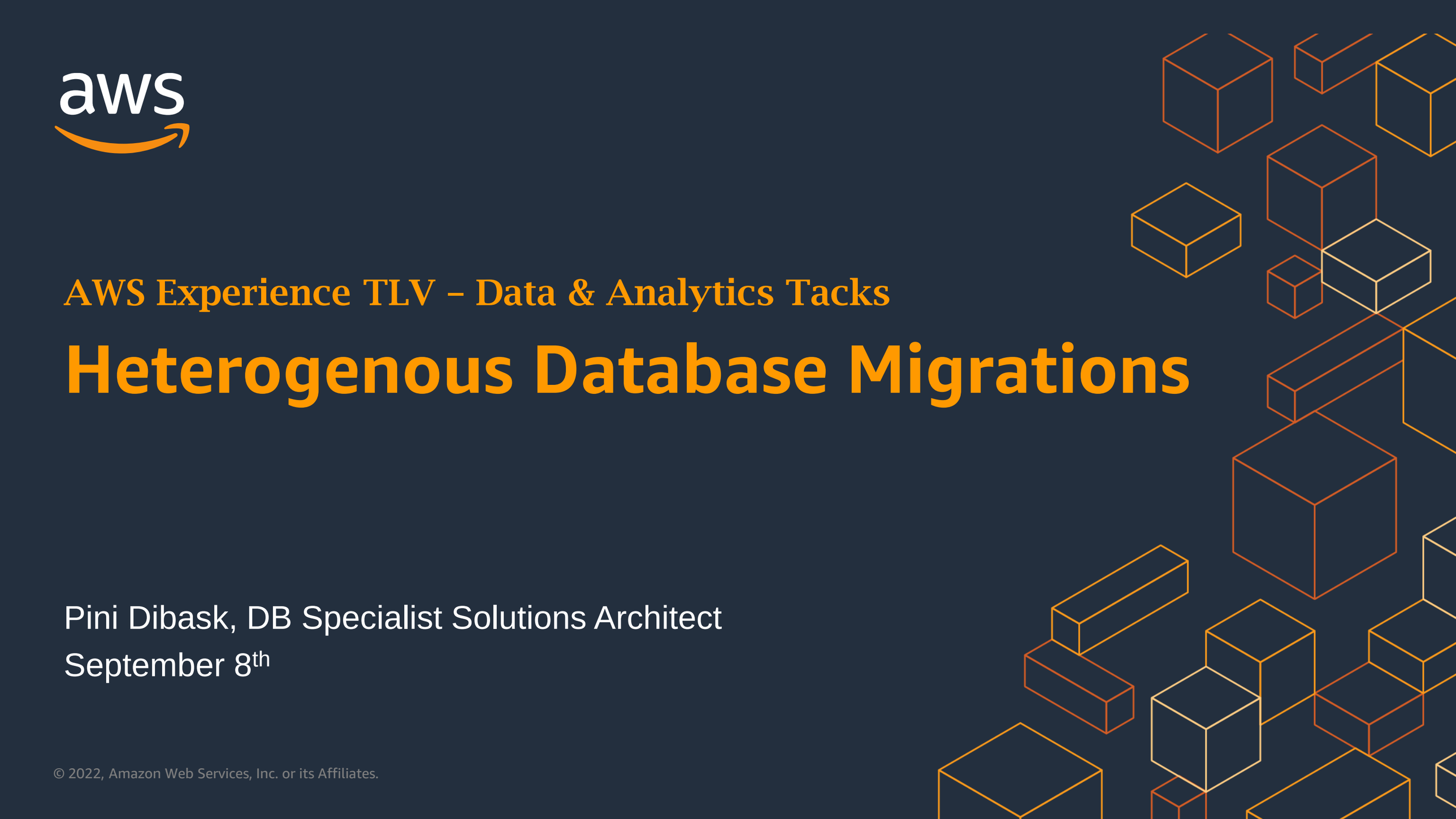


AWS Experience TLV – Data & Analytics Tacks

Heterogenous Database Migrations

Pini Dibask, DB Specialist Solutions Architect

September 8th

Agenda

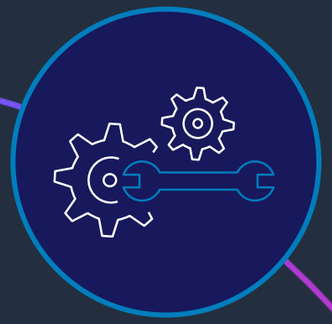
- Introduction to Database Modernization
- Database Modernization Tools
  - Schema Conversion Tool (SCT)
  - Database Migration Service (DMS)
- Oracle to Amazon Aurora Migration Demo
- Key Best Practices

Database Modernization

aws

Developers want the right database to meet their application's unique requirements

Lack of scalability



Lack of developer flexibility

Performance issues

One-size-fits-all approach of using a relational database for everything is **no longer working**

Inability to integrate all data types

aws

AWS delivers the right database for the right job

Relational

Referential integrity, ACID transactions, schema-on-write

Key-value

High throughput, Low latency reads and writes, endless scale

Document

Store documents and quickly access querying on any attribute

In-memory

Query by key with microsecond latency

Graph

Quickly and easily create and navigate relationships between data

Time-series

Collect, store, and process data sequenced by time

Ledger

Complete, immutable, and verifiable history of all changes to application data

Wide Column

Scalable, highly available, and managed Apache Cassandra-compatible service

AWS Service(s)	 RDS  Aurora	 DynamoDB	 DocumentDB	 ElastiCache	 Neptune	 Timestream	 QLDB	 Keyspaces Managed Cassandra
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, payments, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	IoT applications, event tracking	Systems of record, supply chain, health care, registrations, financial	Build low-latency applications, leverage open source, migrate Cassandra to the cloud

aws

Lift and Shift, Move to Managed, or Break Free?

Value

Time

Migrate: Lift and Shift
(re-host)

- Quickest way to onboard workloads
- Self managed (i.e. Runs as on-premises)
- Vendor lock-in on commercial databases (e.g. Oracle, SQL Server)

Migrate: Move to Managed
(re-platform)

- Reduce undifferentiated heavy lifting (backup, recovery, patching, etc)
- RDS Commercial Engines (Oracle, SQL Server) & Open Source (PostgreSQL, MySQL, MariaDB)
- Support for growing # of 3rd party apps *

Modernize: Break Free
(re-factor)

- Reduced licensing costs
- Move to AWS modern managed databases like Aurora, DynamoDB, and Redshift for faster innovation
- Break free from legacy databases
- Avoid lock-in with open source database
- Pick the right tool for the right job: choose from 15 purpose-built databases

aws

Database Modernization Tools

aws

Modernization Tools

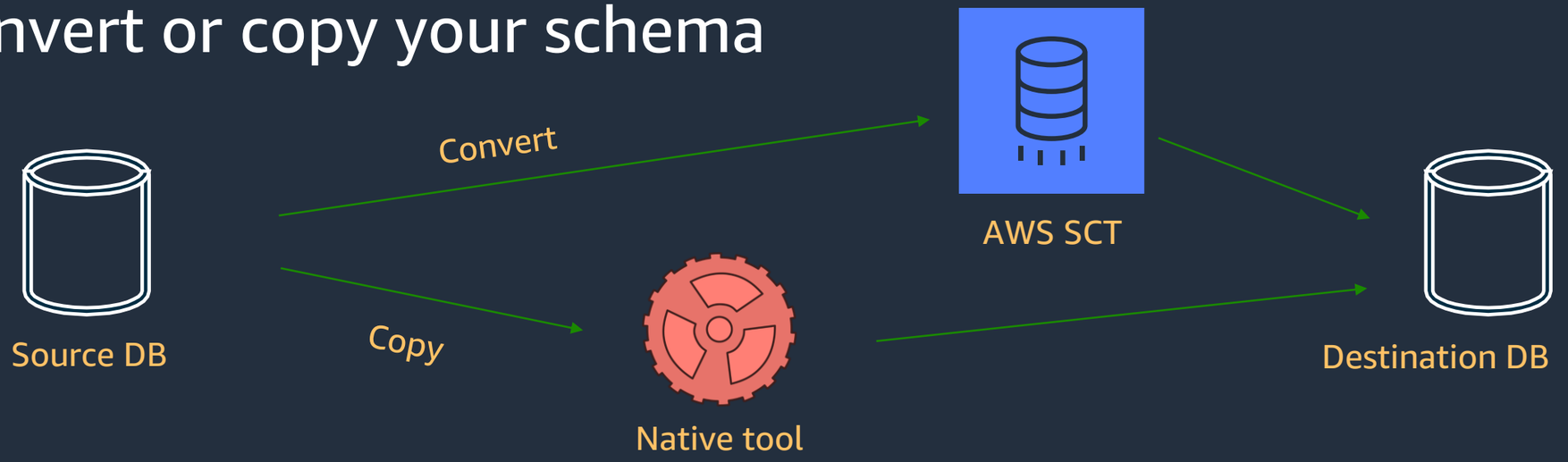
AWS Schema Conversion Tool converts your commercial database and data warehouse schemas to open-source engines or AWS-native services such as Amazon Aurora and Amazon Redshift

AWS Database Migration Service (AWS DMS) easily and securely migrates and/or replicates your databases *and* data warehouses to AWS

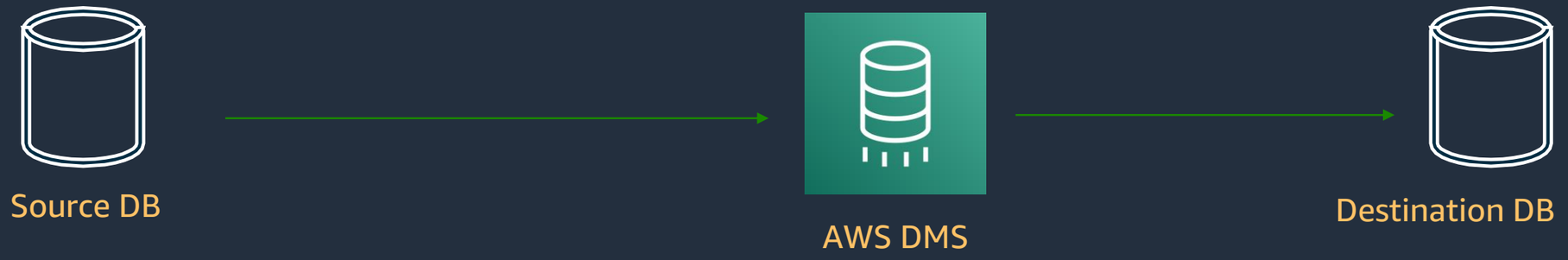
aws

Database migration process

Step 1: Convert or copy your schema



Step 2: Move your data



aws

SCT - Pricing and platform support

$0

You can download AWS Schema Conversion Tool for your platform of choice

Microsoft Windows

Apple Mac (deprecated)

Fedora Linux (rpm)

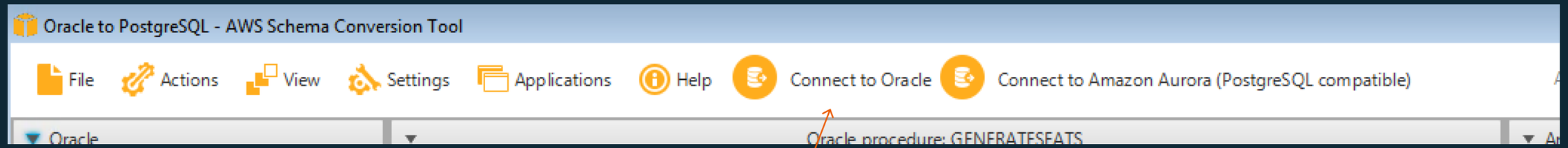
Ubuntu Linux (deb)

SCT Supported Conversions

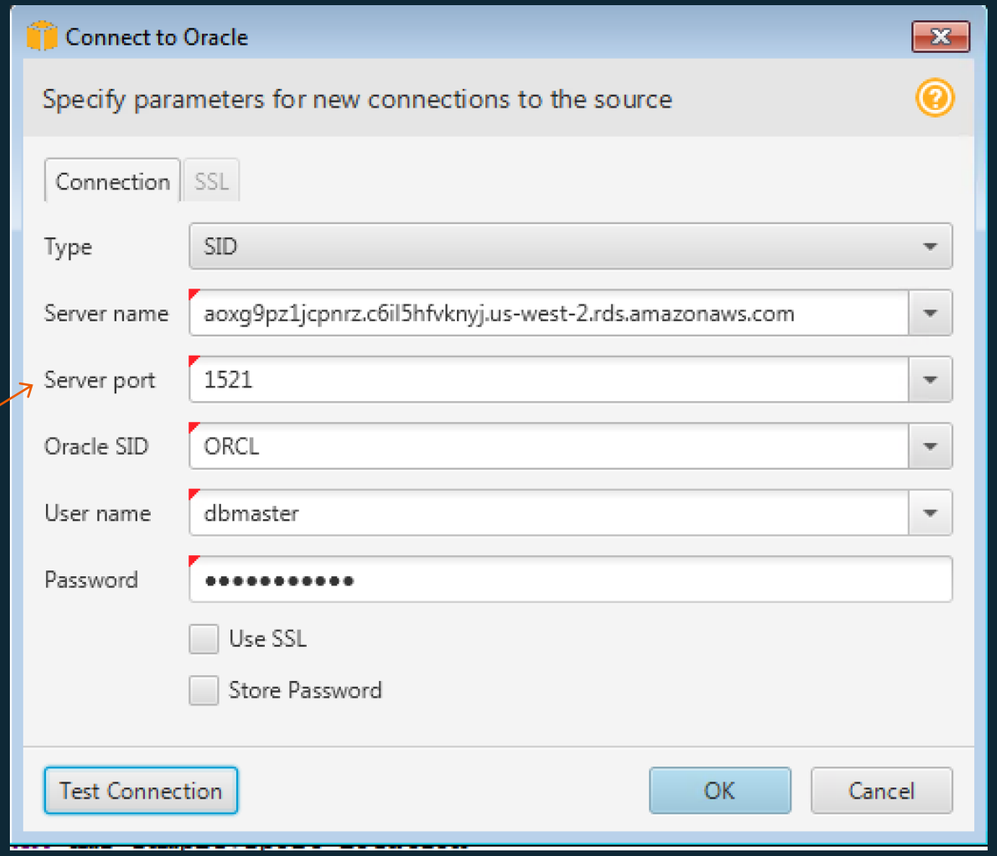
Source* Database	Target* Database on AWS
Oracle database	Amazon Aurora, MySQL/MariaDB, PostgreSQL, Oracle
Oracle data warehouse	Amazon Redshift
Azure SQL	Amazon Aurora, MySQL, PostgreSQL
Microsoft SQL Server	Amazon Aurora, Amazon Redshift, MySQL PostgreSQL
Teradata	Amazon Redshift
IBM Netezza	Amazon Redshift
Greenplum	Amazon Redshift
HPE Vertica	Amazon Redshift
MySQL and Maria DB	PostgreSQL
PostgreSQL	Amazon Aurora, MySQL
IBM DB2 LUW	Amazon Aurora, MySQL, PostgreSQL
Apache Cassandra	Amazon DynamoDB

https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Source.AzureSQL.html

Connecting to your database



1. Create a new project
2. Connect to your source database
3. Enter the connection details



Migration assessment report

Database Migration Assessment Report

Source Database: RDS_ADMINISTRATION.rds_administrator@ec2-54-172-36-60.compute-1.amazonaws.com:8192:ORCL

Oracle Database 12c Enterprise Edition 12.1.0.1.0 (64bit Production)

amazon web services

Executive Summary

We completed the analysis of your Oracle source database and estimate that 91% of the database storage objects and 100% of database code objects can be converted automatically or with minimal changes if you select Amazon Aurora as your migration target. Database storage objects include schemas, tables, columns, constraints, indexes, sequences, synonyms, user define types and types. Database code objects include functions, procedures, packages, triggers, views, materialized views, events, SQL scalar functions, SQL inline functions, SQL table functions, attributes, variables, constants, table types, public types, private types, cursors, exceptions, parameters and other objects. Based on our analysis of SQL syntax elements of your source database schema, we estimate that 99.9% of your entire database schema can be converted automatically to Amazon Aurora. To complete the migration, we recommend 597 conversion action(s) ranging from simple tasks to medium-complexity actions to significant conversion actions.

Database Objects with Conversion Actions for Amazon Aurora

Of the total 1,576 database storage object(s) and 155 database code object(s) in the source database, we were able to identify 1,427 (91%) database storage object(s) and 155 (100%) database code objects that can be converted automatically or with minimal changes to Amazon Aurora.

149 (9%) database storage object(s) required 149 significant user action(s) to complete the conversion.

Figure: Conversion statistics for database storage objects

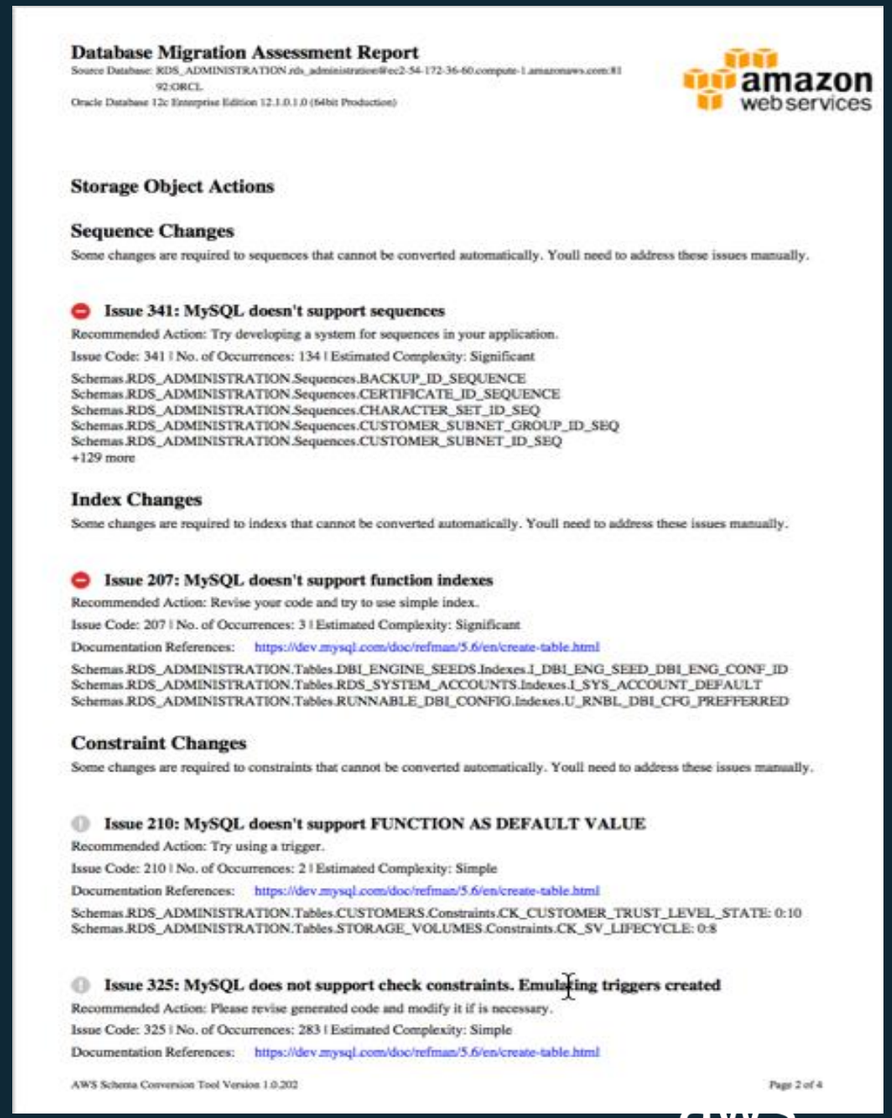
Figure: Conversion statistics for database code objects

Detailed Recommendations for Amazon Aurora Migrations

If you choose to migrate your Oracle database to Amazon Aurora, we recommend the following actions.

AWS Schema Conversion Tool Version 1.0.202 Page 1 of 4

4. Run assessment report
5. Review executive summary
6. Review detailed instructions



Database Migration Assessment Report

Source Database: RDS_ADMINISTRATION.rds_administrator@ec2-54-172-36-60.compute-1.amazonaws.com:8192:ORCL

Oracle Database 12c Enterprise Edition 12.1.0.1.0 (64bit Production)

amazon web services

Storage Object Actions

Sequence Changes

Some changes are required to sequences that cannot be converted automatically. Youll need to address these issues manually.

Issue 341: MySQL doesn't support sequences

Recommended Action: Try developing a system for sequences in your application.

Issue Code: 341 | No. of Occurrences: 134 | Estimated Complexity: Significant

Schemas.RDS_ADMINISTRATION.Sequences.BACKUP_ID_SEQUENCE
Schemas.RDS_ADMINISTRATION.Sequences.CERTIFICATE_ID_SEQUENCE
Schemas.RDS_ADMINISTRATION.Sequences.CHARACTER_SET_ID_SEQ
Schemas.RDS_ADMINISTRATION.Sequences.CUSTOMER_SUBNET_GROUP_ID_SEQ
Schemas.RDS_ADMINISTRATION.Sequences.CUSTOMER_SUBNET_ID_SEQ
+129 more

Index Changes

Some changes are required to indexs that cannot be converted automatically. Youll need to address these issues manually.

Issue 207: MySQL doesn't support function indexes

Recommended Action: Revise your code and try to use simple index.

Issue Code: 207 | No. of Occurrences: 3 | Estimated Complexity: Significant

Documentation References: https://dev.mysql.com/doc/refman/5.6/en/create-table.html

Schemas.RDS_ADMINISTRATION.Tables.DBI_ENGINE_SEEDS.Indexes.I_DBI_ENG_SEED_DBI_ENG_CONF_ID
Schemas.RDS_ADMINISTRATION.Tables.RDS_SYSTEM_ACCOUNTS.Indexes.I_SYS_ACCOUNT_DEFAULT
Schemas.RDS_ADMINISTRATION.Tables.RUNNABLE_DBI_CONFIG.Indexes.U_RNBL_DBI_CFG_PREFFERRED

Constraint Changes

Some changes are required to constraints that cannot be converted automatically. Youll need to address these issues manually.

Issue 210: MySQL doesn't support FUNCTION AS DEFAULT VALUE

Recommended Action: Try using a trigger.

Issue Code: 210 | No. of Occurrences: 2 | Estimated Complexity: Simple

Documentation References: https://dev.mysql.com/doc/refman/5.6/en/create-table.html

Schemas.RDS_ADMINISTRATION.Tables.CUSTOMERS.Constraints.CK_CUSTOMER_TRUST_LEVEL_STATE: 0:10
Schemas.RDS_ADMINISTRATION.Tables.STORAGE_VOLUMES.Constraints.CK_SV_LIFECYCLE: 0:8

Issue 325: MySQL does not support check constraints. Emulating triggers created

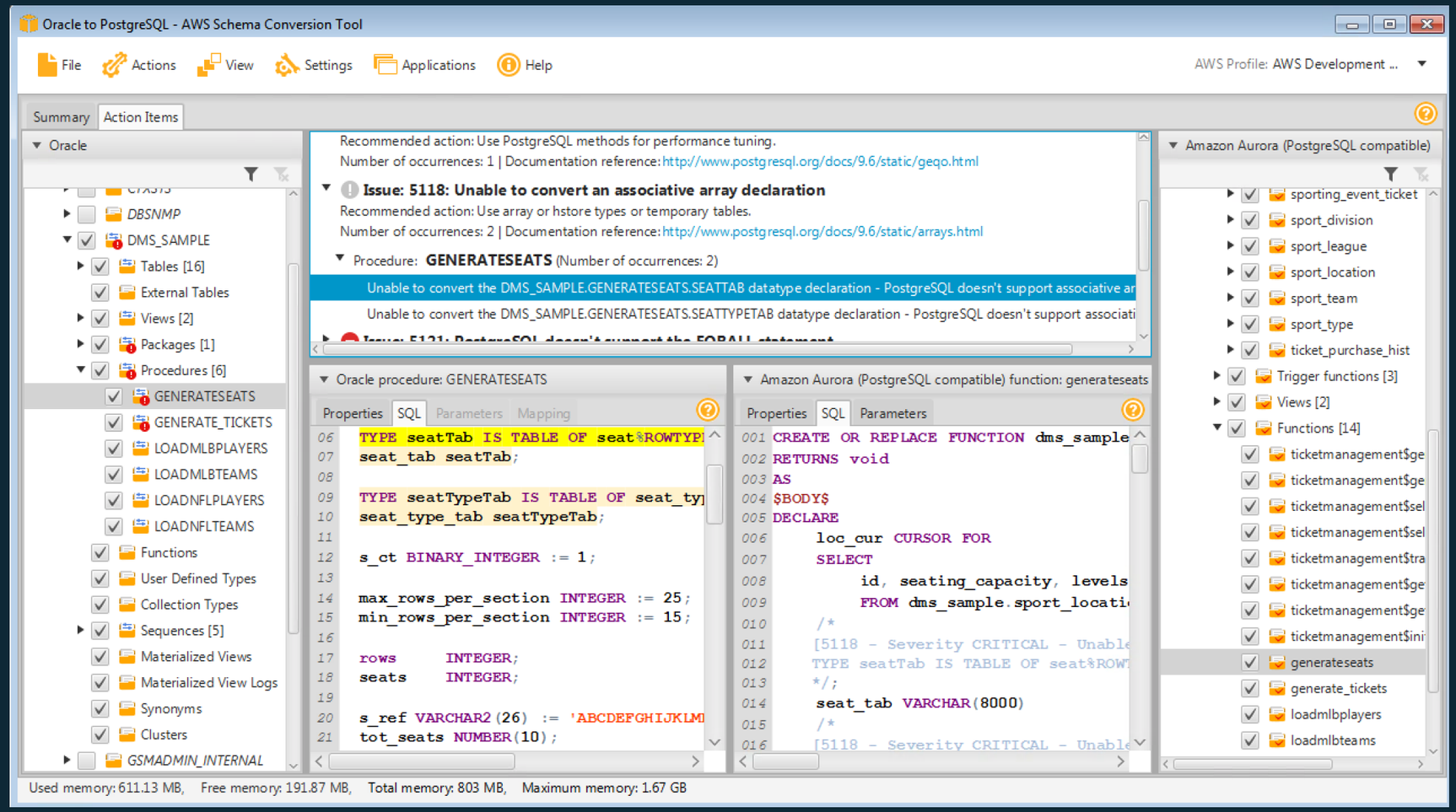
Recommended Action: Please revise generated code and modify it if is necessary.

Issue Code: 325 | No. of Occurrences: 283 | Estimated Complexity: Simple

Documentation References: https://dev.mysql.com/doc/refman/5.6/en/create-table.html

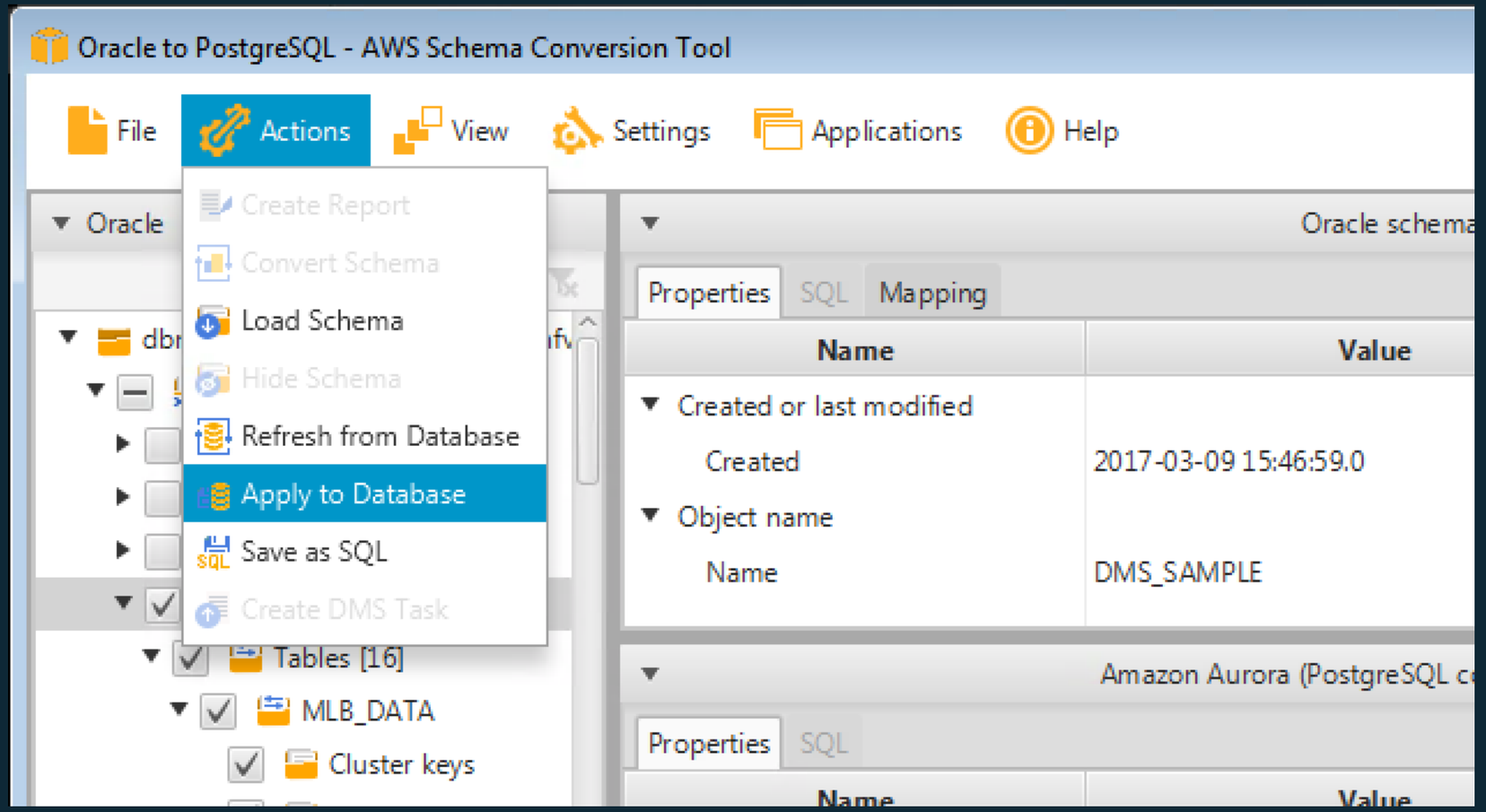
AWS Schema Conversion Tool Version 1.0.202 Page 2 of 4

Convert schema and code objects



- Functions
- Indexes
- Packages
- Tables
- Triggers
- Schemas
- Sequences
- Stored Procedures
- Synonyms
- User Defined Types
- Views

Apply to target database



Select Apply to Database from the Action menu

or

Right click on the target schema to apply

AWS Database Migration Service (DMS)

- No drivers or applications to install
- No changes to the source database in most cases
- Just a few clicks to start a migration from the console
- DMS manages the complexities of migration for you
- Fully supported by CloudFormation

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*	Targets**
Oracle	Oracle
SQL Server	SQL Server
Azure SQL	PostgreSQL
PostgreSQL	MySQL
MySQL or Google Cloud MySQL	Amazon Redshift
SAP ASE	SAP ASE
MongoDB or DocumentDB	Amazon S3
Amazon S3	Amazon DynamoDB
IBM DB2	Amazon Kinesis or Kafka
	Amazon ElasticSearch
	DocumentDB
	Neptune
	Redis / ElastiCache / MemoryDB

Sources link https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.html
Targets link https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.html

Keep your application running during migration

Start a replication instance

Connect to the source and target

Select tables, schemas, or databases

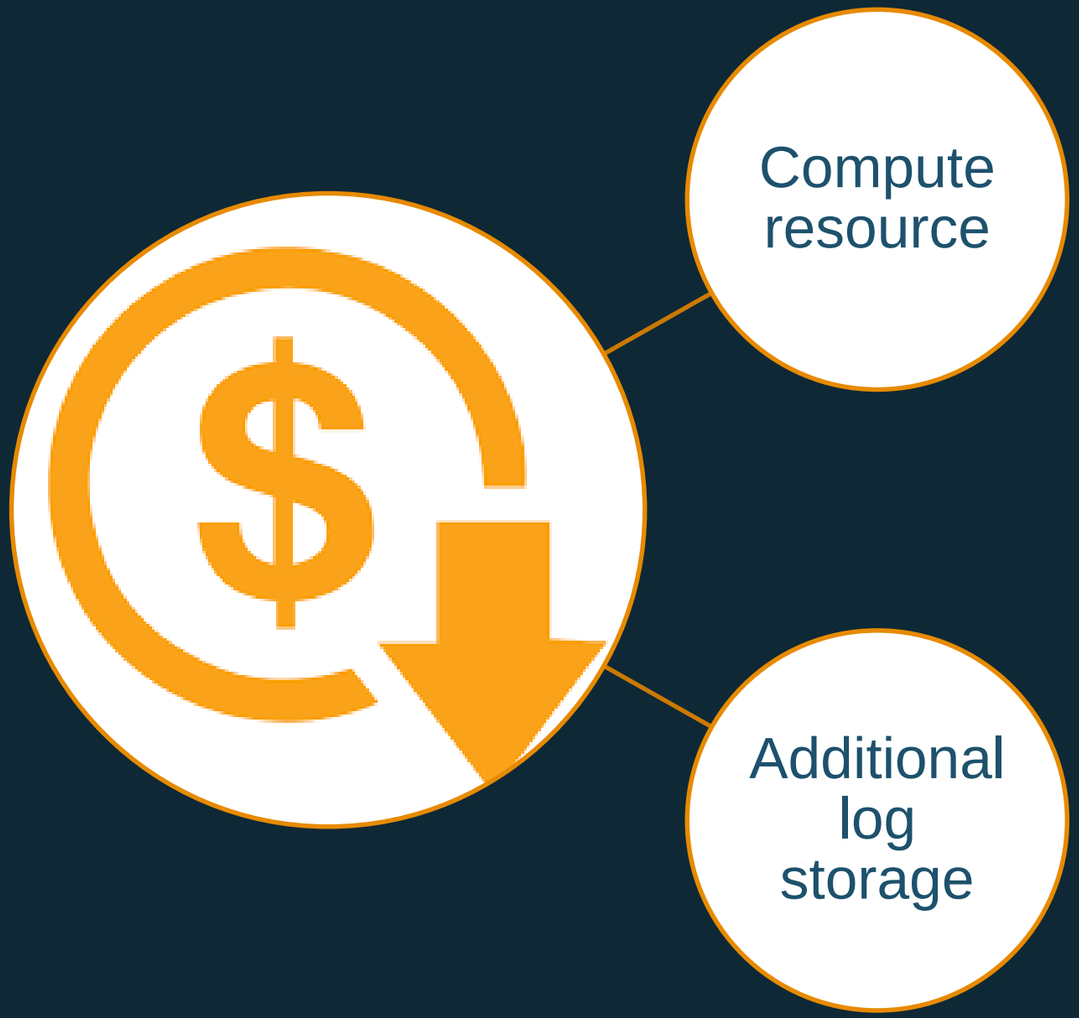
Create target objects

Move data and synchronize objects

Switch applications when ready

Low cost

(https://aws.amazon.com/dms/pricing/)

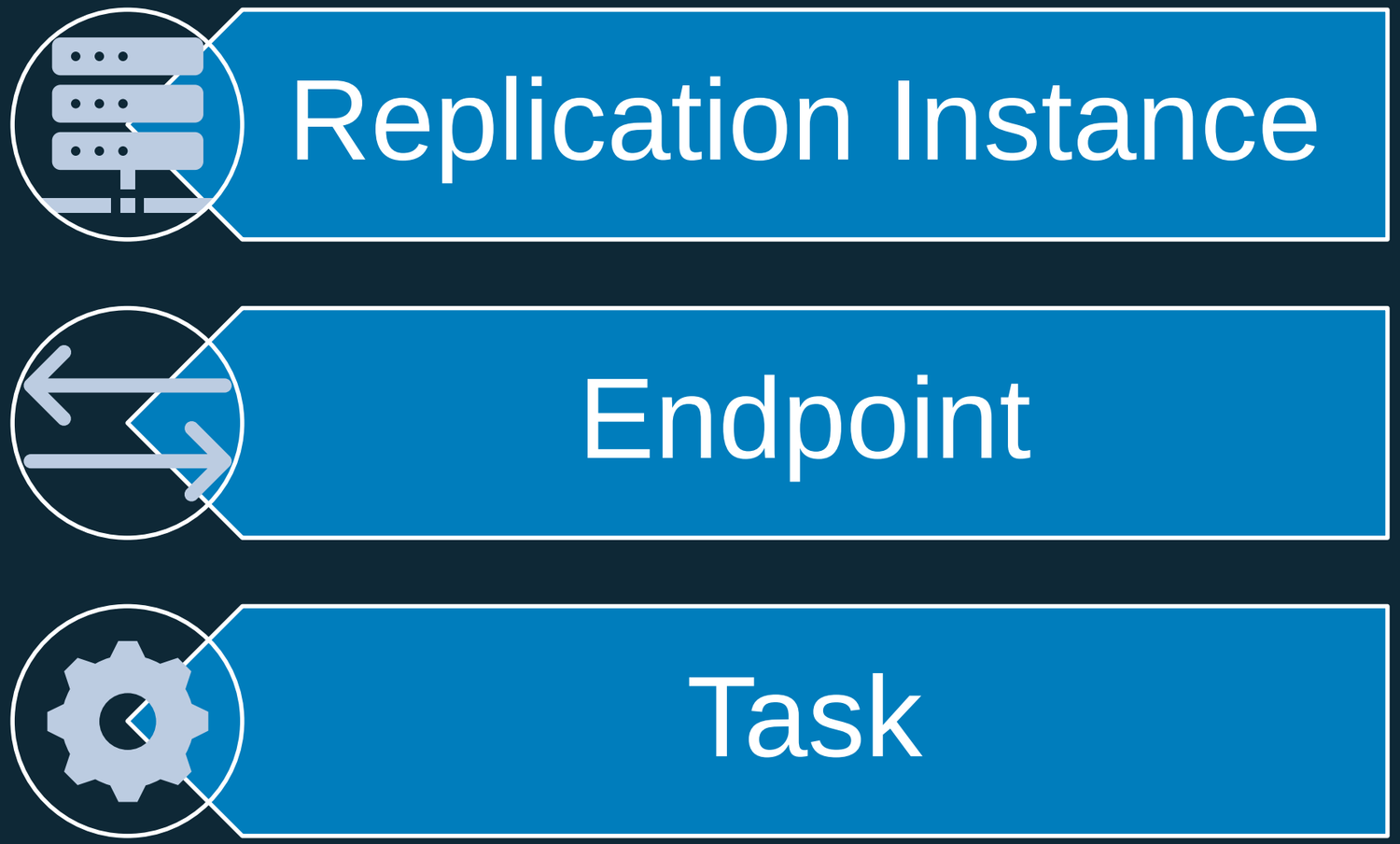


Pricing Example

Instance Type	Hourly Rate	Duration	Activities	Total
t3.small	$0.0371	2 weeks	Testing	$12.50
c5.large	$0.238	2 weeks	Initial Load & CDC Until Cutover (multi-AZ)	$79.97

Migrate a 1 TB DB for under $100 ($92.47)

DMS Components



Replication instance

Managed EC2, Variety of Instance Types

- T2/T3, C4/C5, or R4/R5 instance types

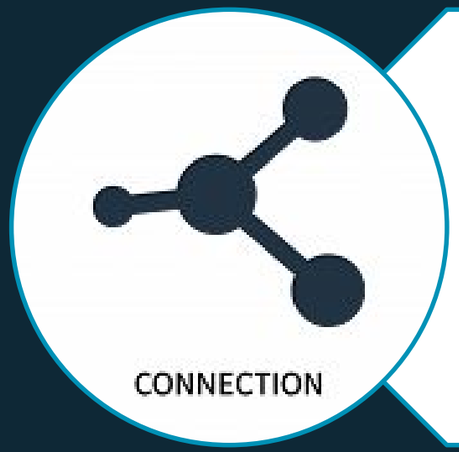
Managed HA

- Standby replica in a different AZ
- Synchronous replication

Task Execution

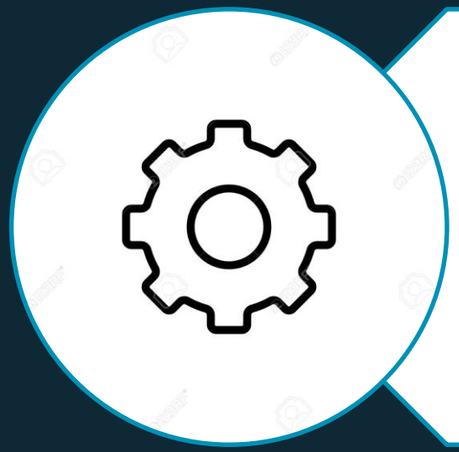
- Moves the data from the source to the target
- Support for multiple tasks

Endpoint



Connection information

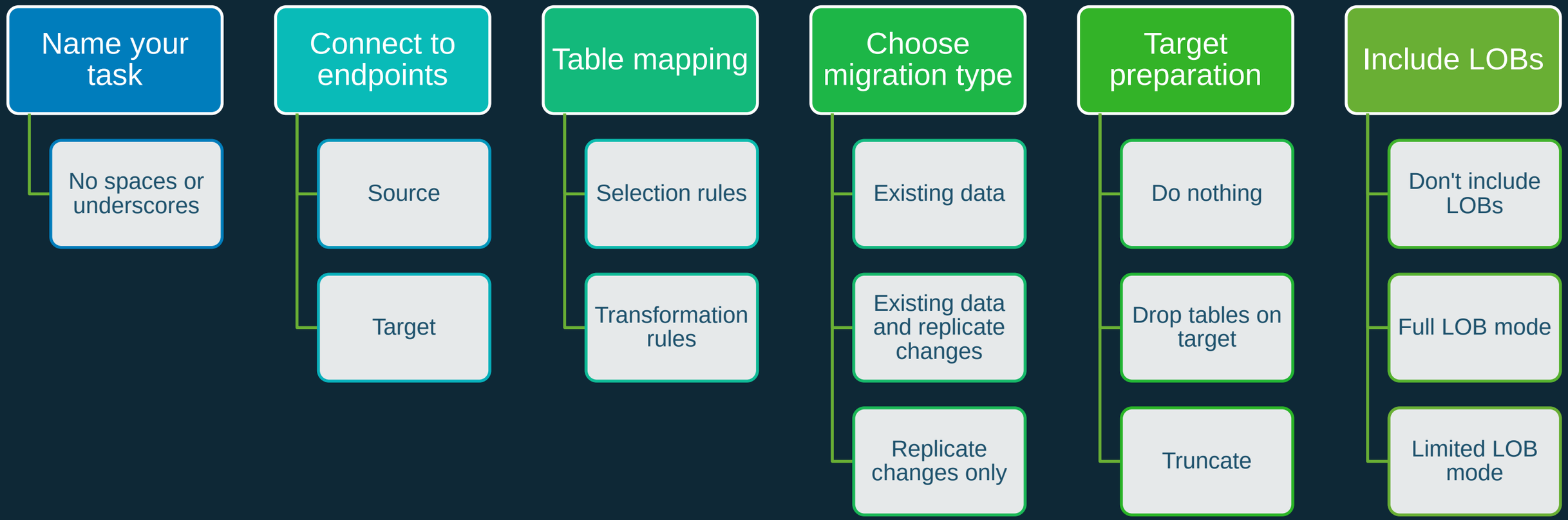
- Endpoint type (source/target), engine
- Server name, port
- Username, password



Advanced options

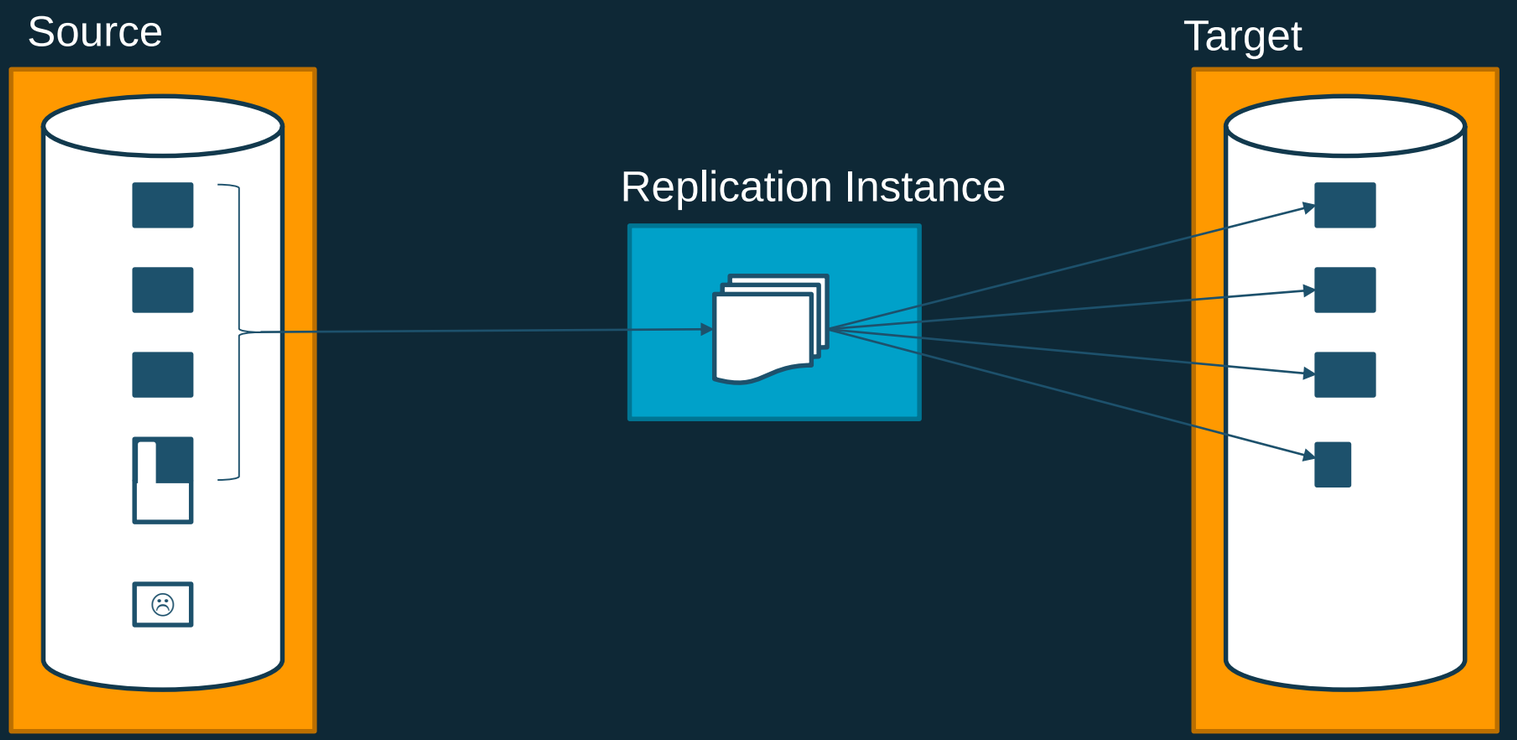
- Encryption
- Engine-specific attributes

Task

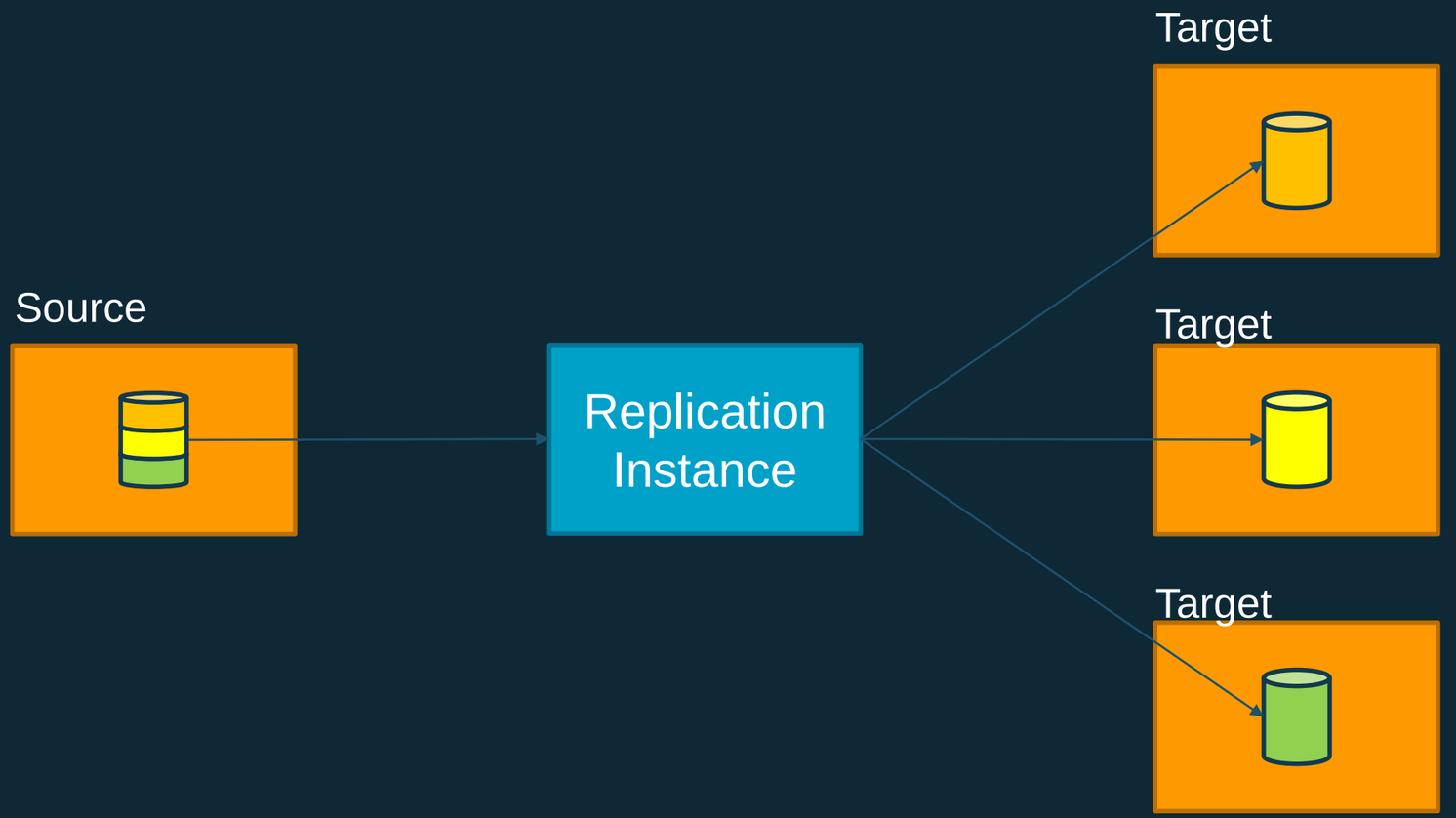


Database Migration Options

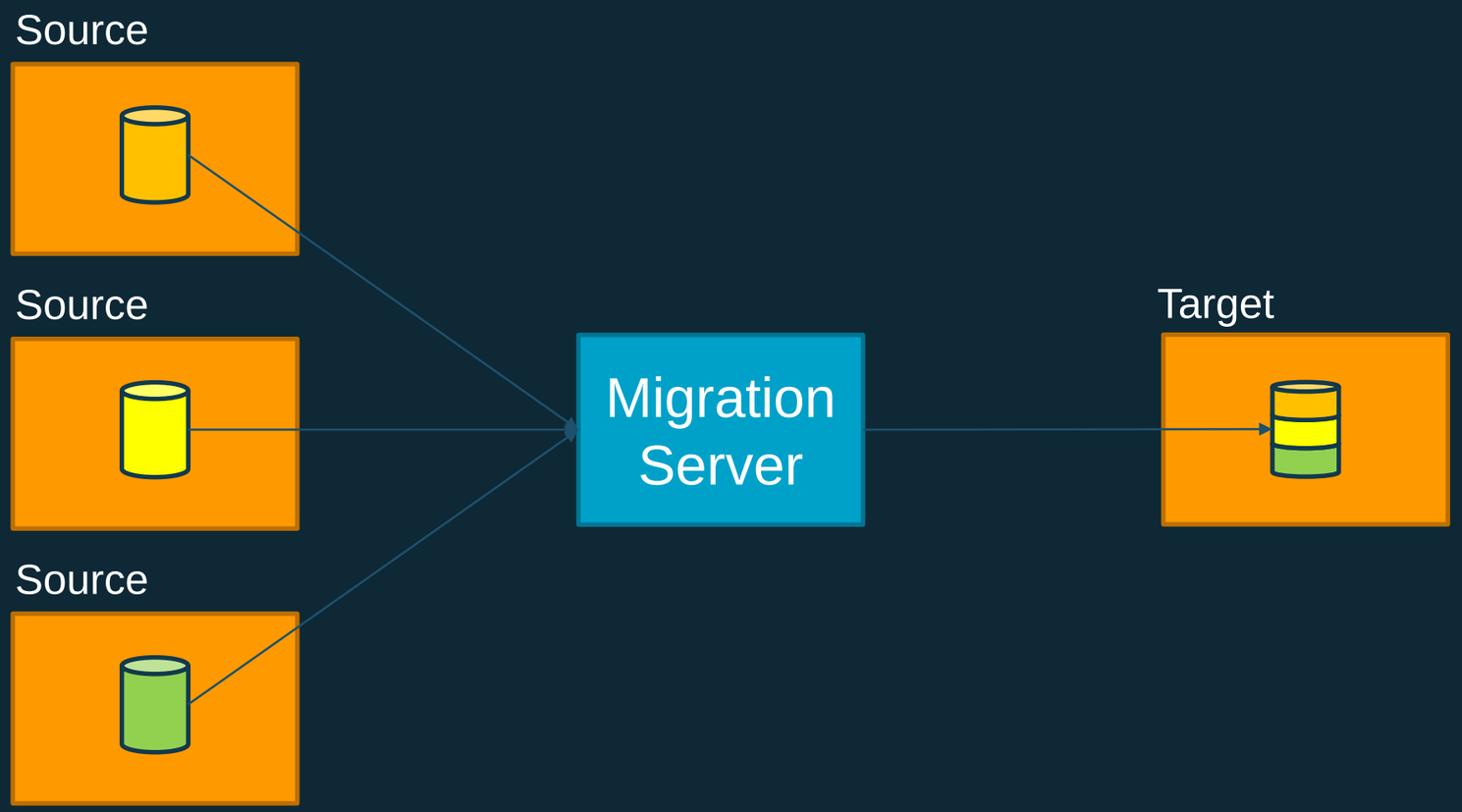
You don’t have to take everything



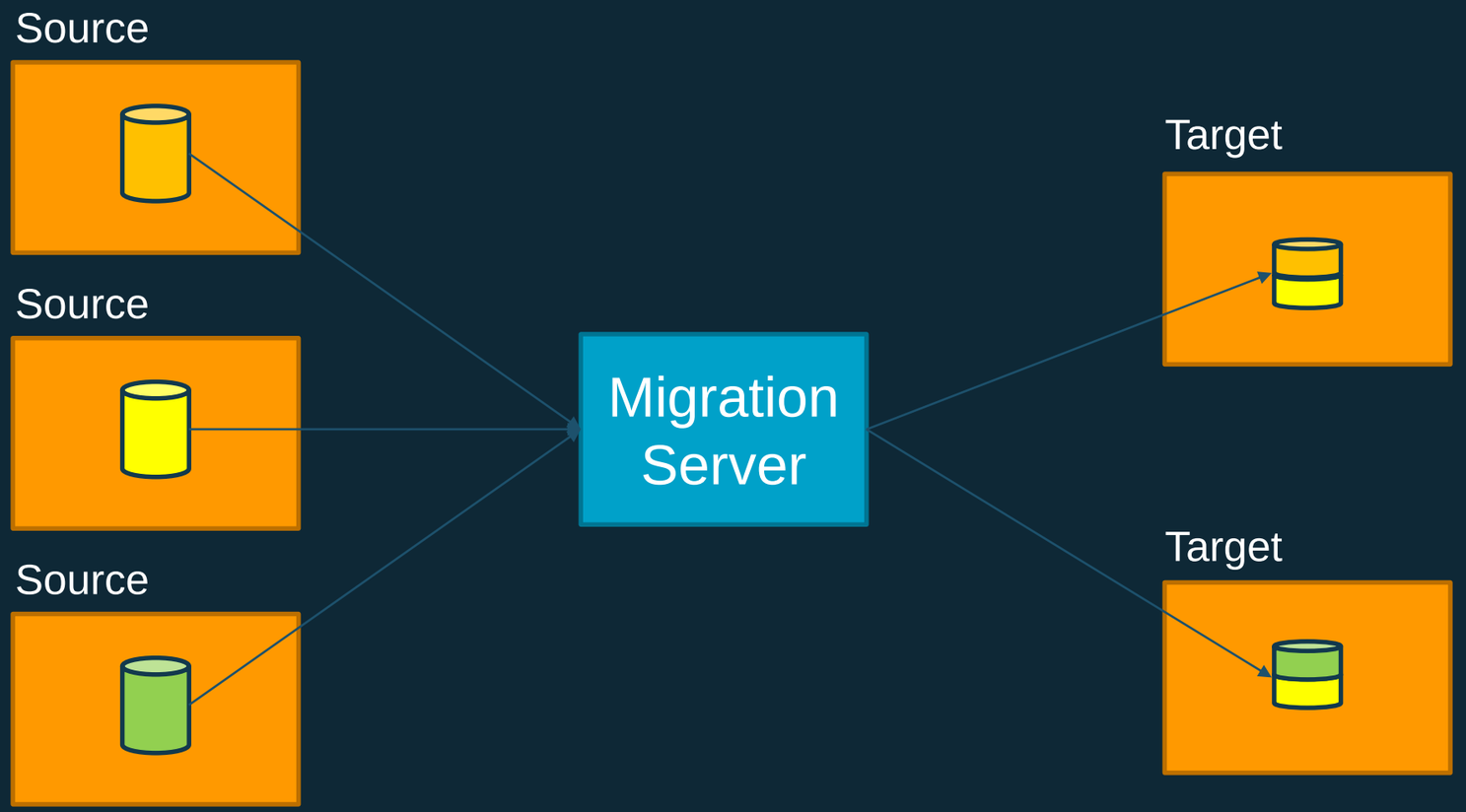
Multiple targets



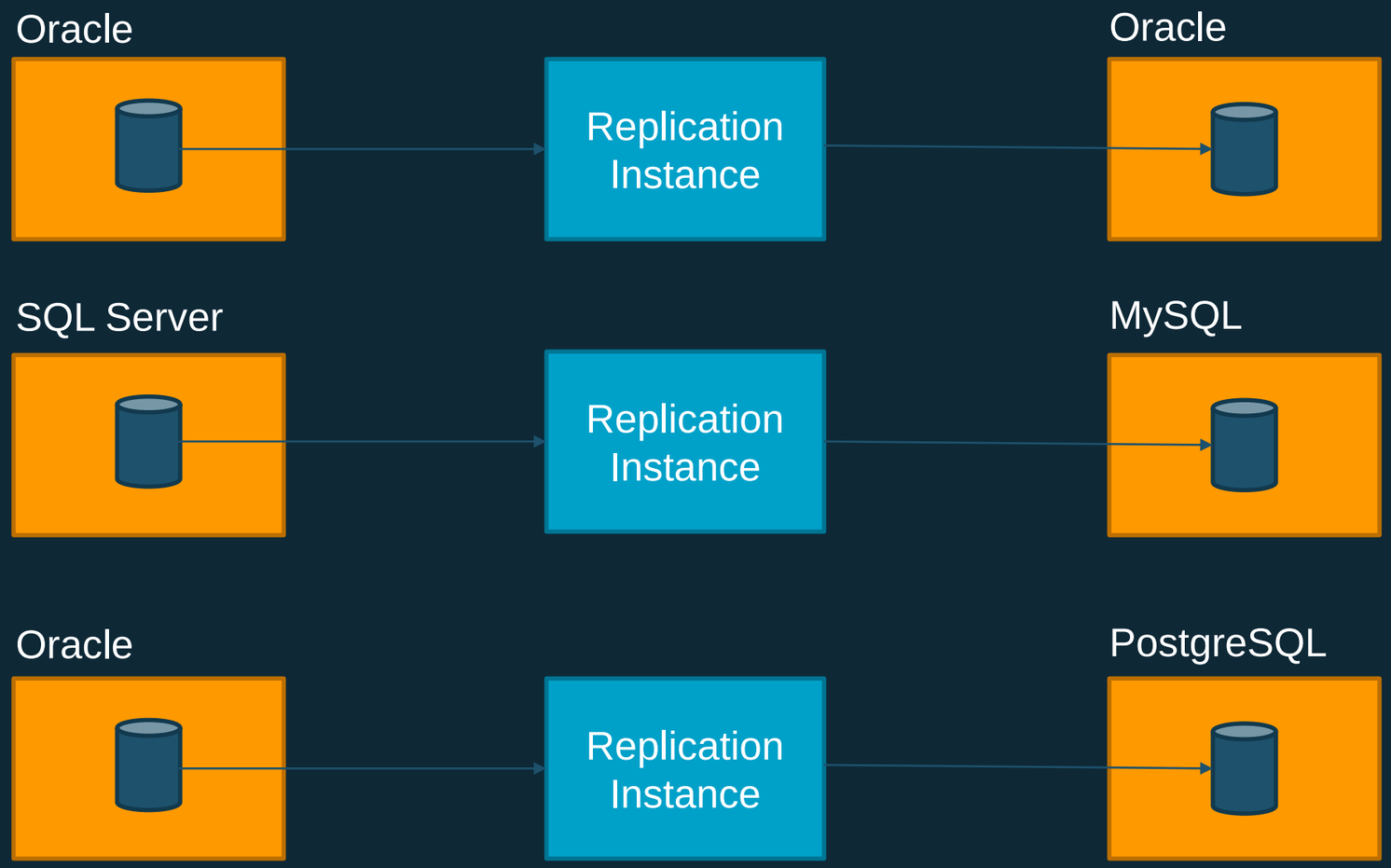
Multiple Sources



Multiple sources and targets



Homogenous or heterogeneous



Best Practices

Key Best Practices

If DMS overburdening your source database

- Full load tasks - reduce number of tables loaded in parallel
- Reduce number of concurrent DMS tasks

Reduce bottlenecks on your target databases

- Prior to full load tasks - drop indexes, constraints, triggers
- Turn off logging, Multi-AZ until cutover

Optimize performance

- Parallel table load, limited lob mode, RI right-sizing, index creation prior to CDC

Monitor your DMS tasks using metrics

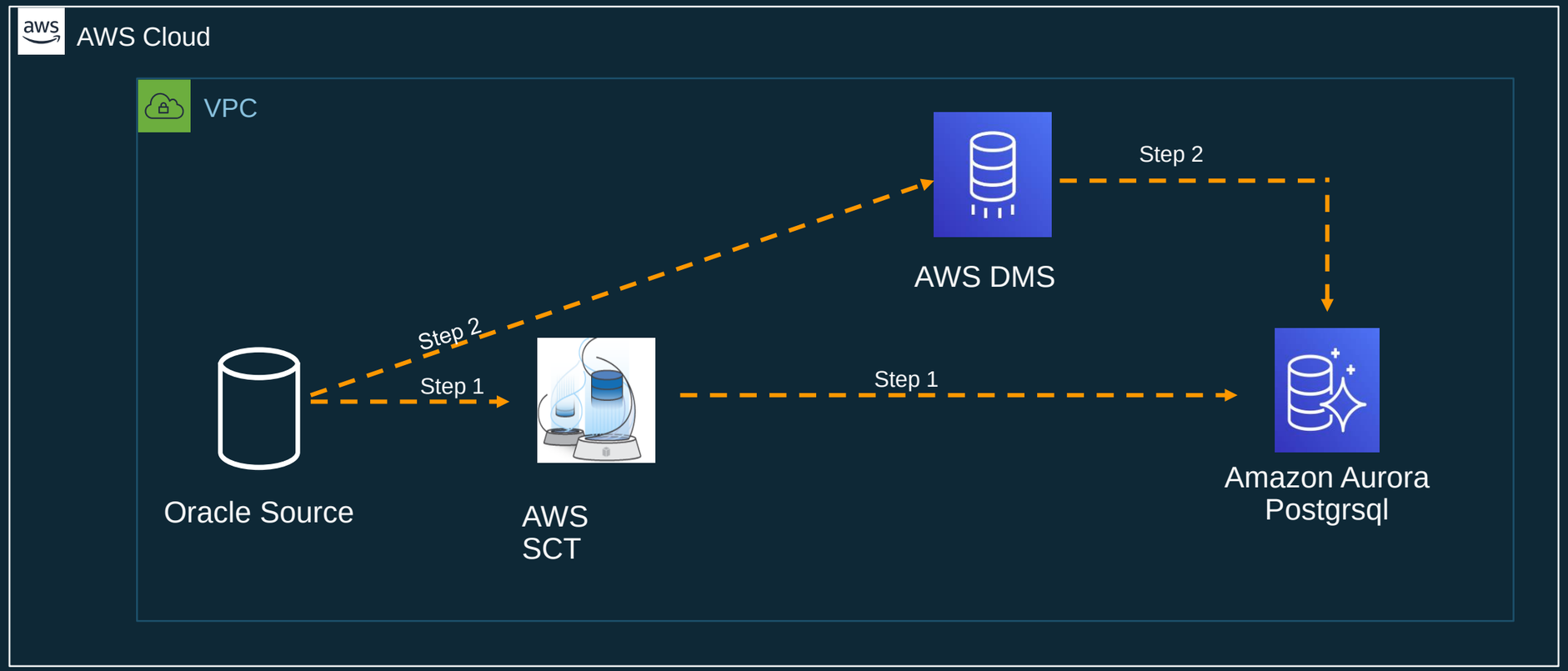
- Host metrics, replication task metrics, table metrics

Run a POC

- Profile your data, get a realistic migration timeline, enable data validation

Demo

Oracle to Amazon Aurora PostgreSQL Demo



Enterprises across all industries are embracing the power of the cloud

AWS and AWS Partners have helped migrate

over 500K

databases to AWS for customers to save, grow, and innovate faster

Customers including

Samsung, Experian, and NASDAQ

have used AWS migration services and programs to ensure their migrations were effective and cost-optimized

A variety of

flexible ways to migrate

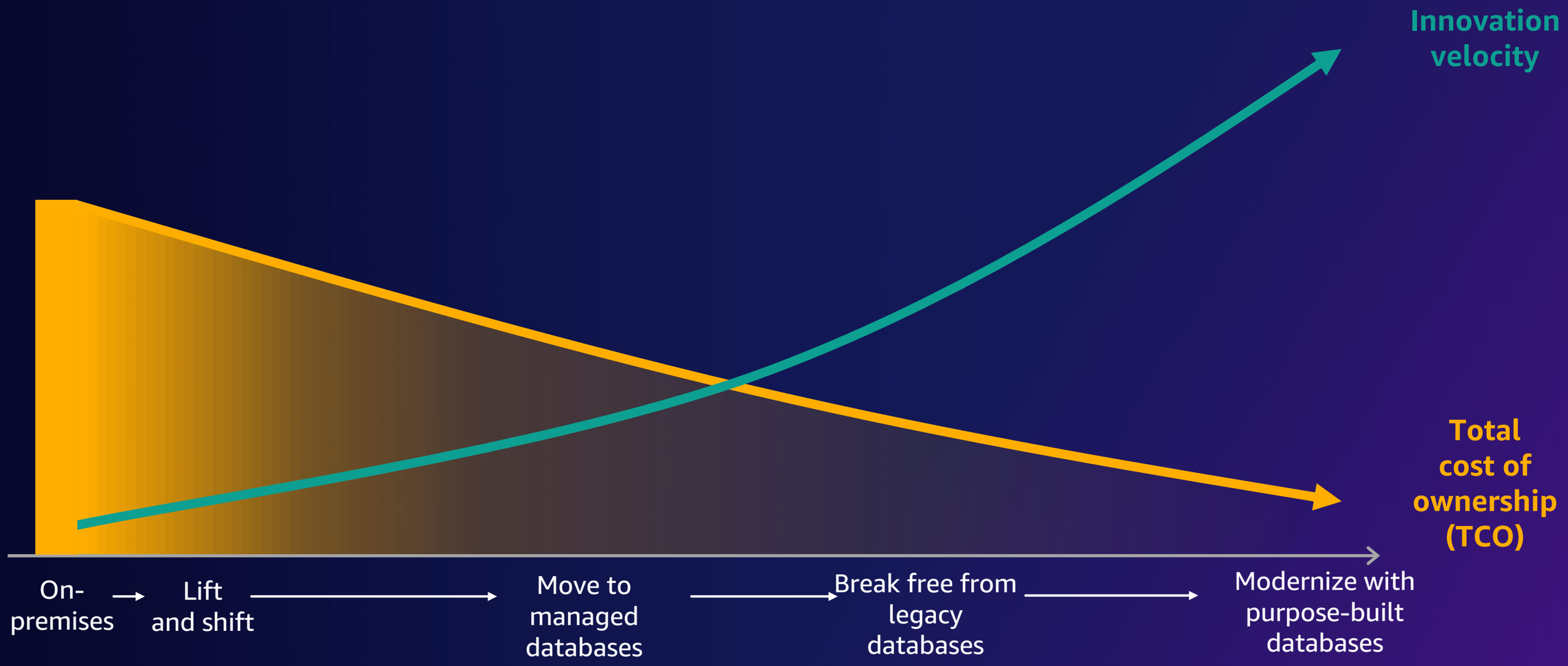
to AWS include AWS self-service migration tools, AWS ProServe, AWS Partners, and AWS database migration programs

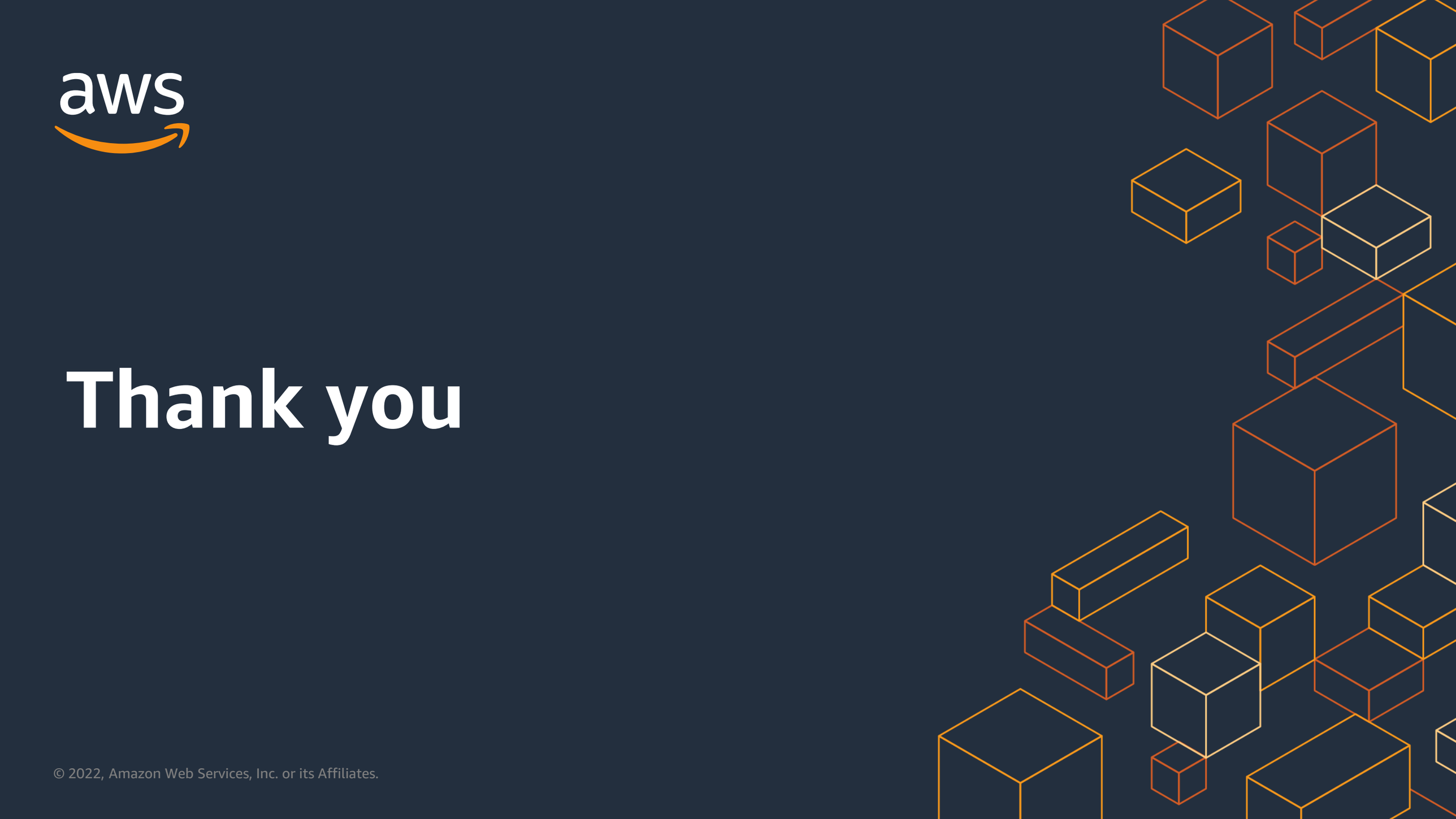
New

post-migration tools include Babelfish for Amazon Aurora PostgreSQL, which makes it possible to run Microsoft SQL Server applications directly on Amazon Aurora and PartiQL for Amazon DynamoDB

aws

Both migration and modernization are important





Thank you