logicworks

Migrating Your Oracle Database to AWS

Understanding Licensing, Costs, and Migration Strategy

www.logicworks.com

Migrating your Oracle database from your datacenter or another vendor to AWS can be daunting. Unlike an application, data can't be recreated or reinstalled, so companies often plan ahead for months to make sure migrations run smoothly.

In this guide, we'll lead you through the most common architectural patterns and processes for building a target database in AWS and de-risking migrations to the cloud. We'll provide architecture diagrams and sample templates, approved by AWS and used by thousands of companies, to help kickstart your migration planning efforts.

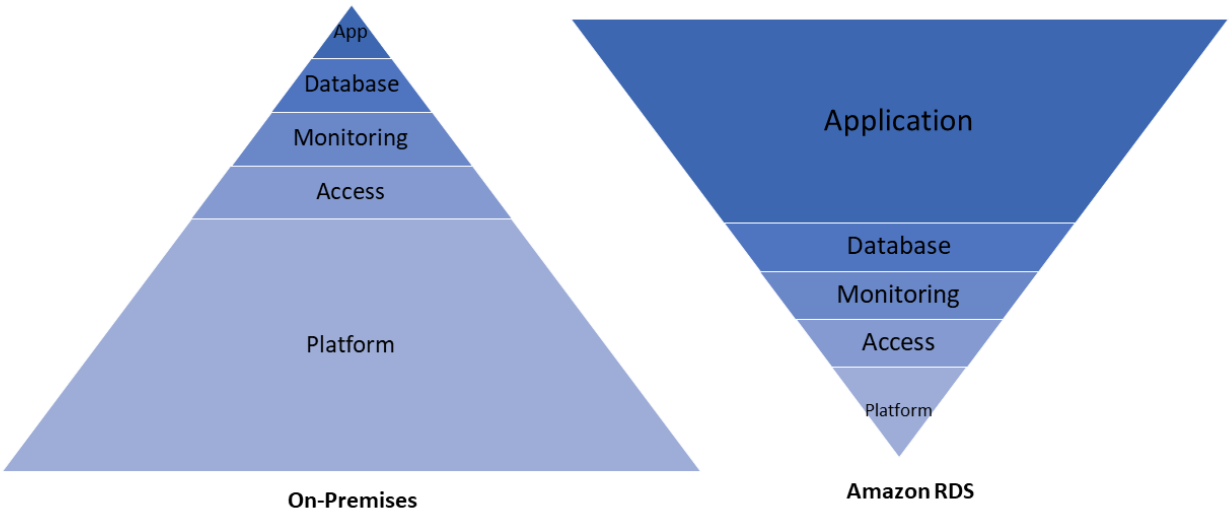
Contents

Why Cloud Databases?

The value proposition of the cloud is well-known: scalability, agility, cost efficiency, etc. But what are the specific reasons why your team, running Oracle or MySQL databases, would want to move those databases to AWS and not keep them on-premises?

Simplify Management and Save Your DBAs Time. As much as DBAs want to spend time on applications and on tuning, it turns out that the majority of time is spent in non-differentiated tasks, like patching, break/fix, and other maintenance. The goal of migrating your database to the cloud is to flip this model on its head, and let the cloud provider manage the platform so you can focus on the application.

DBA Time Spent



Cost. The top reason for companies wanting to move on-premises databases to the cloud is cost, according to a survey by the Independent Oracle Users Group and AWS (2018). However, this should come with a few caveats. We'll explore this in more depth in this Guide, but if you're migrating your Oracle database to run on an Amazon EC2 instance, without any changes, it's unlikely that you're going to see major cost savings vs. an on-premises system. The real cost savings factor of the cloud is when you move to an open source platform (i.e. MySQL, PostgreSQL) and use a cloud-native database like Amazon Aurora.

Licensing Freedom. One of the most common questions we get is how to be "free from Oracle". Part of this is cost-related, part of this is vendor lock-in, and part of this is frustration with Oracle itself.

Scalability. This is the top reported benefit that Oracle users are getting on public cloud platforms. For DBAs accustomed to spending a significant amount of time adding storage and capacity planning in general, cloud-native and fully-managed databases like Amazon RDS and Aurora allow DBAs to increase storage (with no downtime, running on elastic volumes) and compute power (with minimal downtime) on demand.

Step 1: Choose the Right Database Model

Introduction to AWS Database Models

AWS provides a variety of tools and services to help you deploy enterprise-grade relational database solutions. Let's review the three most popular models for running relational databases on AWS.

Option 1: Database on Amazon EC2

In this model, you run your database yourself on top of an Amazon instance/virtual machine, much like running it on your own servers except you don't manage the physical server, networks, or storage. There are 33 different AWS instance types (as of 2020), each optimized for different use cases (ex. storage, compute, memory).

This is the simplest model for running your database on AWS, but doesn't allow you to take full advantage of cloud-native database technologies. We usually don't recommend it unless there's a specific reason why Amazon RDS or Aurora doesn't work.

Option 2: Amazon Relational Database Service (RDS)

Amazon Relational Database Service (RDS) allows you to run your relational database in the cloud without having to worry about certain database administration tasks. Varieties include:

- Amazon RDS for Oracle
- Amazon RDS for MySQL
- Amazon RDS for SQL Server
- Amazon RDS for MariaDB
- Amazon RDS for PostgreSQL

The benefits of RDS are that it scales your database's compute or storage needs automatically, usually without downtime, and performs tasks like patching and backups automatically. RDS Multi-AZ creates a secondary RDS instance in a second AZ with synchronous replication. If volumes or backups have problems, RDS ensures that everything is replaced.

Option 3: Amazon Aurora

Amazon Aurora is a fully-managed database solution that is compatible with MySQL and PostgreSQL. A customer that's running Oracle or SQL Server on-premises requires a schema and code transformation to MySQL or PostgreSQL before database migration occurs.

Why would a company want to do this schema conversion work? Because Amazon Aurora is about 1/10th of the cost of a traditional database. You no longer have licensing costs, it automatically replicates 6 copies of your data across 3 Availability Zones and backs up your data continuously to Amazon S3, and it automatically grows storage as needed. Also performance, scalability and throughput is better in many cases.

Oracle Licensing on AWS

You spend much more on Oracle licenses than you do on any underlying infrastructure, so the impact of AWS on licensing should always be the top consideration. Each of the models listed above have different licensing models.

Model	Licensing Options
Oracle on EC2	Bring-Your-Own-License (BYOL) only
Oracle on RDS	Bring-Your-Own-License (BYOL) or License Included (SE1 or SE2)
Aurora for PostgreSQL or MySQL	No License Required

Bring-Your-Own-License (BYOL)

As the name suggests, this model allows you to run Amazon RDS using your existing Oracle Database software licenses. You can also purchase Oracle Database 11g or 12c licenses directly from Oracle and run them on Amazon RDS. You must have the appropriate Oracle Database license (with Software Update License & Support). You can create and manage license configurations in AWS License Manager.

Oracle's Cloud Licensing Policy allows licensing in EC2 and RDS, but counts two vCPUs as equivalent to one Oracle processor licensing if hyper-threading is enabled, and one if not enabled. This means that running on AWS can be more expensive than traditional processor licensing.

If you run multi-AZ, you must have a license for both the primary DB instance and the standby DB instance.

Supported under this model: Enterprise Edition, Standard Edition Two, Standard Edition, Standard Edition One. Charges do not vary by edition for BYOL Amazon RDS pricing.

License Included

In this model, available only for Amazon RDS, AWS holds the license for the Oracle software. Pricing includes software, underlying hardware resources, and Amazon RDS management capabilities. This model is almost always cheaper than purchasing an Oracle license, and involves no direct engagement with Oracle.

If you run multi-AZ, you just pay the per-hour costs for each RDS instance-hour consumed.

Just be aware that the RDS License Included model only supports Standard Edition 1 and 2, which has a smaller feature set than Enterprise Edition. That said, many companies have been able to use RDS with SE2 by replacing Enterprise Edition features with other AWS-

native features.

Supported under this model: Standard Edition One (SE1), Standard Edition Two (SE2)

Choosing the Right Licensing Model

If you've already purchased Enterprise Oracle software and support licenses, the most cost-effective option is usually to BYOL on EC2 or RDS. If you have not already purchased Oracle licenses, it is significantly cheaper to purchase instances with License Included (SE2 on RDS) rather than to purchase Oracle licenses separately. Look below for a more detailed cost breakdown.

Feature Parity: EC2 vs. RDS vs. Aurora

One of the many reasons that enterprises use Oracle are its enterprise-grade feature set and rock-solid availability. It's important to understand which features are or are not supported by each AWS service.

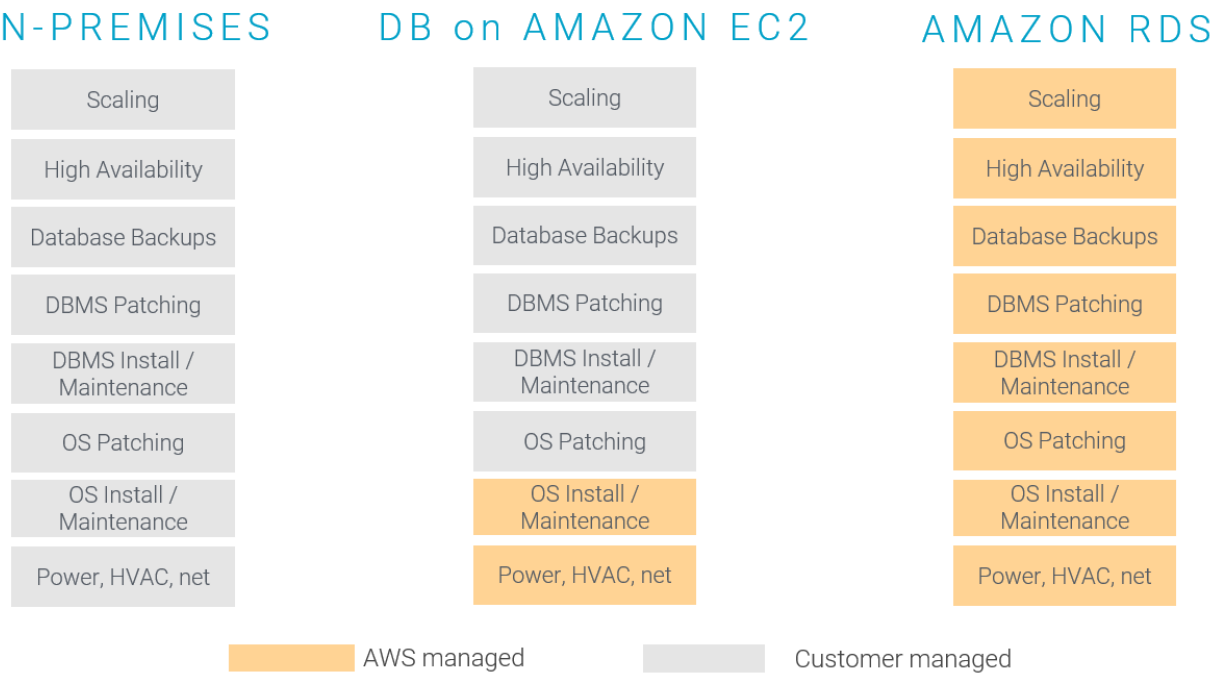
	Feature Support Link	Unsupported Features
Oracle on EC2	All Features Supported	• Oracle RAC
Oracle on RDS	See AWS Documentation	• Automatic Storage Management (ASM) • Database Vault • Flashback Database • Multitenant • Oracle Enterprise Manager Cloud Control Management Repository • Real Application Clusters (Oracle RAC) • Real Application Testing • Unified Auditing, Pure Mode
Aurora PostgreSQL	See AWS Documentation	• UTL_FILE • Exchange and Split Partitions • BITMAP Indexes • There are about 15 additional features that don't have 100% feature parity and require some refactoring.

One of the top reasons why companies tend to run their database on EC2 is that they want 100% feature parity or application does not support RDS, e.g. Oracle Ebusiness Suite. That said, we've found that RDS for Oracle works for the vast majority of use cases.

Aurora PostgreSQL usually has greater feature parity than Aurora MySQL. That said, there are a significant number of features that require a DBA to rethink and refactor the database before migration. We'll review this in detail in the following sections.

Maintenance Responsibility: EC2 vs. RDS vs. Aurora

It's no secret that the "care and feeding" of Oracle databases takes significant time, effort, and expertise. Fully-managed services like RDS and Aurora reduce the maintenance burden on your in-house IT team. In the chart below, you can see the comparison between running your database on-premises, running on EC2, and running on RDS or Aurora.



In Amazon RDS and Aurora, Amazon takes care of all common database infrastructure management tasks; you only have to take care of the data itself. When you run on EC2, you still have to worry about scaling, HA, patching, and backups.

For some teams, outsourcing these tasks to Amazon is very attractive. Other teams that want to be able to control these factors themselves often choose to run their database on EC2. This is especially common for companies that want to avoid automatic software patches that might not be compliant with their applications.

Comparing Sample Costs: EC2 vs. RDS vs. Aurora

Each of the models listed above have very different cost implications. Let's review these briefly.

Model	**Upfront Oracle Licensing Cost for Single DB with 16 vCPUs on AWS*** (includes 8 licenses of Oracle Database Enterprise Edition, Tuning Pack, Diagnostics Pack, and Partitioning)	**Oracle Support Cost - List Price** (Varies widely -- usually this is heavily negotiated)	**Annual Cost for Single DB with 16 vCPUs on AWS** (Single AZ, On Demand, us-east-1)	**1/10th of a DBA** (Annual Salary of DBA is $93,612)	**Annual Cost** (not including Upfront Cost)
Oracle on EC2 (BYOL)	$572,000	$125,840	$8,830.08	$9,361	$144,031.08
Oracle on RDS (BYOL)	$572,000	$125,840	$16.328.64	$0	$142,168.64
Oracle on RDS (License Included)	$0	$0	$33,778.56	$0	$33,778.56
Aurora for PostgreSQL or MySQL	$0	$0	$20,323.20	$0	$20,323.20

***Source:** https://www.oracle.com/assets/cloud-licensing-070579.pdf, https://www.oracle.com/assets/technology-price-list-070617.pdf

The chart demonstrates that running Oracle on EC2 is significantly more expensive than running Aurora. But three final notes on AWS costs.

1. This chart shows On Demand pricing; once you migrate to AWS, you should consider purchasing AWS Reserved Instance or Savings Plan, which reduces per-hour costs by 50-70%.
2. This chart shows the cost of a single AZ deployment. Costs will vary for a multi-AZ deployment.
3. Based on your requirements, some additional AWS charges may apply:
   - Provisioned IOPS SSD Storage (Starts at $0.125/GB per month)
   - Additional backup storage (Starts at $0.095/GB per month)
   - Egress traffic (Starts at $0.05 per GB per month)
   - Multi-AZ cost

Calculating AWS costs is quite complicated. We recommend checking out the AWS Calculator or getting a validated TCO Analysis from an AWS Partner.

A Final Note

The above comparisons reveal that RDS and Aurora are cheaper, require less ongoing maintenance, and can help you avoid ever having to interact with Oracle again. What's the catch?

The trade-off is of course upfront effort to migrate to AWS. You can lift-and-shift your Oracle database to EC2 with little to no effort (see Step 2 below). Migrating to RDS for Oracle requires some effort, but is low risk. Migrating to Aurora requires a schema conversion to MySQL or PostgreSQL before migration -- a high-risk proposition for even the most seasoned DBAs. AWS provides some tools to make this easier, but significant manual work is still required.

That said, migrating from Oracle to Aurora is possible and has been completed by thousands of companies. But many choose a two-step process: first they migrate to RDS, then they migrate to Aurora later. This can help minimize perceived organizational risk. We'll review these migration processes in depth below.

Step 2: Prepare Your On-Premises Database

No matter which AWS database destination platform you choose, the next step is to optimize your existing on-premises database before migration. This is important for a number of reasons:

Cost Efficiency. We often find that 20-40% of storage and compute power on-premises is used for non-essential resources that are no longer required by the business. If you migrate the database as-is, you will likely be overpaying for cloud. Spend a month or two before migration to carefully evaluate your usage and look for opportunities to cut waste.

AWS Service Limits. If you're migrating a large database to AWS, there are certain limits on throughput and database size in AWS that you may hit. Do your research before migration so that you're aware of these limits beforehand.

There are cases where your team may have to divide a single database into multiple smaller databases, or cases where you have to develop workarounds to ensure that your throughput or storage requirements are met on AWS. You should do the work of optimizing your database while it is still on-premises so that you don't pay AWS costs while you're optimizing.

Note Unsupported Features. A thorough evaluation of your database will allow you to note features you're currently using in Oracle that are unsupported by the AWS destination platform, and develop a workaround. We go through this adaptation process in detail below.

Evaluate Partners

During this process, you may find that your in-house team doesn't have the bandwidth or expertise to adapt your on-premises database for the cloud. This is an ideal point to reach out to a partner with expertise in cloud databases.

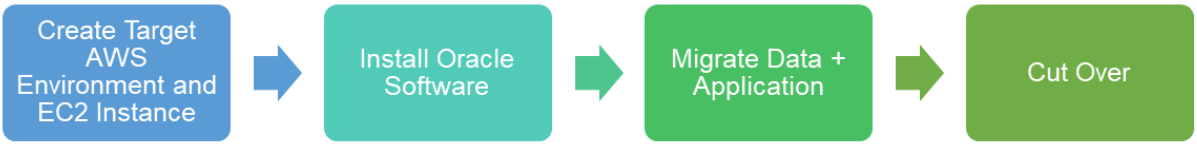
A partner (like Logicworks) can help you analyze your on-premises database and do what's required to be successful on the cloud. A partner's value is that they've done hundreds of

heterogenous and homogenous migrations, know the limits of cloud database platforms, and will help you avoid common roadblocks.

Step 3: Migrate to AWS

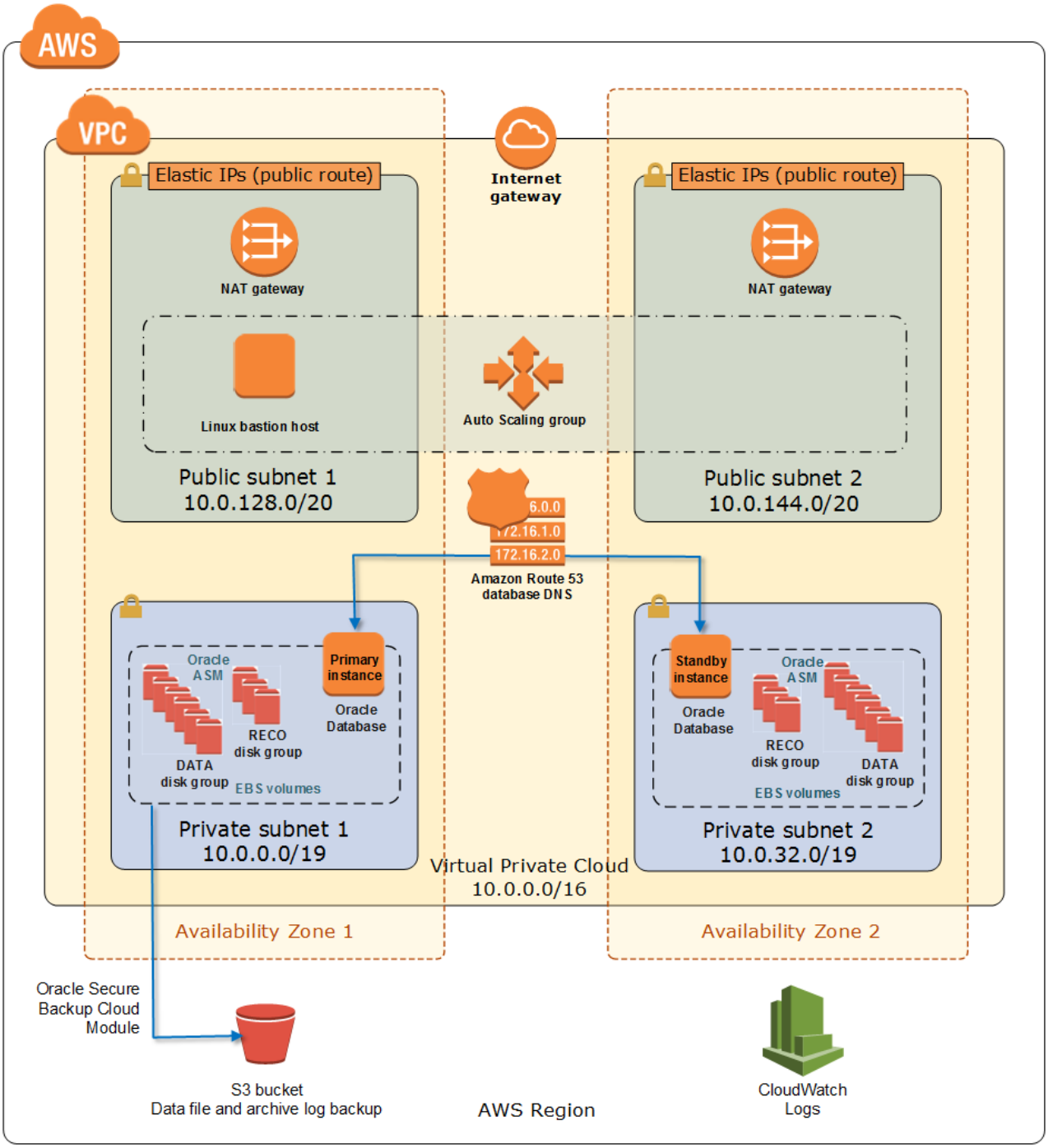
Oracle to Amazon EC2 or RDS

The process for migration from Oracle on-premises to Amazon EC2 or RDS is relatively simple. It involves designing the target database, migrating data, and cutting over to AWS.



Phase 1: Design a Target Architecture

The following diagram shows an EC2 or RDS instance in a private subnet, inside a VPC, replicated to another EC2 or RDS instance inside a 2nd AZ. Of course, a database migration is usually completed in the context of a larger migration, along with web servers and other resources.



You can launch this architecture using a free Quick Start template, available here. The Quick Start installs Oracle ASM for storage management, and Oracle Data Guard for database setup and replication. You can also include the Oracle Secure Backup (OSB) Cloud Module for backups.

As part of this process, you will have to download your Oracle Database software into an AWS S3 bucket.

A note about RDS Multi-AZ. It is recommended to enable Multi-AZ Deployments, where Amazon RDS automatically creates a standby instance in a different AZ with synchronous replication. In the case of infrastructure failure, Amazon RDS performs an automatic failover to the standby without the need for manual administrative intervention. You could also launch an Amazon RDS Read Replica, which can be within the same AZ or cross-AZ, and can be manually promoted to a standalone database instance.

Phase 2: Migrate Data

The process of migrating your data to AWS usually occurs in two phases: first an initial dump of your database, then ongoing (or one-time) replication between the cloud and on-premises database for any new changes/additions to the database.

Options for Initial Data Migration

Over the Internet

The advantages of this method are that it's essentially free and requires little custom configuration. The disadvantage is that it's only possible for small databases or long deadlines.

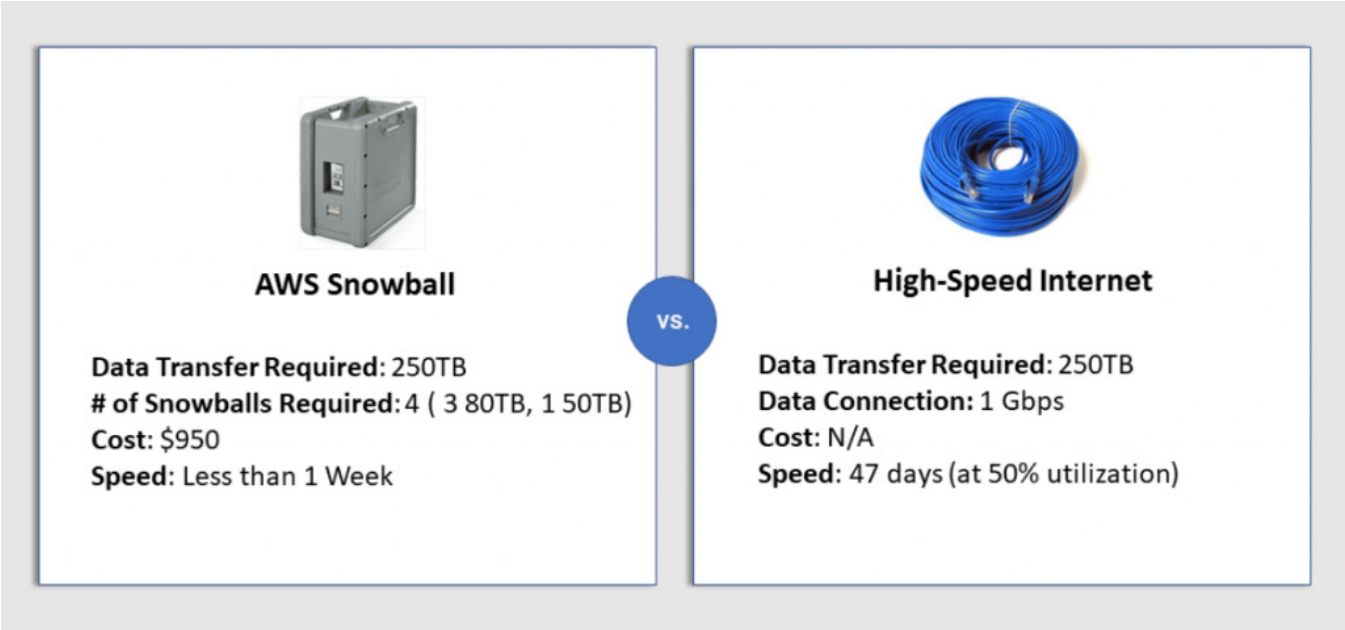
AWS Snowball (Logicworks Recommended)

AWS Snowball is a physical device with either 50 or 80TB of storage space that is designed to transfer large amounts of data in and out of the AWS cloud. You don't have to write code to transfer data. You simply have to create a job in AWS Management Console, and a Snowball device will automatically be shipped out to you. Once received, connect the Snowball to your local network and follow the instructions to select which files you want to transfer.

The biggest benefit of using AWS Snowball is speed. It could take months to transfer 50TB of data even with high-speed internet connections, and transferring that much data over the internet is often not feasible. Using Snowball, you can transfer 50TB onto a Snowball in less than a day (depending on your local environment) and have the data on AWS within a week, including shipping times.

The 50TB Snowball costs $200 per use, and you can keep it for up to 10 days without overage charges. The 80TB Snowball costs $250 per use.

Let's say you want to transfer 250TB to AWS. Here's an estimate of what it would cost:



As a pricing example, to transfer 250 TB using Snowball would only cost a $950 service fee. Amazon claims that this is up to 1/5 of the cost of backing up over the internet. Transferring data into Amazon S3 is free for all users.

AWS Direct Connect

AWS Direct Connect is a dedicated network connection from your datacenter to AWS. Due to its high cost, you should only invest in Direct Connect if you require ongoing replication and connectivity between AWS and your datacenter. If you're making a one-time move to AWS, building a Direct Connect is a waste.

Options for Asynchronous Replication

AWS Database Migration Service

AWS Database Migration Service is a full-featured service that helps you perform homogenous or heterogenous migrations while continuously replicating data using Amazon Redshift and S3.

Read more about how to use AWS Database Migration Service for an Oracle to Oracle migration in AWS' documentation.

Oracle Recovery Manager (RMAN)

RMAN is an Oracle tool that allows full backups plus incremental backups. It can also duplicate a database from an active database. Only works when EC2 is the target. Not compatible with RDS.

Oracle GoldenGate

GoldenGate replicates data between a source and destination database. While it is often used to maintain High-Availability, it can also be used to maintain **asynchronous** replication.

Oracle Data Pump

Oracle Data Pump is a tool that can be used in combination with asynchronous replication tools. If network connectivity is established between AWS and the source database network, Data Pump can be used over the network using database link to copy data or generate dump files and copy the files to AWS.

For a full list of data replication technologies, download AWS' whitepaper on Oracle migration strategies.

Migration Process

Your migration process will vary depending on how much downtime is acceptable for your application:

Minimal Downtime (Logicworks Recommended)

Note: For small databases (less than 100GB) migrating to EC2, you can use Oracle Data Pump to complete the full migration in a short amount of time (less than 2 hours, usually), and it may not require a migration with point-in-time + asynchronous replication.

1. Data is extracted from the source database at point in time.
2. Data is migrated to the destination database.
3. Asynchronous replication is set up between source and destination database (i.e. Oracle GoldenGate, AWS Database Migration Service) or use Oracle Data Guard to setup standby database on EC2 instance
4. Cut over happens at any time because source and destination are always in sync.

Use case: Mission-critical databases that require a few minutes or no downtime.

Several Hours of Downtime

1. Data is extracted from the source database at a point in time.
2. Data is migrated to the destination database. Since the source database is still live, this can happen over a longer period.
3. Data is validated in the destination database and connectivity is tested.
4. Database is shut down during non-peak hours. Any data changed in the source database is propagated to the destination database. (Usually this is minimal compared with the total size of the database, and therefore requires less time.)
5. Destination database is validated.
6. Cut over.

Use case: This is the most common data migration model. Applicable to most use cases.

Maximum Downtime

1. Database is shut down.
2. Data is extracted from the source database and migrated to the destination database.
3. Destination database is validated.
4. Cut over.

Use case: Small, non-production databases.

Oracle to Aurora

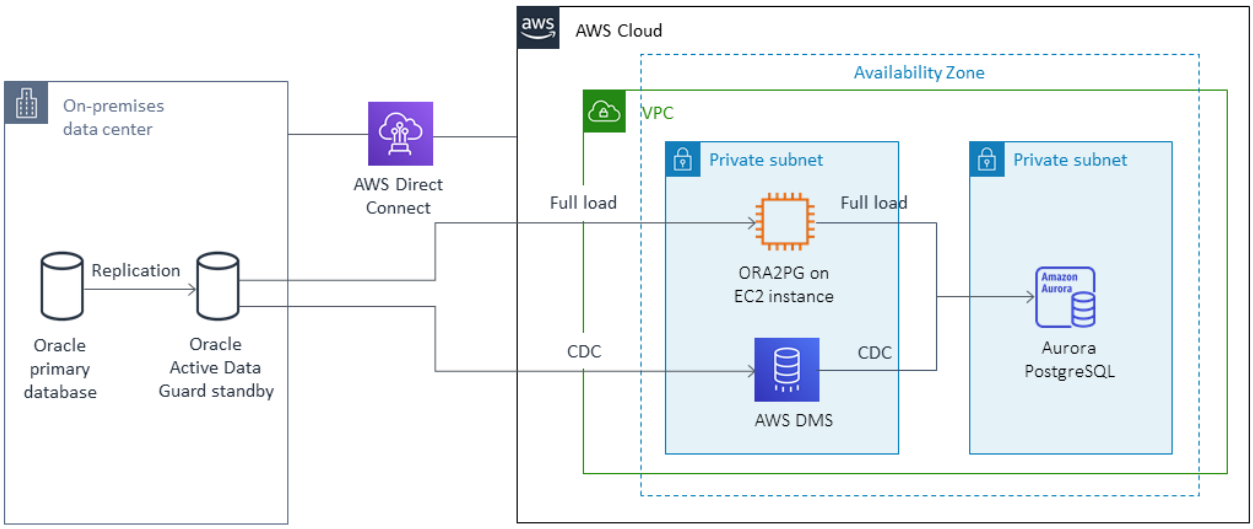
In order to break free from Oracle licensing and other limitations, many companies choose to migrate to Amazon Aurora. This is a significantly more involved and resource-intensive migration process.

The process of migrating data to Aurora is very similar to the process of migrating data to EC2 or RDS. What's different is the upfront work of configuring your Oracle source database and converting schema to MySQL or PostgreSQL. For the purposes of this guide, we'll focus on Aurora PostgreSQL, since this is the more common migration path.

Option 1: Using Ora2Pg to migrate to Aurora PostgreSQL

Ora2Pg is a free tool used to migrate an Oracle or MySQL database to a PostgreSQL compatible schema. It connects your Oracle database, scans it automatically and extracts its structure or data, it then generates SQL scripts that you can load into your PostgreSQL database.

It allows you to export a full database schema (tables, views, sequences, indexes). It reads Oracle's catalog and creates equivalent PostgreSQL objects. It's ideal for small-medium databases. It extracts the database in a SQL file format which can be used to load data into PostgreSQL. It does have some limitations and may be useful where downtime is acceptable for the source system.



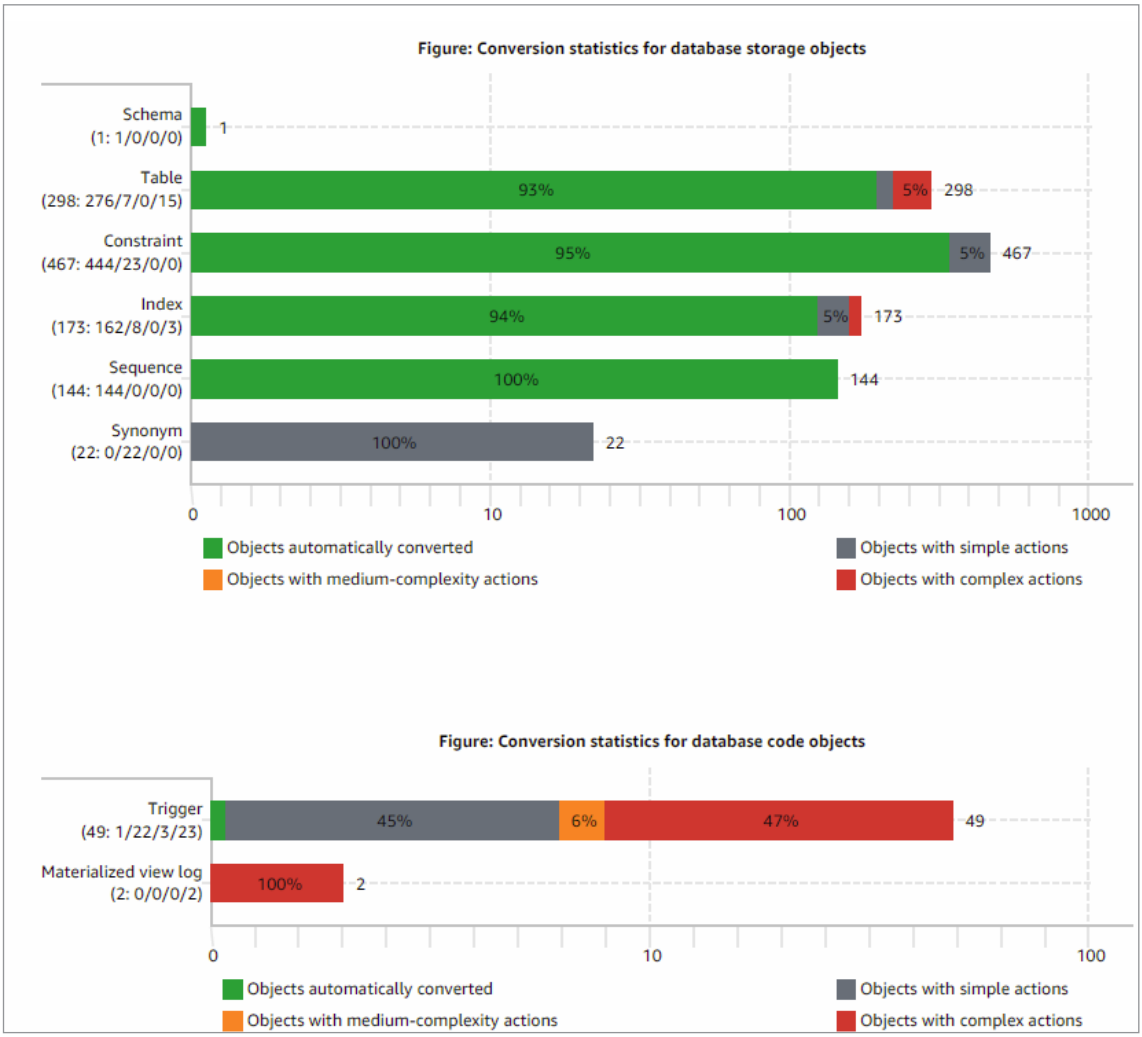
This complex process is well-documented by both AWS, Ora2Pg, and several 3rd parties. If you choose to pursue this option, first read applicable documentation:

- AWS Prescriptive Guidance: On-Prem to Aurora PostgreSQL
- Ora2Pg Documentation

After you complete schema conversion, then you migrate data to a target Aurora PostgreSQL database using any of the data tools described on page X above.

Option 2: Using AWS Schema Conversion Tool (AWS SCT) and Database Migration Service (AWS DMS)

AWS Schema Conversion Tool is a free cloud-native tool that, like Ora2Pg, converts functions, procedures, etc. and marks non-convertible code. The additional benefit of AWS SCT is that it offers a robust Migration Assessment Report, which scans your on-premises database to highlight the potential roadblocks and issues before migration.



Sample AWS SCT Assessment Report. Source: AWS

This complex process is well-documented by both AWS and Ora2Pg. If you choose to pursue this option, first read applicable documentation:

- AWS Whitepaper: Automatic Migration of Oracle Schema Objects Using the AWS Schema Conversion Tool
  - This 317-page guide is a must for any DBA during the migration process, and goes through each potential incompatible object with workarounds.
- AWS Prescriptive Guidance: On-Prem to Aurora PostgreSQL

After you complete schema conversion, then you migrate data to a target Aurora PostgreSQL database using any of the data tools described on page X above.

Phased Migrations to Reduce Risk

While the migration options outlined above assume a single push to AWS, we often find that companies that want to move from Oracle on-premises to Aurora do so in longer phases. A company may choose to migrate first to Oracle on EC2, or Oracle RDS, before they take the final plunge into Aurora.

There are a number of good reasons to pursue this path. First, most IT teams performing the migration are new to the cloud, and it allows teams to use Oracle in a familiar environment. Your team can get familiar with AWS technology before going all-in with an AWS-native solution. You can also the value of AWS quickly without the delay of a schema conversion project.

Summary

Many DBAs find the prospect of migrating their database to AWS to be daunting. They must learn a new set of tools, and in the case of migrating to Aurora, an entirely new database platform. However, as outlined in this Guide, the benefit for both DBAs' workloads and for your budget can be tremendous. Hopefully this guide has helped you choose the migration options that work best for your team.

If your team is new to AWS and would like to rely on a partner for strategic or engineering support, reach out to Logicworks. Our team of DBAs can tackle any type of AWS migration project, reduce time-to-migration, and de-risk schema conversions for your team.

About Logicworks

Logicworks is a cloud consulting and managed services company that helps organizations plan, architect, and manage complex cloud environments. Our team of cloud experts have helped 400+ organizations migrate to AWS with our unique approach to cloud strategy and design, including MassMutual, Major League Soccer, and Choice Hotels.

As an AWS Premier Consulting Partner with HIPAA, HITRUST, PCI, ISO 27001, SOC1, and SOC2 certifications, Logicworks specializes in complex workloads for companies with high security and compliance requirements.

If you're planning new cloud projects and want expert help in avoiding common migration stumbling blocks, visit www.logicworks.com or contact us at (212) 625-5300.

aws partner network

Premier
Consulting Partner

- Security Competency
- Migration Competency
- Healthcare Competency
- DevOps Competency
- MSP Partner

HITRUST CSF Certified

HIPAA COMPLIANCE

PCI

ISO 27001

AICPA SOC 1 — AICPA Service Organization Control Reports — Formerly SAS 70 Reports

AICPA SOC 2 — AICPA Service Organization Control Reports — Formerly SAS 70 Reports