aws | Deloitte.

Accelerating your Database Modernization journey with Deloitte on AWS

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract and Introduction

Abstract

Enterprises and organizations of all sizes are seeing an incredible rate of data growth, data volume and the variety of data that must be stored and acted upon. There is also increasing diversity of data models. The days of having a single database that can appropriately manage all your database needs is less common. Modern architectures in the cloud require having the right tool for the right job, demonstrating the need to leverage purpose-built databases.

This whitepaper explains how purpose-built databases provided by AWS can be leveraged for different application workloads, data shapes and structures, performance requirements, and operational efficiencies. In addition, this whitepaper provides highlights on the history of databases and how applications and databases have evolved over the years. Examples of modern architectures that are made possible by purpose-built database offerings from AWS are reviewed. Concluding with examples of real-world implementations of purpose-built databases from a few of Deloitte's clients. The goal is to provide a strategy on how to modernize their applications and data workloads using AWS purpose-built database services.

Introduction

Many organizations are looking to modernize their application footprint. Data is a part of everyday life in the digital age, with smart phones, IoT (Internet of Things) devices, and digital home appliances generating enormous amounts of data in addition to the large volumes produced by traditional systems. Modern digital organization have needs to ingest, store, and process vast volumes of data to provide useful information and develop insights that can fuel growth and expansion. Databases play a key role in storing and operationalizing a sizable portion of this data.

Organizations want to build applications that are agile without limitations on performance and scaling. Organizations also expect that database services are are self-healing and have reduced day-to-day database management to allow them to focus on solving their business problems. We also see applications that require sub-second latency, and in many cases, even sub-millisecond latencies. The database technologies that underpin this massive expansion have also been evolving to meet the needs of large-scale data processing and storage, as well as building self-aware databases that are purpose built to address modern business challenges.

When talking about data, it comes in various forms, structures, and velocity. Modeling all these structures in a relational database is proving to be inadequate and inefficient. For example, some of the data patterns that are not appropriate for relational databases include: Key-Value, Document, Wide Column, Graph, Time Series etc. To successfully modernizing applications and get ahead of the curve, it is important to understand the data patterns and architectural designs that evolves over time.

Paradigm-shift in databases

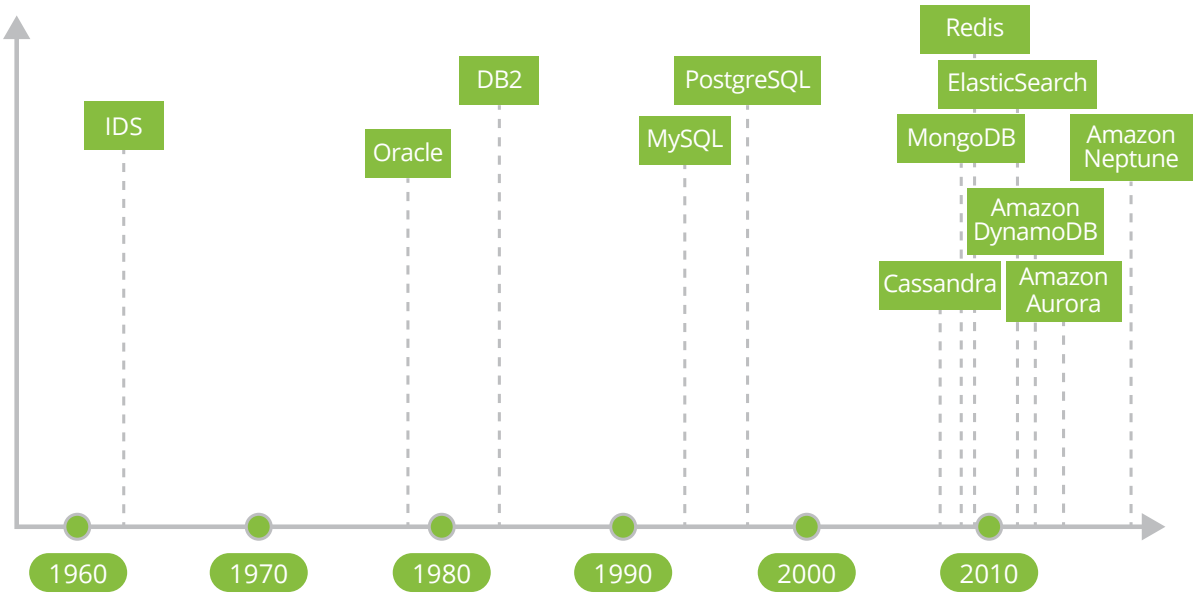


Figure 1: Timeline of databases since 1960s[1]

Over the last few decades, the database landscape has evolved in terms of type of data it can store (variety), the amount of data it can store (volume) and frequency at which it can process and store the data (velocity). The introduction of internet-enabled applications has changed the demands placed on databases. As a result, developers are now reaching for databases that can go faster, and with a global reach. Additionally, the rise of cloud computing has changed what is possible technically, enabling more resilient, and scalable applications in an economical way.

Tracking how application architecture has evolved, Mainframes provided centralized hosting of both applications and databases together. The client-server era separated applications from the database. And in the 1990s, we saw lot of organizations adopting the 3-tier architecture (Web, Application, and Data). At present, we are seeing lot of organizations leveraging microservices and event-driven architectures which are more conducive for splitting database workloads into smaller pieces. Decoupling various facets of applications and even the databases is the key for agility and faster time-to-market. Data growth has been exponential in the last decade, with no signs of slowing down. Gone are the days where a single relational database could manage all requirements for an organization, as demonstrated in Figure 2.

[1] https://s2.smu.edu/~fmoore/timeline.pdf

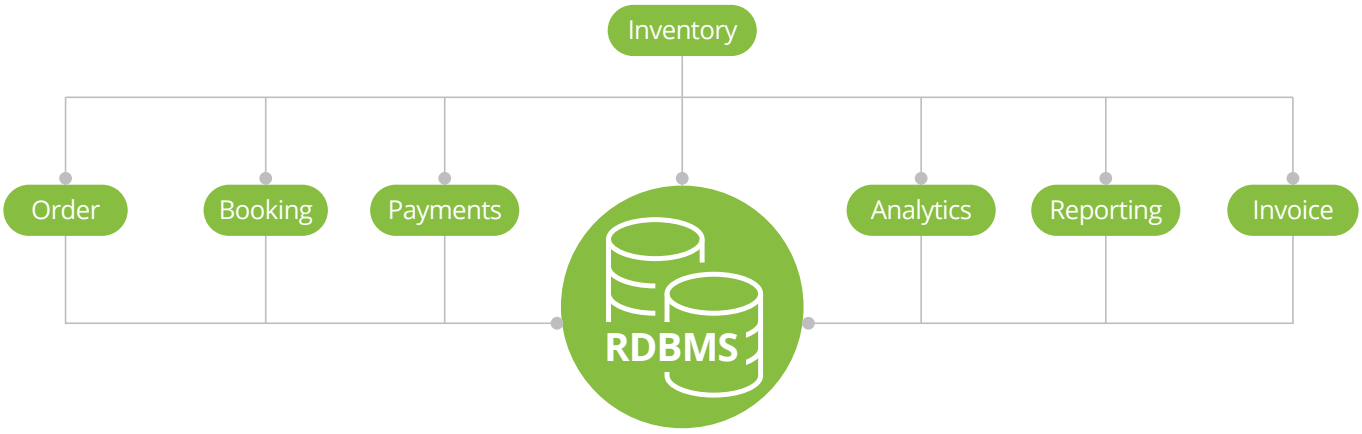


Figure 2: How an RDBMS was a solution for storing all kinds of data

"One size fit all" databases are difficult to address the needs of modern application and analytic ecosystems. Key challenges that necessitate the need for purpose-built databases include:

- Exponential data growth and the need for optimized storage and retrieval
- Automatic scaling
- Automated backups
- Effective handling of multiple types of data (Key-Value pair, Multiple Key-Value pair, TimeSeries)
- Effective handling of multiple structures of data (JSON (JavaScript Object Notation))
- Accommodating various latency requirements with applications

These challenges helped to drive the need for AWS to have 15 database services which are purpose-built for the kind of use cases they cater to. Additionally, with purpose-built, fully managed AWS databases, customers can save time and costs, improve performance at scale, and innovate faster.

Developers do not need to reach for the default relational database. They can consider the needs of their application and choose a database that is the best fit for their requirements.

Choosing a database is a major decision to make as part of your application architecture. The type of database you choose will affect the access patterns you manage, the performance, and the operations for which your team will be responsible. You should consider many factors, including your application workload, data shape, and performance requirements, to name a few.

Different Workloads and Database Alignment

AWS offers 15 purpose-built databases so you can choose the right tool for the job. The fully managed database services provide continuous monitoring, self-healing storage, and automated scaling to help you focus on business drivers for the application. The purpose of this section is to walk through each of these database services that AWS has to offer with each one's primary function, features, and typical use cases.

The following image shows the wide variety of AWS purpose-built database types:

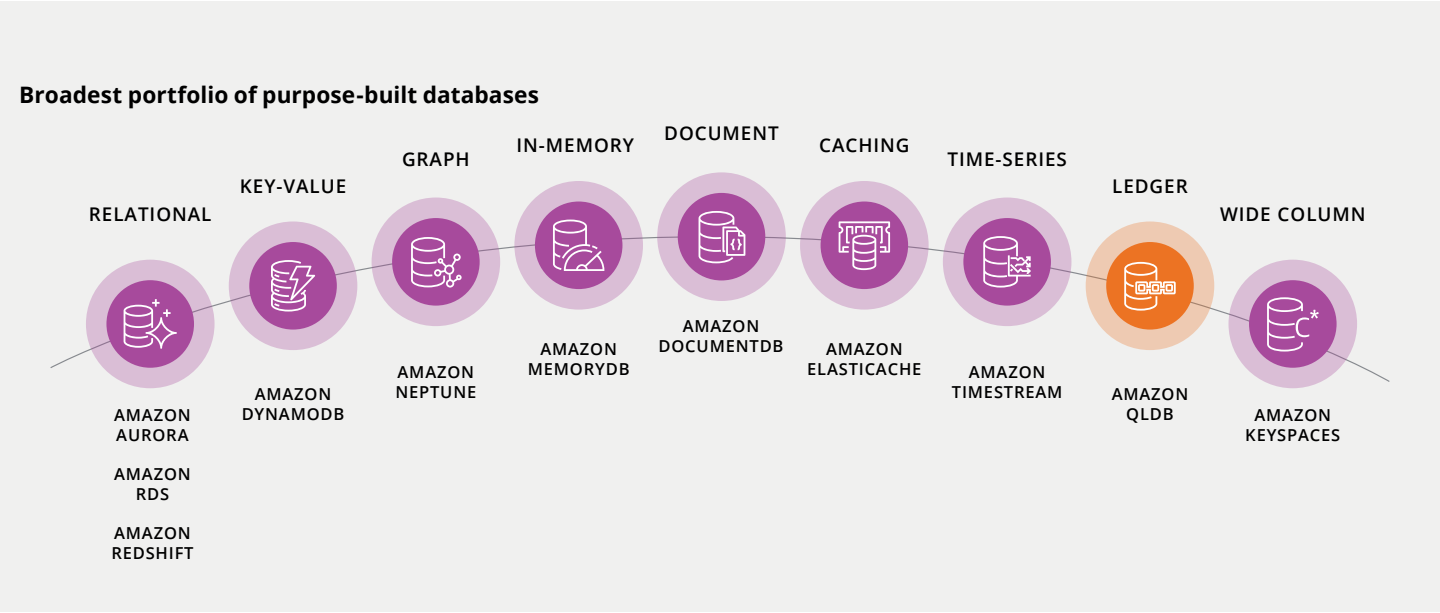


Figure 3: AWS portfolio of purpose-built databases[2]

[2] https://aws.amazon.com/products/databases/

Relational Databases

AWS provides relational databases in two flavors:
1) Amazon Relational Database Service (RDS) which is a relational database service that lets you run five database engines (both commercial and open source) in a managed environment and
2) Amazon Aurora – A MySQL or PostgreSQL compatible database born on the cloud for a high-performance relational database that can scale globally, at significantly less cost compared to commercial engines.

Amazon Relational Database Service (RDS)

Amazon RDS supports six engines to choose from:

- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Microsoft SQL Server
- DB2

This means you can take advantage of using a familiar database engine for which you may already have application, DevOps processes, and tools defined.

Features:

- Easier path to cloud migration
- Multi-AZ (Availability Zone) deployments
- Automatic Backups and patching
- Dials to control performance (General Purpose vs Provisioned IOPS)

There is also Amazon RDS Custom for advanced users who need to customize the underlying hardware. This is typically useful for legacy or packaged applications which may otherwise not be compatible with the cloud.

Use Cases:

- Enterprise resource planning (ERP)
- Customer relationship management (CRM)
- Financial data
- Transactions

Amazon Aurora

Amazon Aurora is a relational database management system built for the cloud with full MySQL and PostgreSQL compatibility. Aurora gives you the performance and availability of commercial-grade databases at one-tenth the cost.

The code, tools, and applications customers use today with existing MySQL and PostgreSQL databases can be used with Amazon Aurora. For certain workloads, Amazon Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to existing applications.

Features:

- Self-healing storage
- Storage autoscaling
- The ability to support up to fifteen read replicas
- Automatic, continuous, incremental backups and point-in-time restore
- Global databases
- Incredible performance and durability
- Serverless offering

Use Cases:

- Mobile and web gaming
- Software as a Service (SaaS) applications
- Enterprise applications

Non-Relational or NoSQL Databases

For decades, the only database choice was a relational database, so no matter the shape or function of the data in the application, using relational database model was the only option.

Is a relational database purpose-built for a normalized schema and to enforce referential integrity in the database? Absolutely, but the key point here is that not all application data models or use cases match the relational model, therefore relational data models are not one size fits for all use cases. We will walk through the NoSQL databases available from AWS:

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database that provides fast, consistent performance at any scale. It has a flexible billing model, tight integration with infrastructure as code, and a hands-off operational model. DynamoDB also provides single digit millisecond performance.

Customers typically consider DynamoDB if they have had scalability problems with other traditional database systems. Also, DynamoDB is ideal when you deal with high performance read and writes and promises predictable performance even during varying loads.

Features:

- Auto scaling
- Time To Live (TTL) for database rows
- On-demand backup and restore
- Single-digit millisecond performance at massive scale

Use Cases:

- Multi-player games for storing stats
- Metadata stores and caches
- Shopping carts
- User tracking

Amazon DocumentDB

A document database is a type of NoSQL database that allows you to store and query rich documents in your application. Amazon DocumentDB is a MongoDB compatible database that provides built-in security best practices, continuous backups, and native integrations with other AWS services. Amazon DocumentDB also offers near instant crash recovery. It uses log-structured storage and does not require crash recovery replay of database redo logs, reducing restart times (typically less than 30 seconds).

Features:

- Fully managed document database on the cloud
- Compatibility with MongoDB
- High scalability and performance
- Automatic, continuous, incremental backups and point-in-time recovery

Use Cases:

- Content management systems
- User profiles
- Product catalogs

Amazon ElastiCache
Some services and applications require a durable, powerful database for their primary data storage. This could mean a relational database, such as Amazon Aurora. Or it could mean a NoSQL database, like Amazon DocumentDB (with MongoDB compatibility). These databases provide powerful query capabilities with strong data guarantees that make them an excellent fit for your primary database.

But sometimes application users need lower latency than what they get from their primary data store. For example, you might have a common request flow in your application that has significantly more reads than writes. For these situations, you may want to use an in-memory cache to help. You can save the results of frequently read queries into a cache to improve response times and reduce pressure on your primary database.

Amazon ElastiCache provides support for two open-source, in-memory cache engines: Redis and Memcached. When you use ElastiCache, you get a fully managed, in-memory cache. You are not responsible for instance failovers, backups and restores, or software upgrades.

Use Cases:

- Session store
- Real-time analytics
- Chat apps
- Product catalogs

Amazon Neptune
Amazon Neptune is a fully managed graph database service provided by AWS, designed for storing and querying graph data, which consists of nodes and edges representing entities and their relationships. It also provides robust security features, including network isolation using Amazon VPC, encryption of data at rest and in transit, and identity and access management (IAM) policies. Neptune is a powerful tool for building and running graph-based applications.

Use Cases:

- Recommendation engines
- Fraud detection
- Social network analysis
- Drug discovery
- Supply chain management

Amazon TimeStream
Building with time-series data is always challenging. If you want to model time series data in a relational database, there are common challenges:

- Unnatural for time series data
- Inefficient in processing time series data
- Rigid schema does not play well with the nature of time series data

Blockers with existing time series databases include:

- Difficult to scale
- Difficult to maintain availability
- Limited data Lifecyle management

Amazon TimeSeries solves for these challenges. It is a fast, scalable, and serverless time-series database service that makes it easier to store and analyze trillions of events per day.

Use Cases:

- Monitoring and taking actions on critical time-based applications
- Storage and analysis of telemetry data
- Storage and analysis of IoT data

Amazon Quantum Ledger Database (QLDB)

Amazon Quantum Ledger Database (QLDB) is a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log. QLDB helps in tracking and maintaining a sequenced history of application data changes using an immutable and transparent journal. You can build correct, event-driven systems with QLDB ACID transactions.

Use Cases:

- Banking and Finance to record immutable transactions
- HR & Payroll to create a complete, centralized record of employee details
- Supply chain system to track and maintain a sequenced history of materials

Amazon KeySpaces

Amazon KeySpaces is a scalable, highly available and managed Apache Cassandra-compatible database service, designed for large scale applications that need fast read and write performance. It is built out of open-source technology and stores data in a wide column format.

It takes specialized expertise to deploy, configure and manage Apache Cassandra. The clunky version upgrades, complexity of managing encryption and optimizing resources typically keep people away from deploying Apache Cassandra based workloads. Amazon KeySpaces offloads these complexities from you so customers can spend more time on solving business problems.

Use Cases:

- Messaging
- Device metadata
- Event logging

Amazon MemoryDB for Redis

Amazon MemoryDB for Redis is a Redis compatible in-memory database service that provides extremely fast performance. Unlike cache, data stored in MemoryDB is durable. It is designed for applications that cannot tolerate even single digit millisecond latencies. MemoryDB is fully managed, and it only takes few clicks to operationalize a highly performant, durable MemoryDB cluster. It provides a distributional transactional log to provide data durability, consistency, and recoverability.

MemoryDB stores your entire dataset in memory, so it can deliver microsecond read latency and single digit millisecond write latency. It can manage more than thirteen trillion requests per day and support peaks of 100s of millions of requests per second.

Use Cases:

- Online games
- Streaming media and entertainment
- Chat and high-performance web applications

Modern Architecture Patterns

This section provides examples on architectural patterns that are possible utilizing AWS' purpose-built databases.

Highly transactional financial services applications

Consider a group of applications in the financial services industry relying on a single, large Oracle database instance in their on-premises environment. Examples of the use cases of these applications are: Financial transactions, trading, accounting, fraud detection etc. It includes very complex database queries, requirements for ACID (Atomicity, Consistency, Isolation and Durability) properties for financial transactions and the need for high volumes of reads and writes. There are 100s of tables which also utilize a caching layer.

There are three options for migrating this type of database to AWS:

01. Migrate as-is to Amazon Elastic Compute Cloud (EC2) instances.
02. Re-platform it to the Amazon Relational Database Service (RDS) – Oracle.
03. Split the large monolith into multiple purpose-built databases.

Options 1 and 2 are often the choice of organizations when migrating workloads to the cloud, but at times, insufficient. Existing problems and challenges of the on-premises environment to the cloud can be migrated with the application. A potential is to look at the various database use cases more closely and split them into purpose-built databases that solve the application's needs more effectively. Let us look at option three in more detail. A proposed architecture could look like the following diagram.

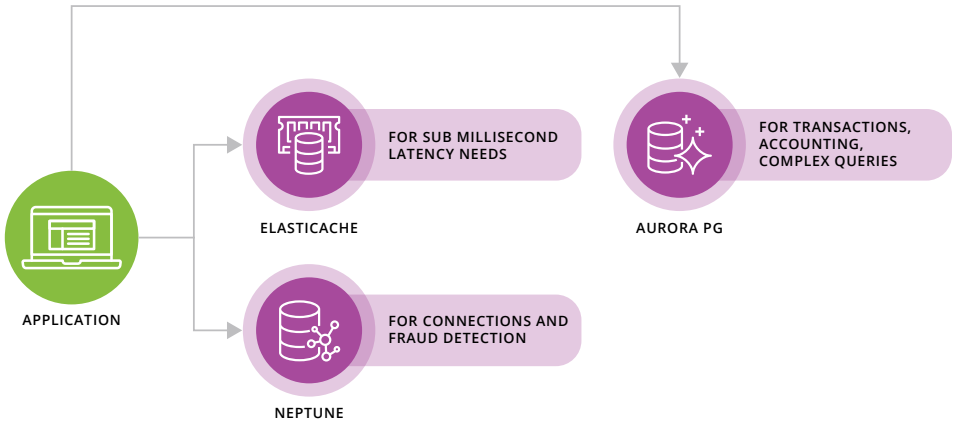


Figure 4: A modern database pattern for a financial services application

Amazon Aurora combines the speed and availability of commercial databases such as Oracle with the simplicity and cost-effectiveness of open-source databases. Aurora will address requirements such as ACID transactions and high volumes of reads and writes. Aurora is more appropriate in this case than say, AWS RDS database engine because the way Aurora can scale IOPS (Input Output operations Per Second), which is incredibly important for an application in the Financial Services industry.

To identify fraud in financial transactions, the transactions themselves will not be of any help. We need to know the actors behind those transactions and how those actors relate to others and how the transactions themselves are related. This is precisely why it is essential to model this as a connected graph. Amazon Neptune can store and query relationships between billions and billions of interconnected entities. By modeling your data like this, you can identify relationships that are not very apparent. And repeatedly, that is where you will be able to find fraudulent transactions.

While Aurora PostgreSQL can provide latencies in double- or triple-digit milliseconds, sometimes there is a need for microsecond latency in applications. And this is essential for applications dealing with financial transactions where every microsecond matters. Amazon ElastiCache offers a high performance, low latency caching solution on the cloud. The mission critical tables that do not change very often in Aurora PostgreSQL can be fronted by ElastiCache for sub-millisecond latency needs. ElastiCache can scale based on demand and can store millions of objects, making it a perfect choice for a caching layer.

IoT based analytics application

Let us explore an application that reads sensor data in real-time and analyzes the data to take a variety of actions. An example could be sensors in an electrical grid sending signals to a central processing unit for analysis and alerting repair crews if there is a voltage fluctuation.

The application can be modeled using two purpose-built databases:

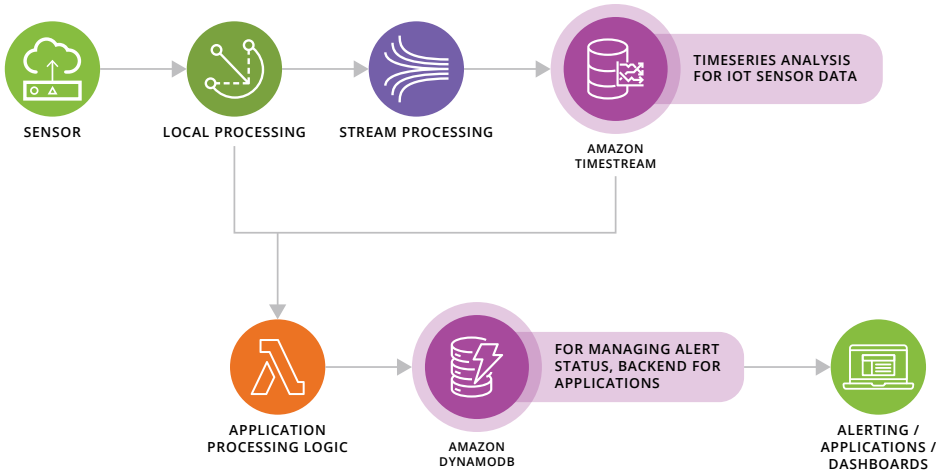


Figure 5: Database pattern for an IoT-based alerting application

Amazon TimeStream is used to ingest and quickly analyze the timestream data is produced by 100s or even 1000s of sensors. In fact, TimeStream can analyze trillions of events a day. TimeStream is entirely serverless and automatically scales up or down based on demand so there is no management required. TimeSeries has built-in analytics capabilities like aggregates, roll-ups, window functions, and various mathematical functions. A combination of these can be used for real-time dashboarding, signal processing, and alerting. Modeling this time-bound data in Amazon TimeSeries makes a lot of sense over using a relation engine or other NoSQL offerings.

The alerts and the entire lifecycle of an alert (e.g., New→Acknowledged→Working→Resolved) can be modeled in a fully managed NoSQL database like Amazon DynamoDB. DynamoDB can support petabytes of data and millions of reads and writes per second. Like TimeStream, DynamoDB also takes the load off customers in terms of server management, patching, software's etc. since DynamoDB is serverless by design. DynamoDB also integrates with AWS Lambda to provide triggers which can be used to run a custom function when the alert status changes.

Real-time multi-player game

Let us evaluate the database choices for a multi-player game. We will limit the use cases to just two: 1/ Storing and retrieving a real-time leaderboard and 2/ Storing and retrieving user profiles

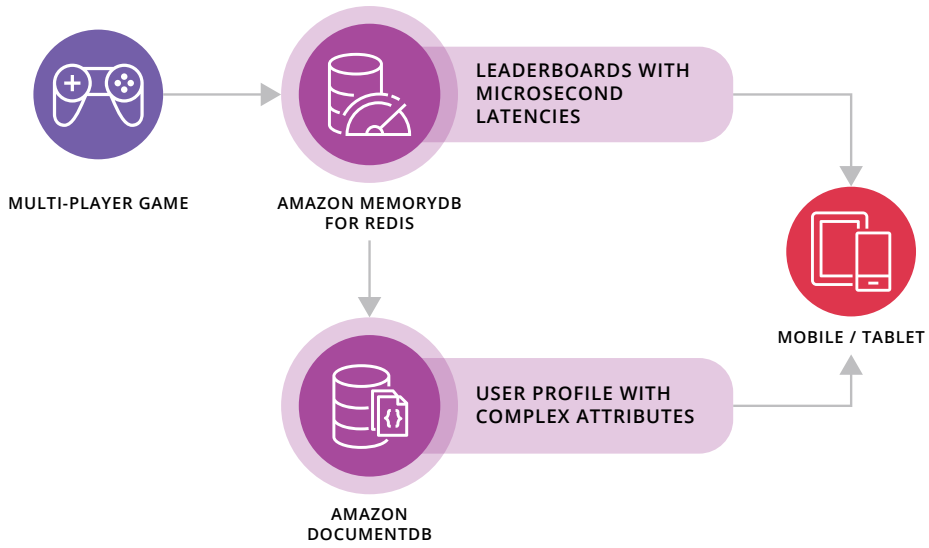


Figure 6: Recommended databases for a real-time gaming application

Leaderboards are a valuable tool in a multiplayer gaming which helps keep the engagement levels high. Durability is the key with leaderboards and latency is important. A brute force approach could be to store the leaderboard in a transactional database like Aurora MySQL and front it with a caching layer like ElastiCache.

While this approach could work, the cost of updating Aurora and the caching layer in a single transaction could become a bottleneck. A better approach is to use the Amazon MemoryDB for Redis. MemoryDB has a multi-AZ transactional log for fast database recovery and restart which gives you strong durability. MemoryDB can scale up to 100s of millions of requests per second which is a key for multi-player games where leaders can change every microsecond.

This massive scale combined with low latency and high concurrency makes MemoryDB an excellent choice for gaming applications.

Simultaneously, user profiles will have simple attributes like username and wins and losses, and complex attributes like an array of achievements earned, an array of inventory the user has etc. JSON is a way to model these attributes, making DocumentDB (with mongoDB compatibility) a great fit to store JSON in its native format. Relational or other NoSQL databases will not perform as well as a database that can manage documents in its natural format. DocumentDB can scale to millions of reads and writes of documents and to petabytes of storage. It is fully managed and does not come with licensing fees.

Serverless Databases
How Serverless takes it one step further?

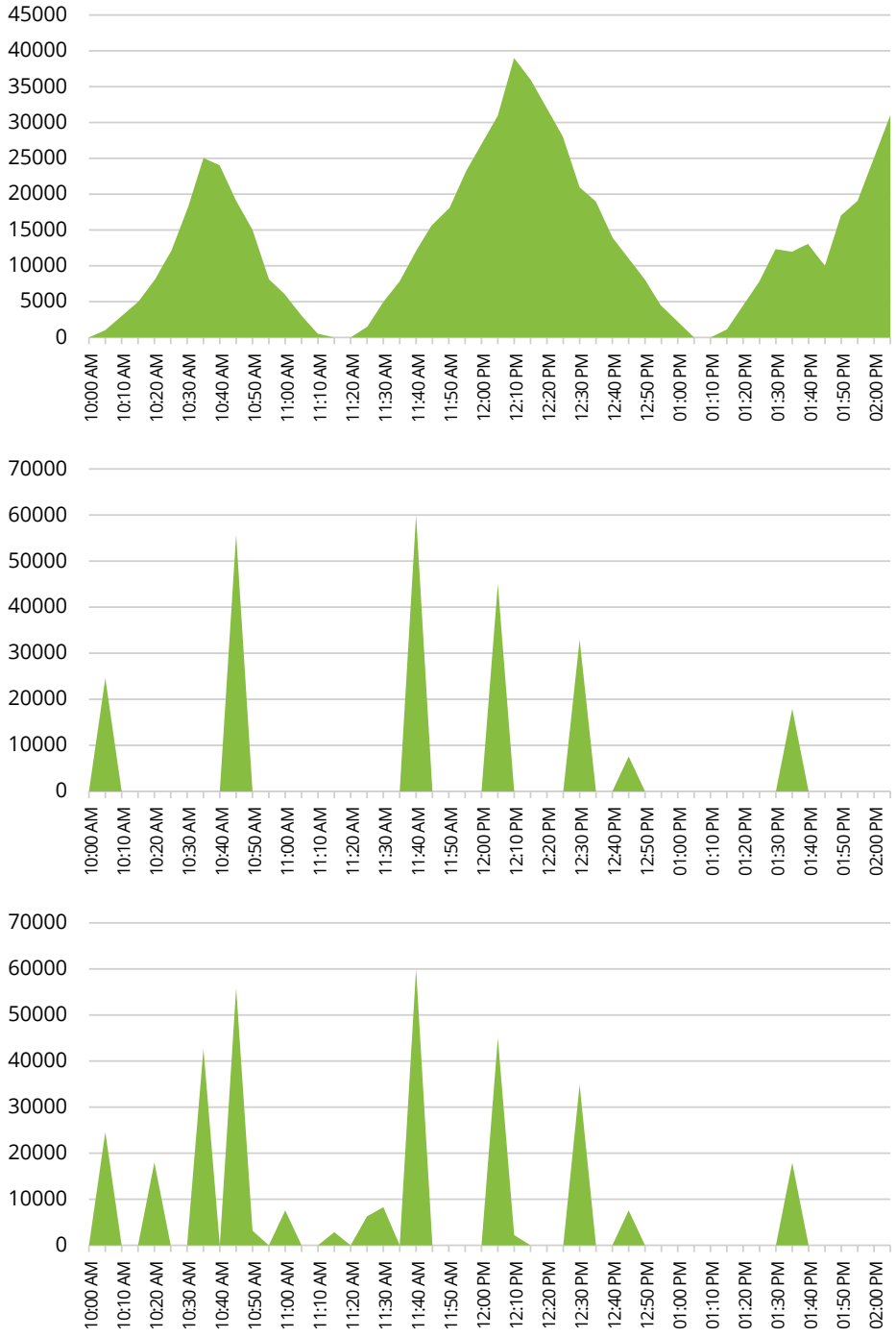
Remember the days when organizations used to fret over the amount of time and resources it would consume to make database changes to match their application demands? Fast forward to the modern database, driven by years of innovation, where database capacity is automatically scaled based on application demand, eliminating the time-consuming planning and upgrades. This is the world of serverless databases.

A serverless database removes the operational overhead associated with managing and maintaining the database infrastructure. This overhead includes scaling, capacity planning, patch management and version upgrades. What this means for customers is that they can focus on their business and innovation opposed to operational tasks (aka. "Keeping the lights on"). There are also benefits of high availability and fault tolerance. With the pay-as-you-go model for these serverless database workloads customers only pay for the duration in which the serverless database is used, nothing more.

The most common use cases for a serverless database are:

- Infrequently used applications
- Development / testing / sandbox workloads
- Unpredictable and variable workloads
- Workloads that need to scale rapidly

Let us take an example of applications that have spiky, infrequent, or periodic workloads, shown here in Figure 7.



[3] https://aws.amazon.com/rds/aurora/serverless/ Figure 7: Serverless database use cases[3]

For those kinds of workloads, we could provision capacity that is not sufficient for the peak. That would result in experience degradation and impacts to performance. The other option is to provide for the peak, which could be expensive resulting in too much capacity outside of peak usage. Yet another option is to deploy an army of people who could continuously monitor the environment and scale up or down as needed. This would involve lot of labor, downtime, and degradation at times. The most logical option for such workloads therefore is to go with serverless databases. Many of the AWS purpose-built database today have a serverless option. Some of them come exclusively serverless like the Amazon DynamoDB, Amazon QLDB (Quantum Ledger Database), Amazon KeySpaces for Cassandra, and Amazon TimeSeries. There are other databases that offer serverless as an option like Amazon Aurora and Amazon Neptune.

Amazon Aurora Serverless

Amazon Aurora is one of the fastest growing services at AWS. Aurora Serverless supports even the most demanding applications and database workloads. Aurora Serverless scales instantly to hundreds of thousands of transactions in a fraction of a second. As it scales, it adjusts capacity in fine-grained increments to provide the right amount of database resources that the application needs. There is no database capacity for you to manage. The scaling happens by adding more CPU (Central Processing Unit) and memory resources and has no impact to current transactions. The compute fleet is continuously monitored and scaled horizontally when needed. The storage also grows and shrinks based on demand as shown in Figure 8.

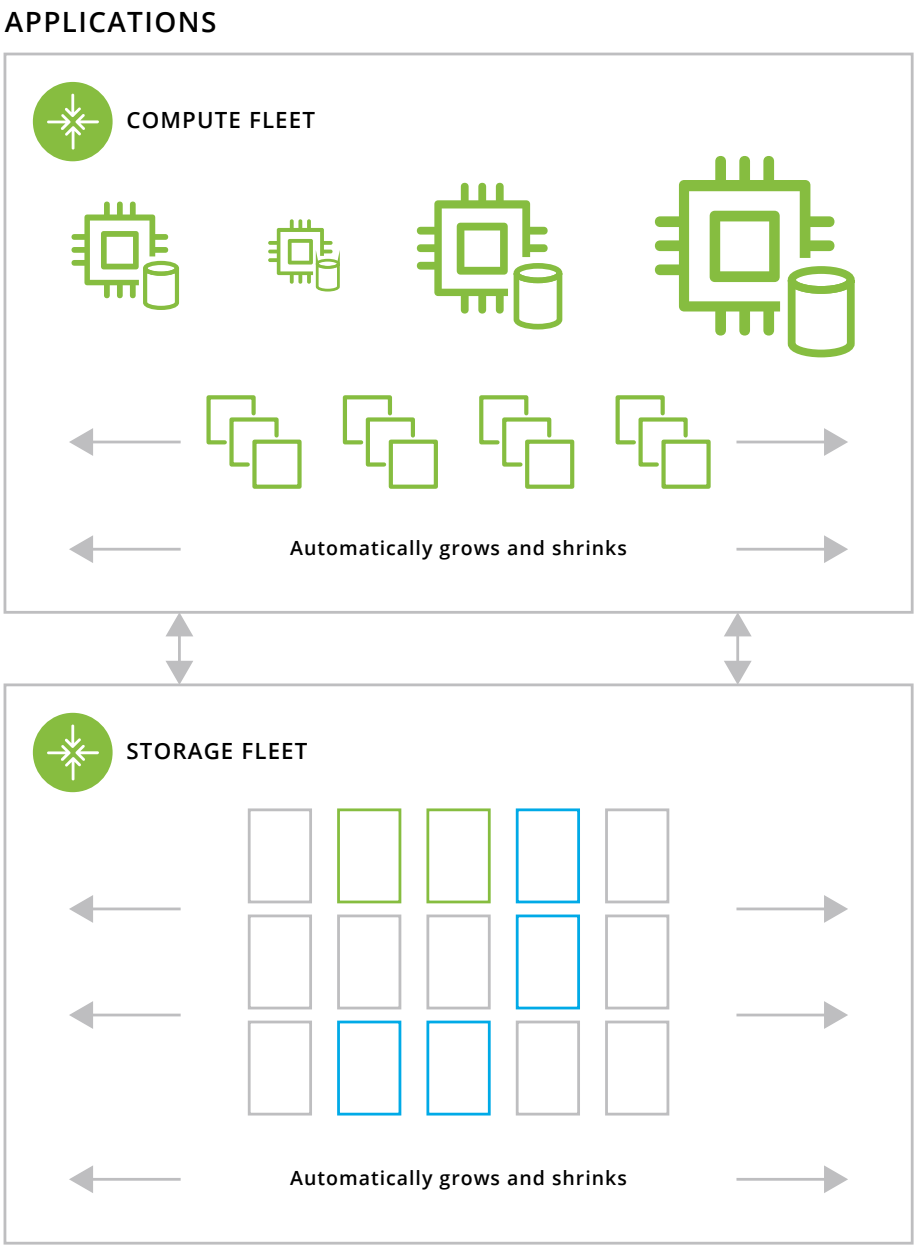


Figure 8: Amazon Aurora Serverless architecture[4]

One of the most important real-world use cases for Aurora serverless are the multi-tenant Software-as-a-Service (SaaS) applications. SaaS vendors typically operate hundreds or thousands of Aurora databases, each supporting a different customer, in a single cluster to improve utilization and cost efficiency. They still need to manage each database individually, including monitoring for and responding to co-located databases in the same cluster that may consume more shared resources than originally planned. With Aurora Serverless v2, SaaS vendors can provision Aurora database clusters for each individual customer without worrying about the costs of provisioned capacity. It automatically scales down to as little as 0.5 Aurora Capacity Units (ACUs) when they are not in use to save costs and instantly adjusts databases capacity to meet changing application requirements.

[4] https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless-v2.how-it-works.html

Amazon Neptune Serverless

Graph databases are still new and we at times see customers running graph workloads on typical OLAP / OLTP databases. Some customers find managing capacity on graph databases to be extremely difficult. That is precisely why Amazon Neptune started offering a serverless database. Amazon Neptune can instantly scale capacity in a fraction of a second. The capacity can scale in fine-grained increments enabling customers to save up to 90% on costs when compared to provisioning for peak capacity.

Amazon Neptune Serverless decides to scale based on the following:

- CPU utilization of both foreground and background processes
- Memory utilization of internal data structures like the buffer pool

Database Migration Case Studies

This section provides three examples of how Deloitte implemented purpose-built databases for their clients.

Deloitte Case Study 1

Executive Summary
A multinational conglomerate providing professionals with trusted content using software to make informed decisions. The client sought to modernize their content and research intelligence capabilities with the overall goal of improving performance and scalability, reducing external dependencies and costs, and enriching the customer experience.

The need for Purpose-built database
As part of the solution the Deloitte team focused on simplifying and classifying content which involved a multi-stage approach of acquisition, ingestion, processing, editing, and publishing the content in a manner that will improve the overall performance and maintain the traceability of the content. To provide metadata for workflows and maintain the content context between service executions, DynamoDB was selected. The global tables feature of DynamoDB was leveraged for eliminating the need for multi-region data synchronization to optimize the performance. Deloitte also deployed RDS PostgreSQL to efficiently process the transactional data (updates and retrievals) for over 300M organizations worldwide.

The Solution
The solution built, fits a digital transformation that many organizations are undertaking to overhaul legacy technology and adapt to the modern world challenges. DynamoDB was selected for the following reasons:

- To store the metadata for workflows, maintaining context between microservices.
- Serve as a configuration store, reducing code deployment dependency and centralizing all configurations
- Global tables for different regions eliminate the need for data synchronization across multiple regions
- Fast and uncomplicated way to make content configuration changes dynamically without having to worry about infrastructure scalability

RDS PostgreSQL was also leveraged to process transactional data for over three hundred million organizations requiring processing simultaneous updates and retrievals without data loss and performance lags.

Results and Benefits
Deloitte successfully transformed a legacy content and research intelligence platform serving client's global customer base (legal, tax and accounting professionals worldwide) using AWS purpose-built databases and cloud native services to empower them with information to shape tomorrow.

Outcomes included:

- A scalable solution that benefited by 40% more transactional volume per day just by using RDS PostgreSQL
- Sub-millisecond response to process over three hundred million organizations content globally
- Improved availability, durability and multi-region fault tolerance using DynamoDB global tables
- Automated cluster management, encryption, privacy, security, and multiple replica instances for speed and disaster recovery using RDS PostgreSQL

Improved consistency and conflict resolution using last-writer-wins reconciliation between concurrent updates to ensure all replicas provide a state for consistent content across the world.

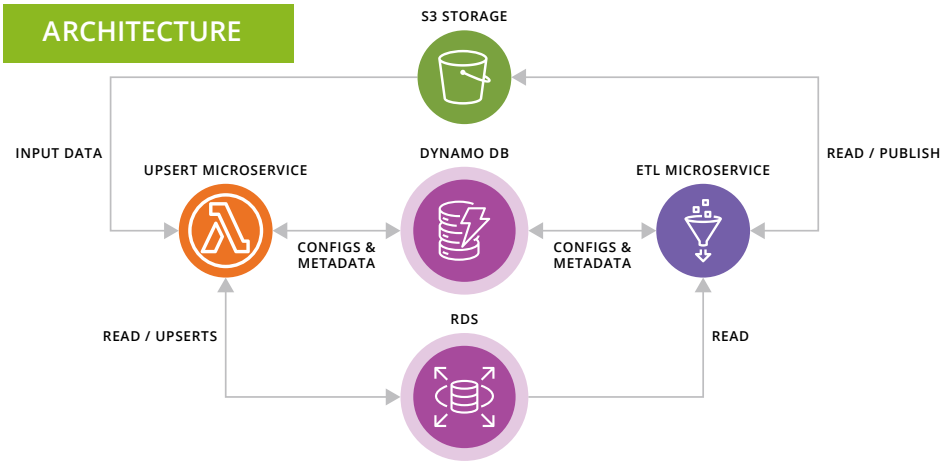


Figure 10: A modern database architecture solution for a global content management client

Deloitte Case Study 2

Executive Summary
Deloitte's ConvergeHEALTH™ CognitiveSpark™ for Clinical is a modular, cloud-based, metadata-driven solution. It is designed to automate data management across the clinical trial lifecycle to digitally generate structured and standardized deliverables from a range of input documents and sources. These elements are then intelligently interpreted and transformed to set up downstream systems, auto-populate required reports and analyses, and generate content for key trial artifacts.

The need for Purpose-built database
The Deloitte team focused on simplifying and classifying the content which involved a muti-stage approach of acquisition, ingestion, processing, editing, and publishing the metadata in a manner that will improve the overall performance, maintain the traceability of the content. To manage semantic relationships between protocol elements, study definitions, biomedical concepts, internal (sponsor) and external managed standards (CDISC and other), Amazon Neptune was selected. RDS PostgreSQL was deployed to efficiently process the transactional data (updates and retrievals) for 20+ clinical data sources.

The Solution
The solution provided an intelligent metadata backbone to drive Digital Data Transformation that many organizations are undertaking to reduce the time it takes to bring a molecule to market. Amazon Neptune was deployed for the following reasons:

- To store the metadata for setting up data collection forms, data tabulation standards and controlled terminologies
- To store machine readable rules for quality, harmonization, and compliance, serving as a single source of truth for entire clinical development lifecycle
- Connect internal and external metadata, providing end-to-end traceability across business processes
- Vertical and horizontal scalability:
  - Integrate with external ontologies, vocabularies, and dictionaries
  - Allows further enrichment from competitive intel (a lot of data available as unstructured content)
  - Really fast and effortless way to make metadata changes dynamically without having to worry about underlying infrastructure scalability

RDS PostgreSQL was also deployed to process transactional data for 20+ clinical data sources requiring processing simultaneous updates and retrievals without data loss and performance lags

Results and Benefits

Deloitte successfully launched the ConvergeHEALTHTM CognitiveSparkTM for Clinical platform using AWS purpose-built databases and cloud native services to empower clinical development stakeholders (Clinicians, Data Management, Biostatistics and Medical Writers) with information to reimagine clinical data flow of future. Outcomes included:

- A scalable data foundation for automated quality profiling, unified storage for raw, enriched, and analytical data sets
- Sub-millisecond response to process data from 20+ clinical data sources
- Improved availability, durability and multi-region fault tolerance using Amazon Neptune
- Automated cluster management, encryption, privacy, security, and multiple replica instances for speed and disaster recovery using RDS PostgreSQL
- Improved consistency and conflict resolution using last-writer-wins reconciliation between concurrent updates to ensure all replicas provide a state for consistent content across the world.

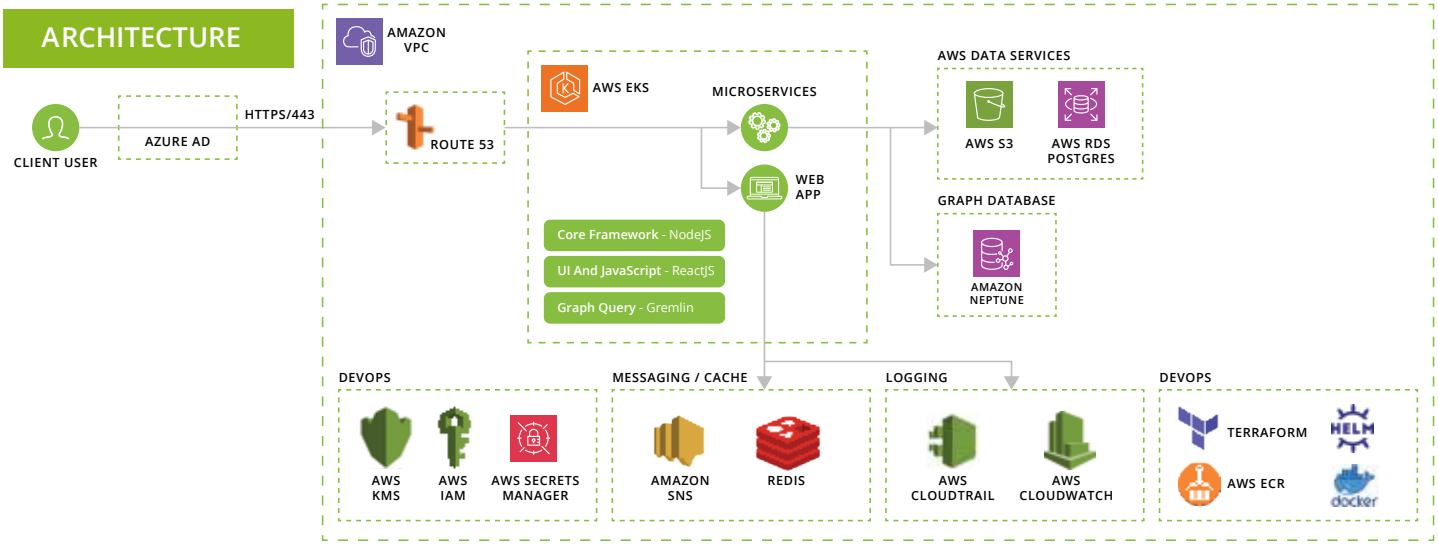


Figure 11: Purpose-built database solution for a clinical development use case

Conclusion

Modernizing an enterprise's legacy data platform that was built many years ago may seem difficult, but if done with the proper approach and technology platform, the rewards can be valuable.

The abundance of choices for different purpose-built databases to aid in cloud modernization in today's market makes modernization efforts more complex, but also represents a wonderful opportunity for organizations to maximize the benefits by using the right tool for the job.

Successfully completing a strategic database modernization requires analysis of the benefits and risk for each type of workload. It is critical to choose technology partners that offer both proprietary and open-source capabilities that strongly embrace the ecosystem. AWS offers an extensive catalog of purpose-built databases to meet the use-case specific demands that accompany modernization goals.

AWS suggest the following guidance when embarking on a database modernization journey:

- Review the purpose-built databases that AWS offers, starting with the use case, and define the requirements to help filter which database services are a good fit
- Overlay selected database options with the desired speed-to-market and team's skillset, along with a balance of the future needs
- Acknowledge that modernization is not just about innovative technology, but a shift in the organization culture that may require upskilling and training

About Deloitte

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private company limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. In the United States, Deloitte refers to one or more of the US member firms of DTTL, their related entities that operate using the "Deloitte" name in the United States and their respective affiliates. Certain services may not be available to attest clients under the rules and regulations of public accounting. Please see www.deloitte.com/about to learn more about our global network of member firms.

Contributors

Contributors to this document include:

Deloitte Consulting LLP

Tony Witherspoon
PPMD
Cloud Engineering (Architecture & Strategy)

Sameer Limaye
Specialist Leader
Cloud Native Development

Amazon Web Services

Kasi Muthu
Senior Partner Solutions Architect
Data, ML

Vishal Srivastava
Senior Partner Solutions Architect
Databases

Grant Schonberg
Principal Tech BD
Databases

Document Revisions

Date	Description
May 30, 2023	First Draft Complete
June 30, 2023	AWS Review Complete
October 26, 2023	Deloitte Review Complete
November 15, 2023	First version ready for review
January 8, 2024	AWS review complete
January 19, 2024	Deloitte review complete
Month, Day, YYYY	First publication

Deloitte.

About Deloitte

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private company limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. In the United States, Deloitte refers to one or more of the US member firms of DTTL, their related entities that operate using the "Deloitte" name in the United States and their respective affiliates. Certain services may not be available to attest clients under the rules and regulations of public accounting.

Please see www.deloitte.com/about to learn more about our global network of member firms.