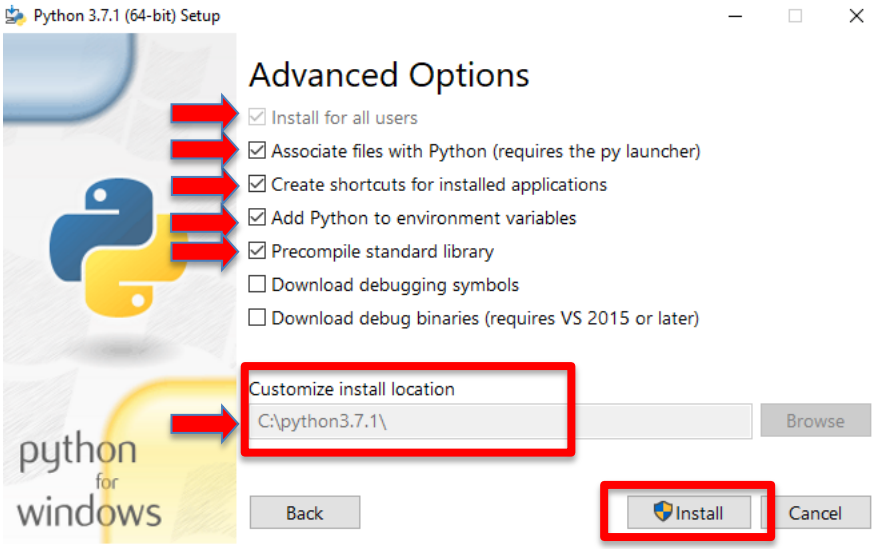


Under Optional Features select **pip, py launcher, and for all users** (see below)

eugenio.galiano@oracle.com



Under Advanced Options choose all options except the last 2. Make sure **Install for all users,** and **Add Python to environment Variables** are checked. Also **under Customize Install Location,** select **c:\python3.7.1 Do NOT** install in the default directory. Click **Install**

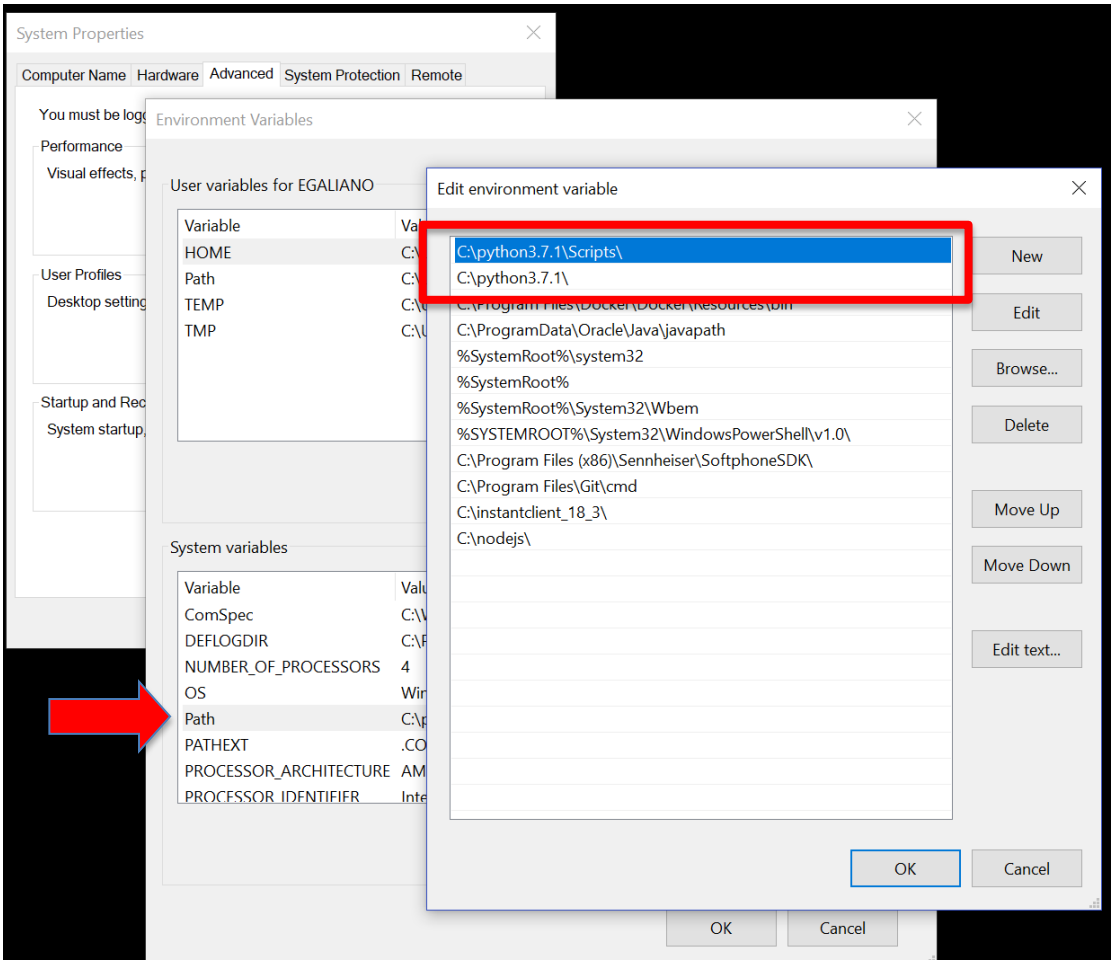


When done, exit the installer. You have installed Python.

Its important to confirm that your PATH has been set correctly othewise Python won't work.

In Windows 10:

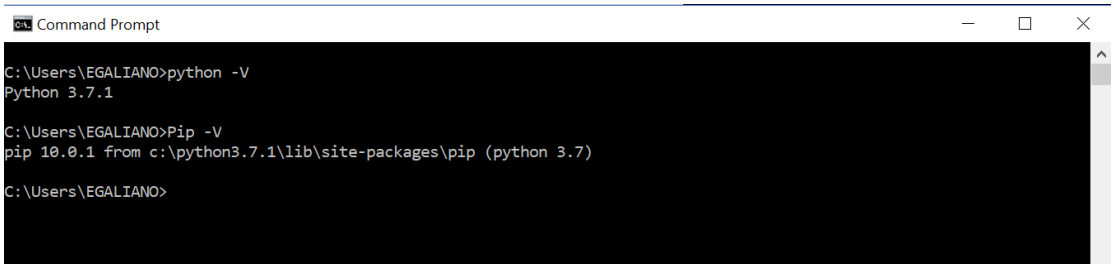
1. In Search, search for and then select: **Advanced System Settings** (control panel)
2. Click **Environment Variables** at bottom of screen
3. In the **System variables** double click **Path**
4. Make sure the Python directories are in the path (see below)
5. If you do not see Python in your path, add both entries as you see below by clicking the New button and manually adding them



Now verify that Python and Pip are working. Open up a Command Prompt and run the following commands to verify **Python** and **Pip** versions (see screenshot):

```
python -V
pip -V
```


eugenio.galiano@oracle.com



9.2 Configuring Python to run with ATP and sample scripts

For Python to work with ATP, the Python Oracle libraries must be installed, Run the following command from you Command Prompt to install the libraries (ignore any warnings about pip):

```
python -m pip install cx_Oracle --upgrade
```

Set the TNS_ADMIN parameter to the location of the wallet (in this case c:\wallets):

```
set TNS_ADMIN=c:\wallets
```

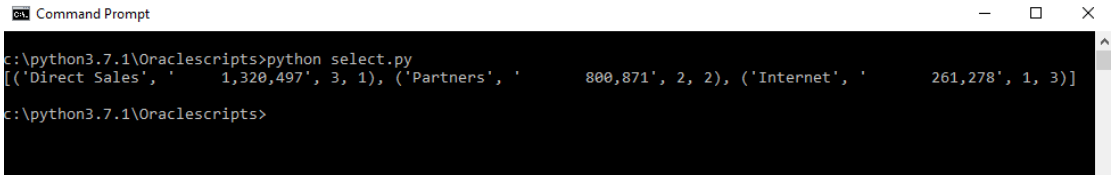
Execute a simple select (same select as on previous labs) through a Python script. Copy and paste the following code into a file called **select.py** in directory you installed python. Change the connection credentials in the con line below to reflect your account/password@connectstring information.

```
import cx_Oracle
con = cx_Oracle.connect('admin/Atpxweek2018@atpxweek_TP')

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='US' \
GROUP BY channel_desc"

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

```
print (result)
cur.close()
con.close()
```

Once you have placed the code in the file, execute it from the directory where the file is located:

```
python select.py
```

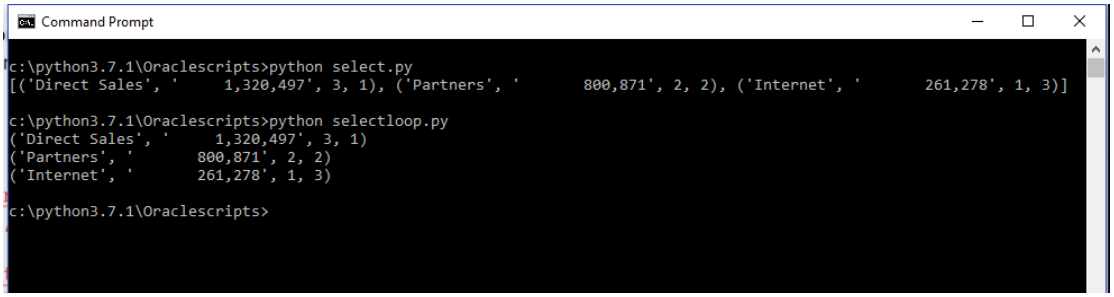


A slight variation to the script will allow the results to be displayed in a loop one a time in a new line until all results are displayed. Unlike the previous code which prints all the results at once. Copy and paste the code into a file named selectloop.py and run it by running the command :

```
python selectloop.py
```

Remember to change the connection credentials in the con line below to reflect your account/password@connectstring information.

```
import cx_Oracle
con = cx_Oracle.connect('admin/Atpxweek2018@atpxweek_TP')
cur = con.cursor()
cur.execute ("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")
for result in cur:
     print (result)
cur.close()
con.close()
```



To understand different ways to query and display results in python, the following script performs a query and then displays results in three different ways. The output of the scripts provides a brief description of the results display process. Copy and paste the code into a file named querytypes.py and run it by running the command:

```
python querytypes.py
```

Remember to change the connection credentials in the con line below to reflect your account/password@connectstring information.

```
from __future__ import print_function

import cx_Oracle

con = cx_Oracle.connect('admin/Atpxweek2018@atpxweek_TP')

sql = """select c_city,c_region,count(*)
from ssb.customer c_high
group by c_city,c_region
order by count(*)"""

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

print("Query one row at a time - do it twice and show 2
results")
cursor.execute(sql)
row = cursor.fetchone()
print(row)
row = cursor.fetchone()
print(row)
```

```
print()

print("Fetch many rows - in this case 3 rows displayed
all at once")
cursor.execute(sql)
res = cursor.fetchmany(numRows=3)
print(res)
```

```
c:\python3.7.1\Oraclescripts>python querytypes.py
Get all rows via iterator - display all results one row at a time - this may take a few minutes
('UNITED KI1', 'EUROPE      ', 119082)
('UNITED ST4', 'AMERICA     ', 119245)
('MOZAMBIQU2', 'AFRICA      ', 119283)
('INDIA    2', 'ASIA        ', 119380)
('ETHIOPIA 5', 'AFRICA      ', 119393)
('BRAZIL   6', 'AMERICA     ', 119393)
('KENYA    6', 'AFRICA      ', 119415)
('CANADA   6', 'AMERICA     ', 120784)
('EGYPT    4', 'MIDDLE EAST ', 120810)

Query one row at a time - do it twice and show 2 results
('UNITED KI1', 'EUROPE      ', 119082)
('UNITED ST4', 'AMERICA     ', 119245)

Fetch many rows - in this case 3 rows displayed all at once
[('UNITED KI1', 'EUROPE      ', 119082), ('UNITED ST4', 'AMERICA     ', 119245), ('MOZAMBIQU2', 'AFRICA      ', 119283)]

c:\python3.7.1\Oraclescripts>
```

Next we will create a table using python. In the script we will first drop the table in case its already there and then create the table. Copy and paste the code into a file named createtable.py and run it by running the command :

```
python createtable.py
```

Remember to change the connection credentials in the con line below to reflect your account/password@connectstring information.

```
import cx_Oracle
con = cx_Oracle.connect('admin/Atpxweek2018@atpxweek_TP')

droptable = "DROP TABLE COUNTRIES"

createtable = "CREATE TABLE COUNTRIES ( \
  COUNTRY_ID            NUMBER       NOT NULL, \
  COUNTRY_ISO_CODE      CHAR(2)      NOT NULL, \
  COUNTRY_NAME          VARCHAR2(40) NOT NULL, \
  COUNTRY_SUBREGION     VARCHAR2(30) NOT NULL, \
  COUNTRY_SUBREGION_ID  NUMBER       NOT NULL, \
  COUNTRY_REGION        VARCHAR2(20) NOT NULL, \
  COUNTRY_REGION_ID     NUMBER       NOT NULL, \
  COUNTRY_TOTAL         NUMBER(9)    NOT NULL, \
  COUNTRY_TOTAL_ID      NUMBER       NOT NULL, \
```

```
  COUNTRY_NAME_HIST    VARCHAR2(40) \
)"

cur = con.cursor()
cur.execute(droptable)
print ("table droped")
cur.execute(createtable)
print ("table created")
cur.close()
con.close()
```

```
c:\python3.7.1\Oraclescripts>python createtable.py
table droped
table created

c:\python3.7.1\Oraclescripts>
```

In the next script you will insert data into the table and then update the data just inserted. The output will be the data inserted and then the updated data. We will also drop and recreate the table so every time you run this script the result is the same. Copy and paste the code into a file named insertandupdate.py and run it by running the command :

```
python insertandupdate.py
```

Remember to change the connection credentials in the con line below to reflect your account/password@connectstring information.

```
import cx_Oracle
con = cx_Oracle.connect('admin/Atpxweek2018@atpxweek_TP')
droptable = "DROP TABLE COUNTRIES"
createtable = "CREATE TABLE COUNTRIES ( \
  COUNTRY_ID           NUMBER, \
  COUNTRY_ISO_CODE     CHAR(2), \
  COUNTRY_NAME         VARCHAR2(40), \
  COUNTRY_SUBREGION    VARCHAR2(30), \
  COUNTRY_SUBREGION_ID NUMBER, \
  COUNTRY_REGION       VARCHAR2(20), \
  COUNTRY_REGION_ID    NUMBER, \
  COUNTRY_TOTAL        NUMBER(9), \
  COUNTRY_TOTAL_ID     NUMBER, \
  COUNTRY_NAME_HIST    VARCHAR2(40) \
)"

#here we create the table
cur = con.cursor()
cur.execute(droptable)
```

```
print ("table droped")
cur.execute(createtable)
print ("table created")

#here we insert data into the table and query it
cur = con.cursor()
statement = 'insert into countries(country_id,
country_name, country_region) values (:2, :4, :7)'
cur.execute(statement, (1, 'Paraguay', 'South America'))
con.commit()
print ("inserting data")
cur.execute("""select * from countries""")
result = cur.fetchall()
print (result)

#here we update data into the table and query it
cur = con.cursor()
statement = 'update countries set country_name = :1 where
country_name = :2'
cur.execute(statement, ("Republic of Paraguay",
"Paraguay"))
con.commit()
print ("updating data")
cur.execute("""select * from countries""")
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

```
c:\python3.7.1\Oraclescripts>python insertandupdate.py
table droped
table created
inserting data
[(1, None, 'Paraguay', None, None, 'South America', None, None, None, None)]
updating data
[(1, None, 'Republic of Paraguay', None, None, 'South America', None, None, None, None)]

c:\python3.7.1\Oraclescripts>
```

You probably noticed that in all previous python scripts, the credential information for the ATP service is hard coded cx_Oracle.connect call. This is makes it non-secure and also hard to manage should something change with the credentials. With the python os library its possible to define an OS environment variable with the credential information and then simply read that variable inside the python script. This provides a secure mechanism to store and provide credentials and an easy way to reduce code changes due to database service changes.

eugenio.galiano@oracle.com

To set the environment variable with your login credentials run the following in your command prompt window (notice this is Windows command, other OS's will have similar processes):

```
set $connectvariable=$dbuser/$dbpassword@$connectstring
```

So for my sample database the command is below (replace the variables above with your database information, the $connectvariable is whatever you want to make it)

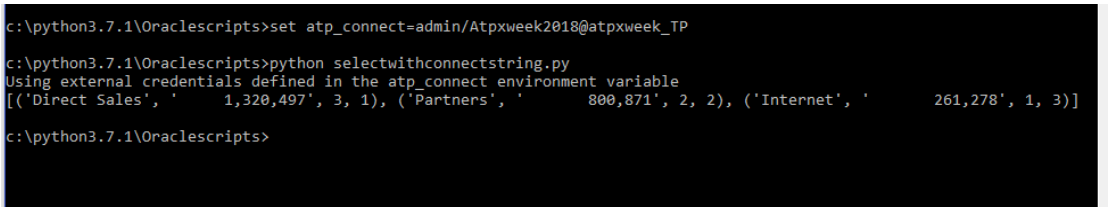
```
set atp_connect=admin/Atpxweek2018@atpxweek_TP
```

In the next script we will run the first select statement we ran, but instead of hardcoding the connection credentials we will call the environment variable to get the connection information. That requires importing the OS library in the script. Copy and paste the code into a file named selectwithconnectstring.py and run it by running the command:

```
python selectwithconnectstring.py
```

code below:

```
import cx_Oracle
import os
connectstring = os.getenv('atp_connect')
con = cx_Oracle.connect(connectstring)
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='US' \
GROUP BY channel_desc"
cur = con.cursor()
cur.execute(sql)
result = cur.fetchall()
print (result)
cur.close()
con.close()
```



In the last python script you will be inserting data into the database. You will be using the external credentials definition used in the exercise above, so make sure the exercise above worked. You will call a python script and passing it parameters. Passing parameters to a python script requires using the sys library so the script includes an import sys statement. This is a very simple script with no syntax checking so parameters must be passed exactly as shown or the script will return an error. The parameters passed will be inserted into the table, and the updated table contents will be displayed. Copy and paste the code into a file named variables.py and run it by running the command:

```
python variables.py $countryid $countryname $countryregion
```

so for example

```
python variables.py 30 Canada NorthAmerica
```

code below:

```
import cx_Oracle
import os
import sys

connectstring = os.getenv('atp_connect')
con = cx_Oracle.connect(connectstring)

print ("When you run this program you must pass command
line parameters or you will get an error")
print ("for example, python variables.py 10 France
Europe")
input("Hit any key to proceed")

id = sys.argv[1]
country = sys.argv[2]
region = sys.argv[3]

print ("inserting your data")
cur = con.cursor()
```



```
statement = 'insert into countries(country_id,
country_name, country_region) values (:2, :4, :7)'
cur.execute(statement, (id, country, region))
con.commit()
print ("selecting data")
cur.execute("""select country_id, country_name,
country_region  from countries""")
result = cur.fetchall()
print (result)
cur.close()
con.close()
```

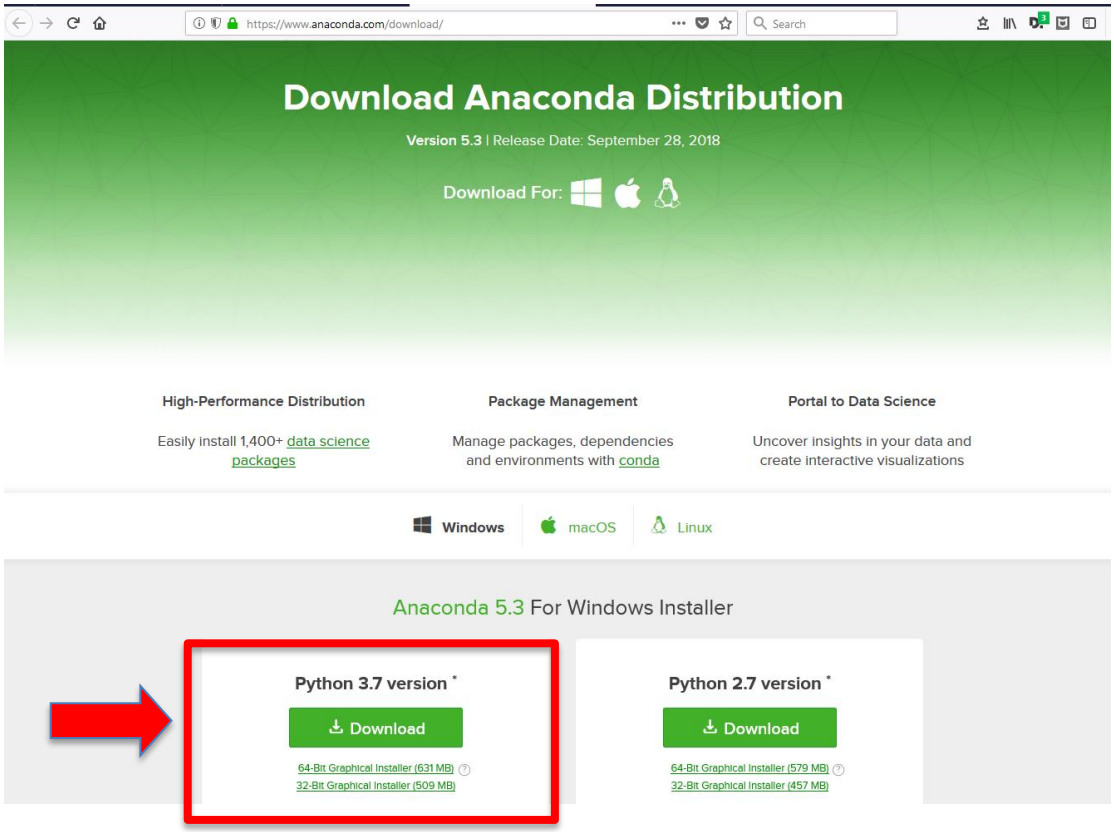
```
c:\python3.7.1\Oraclescripts>python variables.py 30 Canada NorthAmerica
When you run this program you must pass command line parameters or you will get an error
for example, python variables.py 10 France Europe
Hit any key to proceed
inserting your data
selecting data
[(10, 'france', 'europe'), (10, 'france', 'europe'), (1, 'Republic of Paraguay', 'South America'), (25, 'UnitedStates',
'NorthAmerica'), (30, 'Canada', 'NorthAmerica')]
```

9.3 – Optional - Installing Anaconda/Jupyter/Python

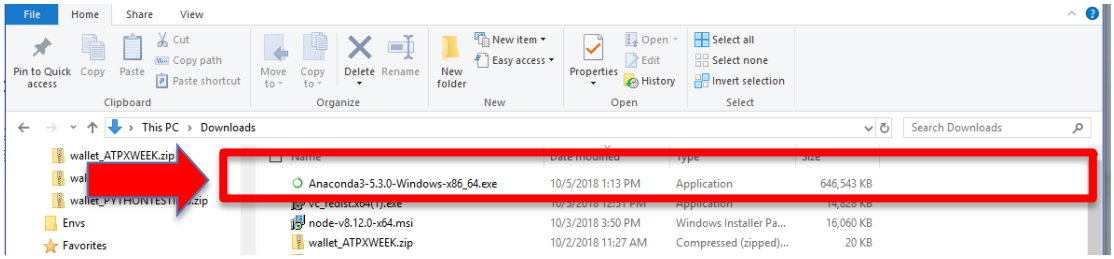
Integrated Development Environment's are very popular with the development community. Among the most popular for Python are the combination of Anaconda/Jupyter or Spyder. If you are interested in experimenting with these environments this lab goes through the process of installing Anaconda/Jupyter to as a Python development environment.
Please note that Jupyter/Python is very sensitive to previous versions, IDE's, and Python installations PATH's associated with previous installations on your computer. You need to be proficient with managing PATH's and libraries to run this optional section.

Download the software from www.anaconda.com/download

Select the Python 3.7 version download highlighted below, make sure you select the one for your correct architecture (32 or 64-bit):

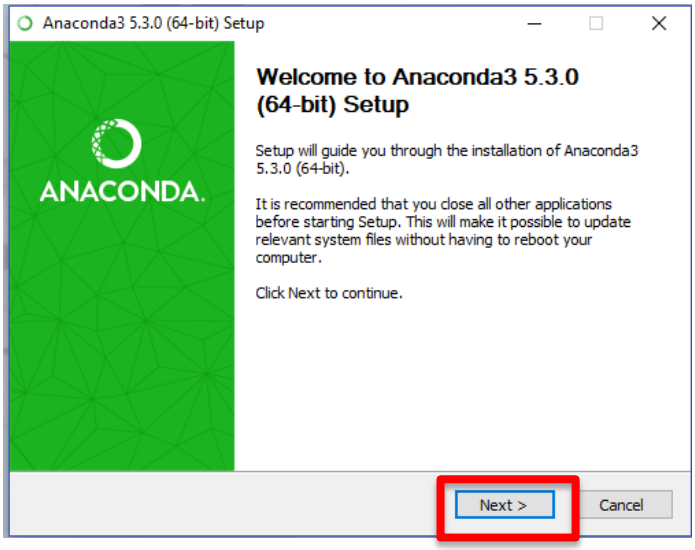


When the download completes open the folder where the file was downloaded and double click it:

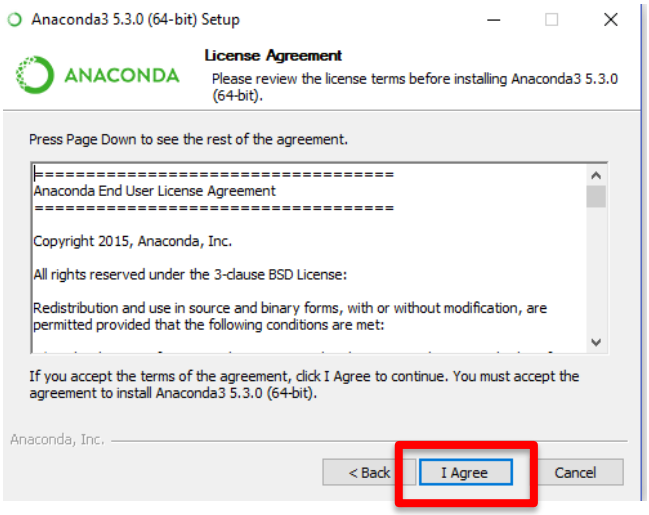


This brings up the Anaconda installation page, click **Next**:

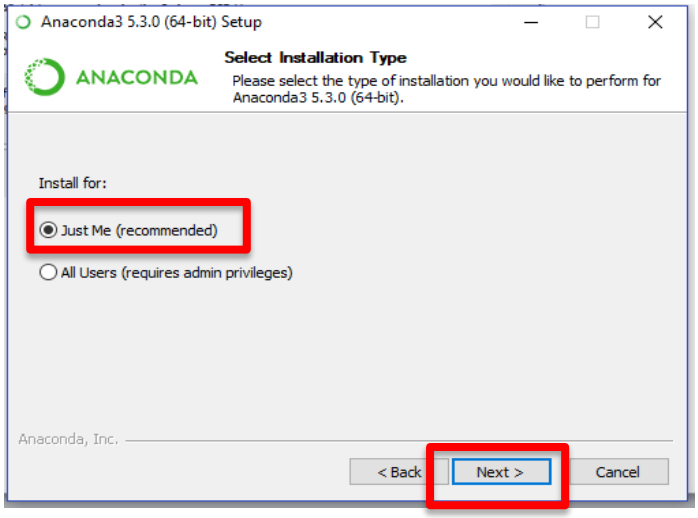
eugenio.galiano@oracle.com



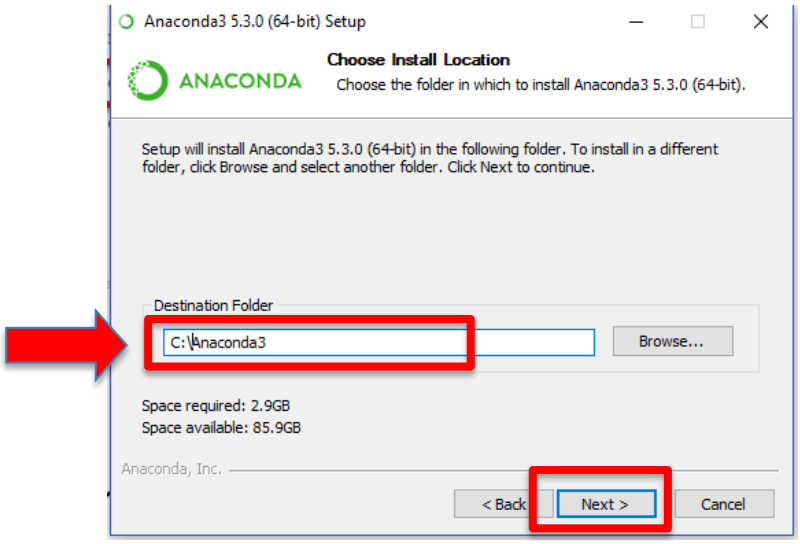
Click **I Agree**:



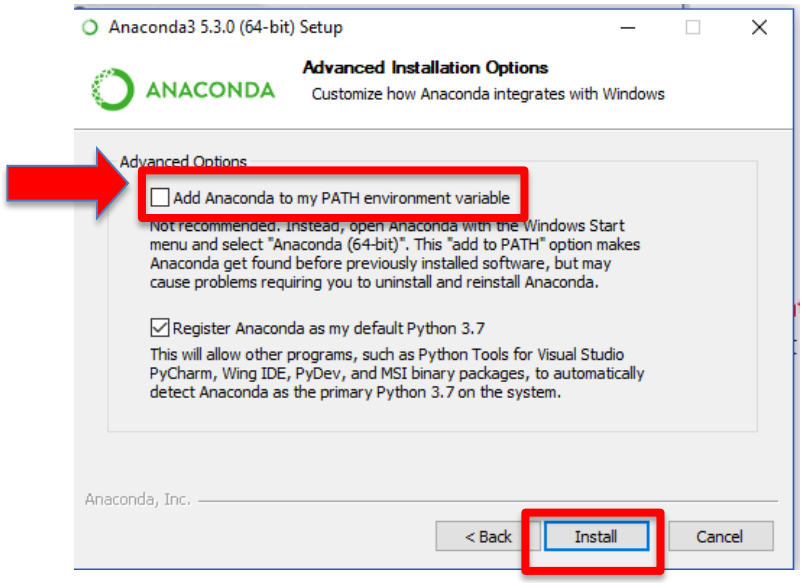
Select **Just Me** and then **Next**:



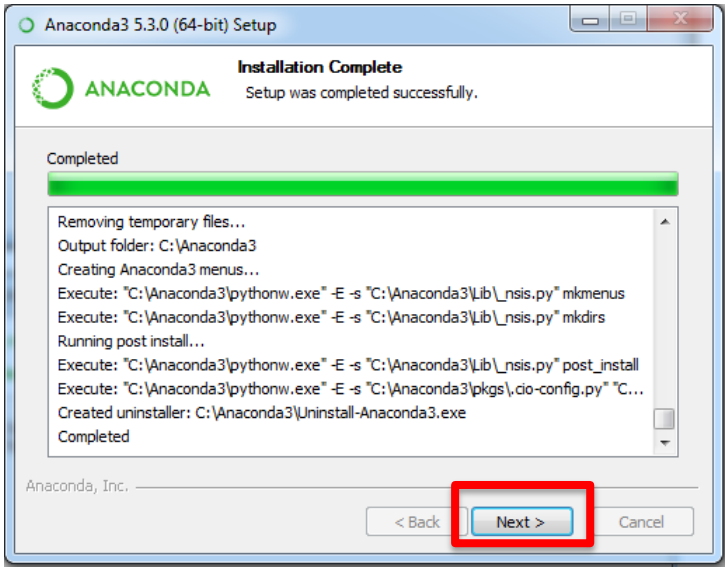
Install in the following directory: **c:\anaconda3 You must create the directory if** the directory does not exist create (the installer will not create it). Click **Next**:



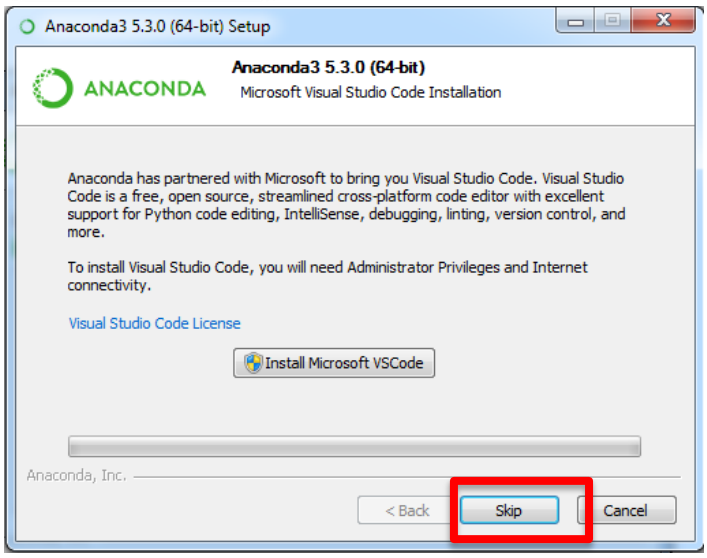
Make sure you select Register Anaconda as my default Python 3.7. Leave Add Anaconda to your PATH environment variable non-selected. Click **Install**



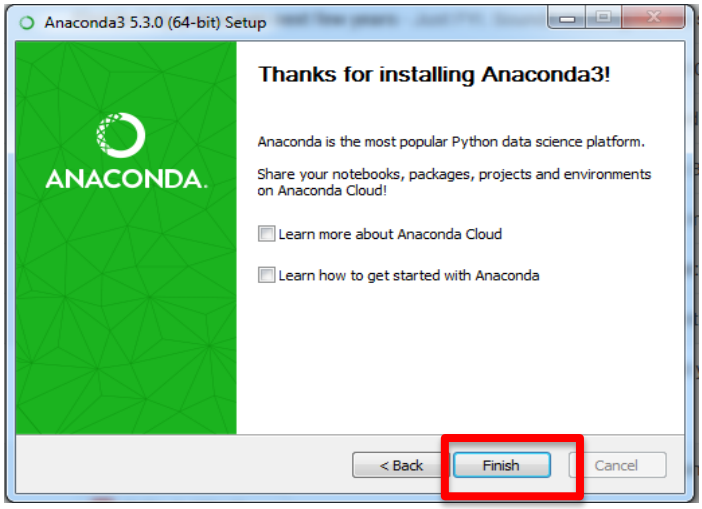
The installation will run for a few minutes, when Completed, click **Next:**



Select **Skip** on this screen:



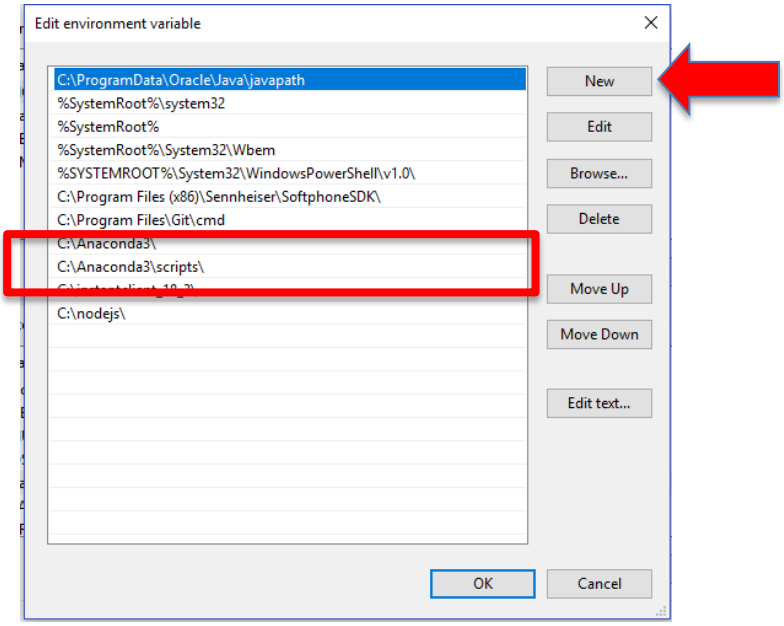
Deselect both options and click **Finish:**



You must add the new install directory into your PATH. Add c:\anaconda3\ and c:\anaconda3\scripts\ to your PATH:

In Windows 10:

6. In Search, search for and then select: **Advanced System Settings** (control panel)
7. Click **Environment Variables** at bottom of screen
8. In the **System variables** double click **Path**
9. In the screen that opens up select **NEW**
10. Add full path to the anaconda directory (**c:\anaconda3)**
11. Add full path to the anaconda scripts directory (**c:\anaconda3\scripts\)**



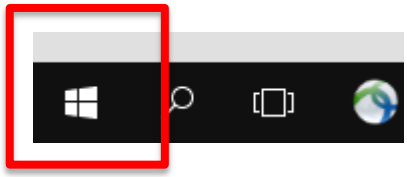
9.4 Using Anaconda/Jupyter/Python with ATP

You have installed Anaconda and Python!! Before running any Python apps that access the database the correct packages must be loaded into the Python environment. Open a Command Prompt Window and navigate to the directory where you installed Anaconda **(c:\anaconda3)** and run the following commands in order. **pip** is a package management system used to install and manage software packages written in **Python**. We will use pip to install the packages:

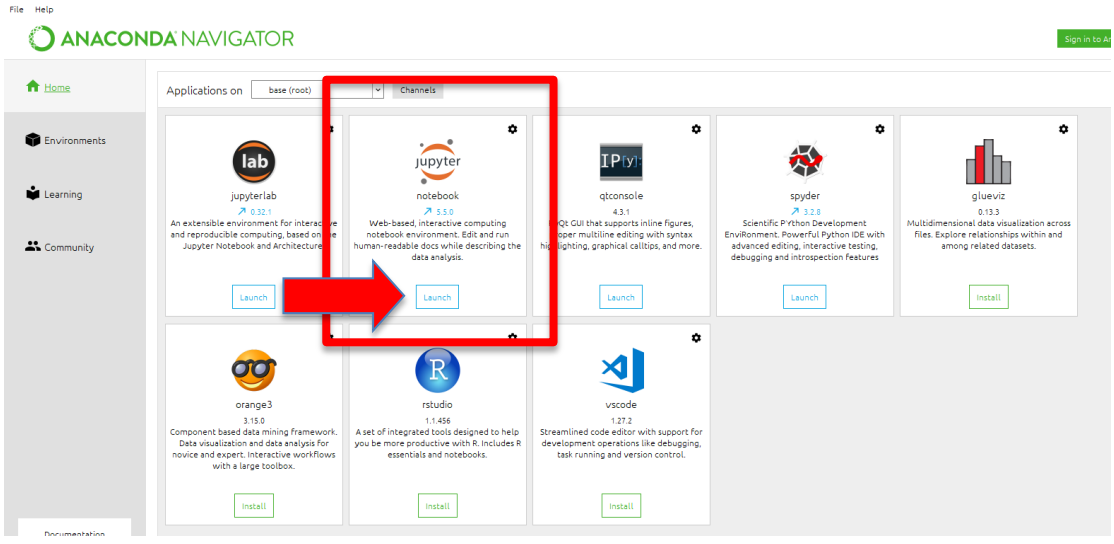
```
python -m pip install --upgrade pip
python -m pip install keyring
python -m pip install cx_oracle
python -m pip install sql
python -m pip install ipython-sql
python -m pip install python-sql
```

```
c:\Anaconda3>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/c2/d7/90f34cb0d83a6c5631cf71dfe64cc1054598c843a92b400e55675cc2ac37/pip-18.
1-py2.py3-none-any.whl (1.3MB)
    100% |                                | 1.3MB 3.3MB/s
Installing collected packages: pip
  Found existing installation: pip 18.0
    Uninstalling pip-18.0:
      Successfully uninstalled pip-18.0
Successfully installed pip-18.1

c:\Anaconda3>python -m pip install keyring
Requirement already satisfied: keyring in c:\anaconda3\lib\site-packages (15.0.0)
Requirement already satisfied: pywin32-ctypes!=0.1.0,!=0.1.1; sys_platform == "win32" in c:\anaconda3\lib\site-packages (from ke
yring) (0.2.0)
Requirement already satisfied: entrypoints in c:\anaconda3\lib\site-packages (from keyring) (0.2.3)

c:\Anaconda3>python -m pip install keyring
Requirement already satisfied: keyring in c:\anaconda3\lib\site-packages (15.0.0)
Requirement already satisfied: entrypoints in c:\anaconda3\lib\site-packages (from keyring) (0.2.3)
Requirement already satisfied: pywin32-ctypes!=0.1.0,!=0.1.1; sys_platform == "win32" in c:\anaconda3\lib\site-packages (from ke
yring) (0.2.0)

c:\Anaconda3>python -m pip install cx_oracle
Requirement already satisfied: cx_oracle in c:\anaconda3\lib\site-packages (6.4.1)

c:\Anaconda3>
```

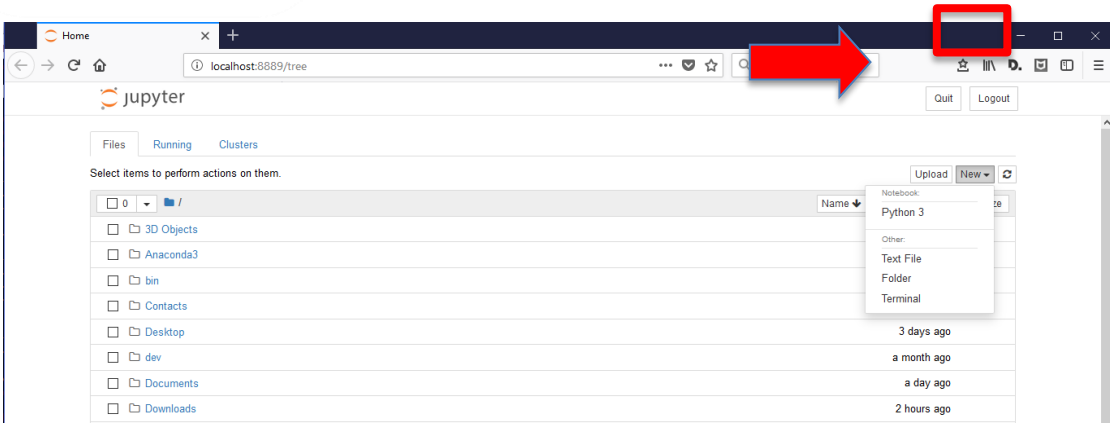
To Start Anaconda/Jupyter, go to the Windows Start Icon, Click and select **Anaconda Navigator** under **Anaconda3**:



From the ANACONDA NAVIGATOR, select **jupyter:**

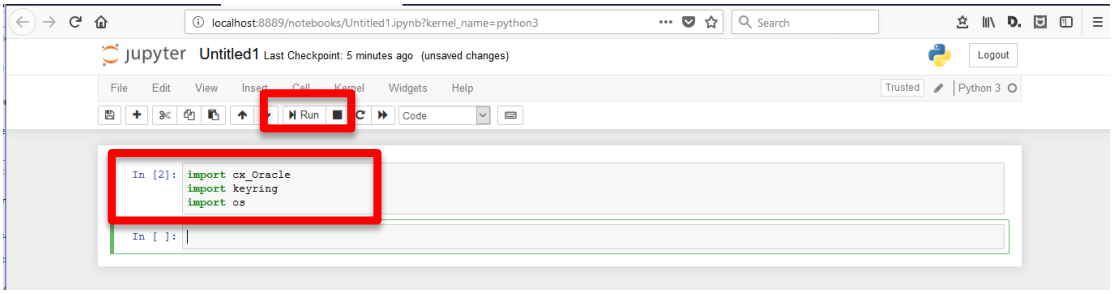


A new browser page will open up, running **jupyter,** select **New** and then **Python 3** highlighted below**:**



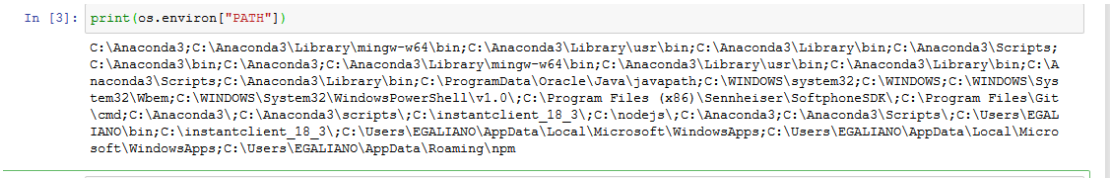
A new Python Notebook will open up. Python is an interpreted language so we must load libraries to use every time an environment is started up. Libraries are loaded with the import command, we will use 3 libraries. Run the following commands as show in the screen. This is similar to the Oracle OML notebook environment used in the initial lab. Copy the 3 lines below and paste them directly in the box next to the **In[]:** prompt, then select **Run**.

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



Run a simple command to display your PATH**.** Run the following command (copy and paste into the box and select **Run)**:

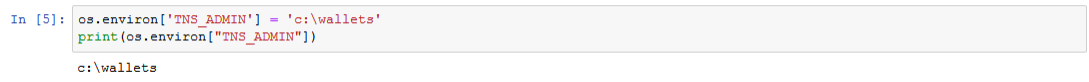
```
print(os.environ["PATH"])
```



The TNS_ADMIN variable must be set. TNS_ADMIN is the location of your unzipped wallet files. Below we set and then check the variable (the first command sets it, the second displays it back). Run the following command (copy and paste into the box and select **Run)**:

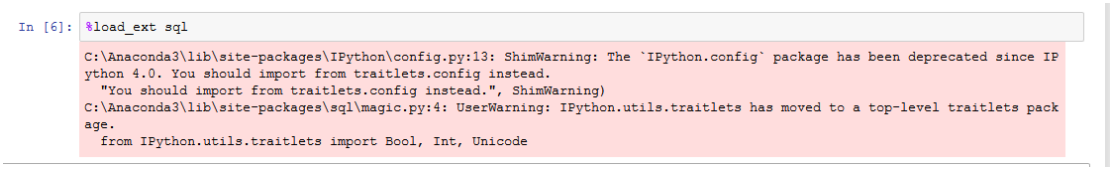
```
os.environ['TNS_ADMIN'] = 'c:\wallets'

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

Below is the result of running the commands which is setting and then displaying the TNS_ADMIN environment variable assignment:



Next we will make external sql calls to ATP, we need to load another library. Run the command below which will load the library needed to call external sql databases (ignore warning/error messages, make sure to include the %):

```
%load_ext sql
```

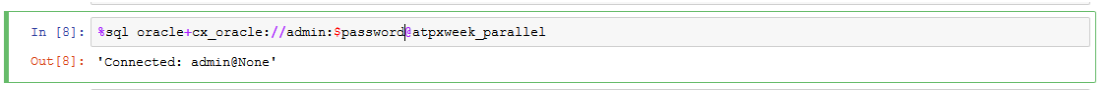


Next we will connect directly to the ATP database using a user name, password and service. Use your admin account and password created when the ATP database was created. The format of the command is (replace the black portion with your database and users information):

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

So your command would look like this:

```
%sql oracle+cx_oracle://admin:atppassword@atpxweek_TP
```



Once connected you will get the message 'Connected: admin@None'

To run a query, once connected use the **oracle+cx library** calls followed by the SQL statement (notice no ; at the end of the statement). The SQL below is the same one we ran in previous labs, copy the statement below and paste it in the box and click **Run**. (replace black portion with your database and user information):

```
%sql oracle+cx_oracle://admin:password@atpxweek_TP

SELECT channel_desc, TO_CHAR(SUM(amount_sold),'9,999,999,999') SALES$,
   RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,
   RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS
custom_rank
FROM sh.sales, sh.products, sh.customers, sh.times, sh.channels,
sh.countries
WHERE sales.prod_id=products.prod_id AND
sales.cust_id=customers.cust_id
  AND customers.country_id = countries.country_id AND
sales.time_id=times.time_id
  AND sales.channel_id=channels.channel_id
  AND times.calendar_month_desc IN ('2000-09', '2000-10')
  AND country_iso_code='US'
GROUP BY channel_desc
```

```
In [18]: %%sql oracle+cx_oracle://admin:$password@atpxweek_parallel
          SELECT channel_desc, TO_CHAR(SUM(amount_sold),'9,999,999,999') SALES$,
            RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,
            RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS custom_rank
         FROM sh.sales, sh.products, sh.customers, sh.times, sh.channels, sh.countries
         WHERE sales.prod_id=products.prod_id AND sales.cust_id=customers.cust_id
           AND customers.country_id = countries.country_id AND sales.time_id=times.time_id
           AND sales.channel_id=channels.channel_id
           AND times.calendar_month_desc IN ('2000-09', '2000-10')
           AND country_iso_code='US'
         GROUP BY channel_desc
         0 rows affected.
```

Out[18]:

channel_desc	sales$	default_rank	custom_rank
Direct Sales	1,320,497	3	1
Partners	800,871	2	2
Internet	261,278	1	3

To create a table use a similar process:

```
In [12]: %%sql oracle+cx_oracle://admin:$password@atpxweek_parallel
         CREATE TABLE COUNTRIES (
           COUNTRY_ID            NUMBER       NOT NULL,
           COUNTRY_ISO_CODE      CHAR(2)      NOT NULL,
           COUNTRY_NAME          VARCHAR2(40) NOT NULL,
           COUNTRY_SUBREGION     VARCHAR2(30) NOT NULL,
           COUNTRY_SUBREGION_ID  NUMBER       NOT NULL,
           COUNTRY_REGION        VARCHAR2(20) NOT NULL,
           COUNTRY_REGION_ID     NUMBER       NOT NULL,
           COUNTRY_TOTAL         NUMBER(9)    NOT NULL,
           COUNTRY_TOTAL_ID      NUMBER       NOT NULL,
           COUNTRY_NAME_HIST     VARCHAR2(40)
         )
         0 rows affected.

Out[12]: []
```

This concludes the optional python lab.

Lab 10. Using Docker Containers with ATP

Objectives

- To build a Docker container running a node.js application microservice
- Use it against an ATP Database service running in the Oracle cloud
- Package your Docker container image for deployment.

Microservices and Docker containers are very popular development and deployment environments Demonstrating that ATP works in a node.js application deployed as a Docker container will demonstrate to the development community that ATP is a viable option for data management in these applications.

In a previous lab you installed and ran node.js scripts to learn the basics of node.js and ATP. In this lab you will deploy a node.js application inside a Docker container that accesses information stored in an ATP database. You will create a Docker container from a build file available from Github. Once your Docker image is built you will customize it to access your ATP database and then package your customized image for deployment on as a Docker image. This lab builds on an Oracle Learning Lab located at here. For any updates please refer to this site. Because we are downloading a pre-created image, some values are hardcoded.

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

10.1 Installing Docker for Windows 10

The Docker environment will need to be installed in your local machine. Below are the steps for installation on Windows 10. For Mac users follow this link Mac Docker Install.

Go to Windows Docker install. Please note the requirements, which are very specific (Virtualization is not enabled in the OBI, but the installation will change that).

- Docker for Windows requires Microsoft Hyper-V to run. The Docker for Windows installer enables Hyper-V for you, if needed, and restart your machine. After Hyper-V is enabled, VirtualBox no longer works, but any VirtualBox VM images remain.
- **System Requirements**:
    - Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).

- Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled.
- At least 4GB of RAM.

Scroll down the screen and **download.docker.com** under Install Docker for Windows Desktop App:

This will save the installation file into your downloads directory. Go to the directory and double click on it:

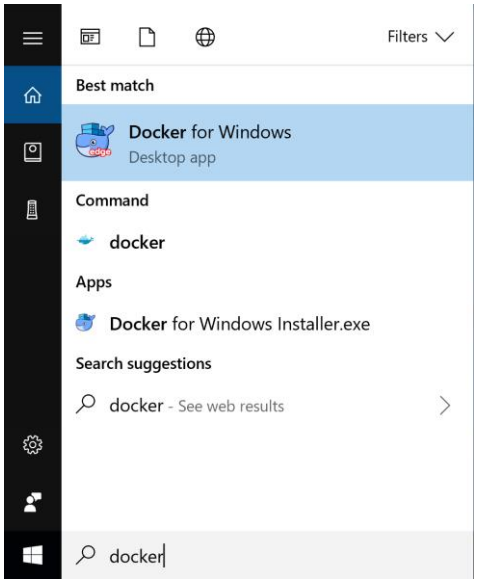
This will save the installation file into your downloads directory. Go to the directory and double click on it. There are two important questions you need to pay attention to. One will ask you if you want to enable Hyper V Virtualization (if it is not already enabled on your computer), you must say YES to this or Docker won't run. The second question is whether you want to run Linux or Windows containers. The

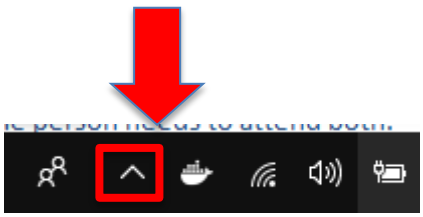
default is **LINUX.** Make sure Linux is selected and Windows is not. The service we will run in this lab is built on Linux.

Follow the install wizard to accept the license, authorize the installer, and proceed with the install. You may be asked to authorize Docker.app with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.

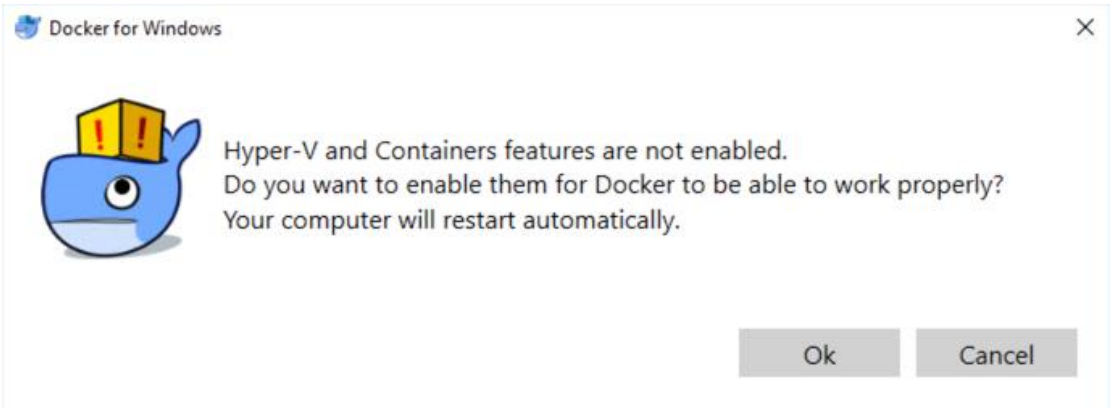
PLEASE NOTE – this process will probably involve one or more reboots.

When the installation is complete, look for Docker for Windows in your Start Menu or search for it in the search tool. Select and start it if it does not automatically start.





When it starts the first time, you will be prompted to install the Hyper-V and Containers features. Go ahead and then reboot.

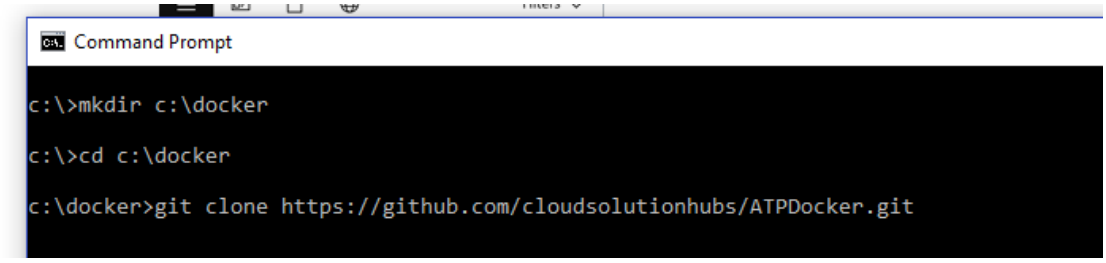


Once Docker is running you will see the familiar whale icon displayed in your tool bar. You may have to expand the "up" arrow to see it in the running applications window.

10.2 Downloading the Github Image for the Lab

Next step is to create a directory to download the github image that contains the files for our Docker container. We will use the "git" command which you already installed for the Swingbench lab. If you don't have git installed, refer to that section in the Swingbench lab. Create a directory called "c:\docker" and navigate to it in a command prompt. Then execute the following command:

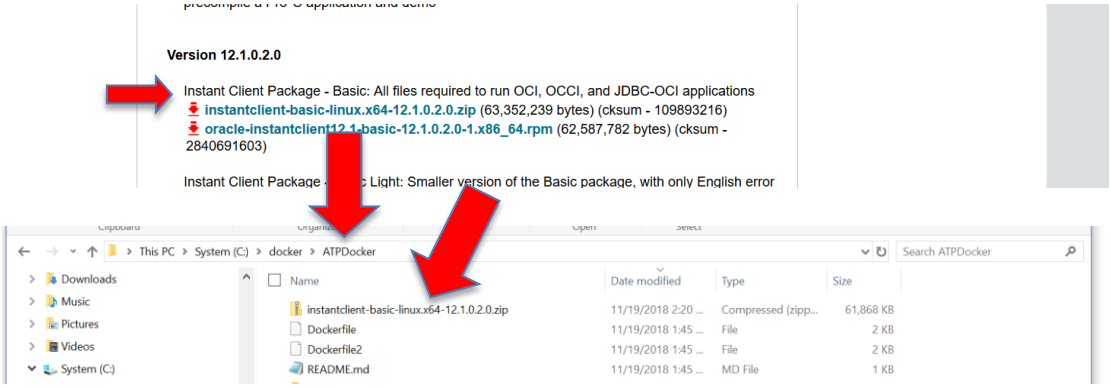
```
git clone https://github.com/cloudsolutionhubs/ATPDocker.git
```



This will create a directory called **ATPDocker** under the Docker directory. Change to that directory and list the contents.


```
c:\docker>cd ATPdocker

c:\docker\ATPDocker>dir
 Volume in drive C is System
 Volume Serial Number is 5C4B-8BF2

 Directory of c:\docker\ATPDocker

10/16/2018  05:14 PM    <DIR>          .
10/16/2018  05:14 PM    <DIR>          ..
10/16/2018  04:34 PM    <DIR>          aone
10/16/2018  04:34 PM    <DIR>          ATPnodeapp
10/16/2018  04:34 PM             1,735 Dockerfile
10/16/2018  04:34 PM             1,756 Dockerfile2
10/16/2018  04:34 PM    <DIR>          node_modules
10/16/2018  04:34 PM               803 README.md
10/16/2018  05:02 PM    <DIR>          wallet_NODEAPPDB2
               3 File(s)          4,294 bytes
               6 Dir(s)  85,746,483,200 bytes free

c:\docker\ATPDocker>
```

This microservice will connect to the ATP database you created using the Oracle instantclient. The service has a hard coded version of the instantclient libraries it uses (12.1.0.2.0), which may not be the latest version published by Oracle. Therefore you have to download this version and place it in the right directory for the image yu will build. Download the **instantclient-basic-linux.x64-12.1.0.2.0.zip** from Oracle OTN here Make sure you download exactly this version or the creation of the container image will fail. Download (or download and then copy to) the **ATPDocker** directory you just created.



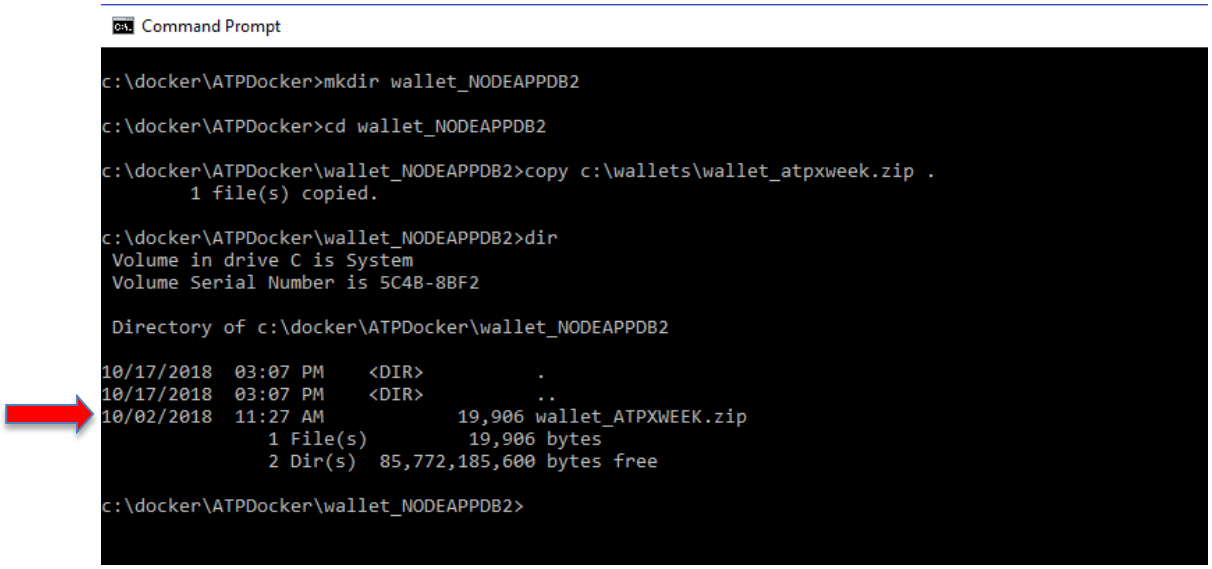
Create a new directory called wallet_NODEAPPDB2 in the same ATPDocker directory where you just downloaded the instant client. The directory MUST be called this because the scripts downloaded from github have this value hard coded in there. Go to this directory and copy your ATP Wallet (zip file downloaded in Lab 2) to this directory. In you command window, in the ATPDocker directory, run the following commands (below change the wallet name to your wallet name in the cp command, and make sure to keep the "." Dot at the end):

```
mkdir wallet_NODEAPPDB2
```



```
cd wallet_NODEAPPDB2

copy c:\wallets\wallet_atpxweek.zip . (notice the "dot")
```

Below you can see the results of running the commands and a listing of the files in the directory. Do not confuse your wallets name and the directory you created.

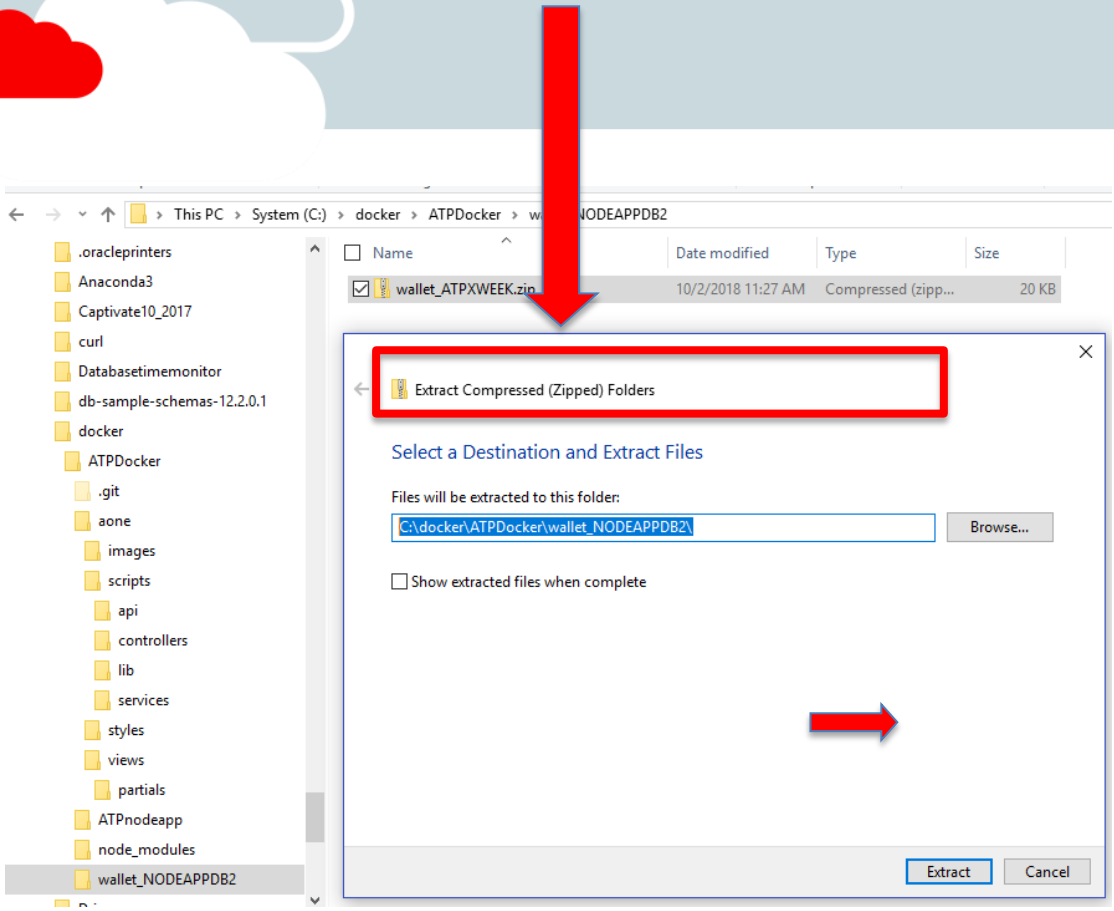


The steps above can also be done through the document viewer as well (not shown here).

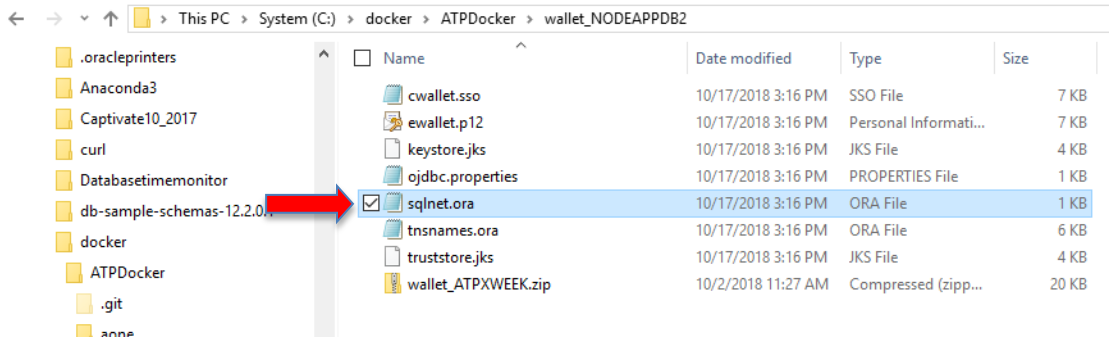
The wallet file needs to be unzipped to make changes to some of the files. Open up the document viewer and go to the c:\docker\ATPDocker\wallet_NODEAPPDB2 folder. Right click on the wallet file and select extract all. In the "Files will be extracted to this folder" make sure a new subdirectory is NOT created. Enter the following in box (and then select Extract):

```
C:\docker\ATPDocker\wallet_NODEAPPDB2\
```


eugenio.galiano@oracle.com



Once the file is extracted you will see all files from the zip wallet file in the directory. The sqlnet.ora file needs to be edited to indicate the location of the of the configuration files in the container image we will create. Right click on sqlnet.ora and select Edit (if no edit option appears, select Open, and open it with Notepad)



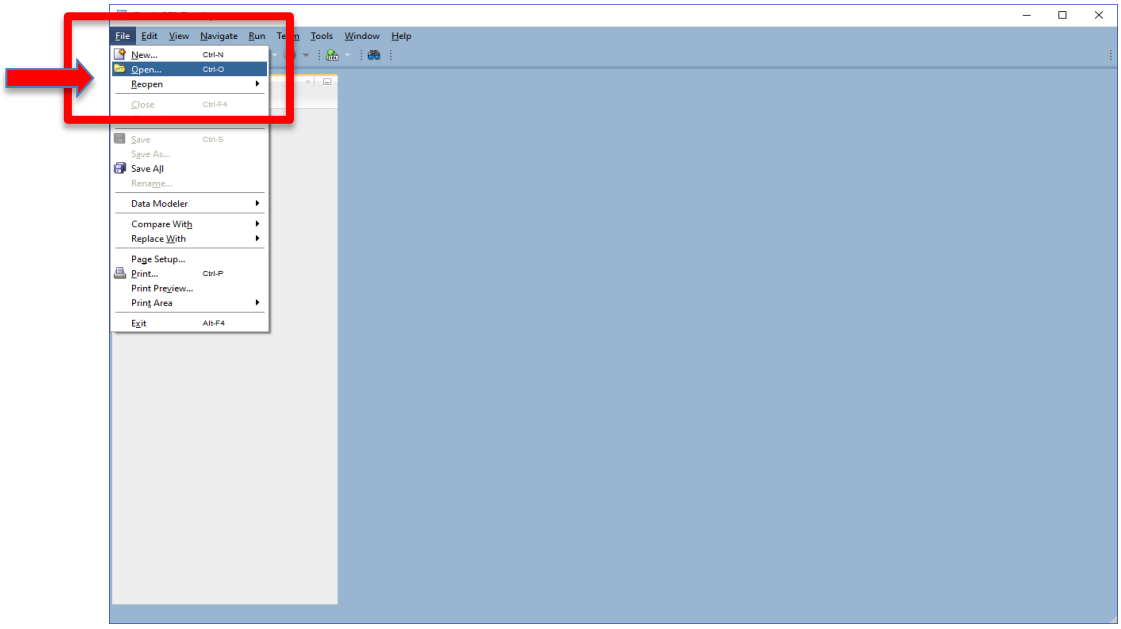
Change and save the file to have the following entry (the entry for DIRECTORY is changed to indicate that the TNS_ADMIN environment variable will point to the wallet directory. The TNS_ADMIN environment variable is hardcoded in the image downloaded from github):

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

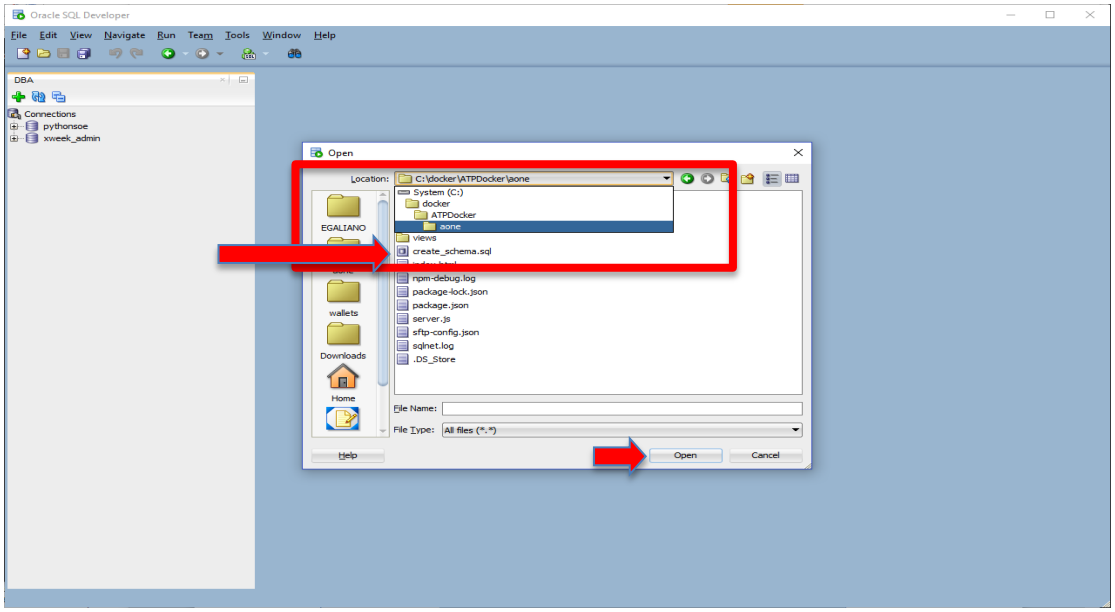
10.3 Populating the ATP Database for the docker/node.js application

The application that will be deployed in the container accesses specific tables in the ATP database that need to be created and populated. The complete SQL to perform this task is located in the `create_schema.sql` file located in the c:\docker\ATPDocker\aone directory.

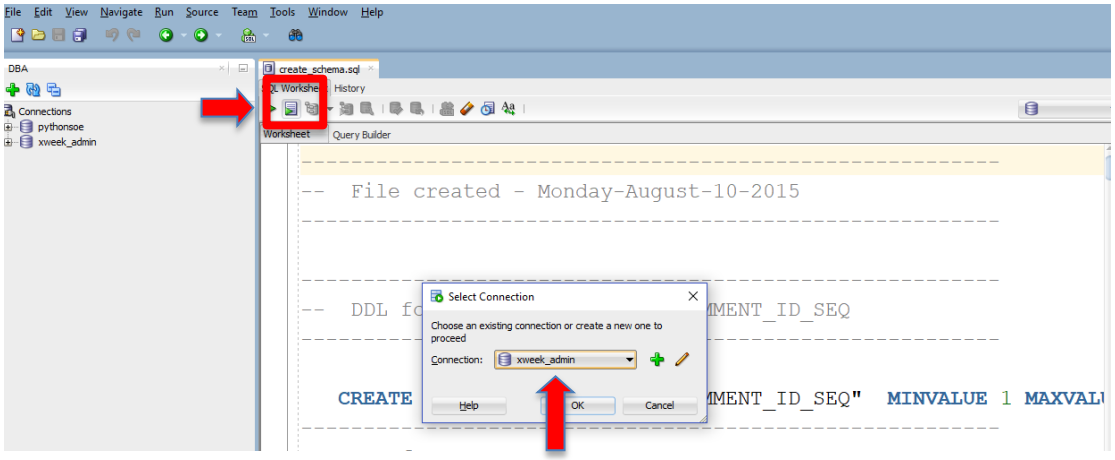
The easiest way to execute this script is to do it through SQL Developer. Open SQL Developer. **Under File in the top left corner, select Open** and connect to your admin connection. Refer to Lab 2 for instructions.



Navigate to c:\docker\ATPDocker\aone and select create_schema.sql and click open

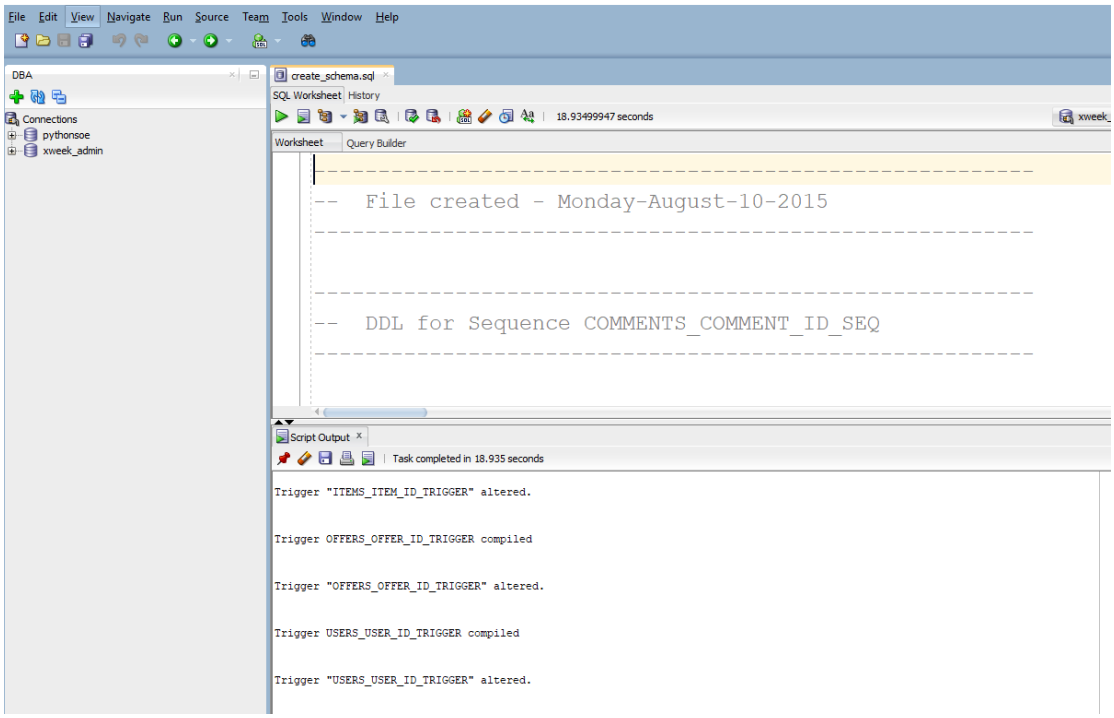


This will load the script into a SQL Worksheet, click Run Script, a window will pop up asking to Select Connection



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

eugenio.galiano@oracle.com



Your database now has the information the application uses.

Alternatively you can open the create_schema.sql file with Notepad, copy the whole content and paste it in the SQL Developer Query Builder box and run it. (Like in previous SQL Developer labs).

10.4 Creating your Docker image

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

However before creating our image explore what images for “Oracle” already exist in the repository. In your Command Prompt window navigate to c:\docker and run the docker command to search the repository:

```
cd c:\docker

docker search "oracle"
```

And you will see a list of images available


```
C:\Users\EGALIANO>docker search "oracle"
NAME                                DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
oraclelinux                         Official Docker builds of Oracle Linux.         509                 [OK]
sath89/oracle-12c                   Oracle Standard Edition 12c Release 1 with d…   402                                     [OK]
frolvlad/alpine-oraclejdk8          The smallest Docker image with OracleJDK 8 (…   339                                     [OK]
alexeiled/docker-oracle-xe-11g      This is a working (hopefully) Oracle XE 11.2…   270                                     [OK]
sath89/oracle-xe-11g                Oracle xe 11g with database files mount supp…   220                                     [OK]
wnameless/oracle-xe-11g             Dockerfile of Oracle Database Express Editio…   109                                     [OK]
jaspeen/oracle-11g                  Docker image for Oracle 11g database            76                                      [OK]
isuper/java-oracle                  This repository contains all java releases f…   55                                      [OK]
oracle/openjdk                      Docker images containing OpenJDK Oracle Linux   48                                      [OK]
airdock/oracle-jdk                  Docker Image for Oracle Java SDK (8 and 7) b…   39                                      [OK]
sath89/oracle-ee-11g                Dockerfile of Oracle Database Enterprise Edi…   33                                      [OK]
cogniteev/oracle-java               Oracle JDK 6, 7, 8, and 9 based on Ubuntu 16…   24                                      [OK]
ingensi/oracle-jdk                  Official Oracle JDK installed on centos.        21                                      [OK]
oracle/nosql                        Oracle NoSQL on a Docker Image with Oracle L…   17                                      [OK]
n3ziniuka5/ubuntu-oracle-jdk        Ubuntu with Oracle JDK. Check tags for versi…   16                                      [OK]
sgrio/java-oracle                   Docker images of Java 7/8/9/10 provided by O…   16                                      [OK]
andreptb/oracle-java                Debian Jessie based image with Oracle JDK in…   7                                       [OK]
flurdy/oracle-java7                 Base image containing Oracle's Java 7 JDK       5                                       [OK]
teradatalabs/centos6-java8-oracle   Docker image of CentOS 6 with Oracle JDK 8 i…   4
davidcaste/debian-oracle-java       Oracle Java 8 (and 7) over Debian Jessie        4                                       [OK]
martinseeler/oracle-server-jre      Oracle's Java 8 as 61 MB Docker container.      4                                       [OK]
publicisworldwide/oracle-core       This is the core image based on Oracle Linux…   1                                       [OK]
softwareplant/oracle                oracle db                                       0                                       [OK]
bitnami/oraclelinux-extras          Oracle Linux base images                        0                                       [OK]
pivotaldata/oracle7-test            Oracle Enterprise Linux (OEL) image for GPDB…   0
```

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

We will not pull an image but build our own from configuration files. You are ready to create your Docker image that will contain the node.js application that connects to the ATP Database. Go to the c:\docker\ATPDocker directory and run the docker build command. We will build an image called "aone" based on the steps and configurations defined in the default "Dockerfile" in this directory (alternate configuration files can be used with the –f parameter):

```
cd c:\docker\ATPDocker
```

```
docker build -t aone . --(notice the "dot" at the end)
```

```
c:\docker\ATPDocker\wallet_NODEAPPDB2>cd c:\docker\atpdocker

c:\docker\ATPDocker>docker build -t aone .
Sending build context to Docker daemon  12.88MB
Step 1/18 : FROM frolvlad/alpine-glibc:alpine-3.8
 ---> 356e3a9f167d
Step 2/18 : RUN apk update && apk add libaio libnsl &&     ln -s /usr/lib/libnsl.so.2 /usr/lib/libnsl.so.1
 ---> Using cache
 ---> 4deee8b9b305
Step 3/18 : RUN apk add --update     nodejs     nodejs-npm     git     python     && rm -rf /var/cache/apk/*
 ---> Using cache
 ---> 6174bf9fb109
Step 4/18 : ENV CLIENT_FILENAME instantclient-basic-linux.x64-12.1.0.1.0.zip
 ---> Using cache
 ---> 0f2ca876b337
Step 5/18 : WORKDIR /opt/oracle/lib
```

This will build the Docker image (ignore errors during the build process)! You can verify by running the following command which lists all your Docker images installed on your computer.

```
docker images –a
```

```
c:\docker\ATPDocker>docker images -a
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
<none>                  <none>              170aa7dd5fdc        2 minutes ago       407MB
<none>                  <none>              858631dc5b93        2 minutes ago       407MB
aone                    latest              ab2bf0c1cff3        2 minutes ago       407MB
<none>                  <none>              bb908118115b        3 minutes ago       400MB
<none>                  <none>              32028a9e2442        3 minutes ago       400MB
<none>                  <none>              5d5436cf410f        3 minutes ago       400MB
<none>                  <none>              e454027c01a5        3 minutes ago       400MB
```

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

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

The image will start up and you will be at a Unix Bash prompt, so you basically created a container service with Linux on your PC.:

```
c:\docker\ATPDocker>docker run -i -p 3050:3050 -t aone sh
/opt/oracle/lib #
```

Open up a new Command Prompt window (leave the Docker image aone up and running as well in a different windows). In the new command window run the following commands to view your image information and what Docker images are running. You will see your aone image running:

```
docker images
```

```
docker ps -all
```

```
Microsoft Windows [Version 10.0.16299.665]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\EGALIANO>docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
aone                    latest              ab2bf0c1cff3        24 hours ago        407MB
<none>                  <none>              1850481ffe68        29 hours ago        407MB
frolvlad/alpine-glibc   alpine-3.8          356e3a9f167d        4 weeks ago         11.3MB

C:\Users\EGALIANO>docker ps -all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
    NAMES
0271129202be        aone                "sh"                12 minutes ago      Up 12 minutes       0.0.0.0:3050->3050/t
cp   admiring_kirch

C:\Users\EGALIANO>
```


10.5 Running the node.js application in your Docker image

If you ran the node.js lab the next set of steps should be familiar. At this point we have a running Docker image with all the components we manually installed and configured for the node.js lab above (except this Docker image is running Linux instead of Windows). All you have to do is configure the ATP Database connection file (dbconfig.js) to contain your ATP connection information. In the window you started the container that now is running the Bash shell, navigate to the directory that has the config file. You will need to know vi or another form of editing in Linux to update the file.

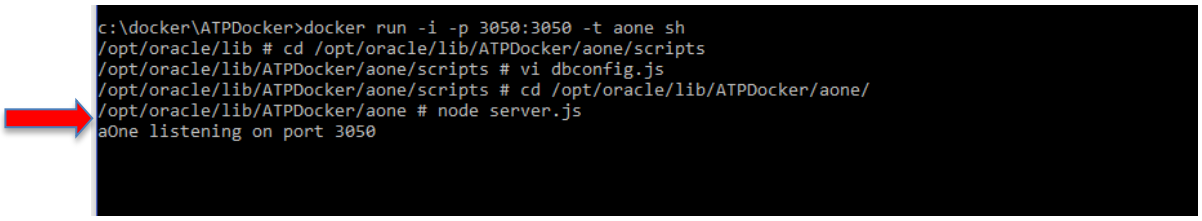
```
cd /opt/oracle/lib/ATPDocker/aone/scripts

vi dbconfig.js
```

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

```
module.exports= {
user:"admin",
password:"Atpxweekpwd",
connectString:"atpxweek_tp"
}
```

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

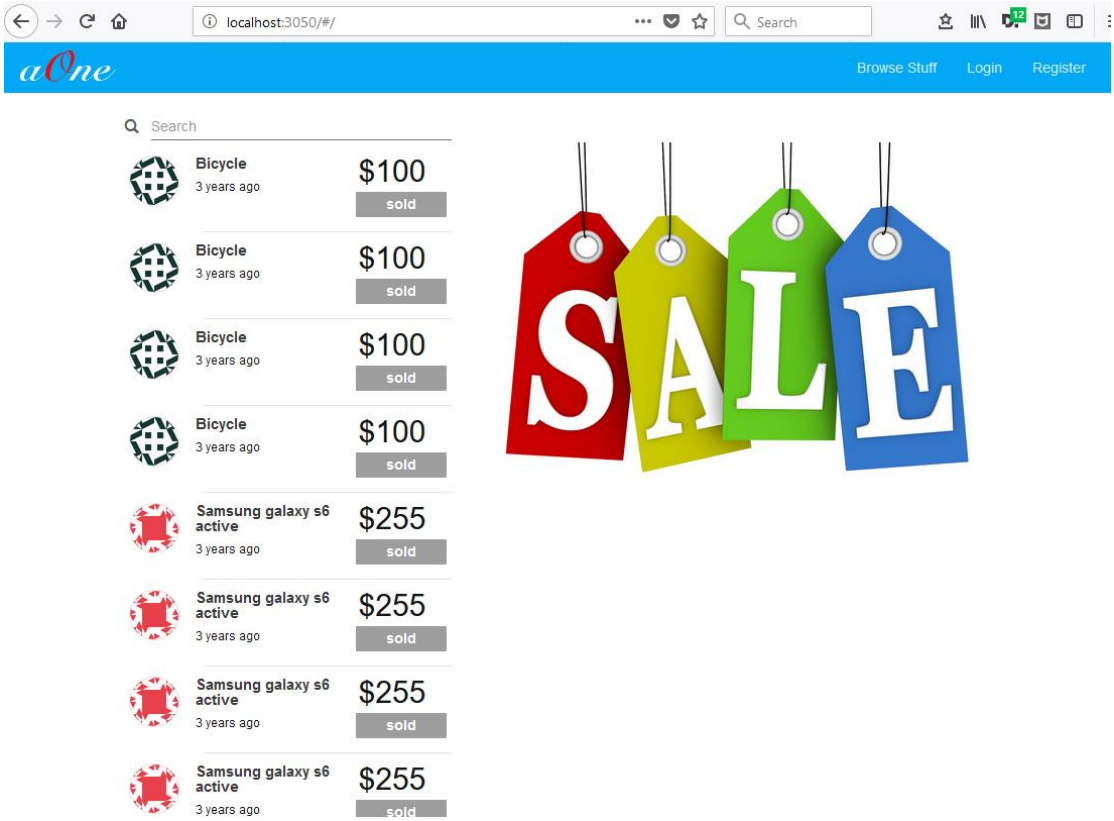
```
cd /opt/oracle/lib/ATPDocker/aone/

node server.js
```



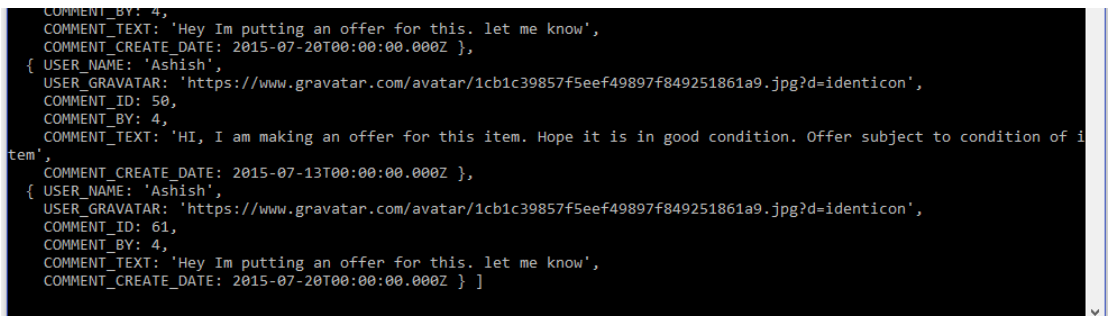
When the application is running you will notice an initial message indicating output on port 3050. Open your browser to:

http://localhost:3050/

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



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

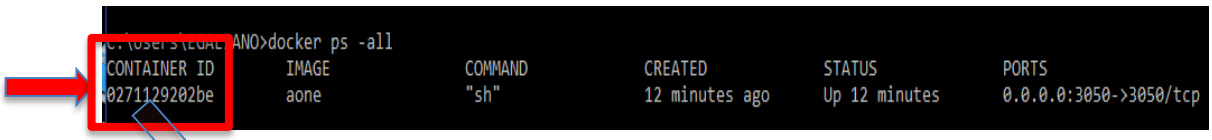


Now that you have customized the Docker image, you can create your own container image based on the customizations you made. You can then take that image and publish to run it on a Docker service without having to make any further customizations (change database parameters).

10.6 Creating and running your own customized Docker Container Image

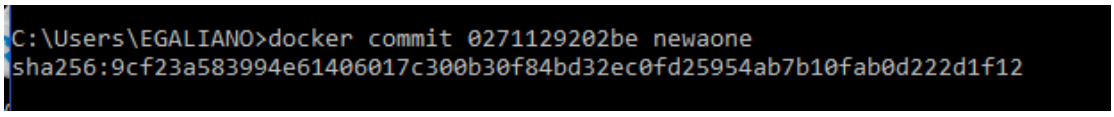
In a Command Prompt window issue the same command we issued above (you need the CONTAINER ID):

```
docker ps -all
```

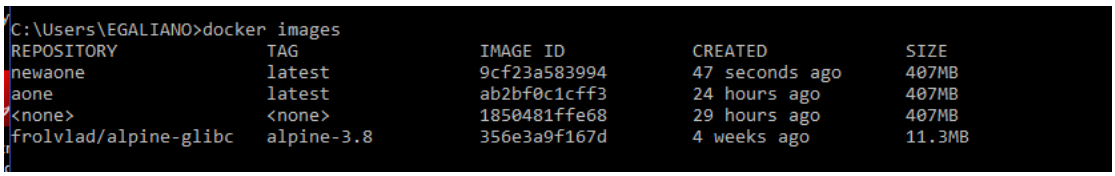


To create an image of your running Docker container image use the `docker commit` command. In this case we are going to commit changes from our running container and make a new container with those changes. The Docker commit statement needs the CONTAINER ID highlighted above, that is the ID associated with the "aone" container we are currently running. Call the new container newaone, running the following command (replace black id below with your id from the docker –ps all command):

```
docker commit 0271129202be newaone
```



To verify the image created run the `docker images` command:



Your "newaone" will show up along with the original aone image. You just created a Docker image! Now run it. Because you did not change any port information on this image, we need to stop the original image before we can run this one. Go back to the original windows where you are running aone and hit Ctrl-C to stop it and it you want you can completely exit the Docker container (exit). Also close the browser page you were using to display the application http://localhost:3050/.

See below for a screenshot where this is done.

```
c:\docker\ATPDocker>docker run -i -p 3050:3050 -t aone sh
/opt/oracle/lib # cd /opt/oracle/lib/ATPDocker/aone/scripts
/opt/oracle/lib/ATPDocker/aone/scripts # vi dbconfig.js
/opt/oracle/lib/ATPDocker/aone/scripts # cd /opt/oracle/lib/ATPDocker/aone/
/opt/oracle/lib/ATPDocker/aone # node server.js
aOne listening on port 3050



^C
/opt/oracle/lib/ATPDocker/aone #
/opt/oracle/lib/ATPDocker/aone # docker
sh: docker: not found
/opt/oracle/lib/ATPDocker/aone # exit

c:\docker\ATPDocker>
```

In the Command Prompt where you ran the Docker commit statement start up your new image, use the same steps as above except now you will run the newaone image:

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

Once in the image go to the scripts directory. Notice that in the new image the directories are still the same but you no longer have to edit the dbconfig.js file as done above because that change was committed to our new image. Once in the scripts directory run the node script:

Go to scripts directory
```
cd /opt/oracle/lib/ATPDocker/aone/scripts
```

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

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

Run node application
```
node server.js
```

```
C:\Users\EGALIANO>docker run -i -p 3050:3050 -t newaone sh
/opt/oracle/lib/ATPDocker/aone # cd /opt/oracle/lib/ATPDocker/aone/scripts
/opt/oracle/lib/ATPDocker/aone/scripts # ls
api          app.js       controllers  dbconfig.js  lib          services
/opt/oracle/lib/ATPDocker/aone/scripts # cat dbconfig.js
module.exports= {
user:"admin",
password:"Atpxweek2018",
connectString :"atpxweek_high"
}
/opt/oracle/lib/ATPDocker/aone/scripts # cd /opt/oracle/lib/ATPDocker/aone/
/opt/oracle/lib/ATPDocker/aone # node server.js
aOne listening on port 3050
```

Go to: http://localhost:3050/

To see the application from your new container running!!

When finished with the Lab, hit Ctrl-C on your Command Prompt window to end the application. At this point you are still in the Docker image running Unix, to exit back to Windows type exit at the command prompt.

Once done with this lab, you can clean up you docker environment which may use a lot of disk space. The following commands can be run from the Command Prompt window. This will maintain your docker installation, but remove any images created. You can always recreate the image above by starting at the docker build command

- docker image prune
- docker container prune
- docker volume prune
- docker network prune

Also, make sure you stop your Docker service when not in use, as it will consume resources on your computer when running. To stop the service, right click on the Whale icon and select Quit Docker:

To summarize this lab, we created a Docker container image and implemented a node.js application within the container that accesses an ATP database. We started from a generic image we downloaded from a library and customized it to our own ATP database credentials. Once the customizations were made we created a new copy of the Docker image with these customizations. This image can now be deployed on any Docker container service and it will run with access to the ATP cloud database from any internet connected location.

Congratulations
You have completed your ATP Hands-on Lab

eugenio.galiano@oracle.com