Moving Data from Oracle Database to Oracle Autonomous Data Warehouse using Oracle GoldenGate 19c Microservices

An On-Premises to Cloud Exercise

By: Bobby L. Curtis, MBA

RheoData

Table of Contents

Introduction

Oracle has produced a cloud services called Oracle Autonomous Database. The Oracle Autonomous Database is a suite of self-driving, self-securing, and self-repairing services that leverage machine learning and automation. The purpose behind the Oracle Autonomous Database is to eliminate human labor, human error, and manual tuning, leading to a reduced cost and complexity, ensuring higher reliability, security, and operational efficiency.

The Oracle Autonomous Database comes in two same but distinct products, the Autonomous Data Warehouse and Autonomous Transaction Processing. Both of which are optimized to meet the specialized needs of each. The Autonomous Data Warehouse delivers reliability, performance, and highly elastic data management. Autonomous Transaction Processing allows for more straightforward application development and deployment of real-time analytics, fraud detection, and personalization. Both offerings within the Oracle Autonomous Database service are excellent and provide flexibility as well as lowering administration time, which allows organizations to be more agile and innovative.

With the ability to spin up an Oracle Autonomous Database come the need for populating it with data. The Oracle Autonomous Database services provide a few methods for moving data into the services; however, there is only one method that allows for real-time processing of data into an Oracle Autonomous Database. This whitepaper will discuss and show the required steps to move data in real-time using Oracle GoldenGate Microservices.

Software

Oracle Autonomous Database Services

The capabilities of the Oracle Database enables the Oracle Autonomous Database to be provision in two editions that are tailored explicitly to workloads for data warehouses (Autonomous Data Warehouse) or online transactional processing (Autonomous Transaction Processing).

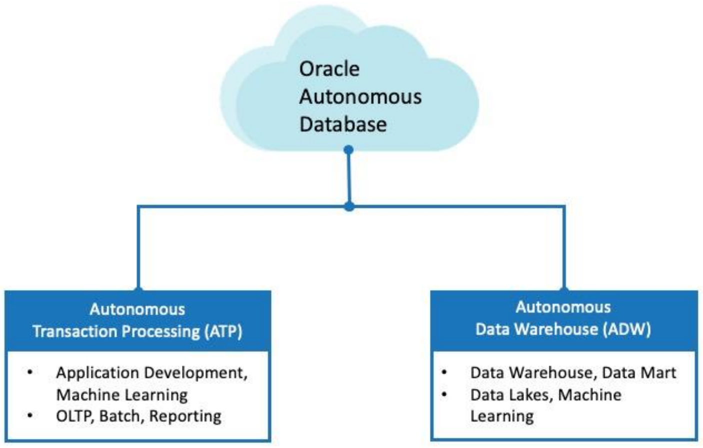

Oracle Autonomous Database
Autonomous Transaction Processing (ATP)
• Application Development, Machine Learning
• OLTP, Batch, Reporting
Autonomous Data Warehouse (ADW)
• Data Warehouse, Data Mart
• Data Lakes, Machine Learning

Figure 1: High-Level Oracle Autonomous Database

Autonomous Data Warehouse

The Autonomous Data Warehouse (ADW) provides a platform for running Data Warehouses, Data Marts, Machine Learning, or Data Lake deployments at scale. Data Warehouses are the foundational structures for data modeling, such as Star Schemas, and other modeling needs to meet business objectives. These types of environments typically rely on the summary representation of the data and leverage high parallel SQL execution to provide fast responses. The Autonomous Data Warehouse is tailored explicitly for these use-cases.

Autonomous Transaction Processing

Similar to the Autonomous Data Warehouse, the Autonomous Transaction Processing bring the autonomous capabilities to the mixed-workload environment. This version of the Autonomous Database provides a platform for complex transaction workloads that include reporting and batch data processing. With the ability to run mixed workloads within a single autonomous database eliminates the need to move data between different types of databases, reducing complexity within processing environments. By automating the creation and management of databases, the Autonomous Transaction Processing also enables more straightforward application development.

Oracle GoldenGate 19c Microservices

Oracle GoldenGate 19c Microservices is an enterprise real-time data integration and replication software. It enables real-time data integration and replication, high-availability solutions, transactional change data capture (CDC) between operational and analytical enterprise systems.

Using Oracle GoldenGate 19c Microservices, you can move transactional data across multiple systems in the enterprise in real-time, providing data where it is needed when it is required. The different capabilities of Oracle GoldenGate 19c Microservices allow you to capture and distribute data between many various data sources, filter transactions within the enterprise, and migrate databases in near zero-downtime.

The Microservices architecture within Oracle GoldenGate 19c builds on top of the existing Oracle GoldenGate framework and makes implementation and usage efficient. By enabling microservices on the various Oracle GoldenGate components and providing REST API endpoints, administration becomes more natural. The enhanced integration within Oracle GoldenGate 19c Microservices opens up administration and execution without having to have server access.

Nginx

Nginx is a free open-source or purchased HTTP/Reverse Proxy utility that is recommended by Oracle to use in front of the Oracle GoldenGate 19c Microservices architecture. The purpose of the reverse proxy is to consolidate the port numbers to a single, well-established port for Oracle GoldenGate 19c Microservices deployments to be accessed over.

Oracle GoldenGate Deployment Model

Oracle GoldenGate 19c Microservices can support many different use-cases or architecture designs. The primary goal of Oracle GoldenGate 19c Microservices is to enable users to build reliable data integration environments on a large scale, quickly establish active/active environments, or scale-out mesh architectures.

One of the popular deployment models is the hub-n-spoke model. This model centralizes the management of Oracle GoldenGate 19c and enables the usage of remote capture and remote apply. There is no need to ship trail files from source to target because everything resides on the central hub of the architecture.

Build

Implementation of Oracle GoldenGate 19c Microservices to replicate to an Oracle Autonomous Database requires three primary things. These items are as follows:

- On-Premises Database – This database or databases is the source of all data
- Oracle GoldenGate 19c Microservices Hub – This is a single compute node with Oracle GoldenGate 19c Microservices and any required Oracle Database Client software needed
- Oracle Autonomous Database (ADB) – an Oracle Autonomous Database deployed in the Oracle Cloud Infrastructure

On-Premises Database

Organizations typically have databases or other data sources that they want to move into a cloud platform or centralize into a data warehouse for reporting purposes. Often this data is gathered, stages, and then consumed via a batch process into other data platforms or data marts. Using Oracle GoldenGate 19c Microservices, the traditional batch process can be streamlined and turned into real-time data feeds.

Before Oracle GoldenGate 19c Microservices can capture data and transport it to another platform, the source database configuration has to change. Configuration changes like Archive Log enablement, database-wide supplemental logging, and force logging has to be enabled. The following are recommended database settings for an Oracle database:

Database Settings:

- Enable Archive Logging

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog
SQL> alter database open
```

- Enable Database full supplemental logging

```
SQL> alter database add supplemental log data;
```

- Enable Force Logging

```
SQL> alter database force logging;
```

- Enable database parameter enable_goldengate_replication

```
SQL> alter system set enable_goldengate_replication=TRUE;
```

Users and Permissions:

Note: The users and permissions for database access are to be as minimal as possible. I have granted the DBA role to these users just for the whitepaper.

- Common GoldenGate User (C##)

```
create user C##GGATE
identified by <password>
default tablespace USERS
temporary tablespace TEMP
quota unlimited on USERS
account unlock;

grant connect to c##ggate;
grant dba to c##ggate;
grant resource to c##ggate;

begin
  SYS.DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE ('C##GGATE',
container=>'ALL');
end;
/
```

- Local GoldenGate User

```
alter session set container = <PDB>;

grant connect to c##ggate;
grant dba to c##ggate;

create user GGATE
identified by <password>
default tablespace USERS
temporary tablespace TEMP
quota unlimited on USERS
account unlock;

grant connect to ggate;
grant dba to ggate;
```

Oracle Autonomous Data Warehouse

The Oracle Autonomous Database, in this case, Autonomous Data Warehouse, has to be established on the Oracle Cloud Infrastructure (OCI) through the OCI Console. After building an Autonomous Data Warehouse, the associated wallet has to be downloaded to ensure a secure connection.

Establishing connections to the Autonomous Data Warehouse are made by using Oracle Net Services. These connection services support a variety of connection types, including OCI, ODBC drivers, JDBC OC, and JDBC Thin Client. For Oracle GoldenGate 19c Microservices to establish a connection to the Autonomous Data Warehouse, the Client Credentials Wallet will securely use these connections.

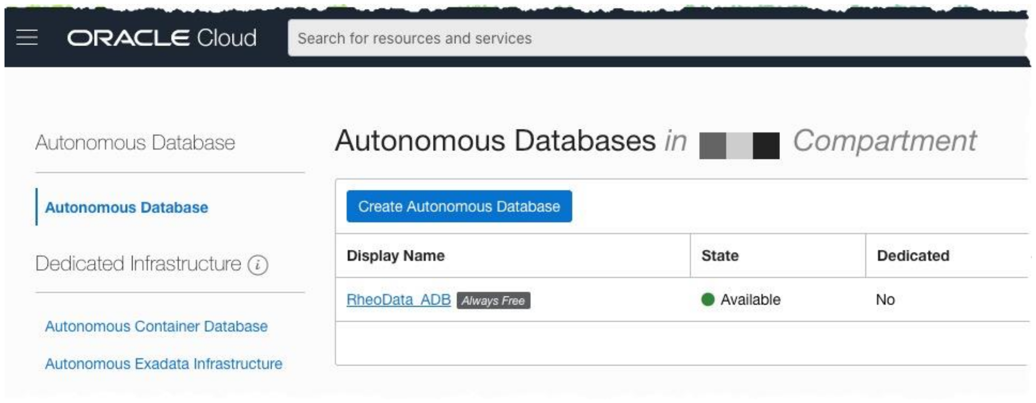
Download Client Credentials (Wallet)

Before Oracle GoldenGate 19c Microservices can make a connection to an Autonomous Data Warehouse, the Client Credentials Wallet must be downloaded and secured on the host. The Client Credentials Wallet (client_credentials.zip) that you download will contain the following files:

- cwallet.sso – Oracle Auto-Login wallet
- rwallet.p12 – PKCS #12 wallet file associated with the Auto-Login wallet
- sqlnet.ora – SQL*Net profile configuration file that includes wallet location and TNSames naming method
- tnsnames.ora – SQL*Net configuration file that contains network service names mapped to connect descriptors for the local naming method
- Java Key Store (JKS) file – Key store files used with JDBC Thin Connections

To download the Client Credentials zip file:

1. Login to the OCI Console and navigate to the Autonomous Database page.

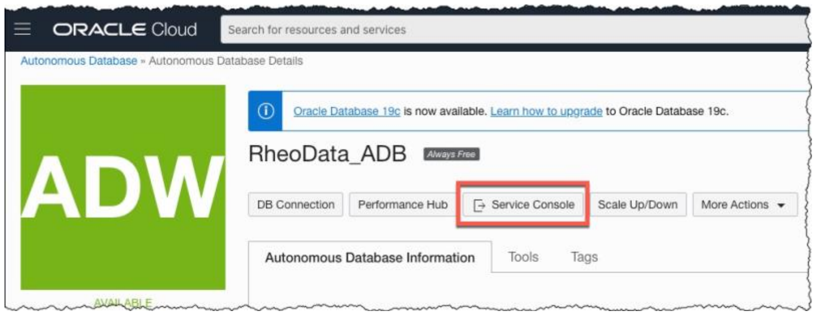

ORACLE Cloud
Search for resources and services
Autonomous Database
Autonomous Database
Dedicated Infrastructure
Autonomous Container Database
Autonomous Exadata Infrastructure
Create Autonomous Database

Autonomous Databases *in* [redacted] *Compartment*

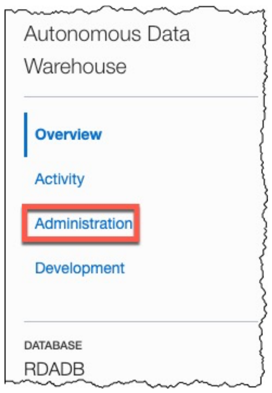
Display Name	State	Dedicated
RheoData_ADB Always Free	● Available	No

2. Click on the Autonomous Database that you want to interact with.

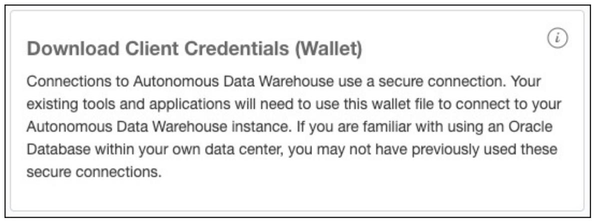
3. Once on the Autonomous Database Details page, select Service Console


ORACLE Cloud
Search for resources and services
Autonomous Database » Autonomous Database Details
ADW
AVAILABLE
Oracle Database 19c is now available. Learn how to upgrade to Oracle Database 19c.
RheoData_ADB
Always Free
DB Connection
Performance Hub
Service Console
Scale Up/Down
More Actions
Autonomous Database Information
Tools
Tags

4. On the Service Console page, select Administration



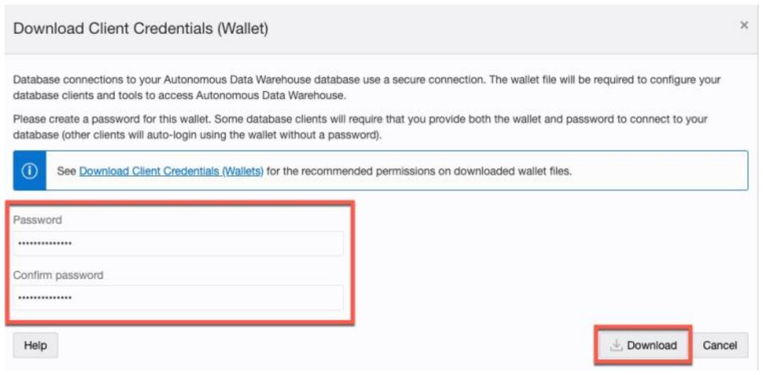
5. Under the Administration option, select Download Client Credentials (Wallet)



Download Client Credentials (Wallet)

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

6. When prompted for a password, provide a password. Password has to conform to the OCI password standards before you can download the zip file.



Download Client Credentials (Wallet)

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

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

See Download Client Credentials (Wallets) for the recommended permissions on downloaded wallet files.

Password: ••••••••••••••

Confirm password: ••••••••••••••

7. After downloading the Client Credentials zip file, it needs to move to the Oracle GoldenGate 19c Microservices hub.

Oracle GoldenGate 19c Microservices Hub

Using Oracle GoldenGate 19c Microservices in a hub configuration is an ideal way of centralizing the management of the Oracle GoldenGate while maximizing the benefits of the product. For this architecture to work, there are a few things that need to be understood.

Depending on the databases that will be used as the source and target significantly affect the look of the Oracle GoldenGate 19c Microservices Hub. If the databases are mixed, e.g., Oracle Database 18c to Oracle Database 19c, then the hub requires four different installs. Two of these installs relate to Oracle GoldenGate 19c Microservices, and the other two installs are for Oracle Database Client software specific to each database in the replication stream.

The Oracle GoldenGate 19c Microservices Hub logically looks like this:

Operating System
|
- Oracle GoldenGate 19c for Oracle Database 19c
- Oracle GoldenGate 19c for Oracle Database 18c
- Oracle Database Client for Oracle Database 19c
- Oracle Database Client for Oracle Database 18c

After the software has been installed on the Oracle GoldenGate 19c Microservices Hub, the capture and apply processes would be configured to capture data from the on-premises database (Oracle 19c) and use the data into the Oracle Autonomous Data Warehouse (Oracle 18c).

Move Client Credentials to Hub

To use the credentials for the Autonomous Data Warehouse, they must move to the Oracle GoldenGate 19c Microservices Hub. Using the following command will move the file on most Linux or Unix environments.

```
$ scp -P 3022 wallet_RDADB.zip oracle@localhost:~
```

Unzip Client Credentials

With the Client Credentials on the Oracle GoldenGate 19c Microservices Hub, they need to be unzipped into the TNS_ADMIN directory. In this case, the TNS_ADMIN location is specific for the deployment that will be used for the Oracle Autonomous Data Warehouse.

```
$ unzip ./wallet_RDADB.zip -d $TNS_ADMIN
```

Update SQLNet.ora

The SQLNet.ora file needs to be updated to point to the correct TNS_ADMIN directory. To update the file, run a sed command and specify the directory where the sqlnet.ora file is located. The below example is provided for you.

```
$ sed -i -e "s|\?|/opt/app/oracle/product/client/instantclient_18_5|"
/opt/app/oracle/product/client/instantclient_18_5/network/admin/sqlnet.ora
```

Source Deployment

The source deployment needs to be configured to ensure capturing of any transaction that is committed on the source database. Items to configure for the source deployment are as follows:

- Configure Source Credentials Store – CDB and PDB
- Add SchemaTrandata
- Add Extract
- Configure Target Credentials Store – Autonomous Data Warehouse

Configure Capture Credential Store – CDB and PDB

The source credential store is where the logins for the container database and the pluggable database will reside. Both connections, container and pluggable database, are used to ensure that capturing from the source is successful. The following cURL commands illustrate how to add these credentials in two simple commands.

Create Container Credential via cURL

```
curl -L -X POST
'https://localhost/Atlanta/adminsrvr/services/v2/credentials/OracleGoldenGate
/SGGATE' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "userid":"c##ggate@10.20.1.2:1539/orcl.oracle.com",
    "password":"WElcome12345##"
}'
```

Create Pluggable Credential via cURL

```
curl -L -X POST
'https://localhost/Atlanta/adminsrvr/services/v2/credentials/OracleGoldenGate
/PDBGGATE' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
--data-raw '{
    "userid":"ggate@10.20.1.2:1539/orclpdb1.oracle.com",
    "password":"WElcome12345##"
}'
```

In the above two cURL commands, credentials are established for both the container and pluggable database (SGGATE and PDBGGATE). These names are identified by looking at the REST API endpoints that are part of the URL within the command. Both of these credentials should be able to login to the database/service identified in the data section using the credentials specified.

Add SchemaTrandata via cURL

Before any transactions are captured, the schema and the tables within the schema have to enable for supplemental logging. Enabling supplemental logging is done by applying *`schematrandata`* to the schema within the pluggable database. The following cURL command can be ran to add schematrandata to the configuration:

Adding SchemaTrandata

```
curl -L -X POST
'https://localhost/Atlanta/adminsrvr/services/v2/connections/OracleGoldenGate
.SGGATE/trandata/schema' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
      "operation":"add",
      "schemaName":"orclpdb1.tstusr",
      "prepareCsnMode":"nowait"
}'
```

There are two things to notice here. The first being that the connection that is established is to the container database via the SGGATE credential created earlier. The other thing is that the schema name being used makes a reference to the pluggable database where the schema resides. Both of these are needed since the redo logs reside at the container database layer within the database.

Add Capture process via cURL

Adding the capture (extract) process via cURL is just as simple as the other steps that have been performed on the source. To add the capture (extract) process use the following cURL command:

```
curl -L -X POST
'https://localhost/Atlanta/adminsrvr/services/v2/extracts/EXT4ADW' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "config":[
        "Extract     EXT4ADW",
        "ExtFile     aa",
        "UseridAlias SGGATE",
        "sourcecatalog orclpdb1;",
        "table tstusr.addresses;",
        "table tstusr.customers;",
        "table tstusr.orders;",
        "table tstusr.order_items;",
        "table tstusr.card_details;",
        "table tstusr.logon;",
        "table tstusr.product_information;",
        "table tstusr.inventories;",
        "table tstusr.product_descriptions;",
        "table tstusr.warehouses;",
        "table tstusr.orderentry_metadata;"
    ],
    "source":{
        "tranlogs":"integrated"
    },
    "credentials":{
        "alias":"SGGATE"
    },
    "registration":{
        "containers": [ "orclpdb1" ],
        "optimized":false
    },
    "begin":"now",
    "targets":[
        {
            "name":"aa",
            "sizeMB":15
        }
    ],
    "status":"running"
}'
```

When looking at the above cURL command, don't let it confuse you. Every step that traditionally would be performed in a GGSCI session is combined into a single command and will produce a running extract process. The *–data-raw* section of this command highlights all these steps. Here is a quick review of what each part does:

Config – This is the parameter file that will be created for the capture process

Source – This sets the capture process to use the transaction logs in integrated mode
Credentials – This tells the capture process what credentials to login to the database with
Registration – This shows the capture process what pluggable databases it should register for usage
Begin – Sets the capture process to begin immediately
Targets – Sets what local trail file should be used and the size of the trail file
Status – Sets what state the capture process should be built-in. Setting it to "running" will bring the extract up upon being built.

Configure Apply Credentials – Autonomous Data Warehouse via cURL

To establish credentials for the Autonomous Data Warehouse, Oracle has already provided a GoldenGate user that can be configured and used for replication, GGADMIN. The only thing configure within the Autonomous Data Warehouse is to set the password and unlock this account.

```
     SQL> alter user ggadmin identified by <password> account unlock;
```

Once the account is unlocked, configuring the apply credentials is similar to what was performed earlier with the capture side. Use the following cURL command to add the apply credentials for the Autonomous Data Warehouse.

```
curl -L -X POST
'https://localhost/ADW/adminsrvr/services/v2/credentials/OracleGoldenGate/TGG
ATE' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "userid":"ggadmin@rdadb_low",
    "password":"WElcome12345##"
}'
```

Add Checkpoint Table

After the apply credentials, the next thing to configure is the checkpoint table. Checkpoint tables keep track of the apply process and where it is at in the apply stream.

Adding a checkpoint table is simple, as illustrated in the below cURL command.

```
curl -L -X POST
'https://localhost/ADW/adminsrvr/services/v2/connections/OracleGoldenGate.TGG
ATE/tables/checkpoint' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "operation":"add",
    "name":"ggadmin.checkpoint"
}'
```

With a successful creation of the checkpoint table on the Autonomous Data Warehouse, the schema and table name will be pre-appended with a string that is random-looking but specific to the Autonomous Data Warehouse.

Add Automatic Heartbeat Table

Automatic heartbeat is the basic way to gauge latency within the replication stream. This table only needs to be set up on the target side (Autonomous Data Warehouse) and will provide information on the performance of the replication. The location of the Automatic Heartbeat table is configured by using the GLOBALS file within the deployment. For configuring the Automatic Heartbeat Table, use the following cURL command:

```
curl -L -X POST
'https://localhost/ADW/adminsrvr/services/v2/connections/OracleGoldenGate.TGG
ATE/tables/heartbeat' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "frequency":30,
    "purgeFrequency":1,
    "retentionTime":1,
    "trackingExtractRestart":true
}'
```

Distribution Path

Since the source database is 19c and the Autonomous Database is 18c, this means that we have to have a way to move the trail files from the 19c environment to the 18c environment. By configuring the Distribution Path in 19c and pointing it to the Receiver Service in 18c, the trail file will be shipped between deployments on the same hub machine without going to a public facing network. One thing to notice when building a distribution path on a hub machine is that you do not need to reference or move the trail file out through the Nginx reverse proxy. Looking at the URI for the distribution path, you will notice that port numbers are specified to make the connection easier to established. Port numbers can be used in the URI as long as you are staying within the same host machine.

```
curl -L -X POST 'https://localhost/Atlanta/distsrvr/services/v2/sources/N2E'
\
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "name": "N2E",
    "status": "running",
    "source": {
        "uri":
"trail://localhost:16002/services/Atlanta/distsrvr/v2/sources?trail=aa"
    },
    "target": {
        "uri": "wss://localhost:17003/services/v2/targets?trail=ab"
    }
}'
```

Initial Load Process

The initial load process is when data in the source database has to be copied to the target database before transactions can be applied. Data loading ensures that all needed data is in the target environment, so applied transactions do not error out, or data becomes missing.

There are multiple ways of loading data into the Autonomous Data Warehouse. Typically, the standard data loading approach is to use Oracle Data Pump Export and Import, which is supported by the Autonomous Data Warehouse. Another way to perform the initial load is to use Oracle GoldenGate 19c Microservices and perform the initial load using Trail Files. Doing an initial load with trail files requires another capture process and apply process that reads an initial load trail file.

The steps to perform an initial load using trail files are:

1. Ensure that the associated table and object structures reside in the target (no data)
2. Build an initial load replicat
3. Build an initial load extract
4. Build an initial load path

These steps will load all data into the Autonomous Data Warehouse and prepare for continuous change data capture (CDC).

Associated Tables on Autonomous Data Warehouse

Moving the data structure into an Autonomous Data Warehouse can be done in different ways. Ideally, you would do an export and import of the data using standard Oracle tools like Oracle Data Pump. For simplicity purposes, since this whitepaper is using a smaller set of tables, just creating the tables and associated keys will work. Leaving you with empty tables on the Autonomous Data Warehouse.

Building Initial Load Replicat

The initial load apply (replicat) process is likened to a “special run” replicat. The cURL command below illustrates how to set this up within the deployment for the Autonomous Data Warehouse.

```
curl -L -X POST 'https://localhost/ADW/adminsrvr/services/v2/replicats/RLOAD'
\
-H 'Content-Type: application/text' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "description": "Create an initial load replicat - File Based",
    "checkpoint": {
        "table": "ggadmin.checkpoint"
    } ,
    "config": [
        "Replicat RLOAD",
        "UseridAlias TGGATE",
        "Map orclpdb1.tstusr.addresses, TARGET tstusr.addresses;",
        "Map orclpdb1.tstusr.customers, TARGET tstusr.customers;",
        "Map orclpdb1.tstusr.orders, TARGET tstusr.orders;",
        "Map orclpdb1.tstusr.order_items, TARGET tstusr.order_items;",
        "Map orclpdb1.tstusr.card_details, TARGET tstusr.card_details;",
        "Map orclpdb1.tstusr.logon, TARGET tstusr.logon;",
        "Map orclpdb1.tstusr.product_information, TARGET
tstusr.product_information;",
        "Map orclpdb1.tstusr.inventories, TARGET tstusr.inventories;",
        "Map orclpdb1.tstusr.product_descriptions, TARGET
tstusr.product_descriptions;",
        "Map orclpdb1.tstusr.warehouses, TARGET tstusr.warehouses;",
        "Map orclpdb1.tstusr.orderentry_metadata, TARGET
tstusr.orderentry_metadata;"
    ],
    "credentials": {
        "alias": "TGGATE"
    },
    "mode": {
        "parallel": false,
        "type": "nonintegrated"
    },
    "registration": "none",
    "source": {
        "name": "zb"
    },
    "status": "running"
}'
```

Build Initial Load Extract

You now need to build an initial load capture process, similar to creating the initial load replicat. The below cURL command will establish the initial load capture (extract) process.

```
curl -L -X POST
'https://localhost/Atlanta/adminsrvr/services/v2/extracts/LOAD' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
"description": "Create an initial load extract - File Based",
"config":
    [
        "Extract LOAD",
        "UseridAlias SGGATE",
        "ExtFile za Megabytes 250 Purge",
        "sourcecatalog orclpdb1",
        "TABLE tstusr.addresses;",
        "TABLE tstusr.customers;",
        "TABLE tstusr.orders;",
        "TABLE tstusr.order_items;",
        "TABLE tstusr.card_details;",
        "TABLE tstusr.logon;",
        "TABLE tstusr.product_information;",
        "TABLE tstusr.inventories;",
        "TABLE tstusr.product_descriptions;",
        "TABLE tstusr.warehouses;",
        "TABLE tstusr.orderentry_metadata;"
    ],
    "source": "tables",
    "status": "running"
}'
```

Build Initial Load Path

Lastly to move all the data for the initial load, you need to build another distribution path that will be used only for the initial load. The following cURL command will build the path:

```
curl -L -X POST
'https://localhost/Atlanta/distsrvr/services/v2/sources/INITLOAD' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "name": "INITLOAD",
    "status": "running",
    "source": {
        "uri":
"trail://localhost:16002/services/Atlanta/distsrvr/v2/sources?trail=za"
    },
    "target": {
        "uri": "wss://localhost:17003/services/v2/targets?trail=zb"
    }
}'
```

Add Apply process via cURL

After the initial load is done, adding the final piece for change data capture (CDC) replication can be done. Adding the apply (replicat) process can be straight forward, but you also want to handle any potential collisions you might come across. The cURL to create and run the Apply (replicat) process is below:

```
curl -L -X POST 'https://localhost/ADW/adminsrvr/services/v2/replicats/R4ADW'
\
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
    "config":[
        "Replicat    R4ADW",
        "UseridAlias TGGATE",
        "Map orclpdb1.tstusr.addresses, TARGET tstusr.addresses;",
        "Map orclpdb1.tstusr.customers, TARGET tstusr.customers;",
        "Map orclpdb1.tstusr.orders, TARGET tstusr.orders;",
        "Map orclpdb1.tstusr.order_items, TARGET tstusr.order_items;",
        "Map orclpdb1.tstusr.card_details, TARGET tstusr.card_details;",
        "Map orclpdb1.tstusr.logon, TARGET tstusr.logon;",
        "Map orclpdb1.tstusr.product_information, TARGET
tstusr.product_information;",
        "Map orclpdb1.tstusr.inventories, TARGET tstusr.inventories;",
        "Map orclpdb1.tstusr.product_descriptions, TARGET
tstusr.product_descriptions;",
        "Map orclpdb1.tstusr.warehouses, TARGET tstusr.warehouses;",
        "Map orclpdb1.tstusr.orderentry_metadata, TARGET
tstusr.orderentry_metadata;"
    ],
    "source":{
        "name":"ab"
    },
    "credentials":{
        "alias":"TGGATE"
    },
    "checkpoint":{
        "table":"ggadmin.checkpoint"
    },
    "mode":{
      "type":"nonintegrated",
      "parallel": true
    },
    "begin":"now",
    "status":"running"
}'
```

Clean Up

After replication has caught up and data is smoothly flowing, then it is time to clean up. The items that you would want to clean up are all the items related to the initial load, including the initial load capture (extract), apply (replicat), and distribution path. Using cURL and the REST API point, clean up can be run manually or scripted. In the example below, you are merely using cURL to remove each of these items.

Removing Initial Load Capture (Extract)

```
curl -L -X DELETE
'https://localhost/Atlanta/adminsrvr/services/v2/extracts/LOAD' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
--data-raw ''
```

Remove Initial Load Apply (Replicat)

```
curl -L -X DELETE
'https://localhost/ADW/adminsrvr/services/v2/replicats/RLOAD' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
--data-raw ''
```

Remove Initial Load Distribution Path

```
curl -L -X DELETE
'https://localhost/Atlanta/distsrvr/services/v2/sources/INITLOAD' \
-H 'Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyMzQ1IyM=' \
-H 'Content-Type: text/plain' \
--data-raw '{
      "distpath":"INITLOAD"
}'
```

Summary

Building data warehouses is nothing new in the industry. Organizations use data warehouses for their reporting purposes and need to have current data on a real-time basis. By using Oracle GoldenGate 19c Microservices, organizations can quickly establish replication paths, capture data, and apply data to environments. By using the Autonomous Database Warehouse organizations and leverage the data that is moving into it for reporting and or apply machine learning to gain greater insight into their data. In combining both of the products, organizations can quickly scale their data warehouse while ensuring consistent data movement for real-time data analysis.

For more information, contact RheoData at `solutions@rheodata.com`.