



Best Practices for Migrating from Oracle to Amazon Aurora

David Bayard

Principal Database Specialist Solutions Architect

Eli Doe

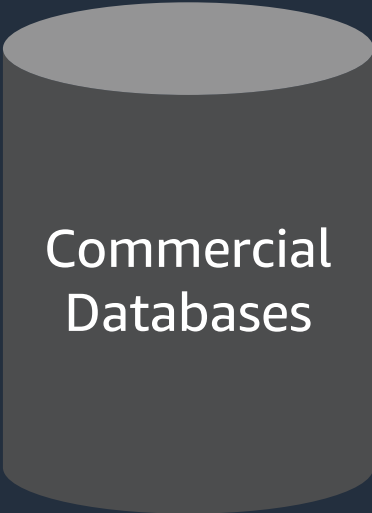
Database Migration Specialist Solutions Architect

Table of contents

- Oracle to Aurora Migration Process Overview
- Demonstration of Schema Conversion Tool (SCT)
- SCT Best Practices
- Demonstration of Database Migration Service (DMS)
- DMS Best Practices
- How AWS Can Help

Oracle to Amazon Aurora Migration

Why customers are migrating?



Expensive



**Restrictive
licensing
terms**



**Reduce
Cost**



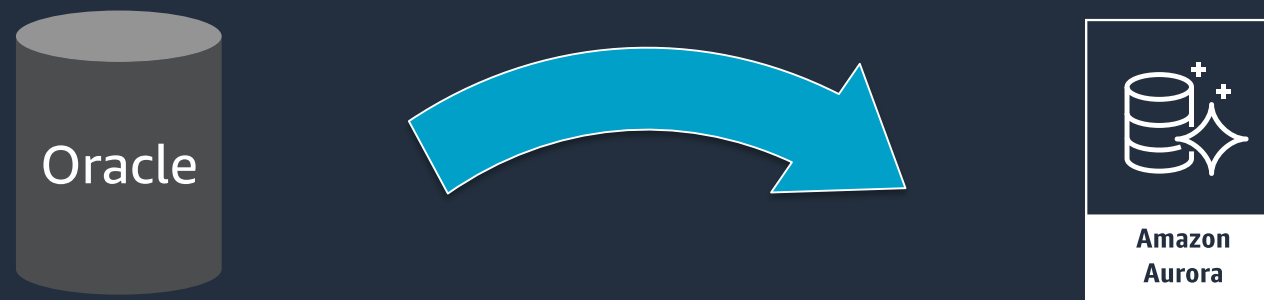
**Increase
Agility**



**Innovate
Faster**

Migrating Oracle to Aurora PostgreSQL

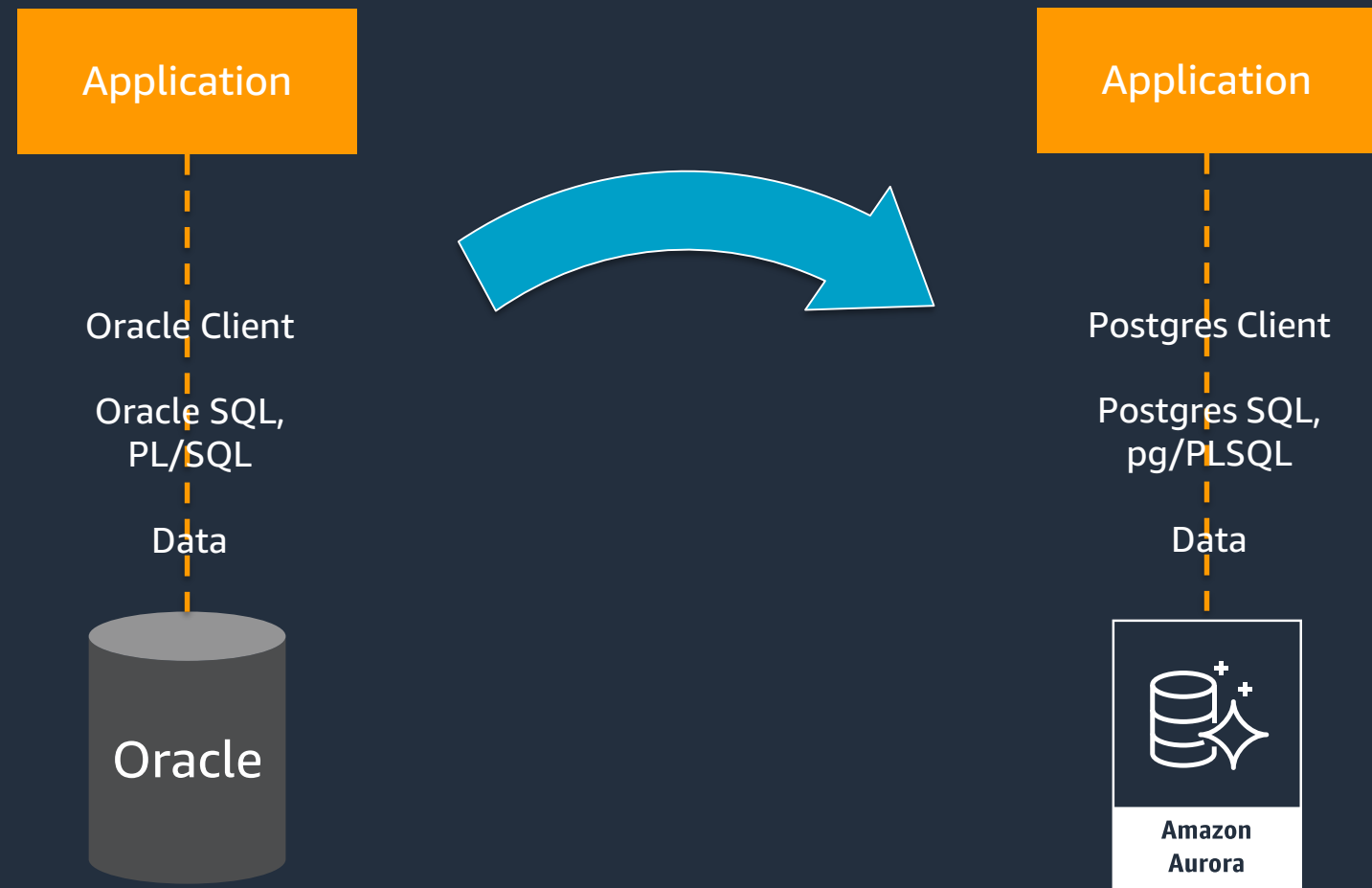
- What does Oracle to Aurora PostgreSQL look like?



- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service

Migrating Oracle to Aurora PostgreSQL

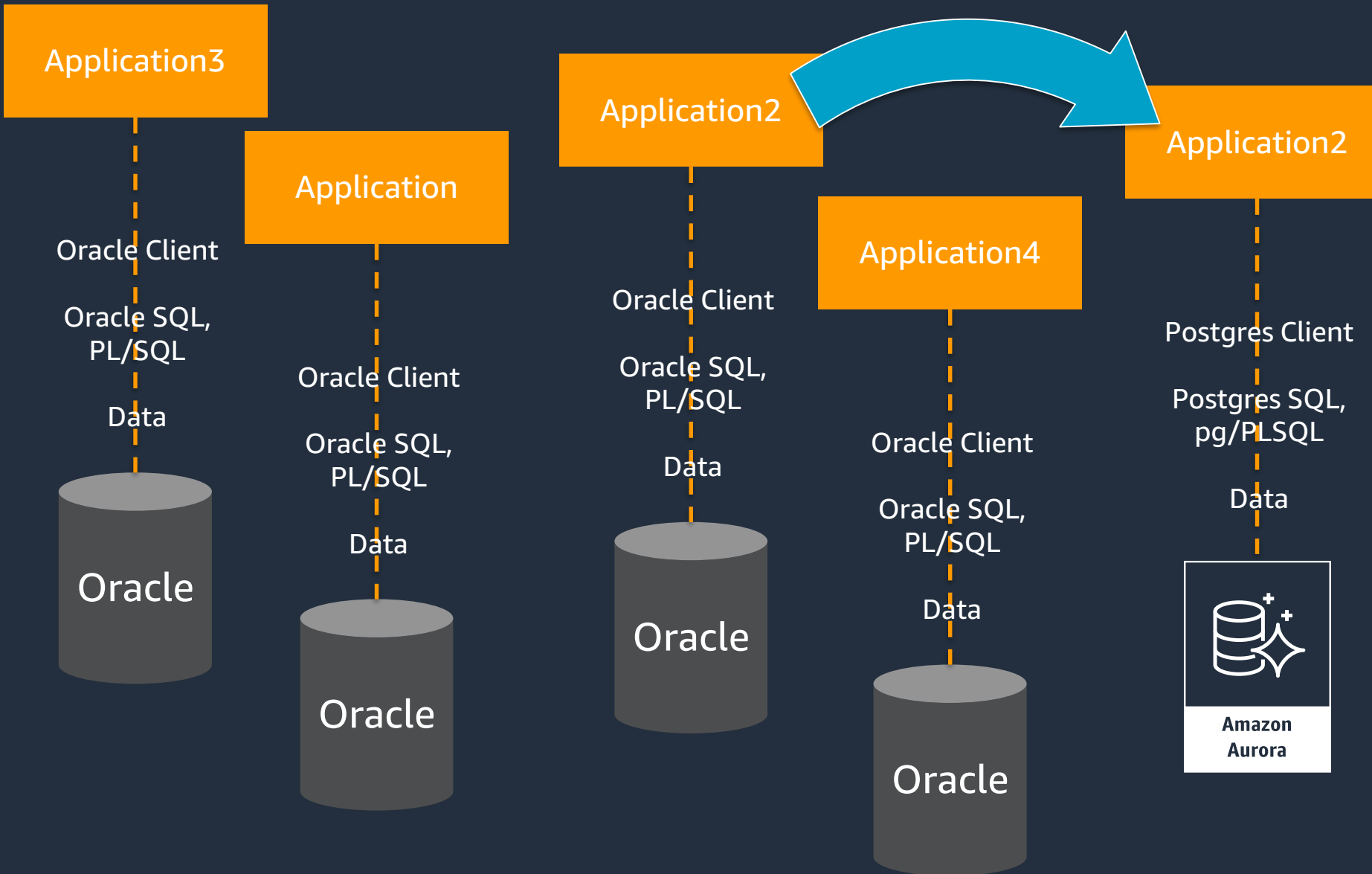
- What does Oracle to Aurora PostgreSQL really look like?



- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL
- Migrate Data from Oracle to Postgres
- Replace Oracle Client with Postgres Client
- Test Application
- Cut over Production

Migrating Oracle to Aurora PostgreSQL

- What does Oracle to Aurora PostgreSQL really look like?



- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL
- Migrate Data from Oracle to Postgres
- Replace Oracle Client with Postgres Client
- Test Application
- Cut over Production

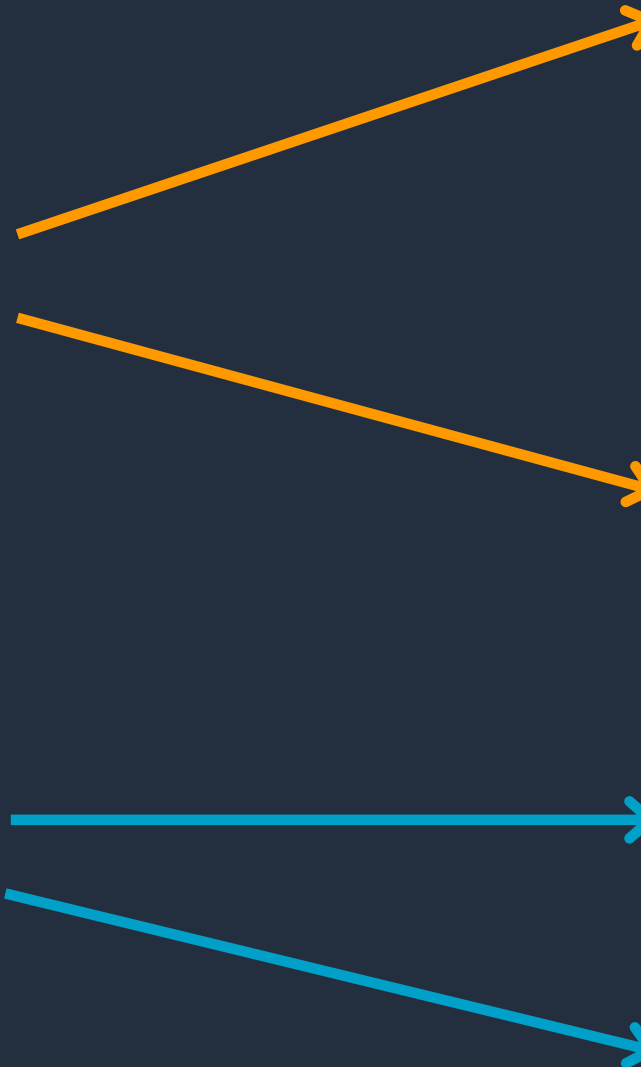
AWS Tools to help Oracle to Aurora Migration



Schema
Conversion Tool



AWS Database
Migration
Service



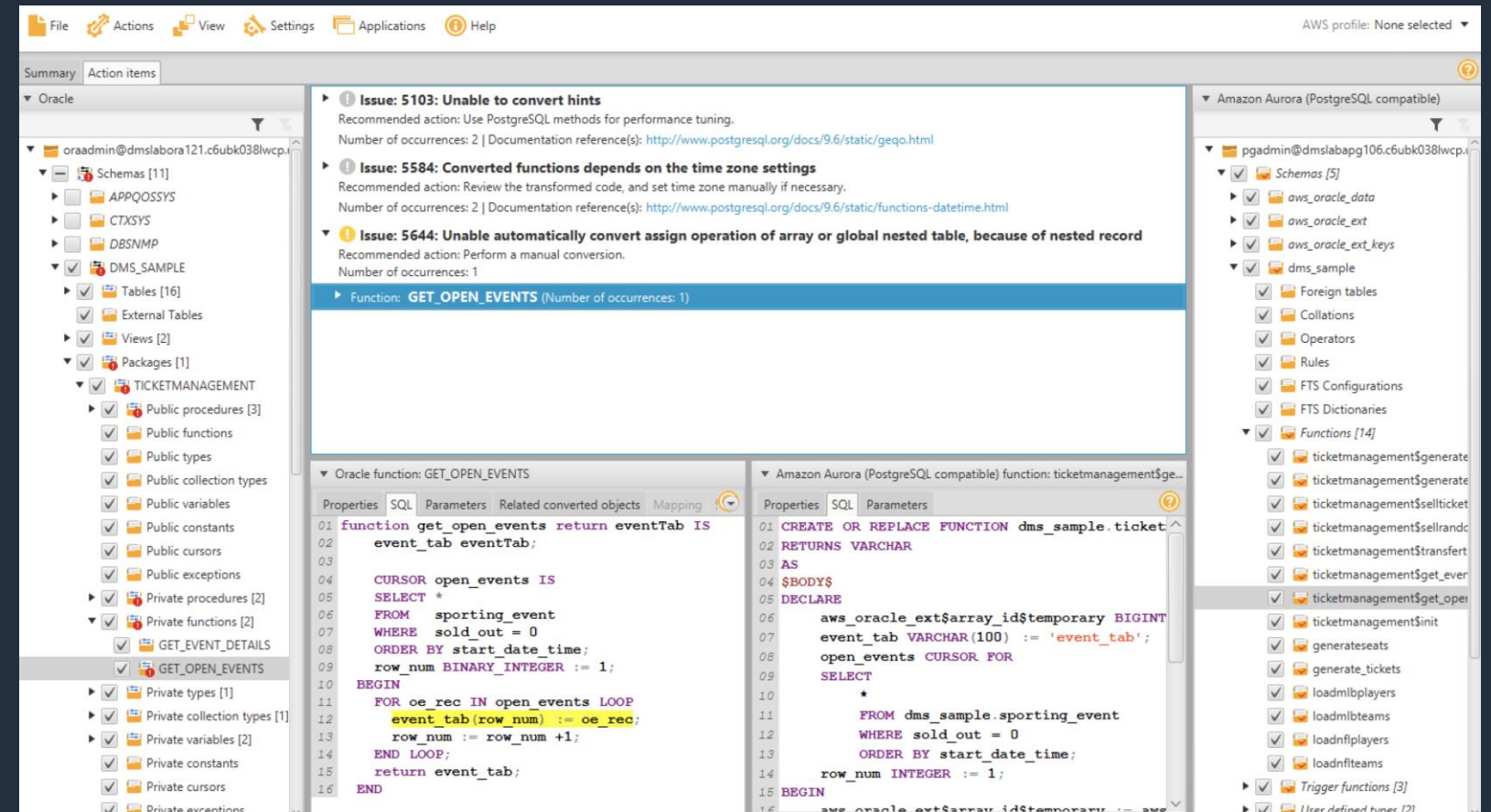
- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL
- Migrate Data from Oracle to Postgres
- Replace Oracle Client with Postgres Client
- Test Application
- Cut over Production

AWS Schema Conversion Tool



Schema
Conversion
Tool

Makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database



Features

Database Migration Assessment report for choosing the right target engine
Automatic conversion for eligible database objects and code
Code browser to highlight places where manual edits are required

AWS Database Migration Service



AWS Database
Migration
Service

- Start your first migration in *10 minutes or less*
- Keep your *apps running* during the migration
- *Replicate* from within, to, or from AWS
- Move data to the same or *different database engine*

Sources*
Oracle
SQL Server
Azure SQL
PostgreSQL
MySQL
SAP ASE
MongoDB
Amazon S3
IBM DB2 (LUW)
Amazon DocumentDB

Targets*
Oracle
SQL Server
PostgreSQL
MySQL
SAP ASE
Amazon Redshift
Amazon S3
Amazon DynamoDB
Amazon Kinesis
Amazon ElasticSearch
Amazon DocumentDB
Amazon Neptune
Apache Kafka

** Consult DMS Documentation for latest DMS sources and targets*

Demonstration, Part 1

Demonstration – Part 1



Schema
Conversion Tool



- Assess Application+Database pairs for migration complexity and classification
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL

Demonstration, Part 1

Eli Doe

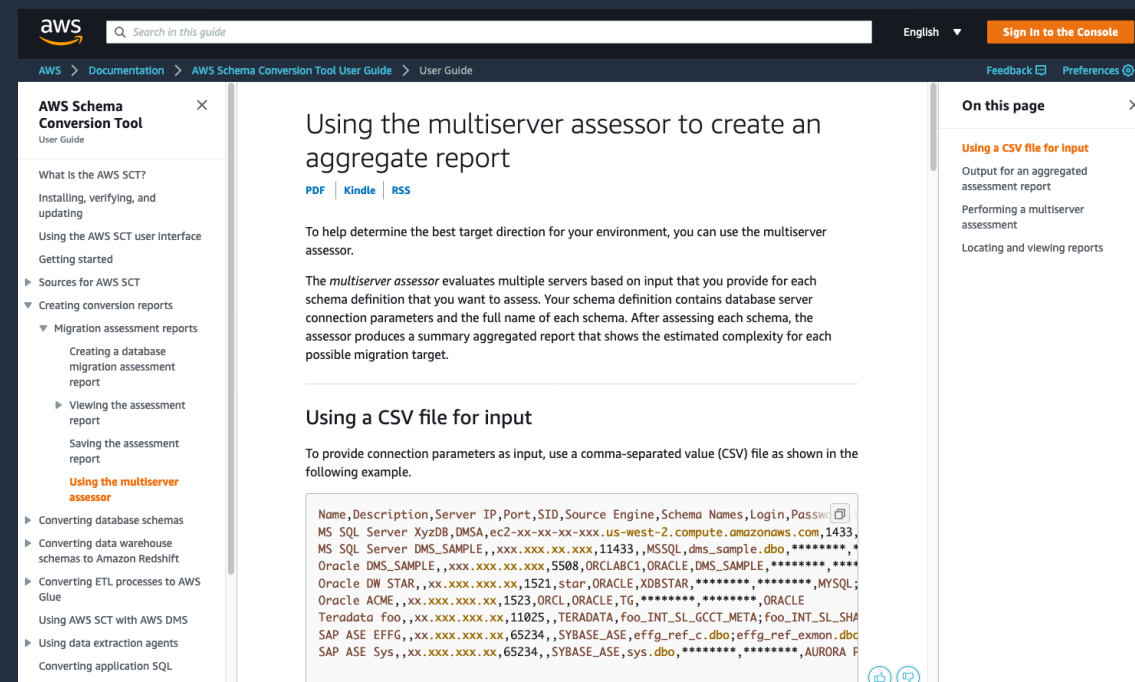
SCT Best Practices

Assessment Phase

SCT Best Practices - Assessment

Use the new SCT Multi-server Assessment feature

- This makes it easier to run assessments against multiple databases and schemas
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_AssessmentReport.Multiserver.html



SCT Best Practices - Assessment

Be sure to save the CSV files when you run an assessment

- The CSV data can be used to create custom reports and used with your classification algorithms
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CH_AP_AssessmentReport.Save.html

Useful blog about classifying database workloads:

- <https://aws.amazon.com/blogs/database/categorizing-and-prioritizing-a-large-scale-move-to-an-open-source-database/>

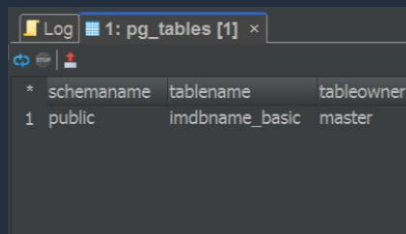
SCT Best Practices

Conversion Phase

SCT Best Practices - Conversion

Don't treat the target like the source. Understand your differences.

Some Basic Examples:



A screenshot of the pg_tables view in PostgreSQL. The table has three columns: schemaname, tablename, and tableowner. The first row shows 'public' as the schema, 'imdbname_basic' as the table, and 'master' as the owner.

	schemaname	tablename	tableowner
1	public	imdbname_basic	master

PostgreSQL is a lowercase data dictionary

```
test=# SELECT CURRENT_DATE;  
  
current_date  
-----  
2017-05-09  
(1 row)
```

No DUAL table needed

Examples

Set the schema search path:

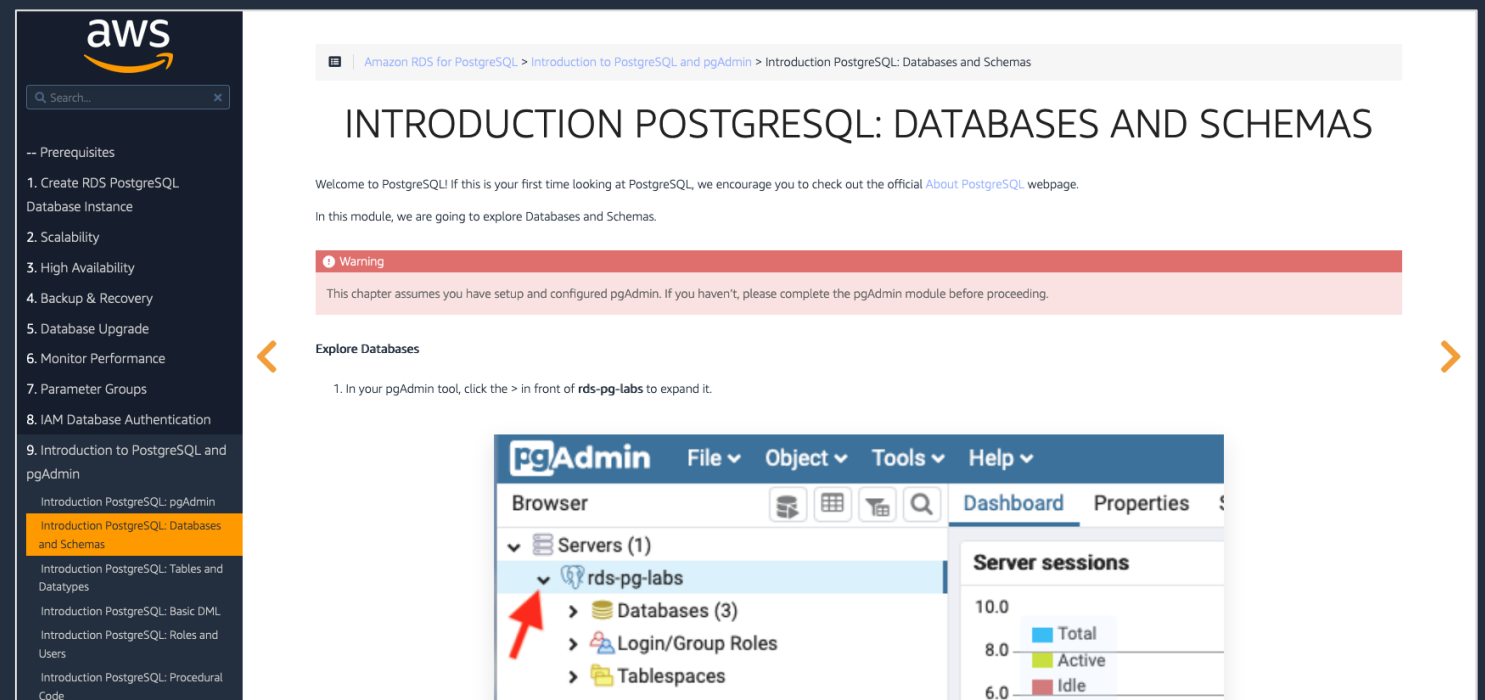
```
SET search_path TO my_schema;
```

search_path replaces PUBLIC SYNONYM

```
test=# SELECT COALESCE(fname, '') || ' ' ||  
COALESCE(mname, '') || ' ' || COALESCE(lname, '') FROM users;  
?column?  
-----  
George Washington  
John Adams  
Thomas Jefferson  
James Madison  
James Monroe  
Andrew Jackson  
Martin Van Buren  
John Tyler  
John Quincy Adams  
William Henry Harrison  
(10 rows)
```

Oracle concatenates nulls strings differently than PostgreSQL

Hint: Just getting started with PostgreSQL? Check out the "Introduction to PostgreSQL" chapter in the AWS PostgreSQL Immersion Day: <https://rdspg.workshop.aws/>

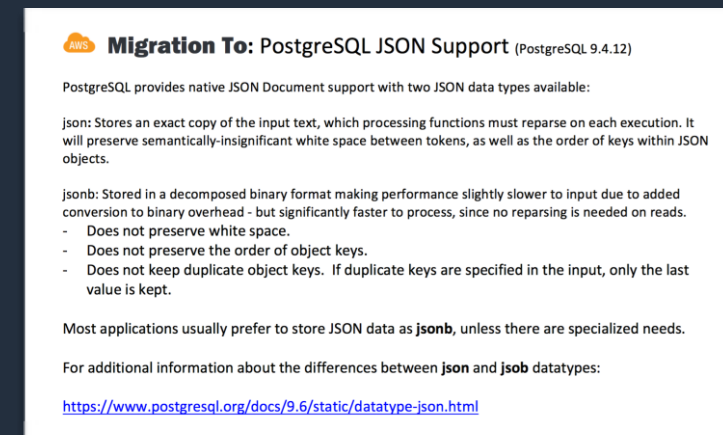
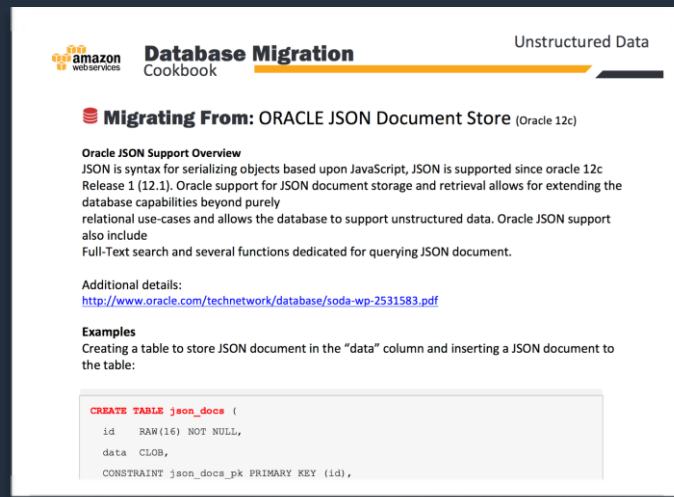


The image shows two side-by-side screenshots. The left screenshot is from the AWS RDS PostgreSQL documentation, titled "INTRODUCTION POSTGRESQL: DATABASES AND SCHEMAS". It includes a table of contents with items like "Prerequisites", "Create RDS PostgreSQL Database Instance", "Scalability", "High Availability", "Backup & Recovery", "Database Upgrade", "Monitor Performance", "Parameter Groups", "IAM Database Authentication", and "Introduction to PostgreSQL and pgAdmin". The right screenshot is a screenshot of the pgAdmin web interface. It shows the "Servers" tree on the left with "rds-pg-labs" selected. The main panel shows the "Server sessions" section with a bar chart showing "Total", "Active", and "Idle" sessions. The "Total" session count is 10.0, "Active" is 8.0, and "Idle" is 6.0.

SCT Best Practices - Conversion

Consult the Oracle to Aurora PostgreSQL Migration Playbook pdf (300+ pages)

- <https://aws.amazon.com/dms/resources/>



Check the AWS Database Blog <https://aws.amazon.com/blogs/database/> for additional topics.

- You can narrow down with the [Amazon Aurora](#) and [RDS For PostgreSQL](#) tags

SCT Best Practices - General

Use the latest version of SCT

- It is updated often and each version enhances conversion capabilities
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_ReleaseNotes.html

If needed, adjust the SCT memory settings higher

- SCT builds an in-memory model of the database objects. More memory equals better performance
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_BestPractices.html

If needed, leverage the SCT log file

- If you have any issues with SCT (such as seeming to hang), check the log file to see the source object that it is trying to convert
- <https://aws.amazon.com/blogs/database/configuring-the-aws-schema-conversion-tool/>

Demonstration, Part 2

Demonstration – Part 2



AWS Database
Migration
Service

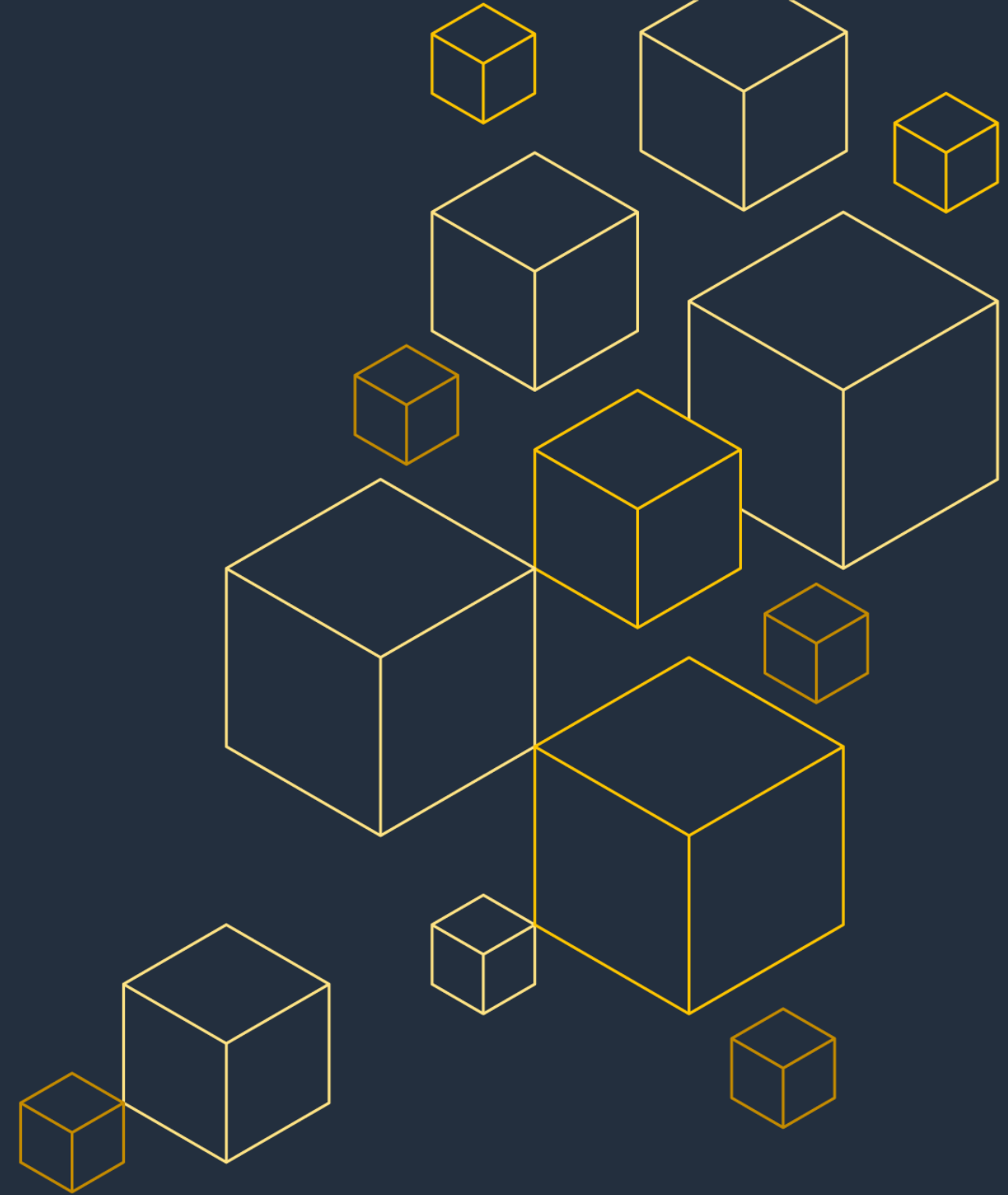


- Migrate Data from Oracle to Postgres
- Cut over Production

Demonstration, Part 2

Eli Doe

DMS Best Practices



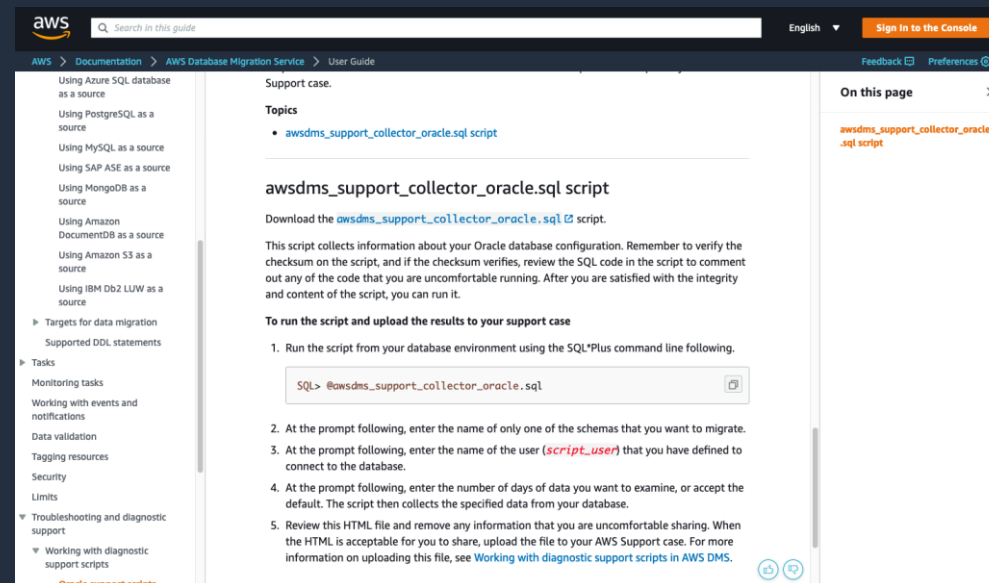
DMS Best Practices – Oracle as a Source

Start with the Oracle as a Source chapter in the DMS documentation. It is important. It is updated regularly.

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html

There is a very good support SQL*Plus script you can use as a pre-check:

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_SupportScripts.Oracle.html



DMS Best Practices – LogMiner or Binary Reader

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.CDC

Feature	LogMiner	Binary Reader
Easy to configure	Yes	No
Lower impact on source system I/O and CPU	No	Yes
Better CDC performance	No	Yes
Supports Oracle table clusters	Yes	No
Supports all types of Oracle Hybrid Columnar Compression (HCC)	Yes	Partially Binary Reader doesn't support QUERY LOW for tasks with CDC. All other HCC types are fully supported.
LOB column support in Oracle 12c	No	Yes
Supports UPDATE statements that affect only LOB columns	No	Yes
Supports Oracle transparent data encryption (TDE)	Yes	Partially Binary Reader supports TDE only for self-managed Oracle databases.
Supports all Oracle compression methods	Yes	No

DMS Best Practices – Handling Oracle LOBS

- What LOB columns do you have?
- What is the biggest LOB size for each LOB column?
- Do any of the tables with LOBs not have PKeys?
- Consider using [per table LOB settings](#) in DMS task

✓ Need to plan migrations for **tables that have no PKs and contain LOBs**. Here is a query to identify those tables:

```
SELECT owner, table_name FROM dba_tables where owner='schema_name' and table_name NOT IN (SELECT table_name FROM dba_constraints WHERE constraint_type = 'P' and owner='schema_name ') and table_name in (select DISTINCT table_name from dba_tab_cols where data_Type IN ('CLOB', 'LOB', 'BLOB') and owner ='schema_name ');
```

✓ Find **the max LOB size** using Oracle system tables:

```
select 'select (max(length(' || COLUMN_NAME || '))/(1024)) as "Size in KB" from ' || owner || '.' || TABLE_NAME || ';' "maxlobsizeqry"
from dba_tab_cols
where owner= 'schema_name' and data_type in ('CLOB','BLOB','LOB');
```

DMS Best Practices – Table Mappings JSON

Understand the richness of the Table Mappings JSON.

- https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.CustomizingTasks.TableMapping.html
- You can specify filters
- You can adjust for the UPPERCASE (Oracle) vs lowercase (PostgreSQL) data dictionary differences
- You can use different settings (LOB, parallel) by table
- You can replicate big tables in parallel chunks
- You can order big tables to load first

DMS Best Practices – Plan time for setup and dry runs

When migrating databases to the cloud, almost every customer's pre-cloud database configuration is unique.

- Allow time in your schedule to work through any site-specific source configuration setup.
- Also, allow time for dry runs to make sure you don't have surprises for the production cutover.
- Reach out to AWS Support if you run into technical issues

DMS Best Practices – Understand the scope

Understand the scope of DMS. For instance,

- It doesn't replicate stored procedures.
- It doesn't replicate sequence values.
- It doesn't enable/disable foreign keys or triggers for you.

Helpful blog showing a realistic workflow:

<https://aws.amazon.com/blogs/database/how-to-migrate-your-oracle-database-to-postgresql/>

1. Create your schema in the target database.
2. Drop foreign keys and secondary indexes on the target database, and disable triggers.
3. Set up a DMS task to replicate your data – full load and change data capture (CDC).
4. Stop the task when the full load phase is complete, and recreate foreign keys and secondary indexes.
5. Enable the DMS task.
6. Migrate tools and software, and enable triggers.

DMS Best Practices – Don't move what you don't need

More good tips from the same blog we just discussed:

<https://aws.amazon.com/blogs/database/how-to-migrate-your-oracle-database-to-postgresql/>

Before we cover SCT and DMS, you should take some preliminary steps that have been proven to be helpful for every migration. One way to make the migration easier is to have what is usually called a *modernization phase* before the migration. This phase involves taking an inventory of objects in your Oracle database and then making a few decisions.

First, deprecate any objects that are no longer needed. Don't waste time migrating objects that no one cares about. Also, purge any historical data that you no longer need. You don't want to waste time replicating data you don't need, for example temporary tables and backup copies of tables from past maintenance. Second, move flat files and long strings stored in LOBs, CLOBs, LONGs, and so on into [Amazon S3](#) or [Amazon Dynamo DB](#). This process requires client software changes, but it reduces the complexity and size of the database and makes the overall system more efficient. The objects, like PL/SQL packages and procedures, need to be manually migrated if SCT cannot translate them, or considered to be moved back to the client software.

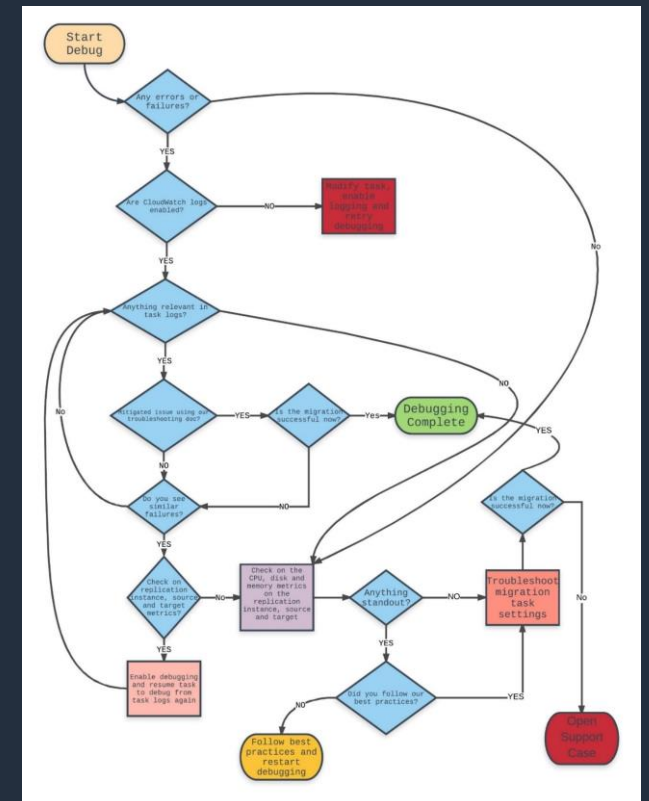
DMS Best Practice - General

There is a good blog for dealing with troubleshooting:

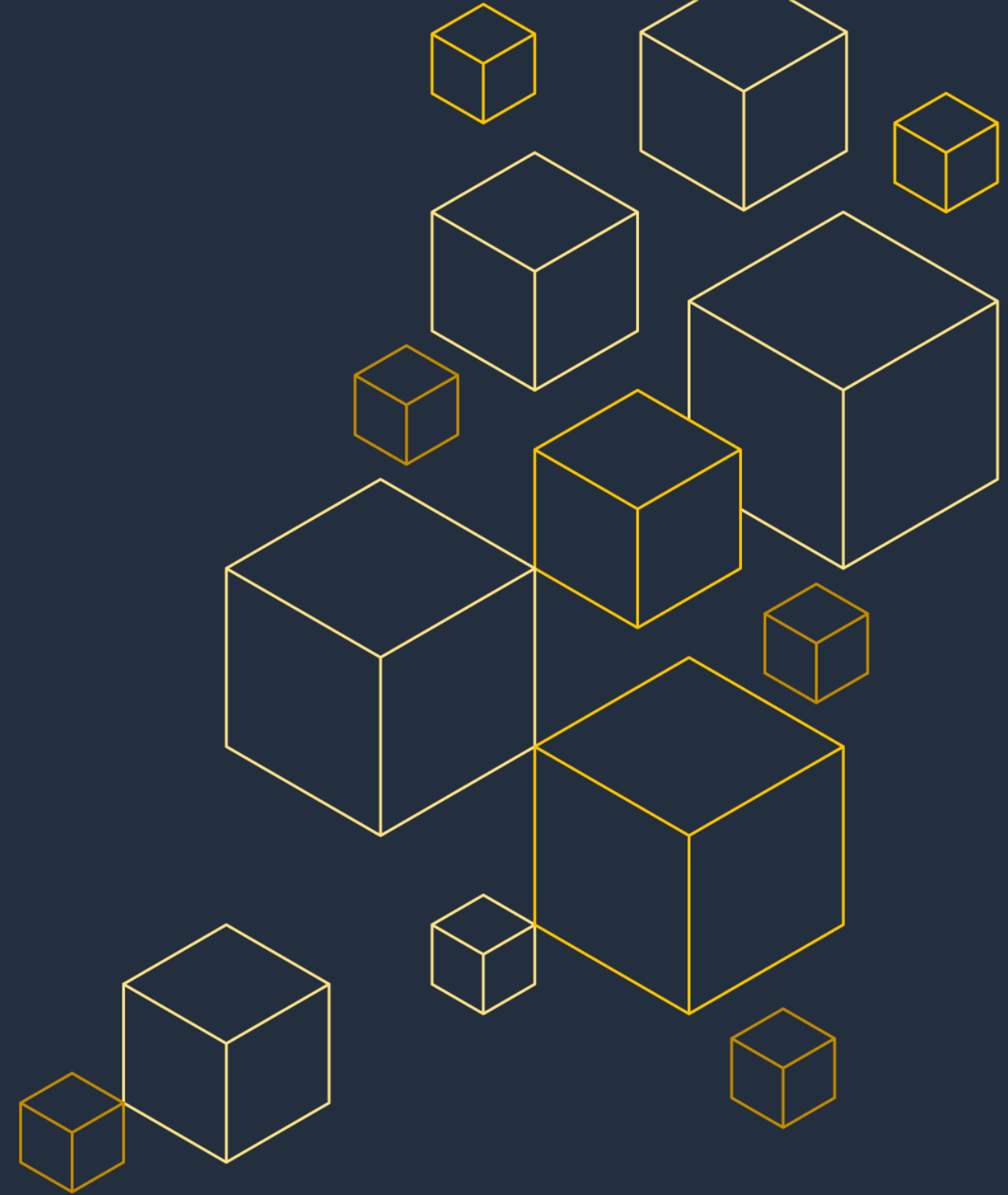
<https://aws.amazon.com/blogs/database/debugging-your-aws-dms-migrations-what-to-do-when-things-go-wrong-part-1/>

As we traverse the flow chart through all the decision boxes, we explore what the importance of each step is in this troubleshooting process. We talk about a few tips and tricks that we have picked up along the way while helping debug thousands of AWS DMS migrations. As stated earlier, migrations are complex and require some configuration tuning and testing based on a number of factors to be successful. As you determine the best possible configuration parameters for your migration using AWS DMS, here are a few factors to consider:

1. Infrastructural issues on the AWS DMS replication instance or source database or target database instances
2. Network issues between the source and replication instance or between the replication instance and the target
3. Data-related issues on the sources
4. AWS DMS limitations (you can find specific limitations for each of our [sources](#) and [targets](#))



How AWS Can Help



AWS Database Freedom

Programs



Database Freedom **reduces the risk and cost** of migrations via technical workshops, POCs, Pilots, and trained Partners

Experts



Extend your talent with AWS Solutions Architects, Professional Services, System Integrators, Training & Certification for your teams

Speed your migration by leveraging **proven practices and guidance**

Innovation



Innovative migration tools such as AWS Database Migration Service ([DMS](#)) and Schema Conversion Tool ([SCT](#)), with high automation to **reduce manual effort**

Amazon Database Migration Accelerator

Risk-mitigated way to convert legacy workloads at fixed price



Leverage AWS database experts

Leverage AWS database experts who have over **100,000 hours of cumulative experience** with technologies such as PostgreSQL, MySQL, and other AWS databases to get your migrations right.



Mitigate Risk

Our **fixed price model**, combined with the flexibility to pay after the migration is complete, reduces your risk of budget and cost overruns.



Migrate Faster

Automated tooling including [AWS Database Migration Service \(DMS\)](#) allows for quick analysis, accurate schema replication, and rapid data transfer.



Customer need

AWS customers need to **accelerate the building of solutions** composed from multiple AWS services, and they want expert guidance from AWS in how to design and validate their architecture. And once they design it, they need a **roadmap for how to put that architecture to work** for their organization.

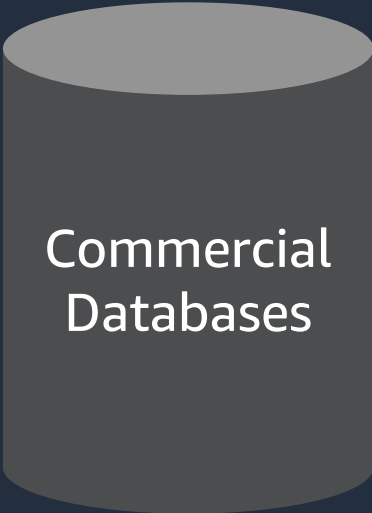


The solution

AWS Data Lab was created to **help bridge the gap** between technology and builders. Having customers engage in a Data Lab and learn directly from Data Architects and AWS experts **greatly reduces barriers** to adoption for quicker and more productive project implementation. Our offerings represent a mutual investment by AWS and our customer in a joint engineering engagement to accelerate cloud projects.

Recap

Why customers are migrating?



Expensive



**Restrictive
licensing
terms**



**Reduce
Cost**



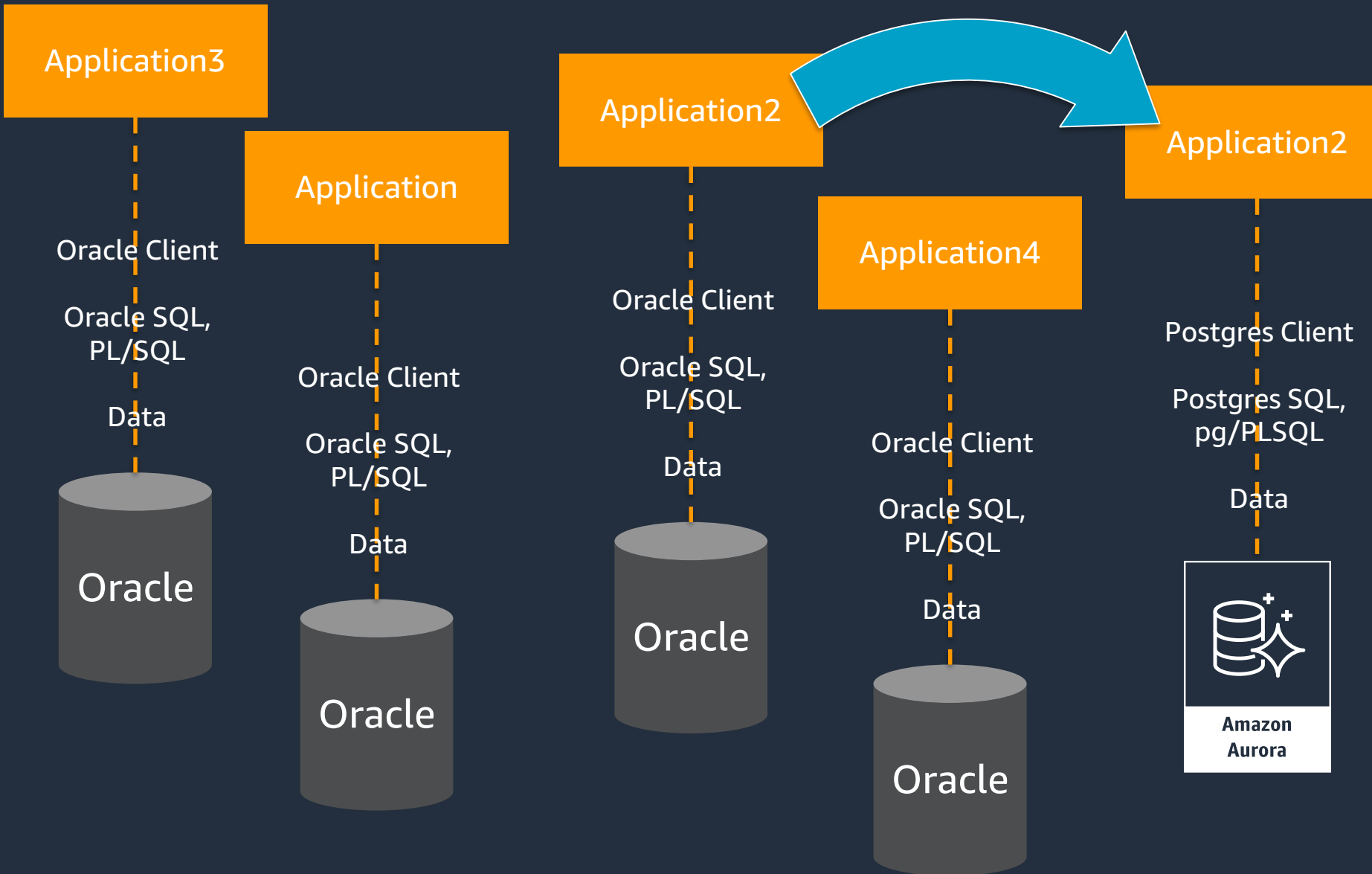
**Increase
Agility**



**Innovate
Faster**

Migrating Oracle to Aurora PostgreSQL

- What does Oracle to Aurora PostgreSQL really look like?



- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL
- Migrate Data from Oracle to Postgres
- Replace Oracle Client with Postgres Client
- Test Application
- Cut over Production

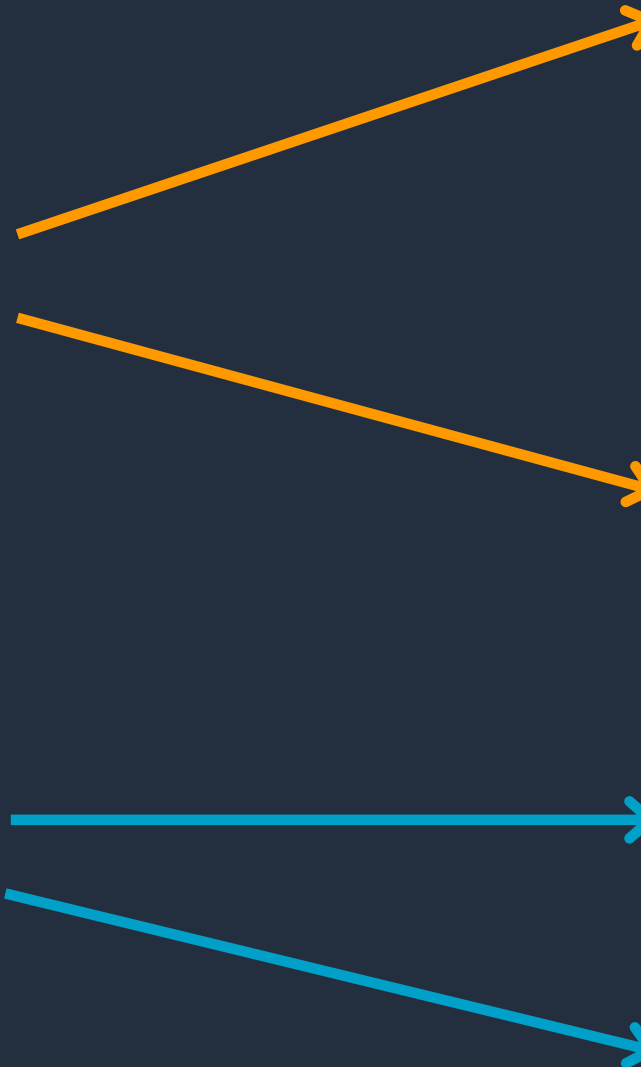
AWS Tools to help Oracle to Aurora Migration



Schema
Conversion Tool



AWS Database
Migration
Service



- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc) from Oracle to Postgres
- Convert Database code-objects (functions, triggers, etc) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to Postgres (ANSI) SQL
- Migrate Data from Oracle to Postgres
- Replace Oracle Client with Postgres Client
- Test Application
- Cut over Production

AWS Database Freedom

Programs



Database Freedom **reduces the risk and cost** of migrations via technical workshops, POCs, Pilots, and trained Partners

Experts



Extend your talent with AWS Solutions Architects, Professional Services, System Integrators, Training & Certification for your teams

Speed your migration by leveraging **proven practices and guidance**

Innovation



Innovative migration tools such as AWS Database Migration Service ([DMS](#)) and Schema Conversion Tool ([SCT](#)), with high automation to **reduce manual effort**

Thank You

David Bayard

Principal Database Specialist Solutions Architect

Eli Doe

Database Migration Specialist Solutions Architect