ORACLE

Oracle OpenWorld 2019

SAN FRANCISCO

Safe Harbor

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

TRN6112
Oracle GoldenGate for Big Data Overview

Randall Richeson

Senior Principal Instructor
Oracle University

What's Ahead

Today

Introduction to GoldenGate for Big Data

Configuring and Using the Oracle GoldenGate HDFS Handler

Configuring and Using the Oracle GoldenGate HBase Handler

Configuring and Using the Oracle GoldenGate Kafka Handler

Configuring and Using the Oracle GoldenGate Cassandra Handler

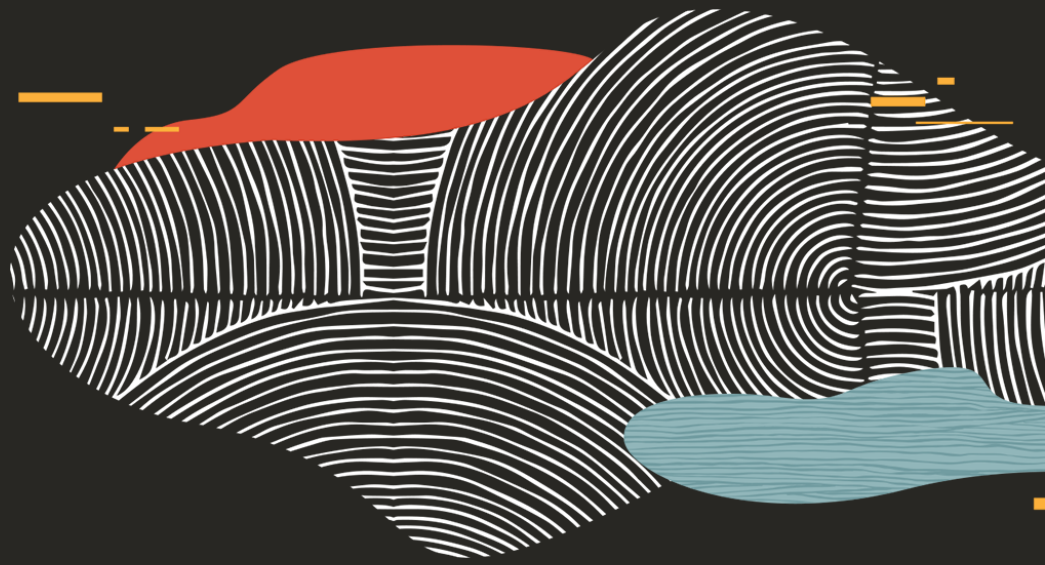
Our Goal

Explain GoldenGate for Big Data Architecture

Show how to configure GoldenGate to replicate changes from Oracle Database redo logs to big data targets including:

- HDFS
- HBase table namespace
- Kafka topic
- Cassandra keyspace

Introduction to GoldenGate for Big Data



- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Product Overview

Oracle GoldenGate for Big Data is a module that adds **pluggable functionality** to the Oracle GoldenGate Java Adapters framework.

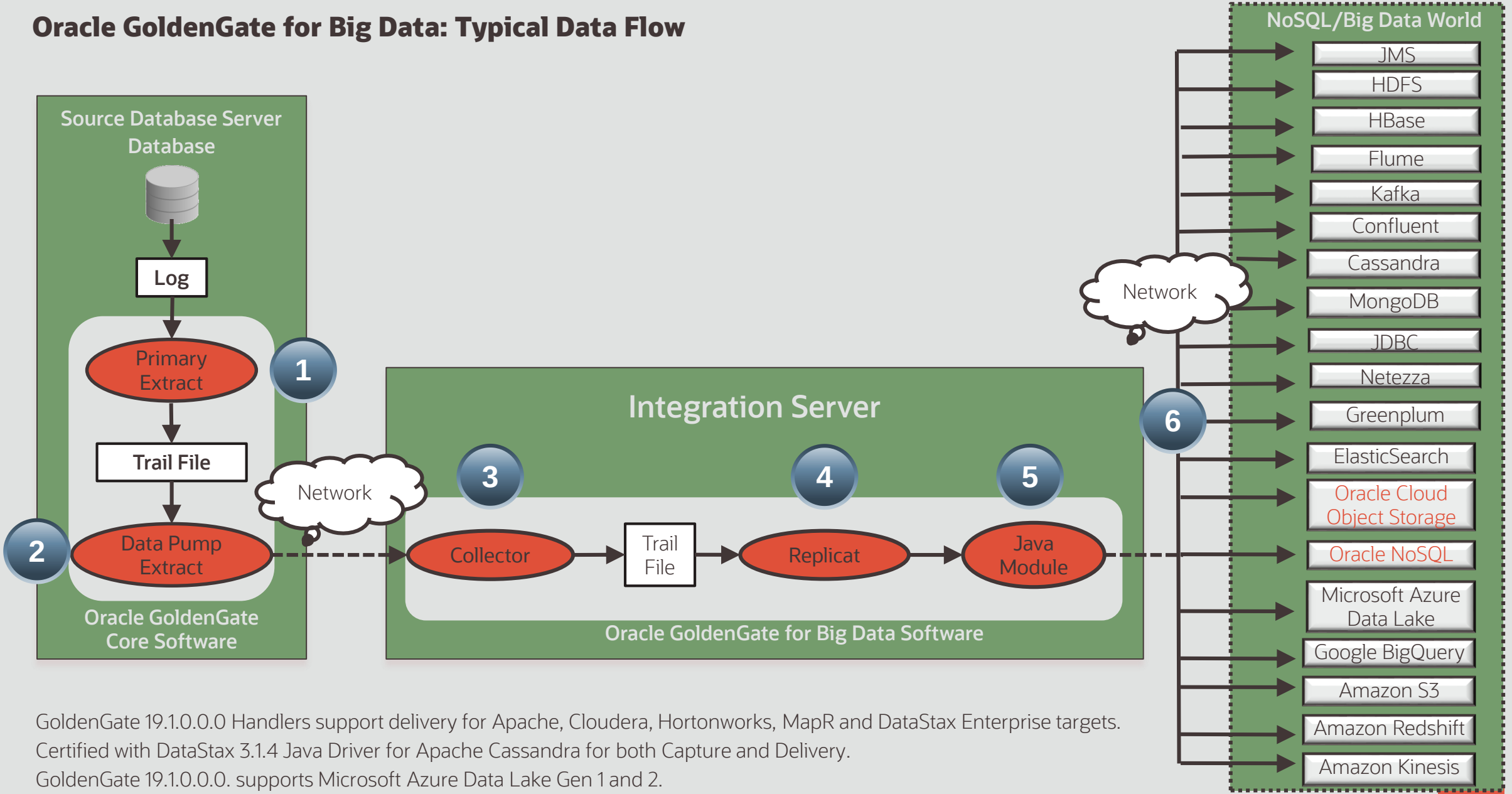
Traditionally, the **Java Adapters** framework read the trail files produced by core GoldenGate instances and helped integration with **JMS queues and/or files**.

Oracle **GoldenGate for Big Data** extends the Java Adapters framework, ensuring interoperability with the Hadoop ecosystem, Apache and DataStax Enterprise Cassandra, MongoDB, Generic JDBC, and other targets.

Integration with Oracle GoldenGate Core Instances

- The Oracle GoldenGate core software enables capturing transactional changes from a source database and sending those changes as a set of database-independent files called trail files.
- Altering the source data by using mapping parameters, functions, and filtering is an option.
- The Oracle GoldenGate Java Adapters framework and the Big Data plug-in can read the trail files and deliver the transactional content to Hadoop, NoSQL databases, JDBC and other big data targets.

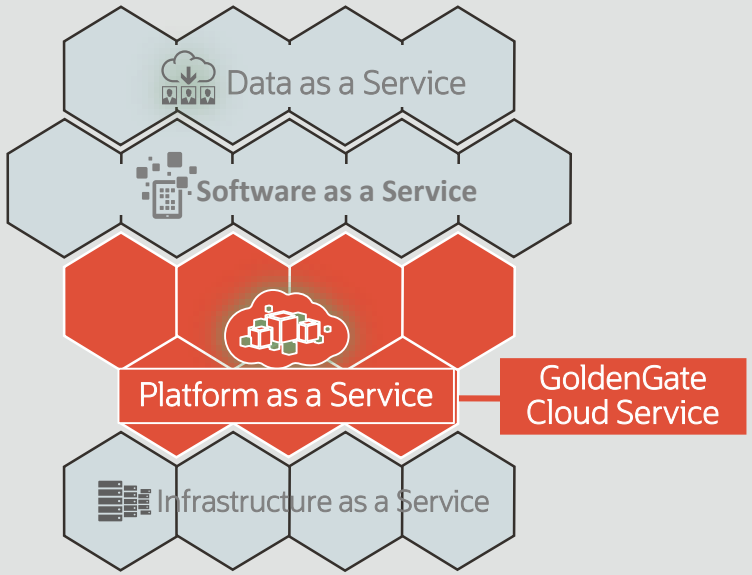
Oracle GoldenGate for Big Data: Typical Data Flow



GoldenGate 19.1.0.0.0 Handlers support delivery for Apache, Cloudera, Hortonworks, MapR and DataStax Enterprise targets.
Certified with DataStax 3.1.4 Java Driver for Apache Cassandra for both Capture and Delivery.
GoldenGate 19.1.0.0.0. supports Microsoft Azure Data Lake Gen 1 and 2.
GoldenGate 19.1.0.0.0 de-supports Flume.

Oracle GoldenGate Cloud Service

PaaS Solution

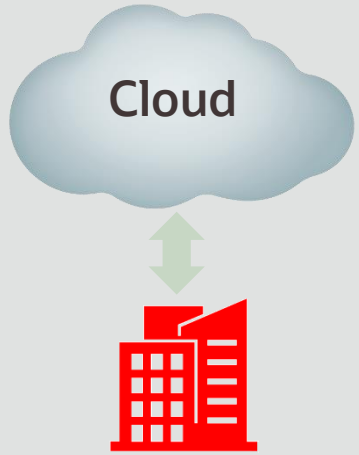


Deep Integration
In a Broader Solution

Built on GoldenGate

High Level Standard
Industry Proven Solution
Secure, Robust, and Scalable

Hybrid Deployment



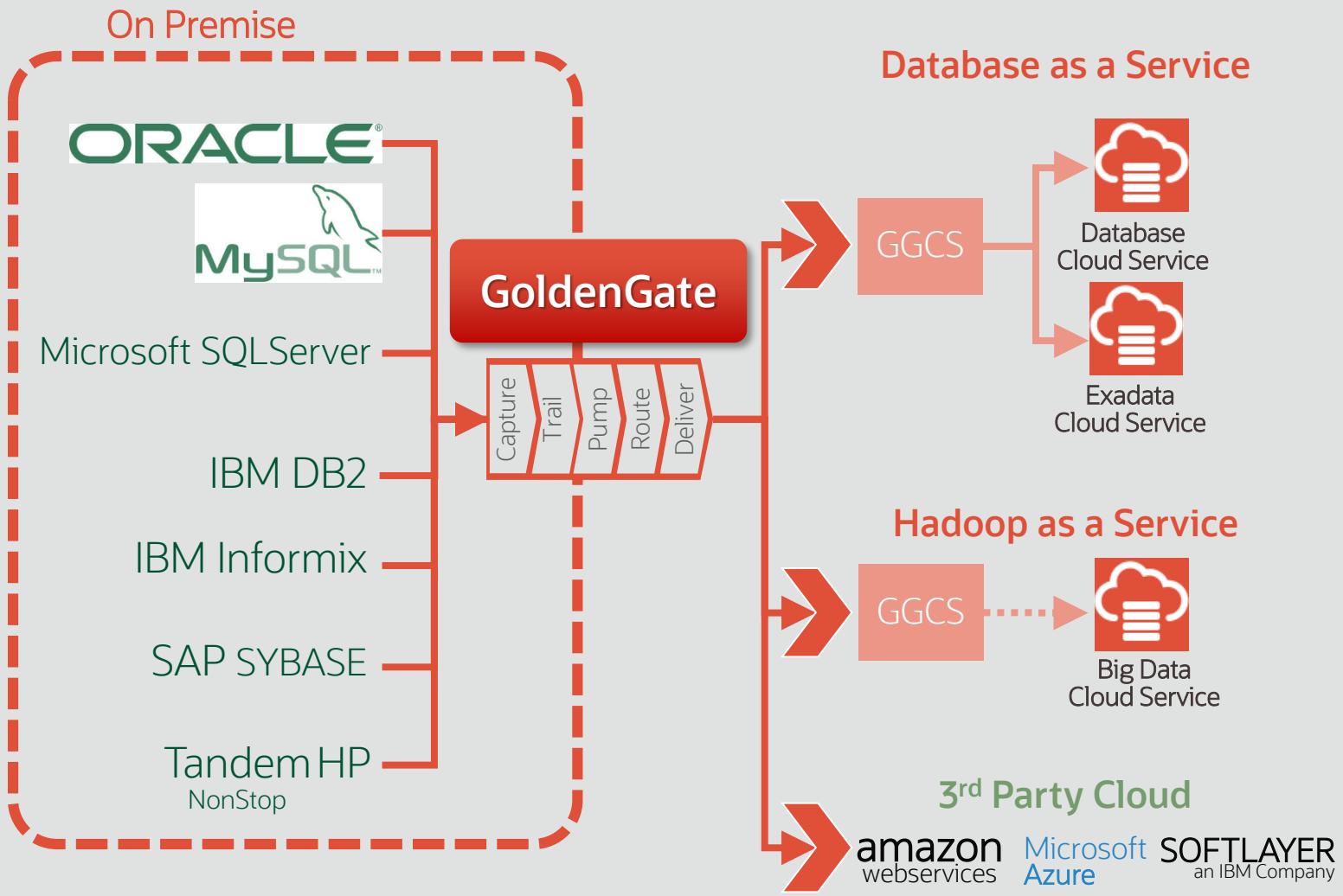
On-Premises

Consistent Standard
Architecture and Products

Oracle GoldenGate Cloud Service

- Real-Time Data Replication for Cloud
- Subscription-based real-time data replication service for Cloud
- Non-intrusive log-based change data capture
- High throughput encrypted and compressed data transport via secure TCP/IP between on-premises and cloud
- High performance parallel data delivery
- Easy to use Web UI for provisioning and managing sandbox environments
- Prebuild integration within Oracle Database Cloud Service

Oracle GoldenGate Cloud Service



GoldenGate Cloud Service

- ✓ Available in the Public Cloud via Subscription or Hourly basis

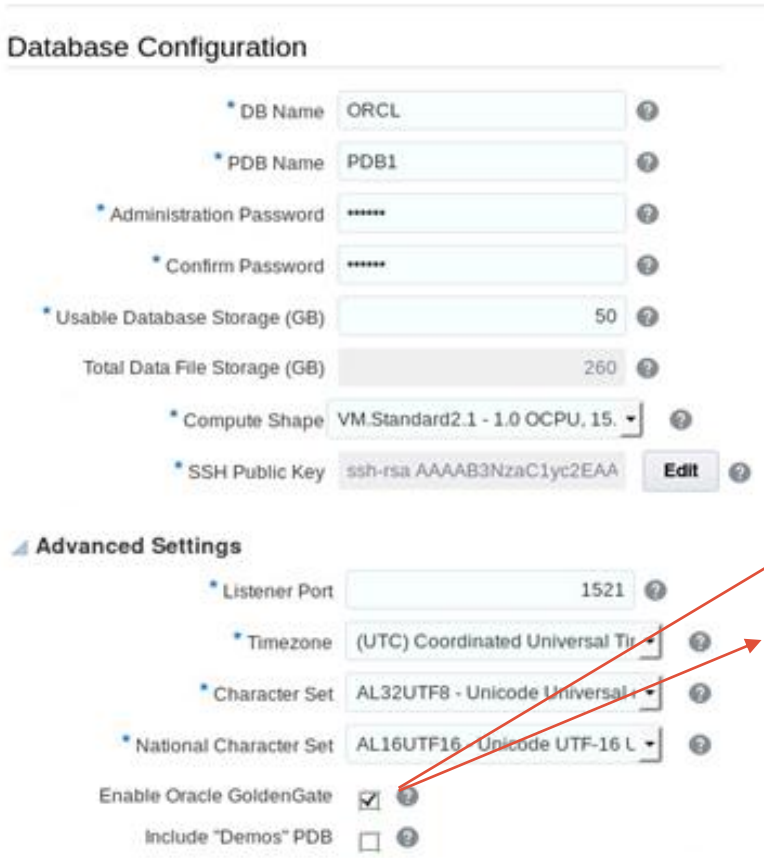
Key Benefits

- ✓ Oracle Database Cloud Service delivery
- ✓ Exadata Cloud Service delivery
- ✓ Big Data Cloud Service delivery to Hadoop and NoSQL

3rd Party Cloud

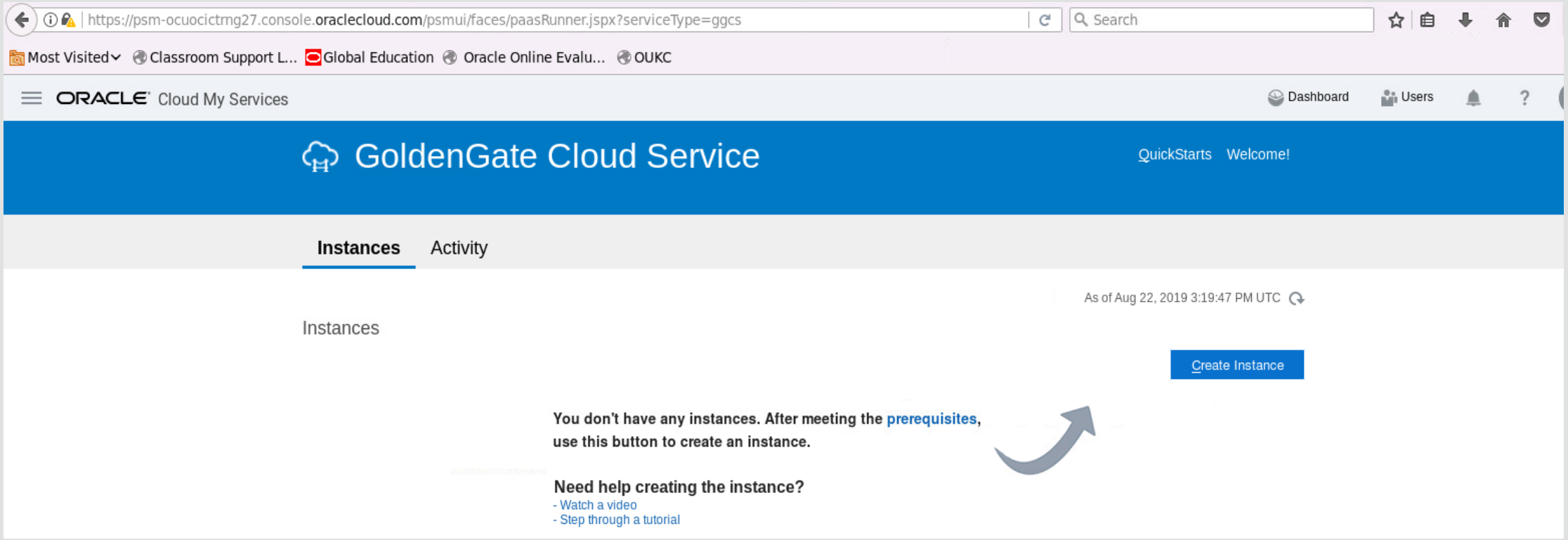
- ✓ More Choices run as BYOL on other Clouds for delivery to any supported Database.

Integration with Oracle Database Cloud Service Example

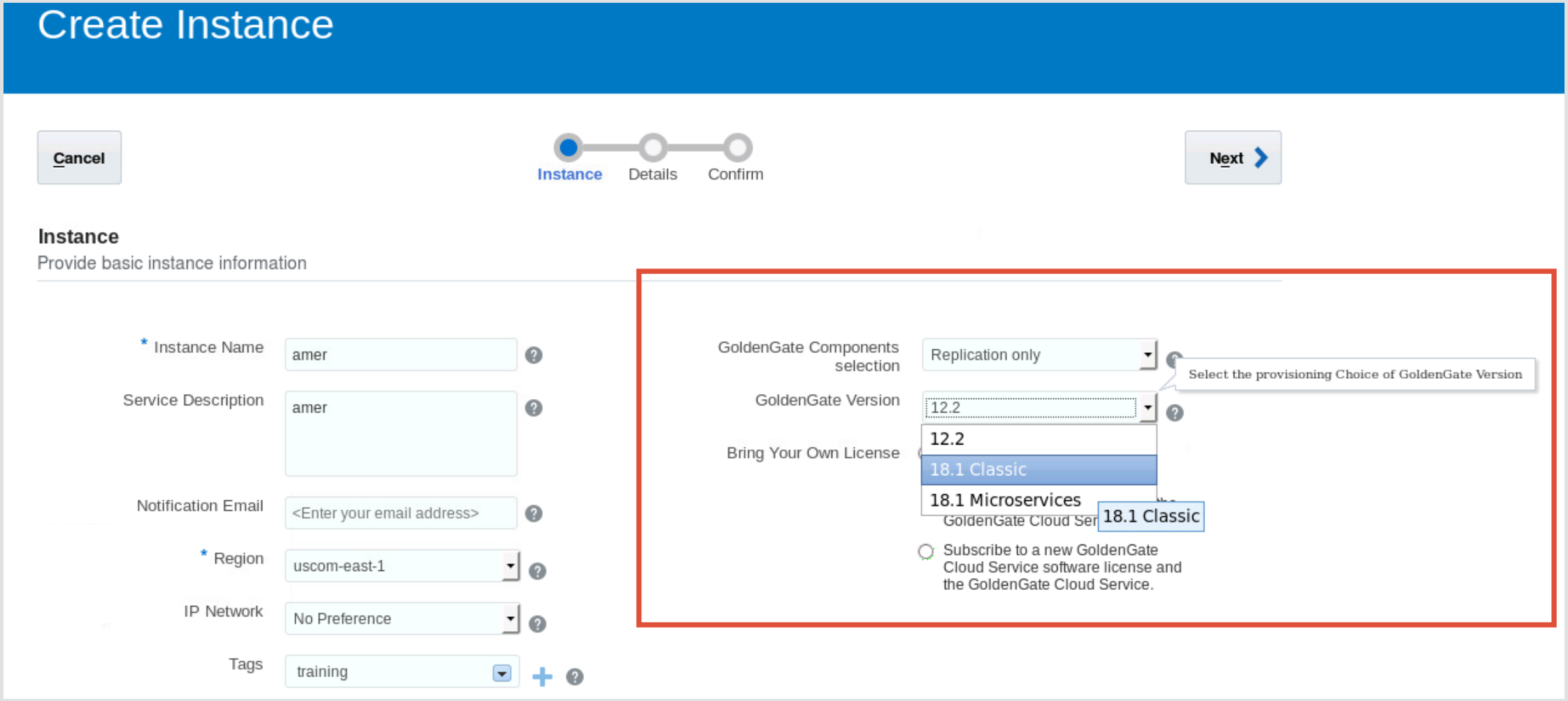



```
SQL> select NAME, SUPPLEMENTAL_LOG_DATA_MIN, CDB, FORCE_LOGGING from v$database;

NAME                 SUPPLEME CDB FORCE_LOGGING
--------- -------- --- ---------------------------------------
ORCL                 YES                  YES YES

SQL> sho parameter enable_gold

NAME                                                                 TYPE                 VALUE
-------------------------------------- ---------- ------------------------------
enable_goldengate_replication          boolean              TRUE

SQL> sho pdbs

    CON_ID CON_NAME                                                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
                    2 PDB$SEED                                       READ ONLY  NO
                    3 PDB1                                           READ WRITE NO
SQL>
```

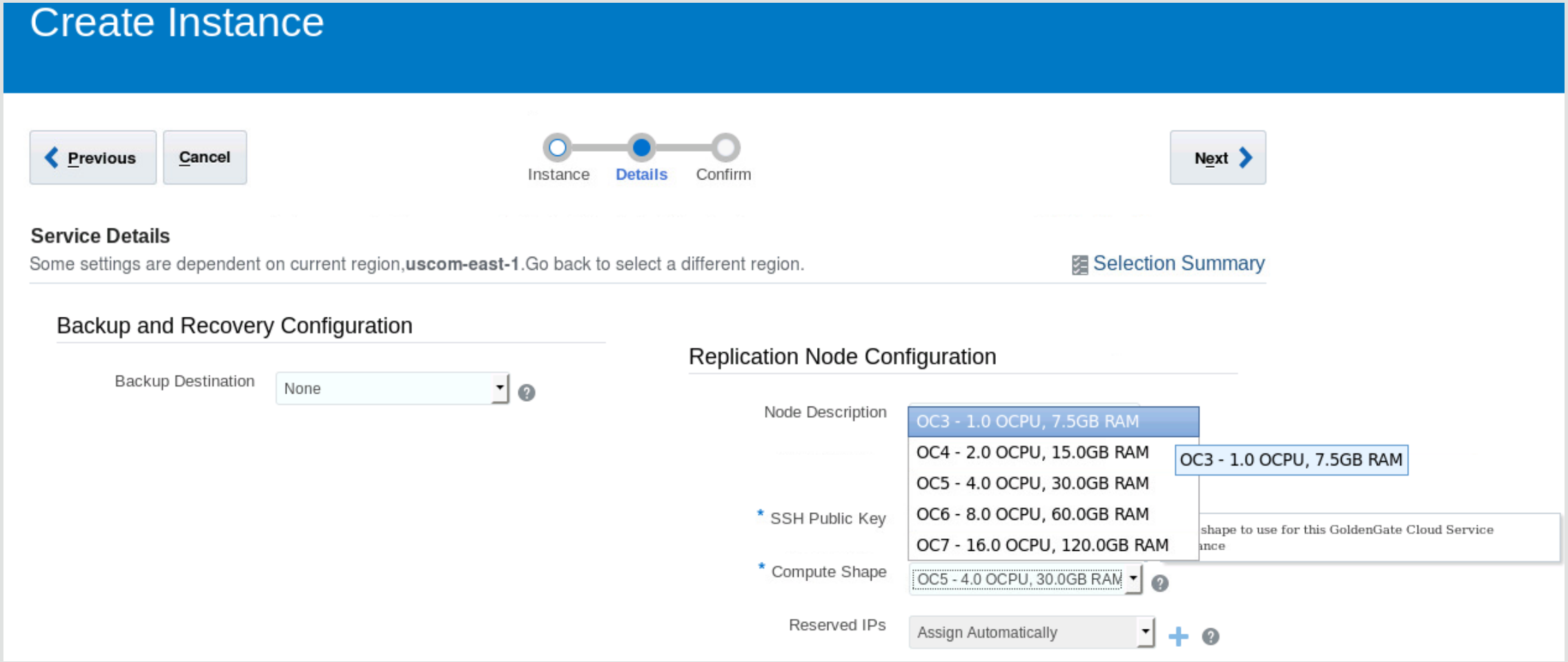
GoldenGate Cloud Service Console Example



GoldenGate Cloud Service supports Classic and Microservices Architectures



GoldenGate Cloud Service Compute Shape Options



GoldenGate Cloud Service Confirmation

Create Instance

Previous | Cancel

Instance — Details — Confirm

Create

Confirmation

Confirm your responses and create this GoldenGate Service instance.

This instance is being created using BYOL terms.

Service

Instance Name:	amer
Service Description:	amer
Bring Your Own License:	BYOL
Service Level:	GoldenGate Cloud Service
SSH Public Key:	ssh-rsa AAAAB3NzaC1yc2EAAAA...
Region:	uscom-east-1
IP Network:	No Preference
Tags:	training ...

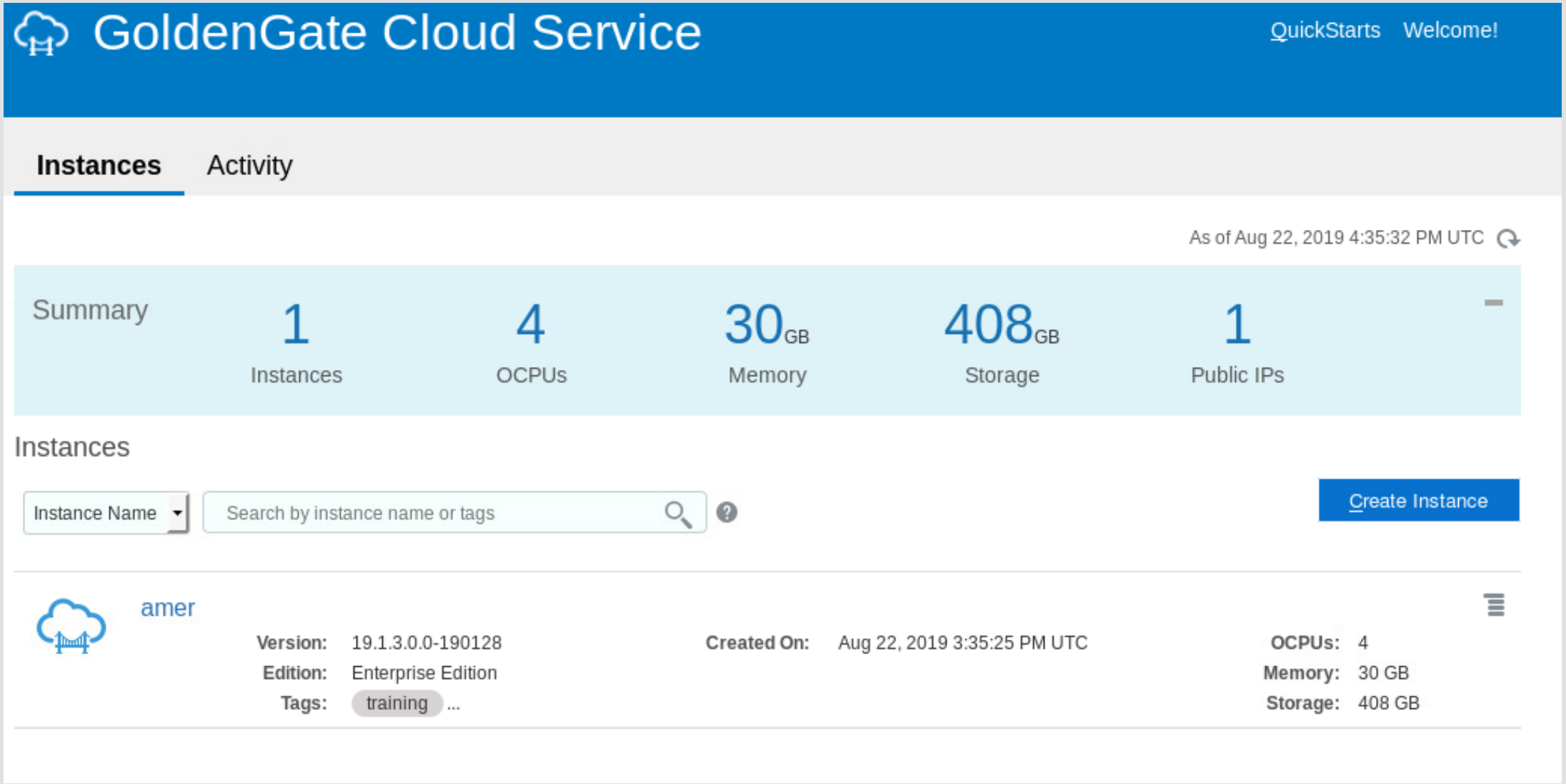
GoldenGate Version Details

GoldenGate Version:	18.1 Classic

Replication Node Configuration

Compute Shape:	OC5 - 4.0 OCPU, 30.0GB RAM

GoldenGate Cloud Service Summary



Introduction to GoldenGate for Big Data

- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Oracle GoldenGate Adapter Architecture

- Two products are based on the Oracle GoldenGate Adapter.

  The Oracle GoldenGate Java Adapter, which is the overall framework, helps you to implement **custom** code to handle trail records according to their specific requirements.

  Oracle GoldenGate for Big Data, which contains built-in support to write operation data from Oracle GoldenGate trail records into various Big Data targets and NoSQL databases.

- You **do not** need to write any custom code to integrate GoldenGate with Big Data and NoSQL applications.

Crossover Functionality

Feature	OGG Java Adapter	OGG for Big Data
Read JMS messages and deliver them as an Oracle GoldenGate trail.	✓	✓
Read an Oracle GoldenGate trail and deliver transactions to a JMS provider or other messaging system or custom application.	✓	✓
Read an Oracle GoldenGate trail and write transactions to a file that can be used by other applications.	✓	✗
Read an Oracle GoldenGate trail and write transactions to Big Data targets.	✗	✓

Oracle GoldenGate for Big Data 12.3.2.1 there is a new Flat file writer designed to load to a local file system and then load completed files to another location like HDFS.

Introduction to GoldenGate for Big Data

- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Oracle GoldenGate Java Adapter: Message Capture and Delivery

Oracle GoldenGate message capture reads messages from a JMS queue and communicates with an Oracle GoldenGate Extract process to generate a set of trail files that contain the processed data.

The message capture processing is implemented as a Vendor Access Module (VAM) plug-in to a generic Extract process.

Oracle GoldenGate Java Adapter delivery applies transactional changes to targets other than a relational database that can be used for ETL tools (DataStage, Ab Initio, Informatica), Integration (JMS messaging), Analytics (Big Data), or custom APIs.

The delivery occurs through a dynamically linked or shared library and a set of Java libraries used to communicate with the Replicat process using the Java Native Interface (JNI).

Oracle GoldenGate Java Adapter: No Custom Code Required !

- All Oracle GoldenGate Java Adapters, Big Data Handlers, and formatters are configured through predefined properties, which are stored in property files located in the standard Oracle GoldenGate directory (dirprm).
- Writing to Big Data and NoSQL targets in various formats is configured by using a set of properties and requires no custom programming.
- Replicating relational tables into JMS queues is also accomplished through predefined properties.
- You can use gendef to create an Oracle GoldenGate source definitions file to parse the resulting trail written by the Extract process.

Introduction to GoldenGate for Big Data

- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Configuring the Vendor Access Module (VAM) Extract

- JMS capture works with the Oracle GoldenGate Extract process.
- The prerequisites to run the Java message capture process include:

  Oracle GoldenGate for Java Adapter
  Extract process
  Extract parameter file configured for message capture
  Source definitions file created with gendef or a COBOL copybook showing incoming format
  Java 8 or higher installed on the host machine

Adding the VAM Extract

- The GGSCI utility is used to add the message capture VAM.

```
ADD EXTRACT extvam, VAM
ADD EXTTRAIL dirdat/id, EXTRACT extvam, MEGABYTES 100
```

- The process name (EXTVAM in the example) can be replaced with any name that is no more than eight characters.

- The trail identifier (id in the example) can be any two characters.

VAM Extract Parameters

Parameter	Explanation
EXTRACT <Extract Name>	Is the name of the Extract process
VAM libggjava_vam.so PARAMS (dirprm/<ext name>.properties)	Specifies the name of the VAM library and the location of the properties file
TranLogOptions VAMCOMPATIBILITY 1	Specifies that the original (1) implementation of the VAM is to be used
TranLogOptions GETMETADATAFROMVAM	Specifies that metadata will be sent by the VAM
ExtTrail dirdat/<xx>	Specifies the identifier of the target trail that the Extract process creates
Table <schema>.<table>;	Specifies metadata information about tables

VAM Properties File

- To process JMS messages:
  - Configure the connection to the JMS interface
  - Retrieve and parse the messages in a transaction
  - Write each message to a trail
  - Commit the transaction
  - Remove its messages from the queue
- You configure message capture using the VAM properties file.
  - This file is identified by the PARAMS option of the Extract VAM parameter.
  - It determines logging characteristics, parser mappings, and JMS connection settings.

VAM Properties File: Example

```
…Omitted lines…
### JMS settings
gg.jms.connectionFactory=ConnectionFactory
gg.jms.destination=dynamicQueues/testWQ
java.naming.provider.url=tcp://localhost:61616?jms.prefetchPolicy.all=1000
java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQFactory
### native JNI library log configuration (output directory,
###                                                 filename,prefix)
goldengate.log.logname=dirrpt/extvam_jni
goldengate.log.level=INFO
goldengate.log.tostdout=false
goldengate.log.tofile=true
### XML parsing properties
parser.type=xml
xml.sourcedefs=dirdef/tc.def
xml.rules=tx_rule
…Many more omitted lines…
```

Introduction to GoldenGate for Big Data

- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Message Parsing: Overview

The parser translates JMS text message data and header properties into an appropriate set of transactions and operations to pass into the VAM interface.

Several **bits** of information must be made available to the message parser:

- Transaction Identifier
- Sequence Identifier
- Time stamp
- Table Name
- Operation Type
- Column Data

The parser obtains this data **from JMS header** properties, system generated values, static values, or in some parser-specific way.

Parser Types

- The Oracle GoldenGate message capture adapter supports three types of parsers:

  Fixed: Messages contain data presented as fixed width fields in contiguous text.
  Delimited: Messages contain data delimited by field and end-of-record characters.
  XML: Messages contain XML data accessed through XPath expressions.

- Configured properties determine how the selected parser gets data and how the source definitions are converted to target definitions.
- These properties are configured in the VAM properties file.

Parser Properties Example

```
…Omitted lines…
### delimited parser properties
parser.type=delimited
delim.schematype=sourcedefs
delim.sourcedefs=dirdef/tc.def
delim.header=schemaandtable,timestamp,optype
delim.optype=optype
delim.seqid=*seqid
delim.quote="
delim.quote.escaped=\"
delim.timestamp=timestamp
delim.txid=*txid
delim.fielddelim=,
delim.fielddelim.escaped=\,
delim.linedelim=\X0a
…Many more omitted lines…
```

Agenda: Introduction to GoldenGate for Big Data

- Oracle GoldenGate for Big Data: Product Overview
- Describing the Java Adapter and Oracle GoldenGate for Big Data
- Configuring and Using the Java Adapter
- Configuring Message Capture
- Message Parsing
- Configuring Message Delivery

Replicat for Java Delivery

Java Delivery is compatible with the Replicat process. Transaction data is read from the trail files and delivered to the Java Delivery module across the JNI interface.

Oracle GoldenGate for Big Data helps streaming data to Big Data, NoSQL, and other targets.

The Java Delivery module can be configured to stream data into various targets (JMS, file writing, and custom integrations).

The SPECIALRUN parameter is supported for JMS and other targets.

Adding the Java Delivery Replicat

- The GGSCI utility is used to add the delivery Replicat.

```
ADD REPLICAT javarep, EXTTRAIL ./dirdat/bb
```

- Replicat consumes a local trail (dirdat/bb) and sends the data to the Java Delivery module which then processes the data and applies it to the target.
- The Replicat process name must be eight characters or less, and the trail file name must be two characters.
- An optional java delivery checkpoint file is used instead of a checkpoint table.

Replicat Parameters

Parameter	Explanation
REPLICAT <Replicat Name>	All Replicat parameter files start with the Replicat name.
SOURCEDEFS ./dirdef/xxx.def	If the input trail files do not contain the metadata records, the Replicat process requires metadata describing the trail data (optional parameter).
TARGETDB LIBFILE libggjava.so SET property=dirprm/<file>	The TARGETDB LIBFILE libggjava.so parameter serves as a trigger to initialize the Java module. The SET subcommand specifies the properties file for the Java Adapter.
MAP schema.*, TARGET *.*;	This specifies the tables to pass to the Java module; tables not included will be skipped.
GROUPTRANSOPS 1000	This groups source transactions into a single larger target transaction for improved performance.

Replicat Parameter File: Example

```
REPLICAT repkfk
TARGETDB LIBFILE libggjava.so SET property=dirprm/kafka.properties
REPORTCOUNT EVERY 1 MINUTES, RATE
GROUPTRANSOPS 10000
MAP QASOURCE.*, TARGET QASOURCE.*;
```

Oracle GoldenGate Java Handlers

Java handlers provide the functionality to push data to integration targets such as JMS or Big Data/NoSQL applications.

Multiple handlers may be separated by commas, and used for debugging or replicating the same information concurrently to multiple targets.

The Java Adapter properties file is used to configure Java Delivery and Java handlers.

Custom handlers can be implemented, in which case the type would be the fully qualified name of the Java class for the handler.

Replicat Java Handler: Kafka Example

```
gg.handlerlist=kafka
gg.handler.kafka.type=kafka
gg.handler.kafka.mode=tx
gg.handler.kafka.topicMappingTemplate=oggtopic
gg.handler.kafka.format=avro_op
gg.handler.kafka.SchemaTopicName=mySchemaTopic
gg.handler.kafka.BlockingSend=false
gg.handler.kafka.includeTokens=false
…Many more omitted lines…
```

Demos: Overview

1-1: Starting and Administering the Hadoop Single-Node Cluster

Verify Big Data single node cluster with Kafka, HBase, and HDFS.

1-2: Verifying the NoSQL Environment

Verify that Cassandra is properly installed.

1-3: Installing Oracle GoldenGate for Big Data

Install the Oracle GoldenGate for Big Data from the distribution kit.

Configuring and Using the Oracle GoldenGate Hadoop HDFS Handler

- Describe the Oracle GoldenGate for Big Data HDFS Handler features
- Identify the HDFS-specific features and parameters
- Configure the HDFS Handler by using the relevant parameters in the property file
- Consider using HDFS partitioning
- Implement Metadata Change Event handling

What is the Hadoop Distributed File System (HDFS) ?

HDFS is the **primary application** for Big Data and is typically installed on multiple machines that work together as a Hadoop cluster.

Hadoop helps you to store very large amounts of data in the cluster that is **horizontally** scaled across the machines in the cluster.

HDFS is a highly **fault-tolerant**, distributed file system and provides support for text and binary files.

HDFS is designed more for **batch** processing rather than interactive use with emphasis on high throughput of data access rather than low latency of data access.

Apache Avro

- Avro is a remote procedure call and data serialization framework developed within Apache's Hadoop project.

  It uses Java Script Object Notation (JSON) for defining data types and protocols, and serializes data in a compact binary format.

  It can provide both a serialization format for persistent data, and a wire format for communication between Hadoop nodes, and from client programs to the Hadoop services.

- The Oracle GoldenGate for Big Data HDFS Handler can write to HDFS in Avro Object Container File (OCF) format.

  Integrates with Apache Hive
  Provides good support for schema evolution

Hadoop Sequence Files

- HDFS supports binary files, including Hadoop Sequence File format, which stores serialized key/value pairs.
- The Oracle GoldenGate for Big Data HDFS Handler supports writing into HDFS in Sequence File format.
  - The key part of the record is set to null by the HDFS Handler.
  - The actual data is set in the value part.
- The Sequence File format offers some distinctive advantages:
  - Optional support is available for compression at different levels (record, block).
  - Files can be split and processed in parallel.
  - HDFS is optimized for large files; Sequence Files can be used as containers for a large number of small files, thus solving Hadoop's drawback of processing a huge number of small files.

Oracle GoldenGate HDFS Handler Supported Formats

- json
- json_row
- xml

- avro_row_ocf
- avro_row_op

- avro_row
- avro_op

- delimitedtext (default)
- sequencefile

Oracle GoldenGate HDFS Handler Classpath Configuration

- The gg.classpath configuration variable for the HDFS must include:
    1. The location of the HDFS **core-site.xml** file
    2. The HDFS client jars
- The **default** location of the HDFS **core-site.xml** file is $HADOOP_HOME/etc/hadoop.
- The **default** locations of the HDFS client jars are the following directories:

```
$HADOOP_HOME/share/hadoop/common/lib/*
$HADOOP_HOME/share/hadoop/common/*
$HADOOP_HOME/share/hadoop/hdfs/lib/*
$HADOOP_HOME/share/hadoop/hdfs/*
```

Relevant Oracle GoldenGate HDFS Handler Configuration Parameters: 1

Parameter	Explanation
gg.handlerlist=hdfs gg.handler.hdfs.type=hdfs	Selects the HDFS Handler for use with Replicat
gg.handler.hdfs.mode	Selects the operation (op) mode or transaction (tx) mode for the Handler. In almost all scenarios, the transaction mode results in better performance.
gg.handler.hdfs.maxFileSize	Selects the maximum file size of the created HDFS files. The default is 1 gigabyte.
gg.handler.name.pathMappingTemplate	You can use keywords interlaced with constants to dynamically generate the HDFS write directory at runtime.
gg.handler.name.fileNameMappingTemplate	You can use keywords interlaced with constants to dynamically generate unique HDFS file names at runtime.

Relevant Oracle GoldenGate HDFS Handler Configuration Parameters: 2

Parameter	Explanation
gg.handler.hdfs.partitionByTable	Determines if data written into HDFS should be partitioned by table (**true** | false)
gg.handler.hdfs.rollOnMetadataChange	Determines if HDFS files should be rolled in the case of a metadata change (**true** | false)
gg.handler.hdfs.format	Selects the formatter for the HDFS Handler for how output data will be formatted
gg.handler.hdfs.includeTokens	Is set to **true** to include the tokens field and tokens key/values in the output, and to **false** to suppress tokens output
gg.handler.hdfs.schemaFilePath	Is set to a legal path in HDFS so that schemas (if available) are written in that HDFS directory. Schemas are currently available only for Avro and JSON formatters.

Oracle GoldenGate HDFS Handler Properties File: Example

```
...Omitted lines…
gg.handlerlist=hdfs
gg.handler.hdfs.type=hdfs
gg.handler.hdfs.mode=tx
gg.handler.hdfs.includeTokens=false
gg.handler.hdfs.maxFileSize=1g
gg.handler.hdfs.pathMappingTemplate=/ogg/${toLowerCase[${fullyQualifiedTableName}]}
gg.handler.hdfs.fileRollInterval=0
gg.handler.name.fileNameMappingTemplate=${fullyQualifiedTableName}_${groupName}_${currentTimeStamp}.txt
gg.handler.hdfs.partitionByTable=true
gg.handler.hdfs.rollOnMetadataChange=true
gg.handler.hdfs.format=delimitedtext
…Many more omitted lines…
```

HDFS Partitioning

- The HDFS Handler supports partitioning of table data by one or more column values.
- Partitioning is handled using the following parameter:

  gg.handler.hdfs.partitioner.<fully_qualified_table_name>

  Example: gg.handler.hdfs.partitioner.dbo.customer=gender
- The preceding partitioning scheme results in the following breakdown in the HDFS structure:

  /ogg/dbo.customer/gender=Female/<data files>
  /ogg/dbo.customer/gender=Male/<data files>

HDFS Metadata Change Events

- Metadata change events are handled by the HDFS Handler.
- The default behavior is to roll the current relevant file in the event of a metadata change event.

  This behavior separates the results of metadata changes into different files.

  File rolling can be turned off with the gg.handler.hdfs.rollOnMetadataChange parameter.

  To support metadata change events, the extract that captures changes in the source database must support **both** DDL changes and metadata in trail.

Oracle GoldenGate HDFS Handler Best Practices

The HDFS Handler calls the HDFS flush method on the HDFS write stream to flush data to the HDFS data nodes at the end of each transaction to maintain write durability.

This is an expensive call and performance can be adversely affected in the case of transactions of one or few operations that result in numerous HDFS flush calls.

Performance of the HDFS Handler can be greatly improved by batching multiple small transactions into a single larger transaction.

The GROUPTRANSOPS property in the Replicat parameter file controls transaction grouping.

Oracle GoldenGate HDFS Handler Best Practices

Set the following to improve the performance of the HDFS Handler:

- Transactional mode (gg.handler.hdfs.mode=tx) in the Java Adapter properties file
- GROUPTRANSOPS in the Replicat parameter file.

Oracle GoldenGate HDFS Handler Performance Considerations

The HDFS client libraries spawn threads for every HDFS file stream opened by the HDFS Handler.

The result is that the number of threads executing in the JVM grows proportionally to the number of HDFS file streams that are open.

Performance of the HDFS Handler can degrade if it writes to many files due to many source replication tables or if there is extensive use of partitioning.

Oracle GoldenGate HDFS Handler Performance Considerations

- Oracle recommends that you enable the roll on time or roll on inactivity features to close HDFS file streams if many tables are simultaneously replicated.
- Closing an HDFS file stream causes the HDFS client threads to terminate and the associated resources can be reclaimed by the JVM.

Demos: Overview

2-1: Replicating RDBMS Generated Customer Data to HDFS Using the HDFS Handler

2-2: Adding a Column on the Source Database, Triggering a Metadata Change Event

Configuring and Using the Oracle GoldenGate Hadoop HBase Handler

- Describe the Oracle GoldenGate for Big Data HBase Handler features
- Identify the HBase-specific features and parameters
- Configure the HBase Handler by using the relevant parameters in the property file
- Consider grouping transactions together to achieve maximum throughput

What is HBase ?

- HBase is a Big Data application, which **emulates** functionality of a relational database management system (RDBMS).
- Hadoop is specifically designed to store large amounts of unstructured data.
- Data stored in databases and replicated through Oracle GoldenGate tends to be highly structured.
- HBase maintains the important **structure** of data, while utilizing the **horizontal scaling** that is offered by the Hadoop Distributed File System (**HDFS**).
- HBase requires **zookeeper.**
- HBase supports **auto-sharding** using **Region** servers, which means tables are dynamically distributed by the system when they get too large.

Oracle GoldenGate HBase Handler Functionality

The HBase Handler takes changes from the trail, creates the corresponding HBase tables, and then loads the changes into the tables. No DDL parameter required.

The case-sensitive table names created in an HBase map to the corresponding table name of the operation from the source trail file.

Two-part table name, (Schema.MyTable), map the schema name to the HBase table namespace.

Three-part table name, (**Catalog.Schema.MyTable**) , map to the HBase table namespace as **Catalog_Schema.MyTable**.

HBase Row Key

HBase has the concept of a row key, similar to an RDBMS primary key.

The HBase row key is the unique identifier for a table row.

HBase supports only a single row key per row and it cannot be empty or NULL.

How does the Oracle GoldenGate HBase Handler handle the key ?

- The Oracle GoldenGate for Big Data HBase Handler maps the primary key value to the HBase row key value.
- If the source table has multiple primary keys, the primary key values are concatenated, separated by a pipe delimiter (|).
- The source table must have at least one primary key column.
- Replication of a table without a primary key causes the HBase Handler to **abend**.

HBase Column Family

A column family is a **grouping** mechanism for column data and only a single column family is supported.

Every HBase column must belong to a single **column family**.

The HBase Handler provides a single column family per table that defaults to **cf**.

After a table is created with a column family name, reconfiguration of the column family name without prior modification or drop of the table provokes the Replicat to abend.

HBase Column Family Example

```
hbase(main):001:0> scan 'OGGSRC:PRODUCT_PROD‘

ROW                COLUMN+CELL
 1                 column=cf:PRODUCT_CODE, timestamp=1503414266190, value=RJM75LEM3UY
 1                 column=cf:PRODUCT_ID, timestamp=1503414266190, value=1
 1                 column=cf:PRODUCT_NAME, timestamp=1503414266190, value=eu, eleifend nec, malesuada
 1                 column=cf:SUPPLIER_ID, timestamp=1503414266190, value=1
```

Oracle GoldenGate HBase Handler Classpath Configuration

- The gg.classpath configuration variable for the HBase Handler must include:

    1. The location of the **hbase-site.xml** file
    2. The location of the HBase client jars

- The default location of the **hbase-site.xml** file is **$HBASE_HOME/conf**.
- The default location of the HBase client jars is **$HBASE_HOME/lib/*.**

Relevant Oracle GoldenGate HBase Handler Configuration Parameters: 1

Parameter	Explanation
gg.handlerlist=hbase gg.handler.hbase.type=hbase	Selects the HBase Handler for use with Replicat
gg.handler.hbase.hBaseColumnFamilyName	Defaults to **cf.** Column family is a grouping mechanism for columns in Hbase.
gg.handler.hbase.includeTokens	Indicates True to have token values included in the output to Hbase
gg.handler.hbase.keyValueDelimiter	Provides a delimiter between the key values in a map; defaults to “=”
gg.handler.hbase.keyValuePairDelimiter	Provides a delimiter between the key value pairs in a map; defaults to “,”
gg.handler.hbase.pkUpdateHandling	Provides configuration for how the Hbase Handler should handle update operations that change a primary key

Relevant Oracle GoldenGate HBase Handler Configuration Parameters: 2

Parameter	Explanation
gg.handler.hbase.rowkeyDelimiter	Configures the delimiter between the primary key values from the source table when generating the HBase row key; defaults to |
gg.handler.hbase.encoding	Determines the encoding of values written to the HBase. HBase values are written as bytes.
gg.handler.hbase.nullValueRepresentation	Helps you to configure what will be sent to HBase in the case of a NULL column value. The default is NULL. The configuration value supports CDATA[] wrapping.

Oracle GoldenGate HBase Handler Properties File: Example

```
…Omitted lines…
gg.handlerlist=hbase
gg.handler.hbase.type=hbase
gg.handler.hbase.mode=tx
gg.handler.hbase.hBaseColumnFamilyName=cf
gg.handler.hbase.includeTokens=true
gg.handler.hbase.keyValueDelimiter=CDATA[=]
gg.handler.hbase.keyValuePairDelimiter=CDATA[,]
gg.handler.hbase.encoding=UTF-8
gg.handler.hbase.pkUpdateHandling=abend
gg.handler.hbase.nullValueRepresentation=CDATA[NULL]
gg.handler.hbase.authType=none
…Many more omitted lines…
```

HBase Write Durability Can Affect Performance !

- At each transaction **commit**, the HBase Handler performs a **flush call** to flush any buffered data to the HBase region server, to maintain write **durability**.
- Flushing to the HBase region server is an **expensive** call.

Oracle GoldenGate HBase Handler Best Practices

- Performance can be improved using the Replicat GROUPTRANSOPS parameter to group multiple smaller transactions in the source trail file into a larger single transaction that is applied to HBase.
- Replicat base-batching can be used by adding the configuration syntax in the Replicat configuration file.
- Operations from multiple transactions are grouped together into a larger transaction, and it is only at the **end of the grouped transaction** that transaction **commit** is executed.

Oracle GoldenGate HBase Metadata Change Events

Replicat

For metadata change events to work properly, the minimum Oracle GoldenGate for Big Data release is 12.2.

Java Adapter

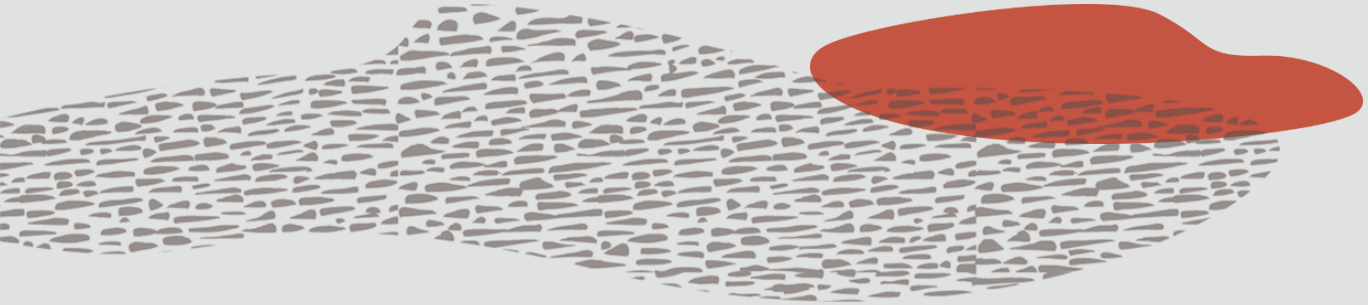
Oracle GoldenGate 12.3 and higher includes metadata in trail, and can handle metadata change events at run time.

Hbase Handler

The HBase Handler can handle metadata change events at run time as well.

HBase

The addition of a column results in the new column and its data being streamed to HBase after the metadata change event.



Demos: Overview

3-1: Replicating RDBMS Generated Product Data to an HBase Database Using the HBase Handler

3-2: Performing and Verifying Updates

Configuring and Using the Oracle GoldenGate Apache Kafka Handler

- Describe the Oracle GoldenGate for Big Data Kafka Handler features
- Identify the Kafka-specific features and parameters
- Configure the Kafka Handler by using the relevant parameters in the property file
- Configure the transactional and operational modes that are available in the Kafka Handler
- Configure the Kafka Handler in Blocking or Non-Blocking mode

What is Apache Kafka ?

- Apache Kafka is a distributed, partitioned, and replicated **guaranteed** messaging service.
- Kafka can be run as a single instance or as a **cluster** on multiple servers.
- Each Kafka server instance is called a **broker**.
- A Kafka **topic** is a category or feed name to which messages are published by the producers and retrieved by consumers.

Oracle GoldenGate Kafka Handler Overview

- The Kafka Handler implements a Kafka **producer**
- The producer writes **serialized** change data capture from multiple source tables to either of the following:

  A **single** configured topic

  Different Kafka topics when the topic name corresponds to the fully qualified source table name

Kafka Core APIs

1. The **Producer API** helps an application to publish a stream of records to one or more Kafka topics.

2. The **Consumer API** helps an application to subscribe to one or more topics and process the stream of records produced to them.

3. The **Streams API** helps an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input stream to an output stream.

4. The **Connector API** helps for building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems.

Kafka Connect API

5. The Kafka Connect API is a functional layer on top of standard Kafka Producer and Consumer APIs.

   Confluent is primary adopter of Kafka Connect.

   Starting with GoldenGate for Big Data 12.3.1.1, the GoldenGate Kafka Connect Handler can create schemas and messages in memory and pass them to the Kafka Connect converter to convert bytes to send to Kafka. It supports JSON and AVRO converters.

Kafka REST proxy API

6. The Kafka REST proxy API provides a RESTful interface to a Kafka cluster.

   The Kafka REST Proxy Handler helps Kafka messages to be streamed using the HTTPS protocol and can be used to stream Kafka messages from an Oracle GoldenGate On Premises installation to cloud or alternately from cloud to cloud.

   Kafka REST Proxy is part of the Confluent Open Source and Confluent Enterprise distributions and **not** available in the Apache Kafka distribution.

Oracle GoldenGate Kafka Handler Modes

- Transaction Mode serializes data for **every operation** in a **transaction** from the source Oracle GoldenGate trail files and **concatenates** them into a single value of the Kafka **ProducerRecord** object.
- Operation Mode serializes data for each **operation** is placed into an **individual ProducerRecord** object as the value. The ProducerRecord is immediately sent by using the Kafka Producer API.
- Blocking Mode delivers messages to Kafka on a **synchronous** basis.
- Non-Blocking Mode delivers messages to Kafka on an **asynchronous** basis.

Oracle GoldenGate Kafka Handler Classpath Configuration

- The gg.classpath configuration variable for the Kafka Handler must include:
    1. The location of the Kafka Producer **properties** file
    2. The location of the Kafka **client JARs**
- The recommended storage location for the Kafka Producer properties file is the Oracle GoldenGate dirprm directory.
- The default location of the Kafka client JARs is $KAFKA_HOME/libs/*.

Relevant Oracle GoldenGate Kafka Handler Configuration Parameters: 1

Parameter	Explanation
gg.handlerlist=kafka gg.handler.kafka.type=kafka	Selects the Kafka Handler for use with Replicat
gg.handler.kafka. KafkaProducerConfigFile	Specifies the file name in the classpath that holds the Apache Kafka properties to configure the Apache Kafka producer
gg.handler.kafka.topicMappingTemplate	Specifies the name of the Kafka topic where payload records will be sent. Templates allow you to configure static values and keywords.
gg.handler.kafka.Format	Specifies the formatter to use to format payload; can be one of xml, **delimitedtext**, json, json_row, avro_row, or avro_op
gg.handler.kafka.SchemaTopicName	Specifies the topic name where schema data will be delivered. Schemas will be propagated only for Avro formatters.

Relevant Oracle GoldenGate Kafka Handler Configuration Parameters: 2

Parameter	Explanation
gg.handler.kafka.BlockingSend	Is set to True or False. If this property is set to true, delivery to Kafka is made to work in a completely synchronous model.
gg.handler.kafka.mode	Is set to tx or op. With Kafka Handler operation mode, each change capture data record payload will be represented as a Kafka Producer Record and will be flushed one at a time.
gg.handler.kafka.topicMappingTemplate	A template string value to resolve the Kafka topic name at runtime.

Oracle GoldenGate Kafka Handler Properties File: Example

```
…Omitted lines…
gg.handlerlist=kafka
gg.handler.kafka.Type=kafka
gg.handler.kafka.KafkaProducerConfigFile=custom_kafka_producer.properties
gg.handler.kafka.topicMappingTemplate=oggtopic
gg.handler.kafka.Format=avro_op
gg.handler.kafka.SchemaTopicName=oggSchemaTopic
gg.handler.kafka.Mode=tx
gg.handler.kafka.BlockingSend=true
…More omitted lines…
```

Oracle GoldenGate Kafka Handler Producer Configuration File

The Kafka Handler must access a Kafka producer configuration file to publish messages to Kafka (KafkaProducerConfigFile parameter).

- The Kafka Handler locates the Kafka producer configuration file using the Java classpath.
- The Java classpath must include the directory containing the Kafka producer configuration file.
- The Kafka Handler uses the properties specified in the Kafka producer configuration file to:

  Resolve the host and port of the Kafka **brokers**
  Control the behavior of the interaction between the Kafka producer client and the Kafka brokers

Oracle GoldenGate Kafka Handler Schema Propagation

- The Kafka Handler can publish schemas to a schema topic .
- **Avro** Row and Operation formatters are the **only** formatters enabled for schema publishing.

Oracle GoldenGate Kafka Handler Schema Propagation continued

When the Kafka Handler schemaTopicName property is set:

- The Avro schema for a specific table will be **published** the first time an operation for that table is encountered
- If the Kafka Handler receives a metadata change event, the schema is **flushed**
- If the Avro wrapping functionality is enabled, the generic wrapper Avro schema is published the first time any operation is encountered

Oracle GoldenGate Kafka Handler Best Practices

- Oracle recommends avoiding setting the **linger.ms** parameter in the Kafka producer configuration file when gg.handler.kafka.BlockingSend is set to true.
- When the value is true, each send blocks for at least linger.ms minutes leading to major **performance issues**. This is set in milliseconds.
- Oracle recommends **preconfigured topics** by manually creating them.

Oracle GoldenGate Kafka Handler Best Practices

For best performance, you should set:

- gg.handler.kafka.mode=op
- gg.handler.kafka.BlockingSend=false
- Replicat GROUPTRANSOPS to 10,000.

Oracle GoldenGate Kafka Handler Advanced Configuration for Security

You can secure the Kafka Handler using:

- Transport Layer Security (TLS)
- Secured Socket Layer (SSL)
- Kerberos

Oracle GoldenGate Kafka Handler Advanced Configuration for Metadata Change Events

- If a **schema topic** is configured and the formatter used supports schema propagation (**Avro** row and **Avro** Operation formatters), the next time an operation is encountered for a table for which the schema has changed, the updated schema is **published** to the schema topic.

- To support metadata change events, the Oracle GoldenGate process that is capturing changes in the source database must support the Oracle GoldenGate metadata in trail feature (Oracle GoldenGate core release 12.2 or higher).

Kafka Handler Advanced Configuration for Data Compression

The Kafka producer configuration file supports the following compression methods:

- gzip
- Snappy (Oracle recommends this on Linux for performance reasons.)

Demos: Overview

5-1: Replicating Customer Data from an Oracle RDBMS to Apache Kafka

5-2: Performing and Verifying Updates

Configuring and Using the Cassandra Handler

- Describe the Oracle GoldenGate for Big Data Cassandra Handler features
- Identify the Cassandra-specific features and parameters
- Configure the Cassandra Handler by using the relevant parameters in the property file
- Configure the transactional and operational modes that are available in the Cassandra Handler
- Configure the Cassandra Handler in Asynchronous or Synchronous mode

What is Cassandra ?

Cassandra is a NoSQL Database Management System that is designed to store large amounts of data.

It can provide high availability and eliminate a single point of failure by replicating data to multiple nodes within a Cassandra cluster.

A Cassandra cluster configuration provides horizontal scaling and replication of data across multiple machines.

Apache Cassandra is designed to run on low cost commodity hardware.

Cassandra does not have the following concepts !

- No concept of Referential Integrity
- No concept of Foreign Key
- No concept of commit
- No concept of rollback
- No row locking mechanism

ACID Properties Versus Eventual Consistency

- Cassandra **relaxes** the axioms of traditional Relational Database Management Systems regarding atomicity, consistency, isolation, and durability (ACID).
- Cassandra instead provides **eventual consistency**:

  Accessing the state of data for a specific row will eventually return the latest state of the data for that row as defined by the most recent change.

  There may be a latency period between the creation and modification of the state of a row and what is returned when the state of that row is queried.

  The promise of eventual consistency is that the latency period is predictable based on the Cassandra configuration and the level of workload the Cassandra cluster is currently under.

- The Cassandra Handler provides some control over consistency with the configuration of the gg.handler.cassandra.consistencyLevel property.

Oracle GoldenGate Cassandra Handler: Supported Data Types

- The Cassandra Handler supports most scalar data types that are available in Cassandra.
- The following collections, however, are not supported:

  Counter
  List
  Map
  Set
  Tuple
  Custom_Type

Oracle GoldenGate Cassandra Handler: Supported Database Operations

- The Cassandra Handler supports the following database operations:

  Insert
  Update (captured as Insert)
  Delete

- The Cassandra Handler does not support the following database operations:

  Truncate
  DDL (Create, Alter, Drop)

Schema, Table, and Column Mapping

- In relational RDBMS, structured data is organized in tables, and databases are collections of tables.

  RDBMS tables are also tables in Cassandra.

  In Cassandra, **keyspaces** play the role of RDBMS databases and/or schemas.

- **Keyspaces** in Cassandra define a replication factor, the replication strategy, and the topology, because Cassandra is inherently cluster and redundancy oriented.
- The Cassandra Handler cannot create **keyspaces** from the metadata information that is stored in the trail files.

  Cassandra keyspaces must be created by using the **Cassandra client**.

  The Cassandra Handler can, however, create and modify tables.

Cassandra Primary Keys

- Cassandra has two types of primary keys:

  1. **Partitioning keys** define how data for a table is separated into partitions and are not needed by single-node systems.

  2. **Clustering keys** define the order of items within a partition.

- The **default** mapping for automated table creation by the Cassandra Handler is that the **first primary key** is the partition key and any additional primary keys are mapped as clustering keys.
- **Automated** table creation by the Cassandra Handler may result in data definitions that do not scale well !!!

Oracle GoldenGate Cassandra Handler Operation Processing APIs

- The Cassandra Handler pushes operations to Cassandra by using either of the following:
  - Asynchronous API
  - Synchronous API
- In asynchronous mode, operations are flushed at transaction commit to ensure write durability.

Oracle GoldenGate Cassandra Handler Operation Processing is Different than the Oracle Database !

The Cassandra Handler does not manage the interaction with Cassandra in a transactional manner.

Insert, update, and delete operations are processed differently in Cassandra than in a traditional RDBMS.

- INSERT in Cassandra does not check the prior existence of the row by default: the row is created if none existed before, and updated otherwise.
- UPDATE does not check the prior existence of the row by default. The row is created if none existed before, and updated otherwise.
- DELETE deletes columns and rows. If column names are provided directly after the DELETE keyword, only those columns are deleted from the row indicated by the WHERE clause. Otherwise, whole rows are removed.

Compressed Versus Full Image Updates

- The Oracle GoldenGate core helps you to control the data that is propagated to the source trail file in the event of an update. The data can be either of the following:

  1. **Compressed** where only primary keys and the data for columns that changed are stored in the trail file.

  2. **Full Image** where the trail file stores all columns, including primary keys, columns for which the value changed, and columns for which the value did not change.

- The amount of information available in an update is important to the Cassandra Handler.

  If the source trail file contains full images of the change data, the Cassandra Handler can use prepared statements to perform row updates in Cassandra.

  In Cassandra, primary keys are immutable so an update that changes a primary key must be treated as a delete and an insert.

Oracle GoldenGate Cassandra Handler Classpath Configuration

- The Datastax Java Driver for Cassandra does not ship with Oracle GoldenGate for Big Data.
- It must be downloaded and installed.
- The installation directory must be included in the gg.classpath configuration variable of the Java Adapter properties file.

The recommended DataStax Java Driver is 3.1.0.

The **gg.classpath** configuration variable should be set as follows:
gg.classpath=*{download_dir}*/cassandra-java-driver-3.1.0/*:*{download_dir}*/cassandrajava-driver-3.1.0/lib/*

The **gg.classpath** variable supports the wildcard (*) character to select all JARs in a configured directory.

Relevant Oracle GoldenGate Cassandra Handler Configuration Parameters: 1

Parameter	Explanation
gg.handlerlist=cassandra gg.handler.cassandra.type=cassandra	Selects the Cassandra Handler for use with Replicat
gg.handler.cassandra.mode	Is either **op** or tx. In **op** mode, operations are processed as received. In tx mode, operations are cached and processed at transaction commit.
gg.handler.cassandra.contactPoints	Specifies a comma-separated list of Cassandra host machines for the driver to establish an initial connection to the Cassandra cluster
gg.handler.cassandra.compressedUpdates	Sets the Cassandra Handler whether to or not to expect full image updates from the source trail file
gg.handler.cassandra.cassandraMode	Is either **async** or sync

Relevant Oracle GoldenGate Cassandra Handler Configuration Parameters: 2

Parameter	Explanation
gg.handler.cassandra.ddlHandling	Specifies the DDL functionality to provide. Options include CREATE, ADD, and DROP. These options can be set in any combination delimited by commas.
gg.handler.cassandra.consistencyLevel	Sets the consistency level for operations with Cassandra. It configures the criteria that must be met for storage on the Cassandra cluster when an operation is executed.
gg.handler.cassandra.contactPoints	Specifies a comma-separated list of Cassandra host machines for the driver to establish an initial connection to the Cassandra cluster
gg.handler.cassandra.username	Specifies the username for the connection to Cassandra; required if Cassandra is configured to require credentials
gg.handler.cassandra.password	Specifies the password for the connection to Cassandra; required if Cassandra is configured to require credentials

Oracle GoldenGate Cassandra Handler Properties File: Example

```
…Omitted lines…
gg.handlerlist=cassandra
gg.handler.cassandra.type=cassandra
gg.handler.cassandra.mode=op
gg.handler.cassandra.contactPoints=localhost
gg.handler.cassandra.ddlHandling=CREATE,ADD,DROP
gg.handler.cassandra.compressedUpdates=true
gg.handler.cassandra.cassandraMode=async
gg.handler.cassandra.consistencyLevel=ONE
…More omitted lines…
```

Oracle GoldenGate Cassandra Handler Best Practices

- Configuring the Cassandra Handler for **async** mode will provide better performance than sync mode. The Replicat property GROUPTRANSOPS should be set to the default of **1000**.
- Setting a consistency level directly affects performance:

  The higher the consistency level, the more the work that must occur on the Cassandra cluster before the transmission of a given operation can be considered complete.

  Select the minimum consistency level that still satisfies the requirements of your use case.

- The Cassandra Handler can work in either operation (**op**) or transaction (tx) mode. For best performance, operation mode is recommended.

Demos: Overview

6-1: Replicating Product Data from Oracle RDBMS to Apache Cassandra

6-2: Performing and Verifying Updates

What we covered today

- GoldenGate for Big Data Architecture
- How to configure GoldenGate to replicate changes from Oracle Database redo logs to big data targets including:

  HDFS
  HBase table namespace
  Kafka topic
  Cassandra keyspace

Where can I learn more from Oracle University ?

- Oracle GoldenGate 12c: Fundamentals for Oracle (4 days)
- Oracle GoldenGate 12c: Advanced Configuration for Oracle (4 days)
- Oracle GoldenGate 12c: Troubleshooting and Tuning (4 days)
- Oracle GoldenGate 12c: Management Pack Overview (2 days)
- Oracle GoldenGate 12c: Veridata Essentials (1 day)
- Oracle GoldenGate 12c: Integrate Big Data (3 days)

CPE Credit Opportunity

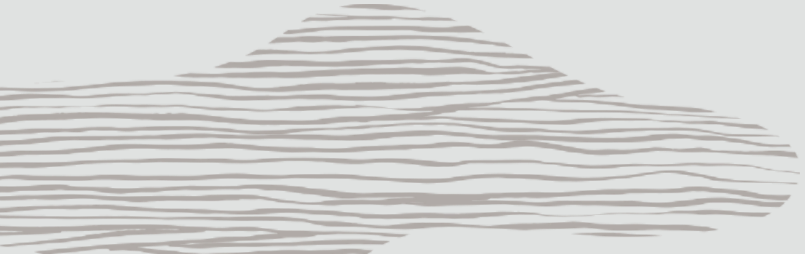
Attendees of Deep Dive sessions can earn CPE (Continuing Professional Education) credits (US-based attendees only).

1. Send an email to oucloud_ww@oracle.com to request a NASBA-approved CPE Certificate of Completion
2. Include event attended and session name/code for the Deep Dive session
3. Once received, provide the certificate to your state board of accountancy for credit.

Email: oucloud_ww@oracle.com

Session Survey

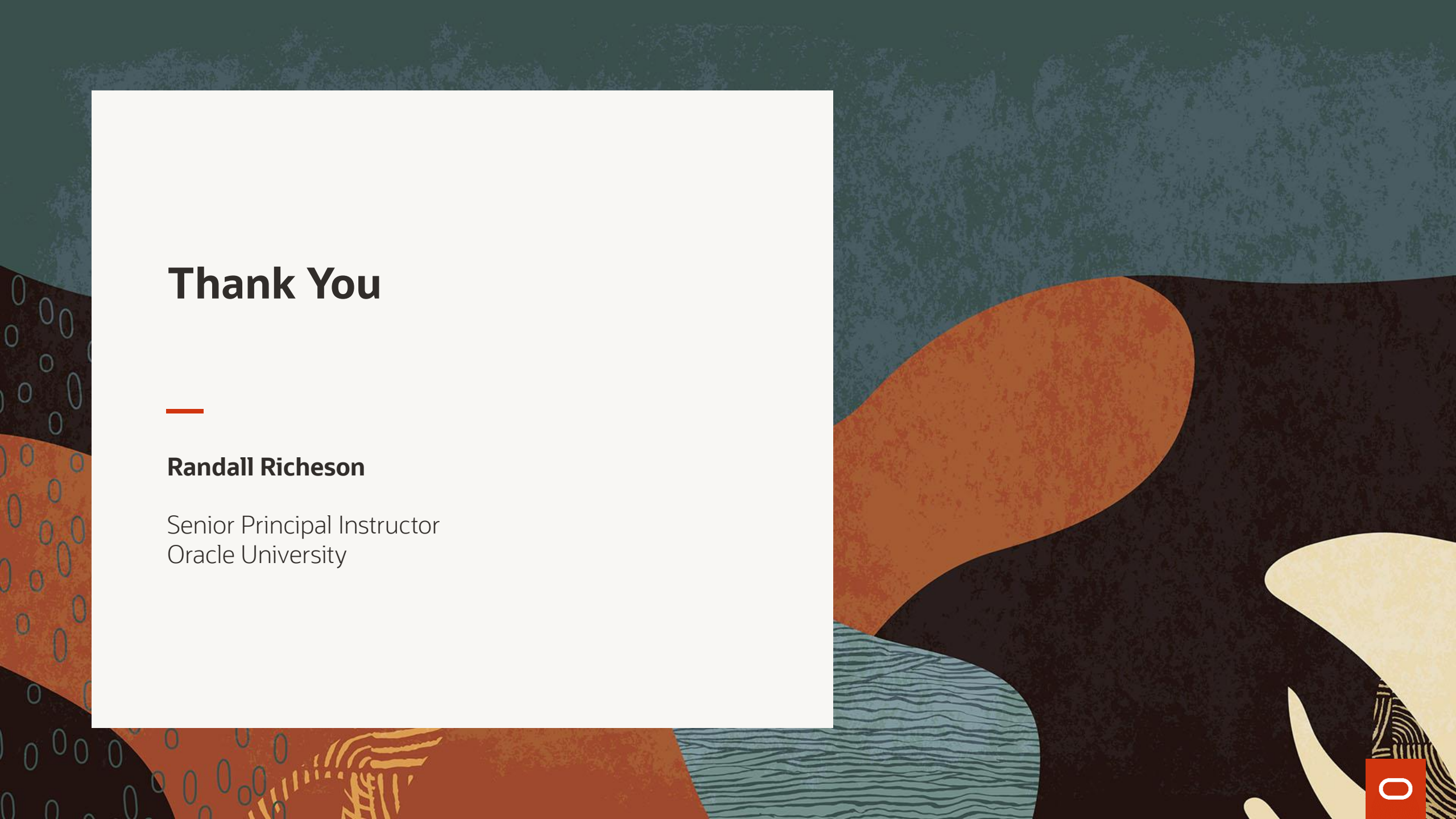
Help us make the content even better. Please complete the session survey in the Mobile App.



Thank You

Randall Richeson

Senior Principal Instructor
Oracle University



Safe Harbor

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

ORACLE