AWS Decision Guide

Choosing an AWS database service

Choosing an AWS database service: AWS Decision Guide

Table of Contents

Choosing an AWS database service

Taking the first step

Purpose	Help determine which AWS database or databases are the best fit for your organizat ion.
Last updated	May 13, 2024
Covered services	• Amazon Aurora • Amazon DocumentDB • Amazon DynamoDB • Amazon ElastiCache • Amazon Keyspaces • Amazon MemoryDB for Redis • Amazon Neptune • Amazon QLDB • Amazon RDS • Amazon Timestream

Introduction

Amazon Web Services (AWS) offers a growing number of database options (currently more than 15) to support diverse data models. These include relational, key-value, document, in-memory, graph, time-series, wide column, and ledger databases.

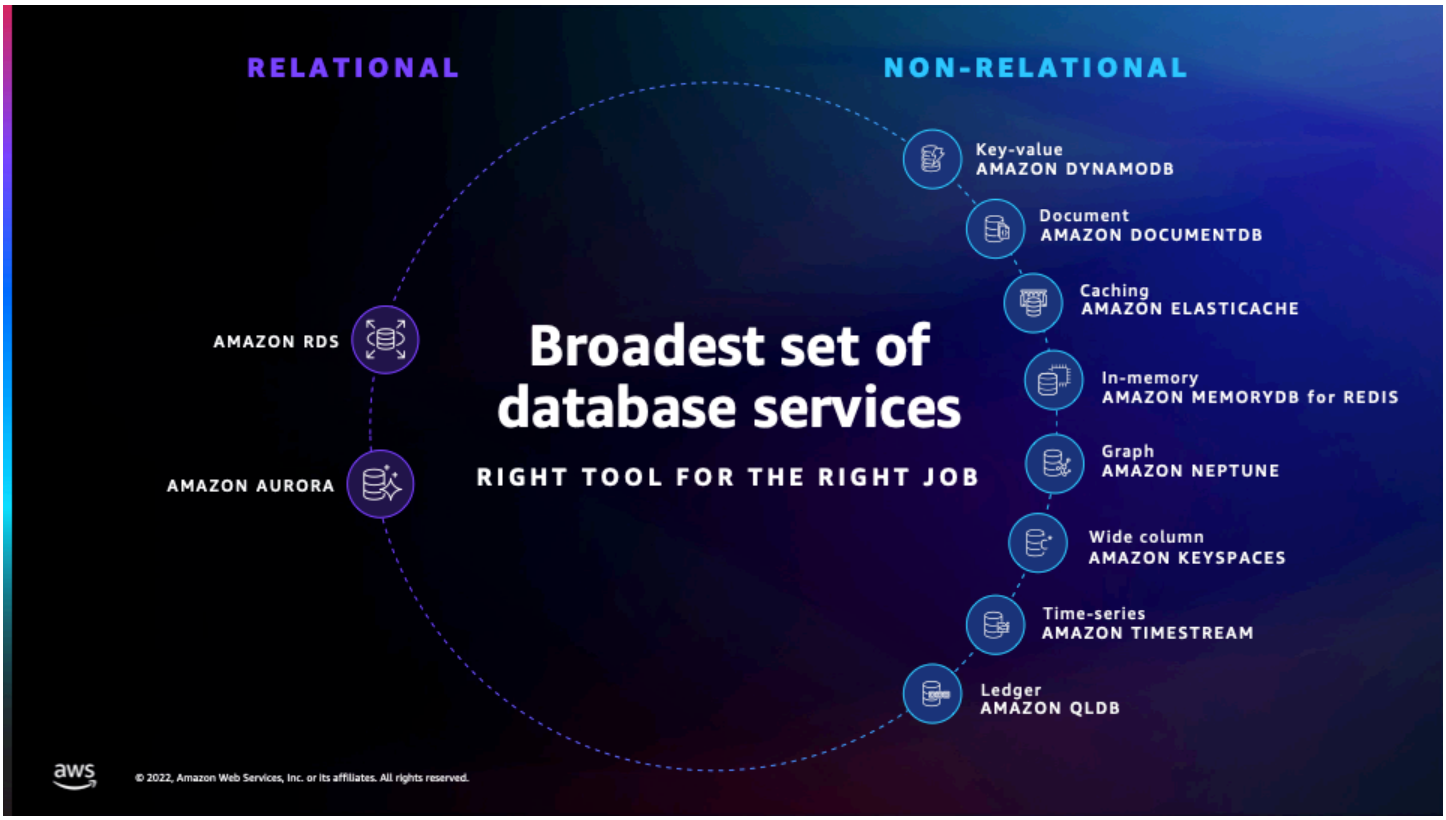
Choosing the right database or multiple databases requires you to make a series of decisions based on your organizational needs. This decision guide will help you ask the right questions, provide a clear path for implementation, and help you migrate from your existing database.

This six and a half minute video explains the basics of choosing an AWS database.

Understand

Databases are important backend systems used to store data for any type of app, whether it's a small mobile app or an enterprise app with internet-scale and real-time requirements.

This decision guide is designed to help you understand the range of choices available to you, establish the criteria that make sense for you to make your database choice, provide you with detailed information on the unique properties of each database—and then allow you to dive deeper into the capabilities that each offers.



What kinds of apps do people build using AWS databases?

- **Internet-scale apps:** These apps can handle millions of requests per second over hundreds of terabytes of data. They automatically scale vertically and horizontally to accommodate your spiky workloads.
- **Real-time apps:** Real-time apps such as caching, session stores, gaming leaderboards, ride-hailing, ad-targeting, and real-time analytics need microsecond latency and high throughput to support millions of requests per second.
- **Enterprise apps:** Enterprise apps manage core business processes, such as sales, billing, customer service, human resources, and line-of-business processes, such as a reservation system at a hotel

chain or a risk-management system at an insurance company. These apps need databases that are fast, scalable, secure, available, and reliable.

- **Generative AI apps:** Your data is the key to moving from generic applications to generative AI applications that create differentiating value for your customers and their business. Often, this differentiating data is stored in operational databases powering your applications.

> **Note**
>
> This guide focuses on databases suitable for Online Transaction Processing (OLTP) applications. If you need to store and analyse massive amounts of data quickly and efficiently (typically met by an online analytical processing (OLAP) application), AWS offers Amazon Redshift, a fully managed, cloud-based data warehousing service that is designed to handle large-scale analytics workloads.

There are two high-level categories of AWS OLTP databases—relational and non-relational.

- The AWS relational database family includes eight popular engines for Amazon RDS and Amazon Aurora. The Amazon Aurora engines include Amazon Aurora with MySQL compatibility, and Amazon Aurora with PostgreSQL compatibility. The other RDS engines include Db2, MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server. AWS also offers deployment options such as Amazon RDS Custom and Amazon RDS on Outposts.
- The non-relational database options are designed for specific data models including key-value, document, caching, in-memory, graph, time series, wide column, and ledger data models.

We explore all of these in detail in the Choose section of this guide.

Database migration

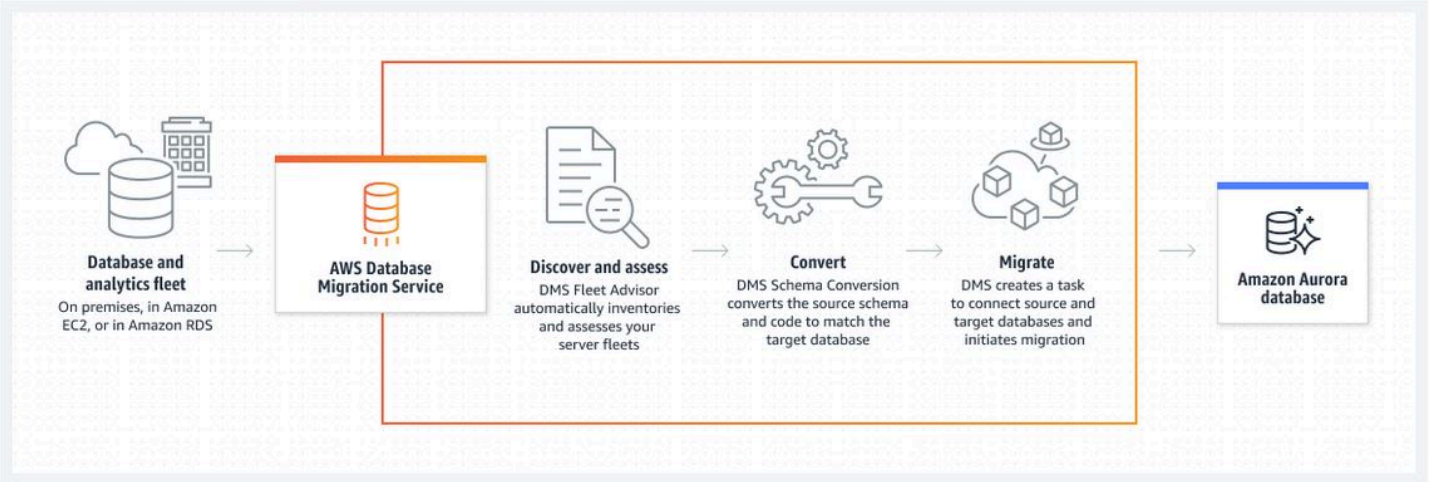
Before deciding which database service you want to use to work with your data, you should spend time thinking about your business objective, database selection, and how you are going to migrate your existing databases.

The best database migration strategy helps you take full advantage of the AWS Cloud. This might involve migrating your applications to use purpose-built cloud databases. You might just want the benefit of using a fully managed version of your existing database, such as RDS for PostgreSQL or RDS for MySQL. Alternatively, you might want to migrate from your commercial

licensed databases, such as Oracle or SQL Server, to Amazon Aurora. Consider modernizing your applications and choosing the databases that best suit your applications' workflow requirements.

For example, if you choose to first transition your applications and then transform them, you might decide to re-platform (which makes no changes to the application you use, but lets you take advantage of a fully managed service in the cloud). When you are fully in the AWS Cloud, you can start working to modernize your application. This strategy can help you exit your current on-premises environment quickly, and then focus on modernization.

The following image shows how the AWS Database Migration Service is used to move data to Amazon Aurora. For resources to help with your migration strategy, see the Explore section.



Consider

You're considering hosting a database on AWS. This might be to support a greenfield/pilot project as a first step in your cloud migration journey, or you might want to migrate an existing workload with as little disruption as possible. Or perhaps you would like to port your workload to managed AWS services or even refactor it to be fully cloud-native.

Whatever your goal, considering the right questions will make your database decision easier. Here's a summary of the key criteria to consider.

Business objective

> The first major consideration when choosing your database is your business objective. What is the strategic direction driving your organization to change? As suggested in the 7 Rs of AWS, consider whether you want to rehost an existing workload, or refactor to a new platform to shed commercial license commitments.

Migration strategy

You can choose a rehosting strategy to deploy to the cloud faster, with fewer data migration headaches. Install your database engine software on Amazon EC2, migrate your data, and manage your database much as you do on-premises. While rehosting is a fast path to the cloud, you are still left with the operational tasks such as upgrades, patches, backups, capacity planning and management, maintaining performance, and availability targets.

Alternatively, you can choose a re-platform strategy where you migrate your on-premises relational database to a fully managed Amazon RDS instance.

You may consider this an opportunity to refactor your workload to be cloud-native, making use of Amazon Aurora or purpose-built NoSQL databases such as Amazon DynamoDB, Amazon Neptune, or Amazon DocumentDB.

Finally, AWS offers serverless databases, which can scale to an application's demands with a pay-for-use pricing model and built-in high availability. Serverless databases increase your agility and optimize costs. In addition to not needing to provision, patch, or manage servers, many AWS serverless databases provide zero downtime maintenance.

AWS serverless offerings include Amazon Aurora Serverless, Amazon DynamoDB, Amazon ElastiCache, Amazon Keyspaces, Amazon Timestream and Amazon Neptune serverless, the graph database.

Data considerations

The core of any database choice includes the characteristics of the data that you need to store, retrieve, analyze, and work with. This includes your data model (is it relational, structured or semi-structured, using a highly connected dataset, or time-series?), data access (how do you need to access your data?), the extent to which you need real-time data, and whether there is a particular data record size you have in mind.

Operational considerations

Your primary operational considerations are all about where your data is going to live and how it will be managed. The two key choices you need to make are:

- **Whether it will be self-hosted or fully managed**: The core question here is where is your team going to provide the most value to the business? If your database is self-hosted, you will be responsible for the day-to-day maintenance, monitoring and patching of the database. Choosing a fully managed AWS database simplifies your work by removing undifferentiated database management tasks, allowing your team to focus on delivering value such as schema

design, query construction, query optimization, and also responsible for supporting the development of applications that align with your business objectives.

- **Whether you need a serverless or provisioned database**: DynamoDB, Amazon Keyspaces, Timestream, ElastiCache, Neptune, and Aurora provide models for how to think about this choice. Amazon Aurora Serverless v2, for example, is suitable for demanding, highly variable workloads. For example, your database usage might be heavy for a short period of time, followed by long periods of light activity or no activity at all.

Resiliency considerations

Database resiliency is key for any business. Achieving it means paying attention to a number of key factors, including capabilities for backup and restore, replication, failover, and point-in-time recovery (PITR).

Performance considerations

Consider whether your database will need to support a high concurrency of transactions (10,000 or more) and whether it needs to be deployed in multiple geographic regions.

If your workload requires extremely high read performance with a response time measured in microseconds rather than single-digit milliseconds, you might want to consider using in-memory caching solutions such as Amazon ElastiCache alongside your database, or a database that supports in-memory data access such as MemoryDB.

Security considerations

Security is a shared responsibility between AWS and you. The AWS shared responsibility model describes this as security of the cloud managed by AWS, and security in the cloud managed by the customer. Specific security considerations include data protection at all levels of your data, authentication, compliance, data security, storage of sensitive data and support for auditing requirements.

Choose

Now that you know the criteria by which you are evaluating your database options, you are ready to choose which AWS database services might be a good fit for your organizational requirements.

This table highlights the type of data each database is optimized to handle. Use it to help determine the database that is the best fit for your use case.

Database families	When would you use it?	What is it optimized for?	Related database engines or services
Relational	Use when you're migrating or modernizing an on-premises relational workload or if your workload has less predictable query patterns.	Optimized for structured data stored in tables, rows, and columns. They support complex queries through joins.	Amazon Aurora Amazon RDS
Key-value	Use for workloads such as session stores or shopping carts. Key-value databases can scale to large amounts of data and extremely high throughput of requests, while servicing millions of simultaneous users through distribut ed processing and storage.	Optimized for consistent single-digit millisecond performance at any scale (meaning any number of writes and reads).	Amazon DynamoDB
In-memory	Use Amazon ElastiCac he when you need a caching layer to improve read performance. Use Amazon MemoryDB for Redis when you need full data persistence, but still	ElastiCache is optimized to support microsecond reads and sub-millisecond writes. MemoryDB supports microseco nds reads and single-digit milliseconds writes. ElastiCache is an ephemeral cache	Amazon ElastiCache Amazon MemoryDB for Redis

Database families	When would you use it?	What is it optimized for?	Related database engines or services
	need sub-millisecond read latencies.	while MemoryDB is an in-memory database.	
Document	Use when you want to store JSON-like documents with rich querying abilities across the fields of the documents.	Optimized for storing semi-structured data as documents with multi-layered attributes.	Amazon DocumentDB (with MongoDB compatibility)
Wide column	Use when you need to migrate your on-premises Cassandra workloads, or when you need to process data at high speeds for applications that require single-digit-millisecond latency.	Optimized for workloads that require heavy reads/writes and high throughput coupled with low latency and linear scalability.	Amazon Keyspaces (for Apache Cassandra)
Graph	Use when you have to model complex networks of objects, such as social networks, fraud detection and recommendation engine use cases.	Optimized for traversing and evaluating large numbers of relationships, and identifying patterns with minimal latency.	Amazon Neptune

Database families	When would you use it?	What is it optimized for?	Related database engines or services
Time series	Use when you have a large amount of time series data, potentially from a number of sources, such as Internet of Things (IoT) data, application metrics, and asset tracking.	Optimized for storing and querying data that is associated with timestamps and trend lines.	Amazon Timestream
Ledger	Use when your organization has to communicate with other entities (businesses, customers) and you need a way to verify and trust each other, or when you not only need to retrieve the current state of data, but need to prove how data mutated into the current state.	Optimized for maintaining a complete, immutable, and verifiable history of database changes.	Amazon Quantum Ledger Database (Amazon QLDB)

Use

Just as there is no single database that can satisfy all possible use cases effectively at the same time, any particular database type discussed above, may not satisfy all your requirements perfectly.

Consider your needs and workload requirements carefully and prioritize based on the considerations covered above, the requirements you must meet to the highest standard, the ones you might have some flexibility on, or even the ones you can do without. This can form a system of

values, that will help you make effective trade-offs, that lead to the best possible outcome for your unique circumstances.

Also consider that, usually, you will be able to cover your application requirements with a mix of best-fit databases. Building a solution with multiple database types allows you to lean on each, for the strengths it provides.

For example, in an e-commerce use case, you may use DocumentDB (for product catalogs and user profiles), leaning on the flexibility provided by semi-structured data (but also the low, predictable latency afforded by DynamoDB, when your users are browsing your product catalog). You may also use Aurora for inventory and order processing, where a relational data model and transaction support may be more valuable to you.

To help you learn more about each of the available AWS database services, we have provided a pathway to explore how each of the services work. The following section provides links to in-depth documentation, hands-on tutorials, and resources to get you started.

Amazon Aurora

Getting started with Amazon Aurora

This guide includes tutorials and covers more advanced Aurora concepts and procedures, such as the different kinds of endpoints and how to scale Aurora clusters up and down.

Explore the guide

Create a highly available database

Learn how to configure an Amazon Aurora cluster to create a highly available database. This database consists of compute nodes that are replicated across multiple Availabil ity Zones to provide increased read scalabili ty and failover protection.

Get started with the tutorial

Use Amazon Aurora global databases

We help you get started using Aurora global databases. This guide outlines the supported engines and AWS Region availability for Aurora global databases with Aurora MySQL and Aurora PostgreSQL.

Explore the guide

Amazon RDS

Getting started with Amazon RDS

We explain how to create and connect to a DB instance using Amazon RDS. You learn to create a DB instance that uses DB2, MariaDB, MySQL, Microsoft SQL Server, Oracle, or PostgreSQL.

Explore the guide

Create and Connect to a PostgreSQL Database

We show you how to create an environme nt to run your PostgreSQL database (we call this environment an instance), connect to the database, and delete the DB instance.

Get started with the tutorial

Create a web server and an Amazon RDS DB instance

Learn how to install an Apache web server with PHP and create a MySQL database. The web server runs on an Amazon EC2 instance using Amazon Linux, and the MySQL database is a MySQL DB instance.

Get started using the tutorial

Amazon DocumentDB

Getting started with Amazon DocumentDB

We help you get started using Amazon DocumentDB in just seven steps. This guide uses AWS Cloud9 to connect and query your cluster using the MongoDB shell directly from the AWS Management Console.

Explore the guide

Setting up a document database with Amazon DocumentDB

This tutorial helps you get started connectin g to your Amazon DocumentDB cluster from your AWS Cloud9 environment with a MongoDB shell and run a few queries.

Get started with the tutorial

Best practices for working with Amazon DocumentDB

Learn best practices for working with Amazon DocumentDB (with MongoDB compatibility), along with the basic operational guidelines when working with it.

Explore the guide

Migrate from MongoDB to Amazon DocumentDB

Learn how to migrate an existing self-mana ged MongoDB database to a fully managed database on Amazon DocumentDB (with MongoDB compatibility).

Get started with the tutorial

Assessing MongoDB compatibility

Use the Amazon DocumentDB compatibility tool to help you assess the compatibility of a MongoDB application by using the applicati

on's source code or MongoDB server profile logs.

Use the tool

Amazon DynamoDB

What is Amazon DynamoDB?

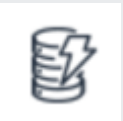
This guide explains how this fully managed NoSQL database service lets you offload the administrative burdens of operating and scaling a distributed database (including hardware provisioning, setup and configura tion, replication, software patching, and cluster scaling).

Explore the guide

Getting started with DynamoDB and the AWS SDKs

We help you get started with Amazon DynamoDB and the AWS SDKs. This guide includes hands-on tutorials that show you how to run code examples in DynamoDB.

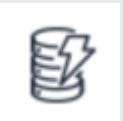
Explore the guide



Create and Query a NoSQL Table with Amazon DynamoDB

Use these hands-on tutorials to get started with Amazon DynamoDB and the AWS SDKs. You can run the code examples on either the downloadable version of DynamoDB or the DynamoDB web service.

Get started with the tutorials



Create an Amazon DynamoDB table

We show you how to create a DynamoDB table and use the table to store and retrieve data. This tutorial uses an online bookstore application as a guiding example.

Get started with the tutorial

Amazon ElastiCache

Documentation for Amazon ElastiCache

Explore the full set of Amazon ElastiCache documentation, including user guides for ElastiCache for Redis and ElastiCache for Memcached, as well as specific AWS CLI and API references.

Explore the guide

Getting started with Amazon ElastiCache for Redis

Learn how to create, grant access to, connect to, and delete a Redis (cluster mode disabled) cluster using the Amazon ElastiCac he console.

Explore the guide

Build a fast session store for an online application

Learn how to use Amazon ElastiCache for Redis as a distributed cache for session management. You will also learn the best practices for configuring your ElastiCache nodes and how to handle the sessions from your application.

Get started with the tutorial

Setting up a Redis Cluster for scalability and high availability

Learn how to create and configure a Redis Cluster with ElastiCache for Redis version 7.0 with TLS-encryption enabled. With cluster mode enabled, your Redis Cluster gains enhanced scalability and high availability.

Get started with the tutorial

Amazon MemoryDB for Redis

Getting started using Amazon MemoryDB

Getting started with Amazon MemoryDB for Redis

We guide you through the steps to create, grant access to, connect to, and delete a MemoryDB cluster using the MemoryDB Management Console.

Use the guide

Learn how to simplify your architecture and use MemoryDB as a single, primary database instead of using a low-latency cache in front of a durable database.

Read the blog

Integrating Amazon MemoryDB for Redis with Java-based AWS Lambda

We discuss some of the common use cases for the data store, Amazon MemoryDB for Redis, which is built to provide durability and faster reads and writes.

Read the blog

Amazon Keyspaces

Getting started with Amazon Keyspaces (for Apache Cassandra)

This guide is for those who are new to Apache Cassandra and Amazon Keyspaces (for Apache Cassandra). It walks you through installing all the programs and drivers that you need to successfully use Amazon Keyspaces (for Apache Cassandra).

Beginner course on using Amazon Keyspaces (for Apache Cassandra)

Learn the benefits, typical use cases, and technical concepts of Amazon Keyspaces (for Apache Cassandra). You can try the service through the sample code provided or the interactive tool in the AWS Management Console.

Explore the guide

Take the course (requires sign-in)

Amazon Neptune

Getting started with Amazon Neptune

We help you get started using Amazon Neptune, a fully managed graph database service. This guide shows you how to create a Neptune database.

Explore the guide

Build a fraud detection service using Amazon Neptune

We walk you through the steps to create a Neptune database, design your data model, and use the database in your application.

Explore the guide

Build a recommendation engine with Amazon Neptune

We show you how to build a friend recommendation engine for a multiplayer game application using Amazon Neptune.

Explore the guide

Amazon Timestream

Getting started with Amazon Timestream

Best practices with Amazon Timestream

We help you get started with Amazon Timestream. This guide provides instructi ons for setting up a fully functional sample application.

Explore the guide

We explore best practices, including those relating to data modeling, security, configura tion, data ingestion, queries, client applicati ons and supported integrations.

Explore the guide

Accessing Amazon Timestream using AWS SDKs

Learn how to access Amazon Timestream using the AWS SDKs in the language of your choice: Java, Go, Python, Node.js, or .NET.

Explore the guide

Amazon QLDB

Getting started with Amazon QLDB

In Amazon Quantum Ledger Database (Amazon QLDB), the journal is the core of the database. This guide provides a high-level overview of Amazon QLDB service components and how they interact.

Explore the guide

Creating your first Amazon QLDB ledger

We guide you through the steps to create your first Amazon QLDB sample ledger and populate it with tables and sample data.

Get started with the tutorial

Using an Amazon QLDB driver with an AWS SDK

Learn how to use the Amazon QLDB driver with an AWS SDK to create a QLDB ledger and populate it with sample data. The driver lets your application interact with QLDB using the transactional data API.

Get started with the tutorial

Explore

For your role

Developers

Solution architects

Professional development

Startups

Decision makers

Migration strategy

Getting Started with AWS Database Migration Service

Using the AWS Schema Conversion Tool

Selecting the Right Database and Database Migration Plan for your Workloads

Architecture diagrams

Explore reference architecture diagrams to help you develop, scale, and test your databases on AWS.

Whitepapers

Explore whitepapers to help you get started, learn best practices, and migrate your databases.

AWS Solutions

Explore vetted solutions and architectural guidance for common use cases for databases.

Explore architecture diagrams	Explore whitepapers	Explore solutions

Document history

The following table describes the important changes to this decision guide. For notifications about updates to this guide, you can subscribe to an RSS feed.

Change	Description	Date
Guide updated	Updated Amazon RDS description to include Db2 support, and provided additional links to service-specific documentation.	May 13, 2024
Initial publication	Guide first published.	September 11, 2023