Data Engineering Day 24

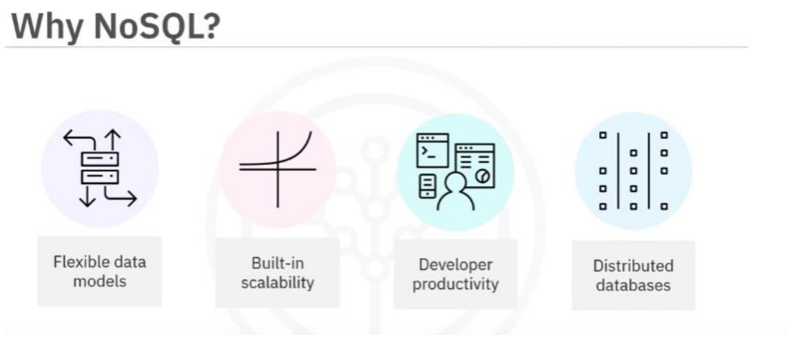
The credit for this course goes to Coursera. Click More

Another link : Azure data Engineer

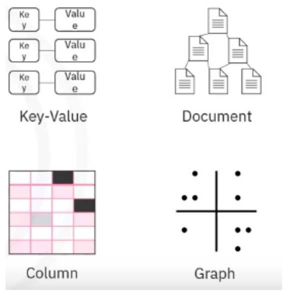
Introduction to NoSQL databases

# Introducing NoSQL(Not Only SQL): click me to read more

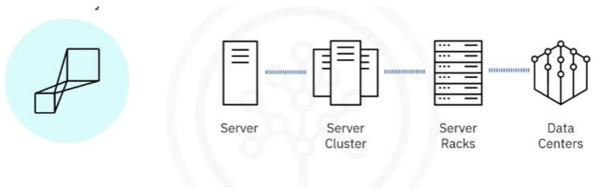
- NoSQL databases (aka " not only SQL") are non-tabular databases and store data differently than relational tables.
- NoSQL databases come in a varity of types based on their data model.The main types are dicument, key-value wide-column, and graph.
- It provide flexible schemas and scale easily with large amounts of data and higher user loads.



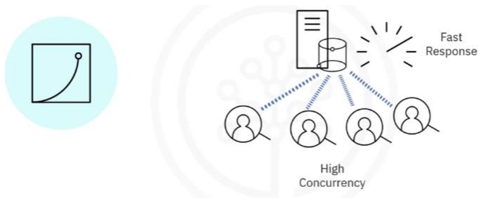
- **Characteristics of NoSQL Databases:**
  - NoSQL databases fit into four categories such as key-value, Document, Column, and the graph.
  - Are built to scale horizaontally
  - Share data more easily than RDBMS
  - Use the global unique key to unify the data shading
  - Are more used case specific than the RDBMS
  - Are more developer friendly



- **Advantage of using NoSQL:**
  - Scalability



  - Performance



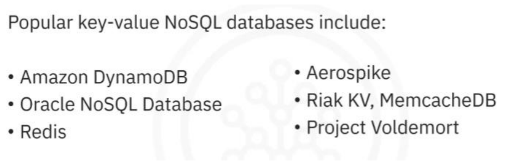
- Availability:

- Cloud Architecture:

- Cost

- Flexibility

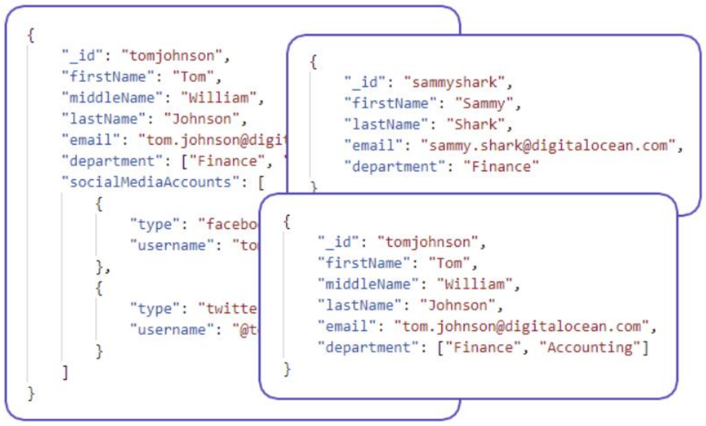
- **Key-Value NoSQL Databases:**
    - All data are stored as key and associated with the value in it.
    - Least complex
    - Stored and displayed as a HashMap.
    - Are ideal for basic CRUD operations.
    - Scales well
    - Shards easily
- **Use suitable cases of Key-Value NoSQL Databases:**
    - Are used for baisc CRUD operations
    - Storing in app user profiles and preferences
- **Use unsuitable cases of Key-Value NoSQL Databases:**
    - Not suitable for many to many interconnected data (social networks, recommendations engines)
    - When a high level of consistency is required for multi-oerations transistions with multiple keys



Popular key-value NoSQL databases include:

- Amazon DynamoDB
- Oracle NoSQL Database
- Redis
- Aerospike
- Riak KV, MemcacheDB
- Project Voldemort

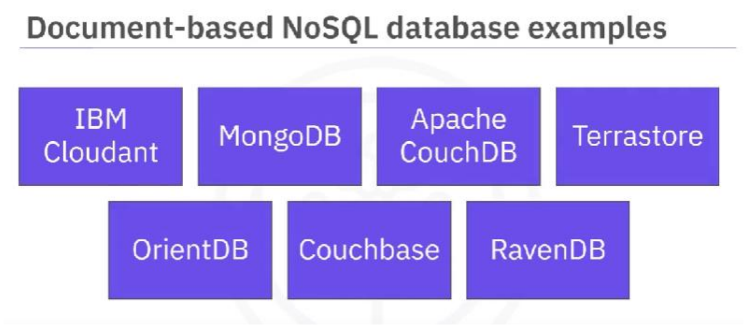
- **Document-Based NoSQL Databases:**
    - Use document to store data.
    - Values are visible and can be queried.
    - Each piece of data is considered as the document.
    - Each document offers flexible schema.
    - Suitable use cases include event login for apps and processing it, for online blogs such as create, comment, operations data for web and the mobile applications.
    - Unsuitable use cases are in ACID transitions and when data is in aggregate-oriented design.
    - The following figure represents the example of NoSQL database. The image can be found in this link



```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digi
    "department": ["Finance", "
    "socialMediaAccounts": [
        {
            "type": "facebo
            "username": "to
        },
        {
            "type": "twitte
            "username": "@t
        }
    ]
}
```

```
{
    "_id": "sammyshark",
    "firstName": "Sammy",
    "lastName": "Shark",
    "email": "sammy.shark@digitalocean.com",
    "department": "Finance"
}
```

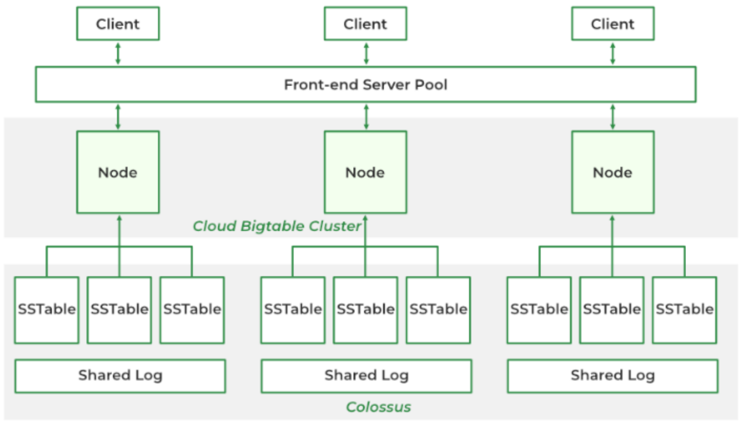
```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digitalocean.com",
    "department": ["Finance", "Accounting"]
}
```



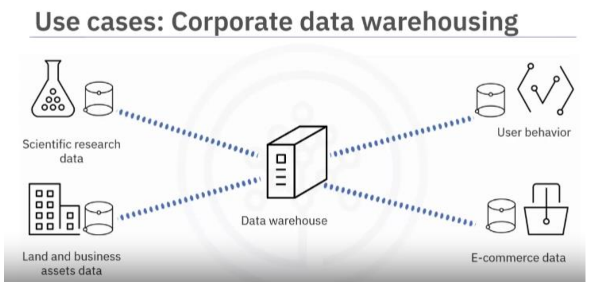
Document-based NoSQL database examples

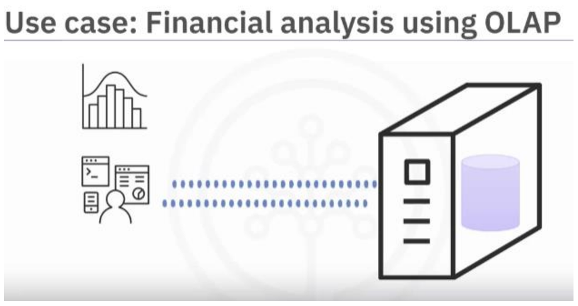
IBM Cloudant | MongoDB | Apache CouchDB | Terrastore | OrientDB | Couchbase | RavenDB

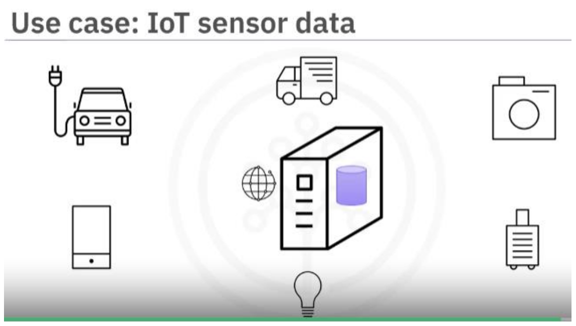
- **Column-Based NoSQL Databases:**
  - Is spawned from an architecture that google created and called Bigtable. click me to read more from geeks for geeks



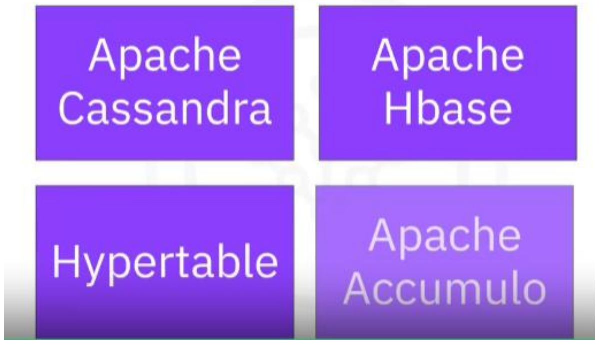
  - Are also named as Bigtable clones, columnar, wide column databases.
  - Data are stored as a column or a group of columns.
  - Each column will have unique keys or an identifier.
  - Some of the use cases are as it is great for large amounts of spares data, stored as a column database, handle deployments very smoothly across the clusters node, used for event logging and the blogs,
  - Is not suitable for traditional ACID transitions etc.


Use cases: Corporate data warehousing
Scientific research data
User behavior
Data warehouse
Land and business assets data
E-commerce data



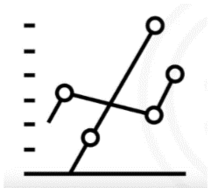


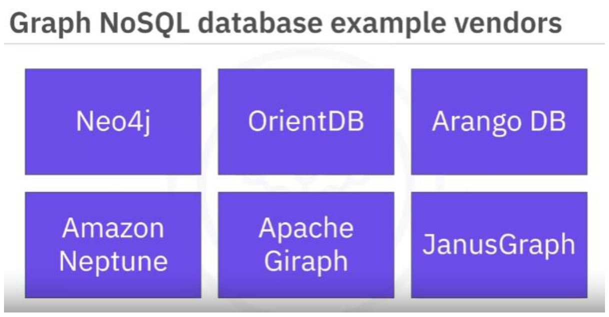
- Popular column based No SQL databases are shown below.



Apache Cassandra | Apache Hbase | Hypertable | Apache Accumulo

- **NoSQL Graphical Databases:**
    - Stores information in entities (or a node) and relationships (or edges)
    - Graph databases do not share well.
    - Are ACID transactions complaint.
    - Used cases includes beneficial for highly connected and related data, social networking, used for routing, spatial, map apps, recommendations engine building cases, etc.
    - Unused cases include such as can have limitations in querying, performance, and modeling, with a steep learning curve and dependency on vendor tools.





Graph NoSQL database example vendors

Neo4j	OrientDB	Arango DB
Amazon Neptune	Apache Giraph	JanusGraph

NoSQL Database Deployment Options

Database deployment options refer to the various methods and strategies for implementing and managing databases within an organization. Choosing the right deployment option is crucial for optimizing performance, scalability, and efficiency. Here, you'll explore some common database deployment options and their key characteristics

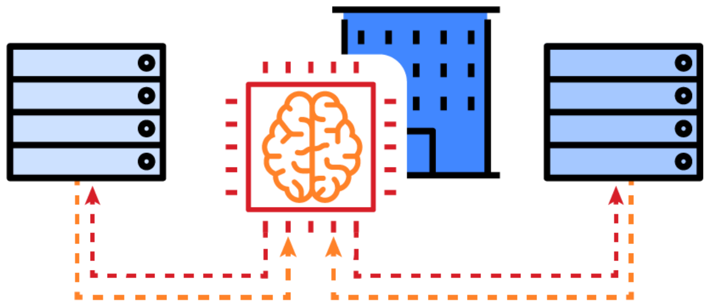
What you will learn

After completing this reading, you'll be able to

- Identify methods for deploying NoSQL databases
- List examples of NoSQL database deployments
- Describe some of the advantages and challenges associated with deploying NoSQL databases

On-premises deployment

On-premises deployment involves hosting the entire database infrastructure within the organization's physical location or a dedicated data center.



Example

An organization manages customer relationship data using an on-premises Oracle Database installed on dedicated servers within its corporate data center.

Advantages

Full control: Organizations have complete control over hardware specifications, software configurations, and security measures.

Compliance: On-premises solutions might be necessary for industries with stringent regulatory or compliance requirements.

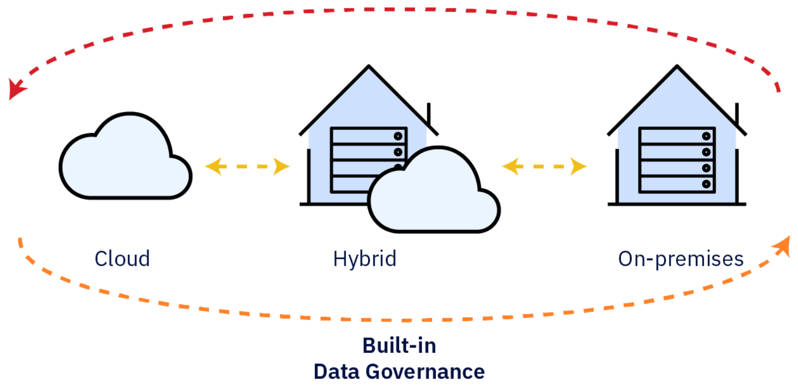
Challenges

Upfront costs: High initial investments in hardware, infrastructure, and skilled personnel.

Scalability: Limited scalability compared to cloud alternatives, which might result in challenges during periods of rapid growth.

Cloud deployment

Cloud deployment involves utilizing cloud service providers to host and manage databases over the internet.



Example

A startup leverages Amazon Web Services (AWS) to deploy and host its e-commerce database, utilizing Amazon relational database service (RDS) for scalability and managed database services.

Advantages

Scalability: Resources can be scaled up or down based on demand, providing flexibility and cost savings.

Cost-effectiveness: Pay-as-you-go pricing allows organizations to pay only for the resources they consume.

Automation: Cloud providers handle routine maintenance tasks, updates, and backups.

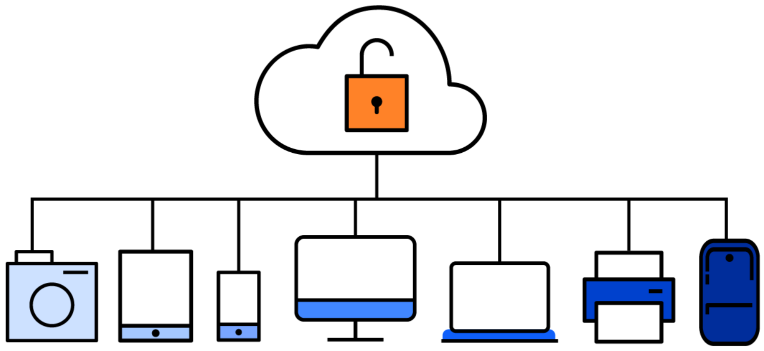
Challenges

Connectivity dependencies: Relies on internet connectivity, which might pose issues in areas with limited or unreliable access.

Security concerns: Requires robust security measures to protect sensitive data from unauthorized access.

Hybrid deployment

Hybrid deployment combines on-premises and cloud solutions, allowing organizations to distribute data and workloads based on specific needs.



Example

A financial institution stores sensitive customer information, such as account balances, in an on-premises database while utilizing a cloud-based service, like AWS Lambda, for processing non-sensitive data analytics workloads.

Advantages

Flexibility: Companies can keep sensitive data on-premises while using the cloud for scalable and dynamic workloads.

Disaster recovery: Companies can use cloud storage for backups, enhancing disaster recovery capabilities.

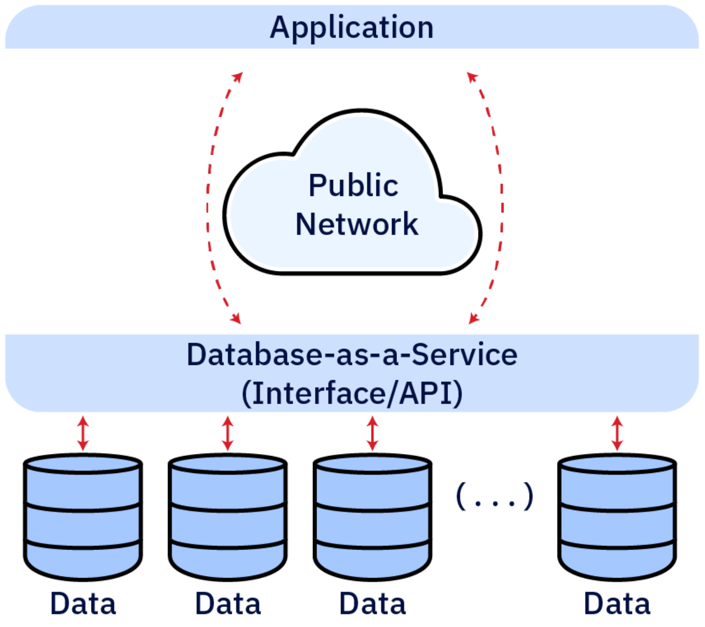
Challenges

Integration complexity: Integration complexities require effective integration between on-premises and cloud environments for seamless operation.

Management overhead: Monitoring and managing two different environments might increase administrative complexity.

Database as a service (DBaaS)

DBaaS provides a fully managed database solution where the service provider handles administrative tasks, allowing organizations to focus on application development.



Example

A software development company uses Microsoft Azure SQL Database as a DBaaS solution to host and manage its relational databases, allowing developers to focus on application development rather than database administration tasks.

Advantages

Reduced overhead: Organizations benefit from reduced administrative tasks, including maintenance, backups, and updates.

Rapid deployment: Quick deployment without the need for in-depth database expertise.

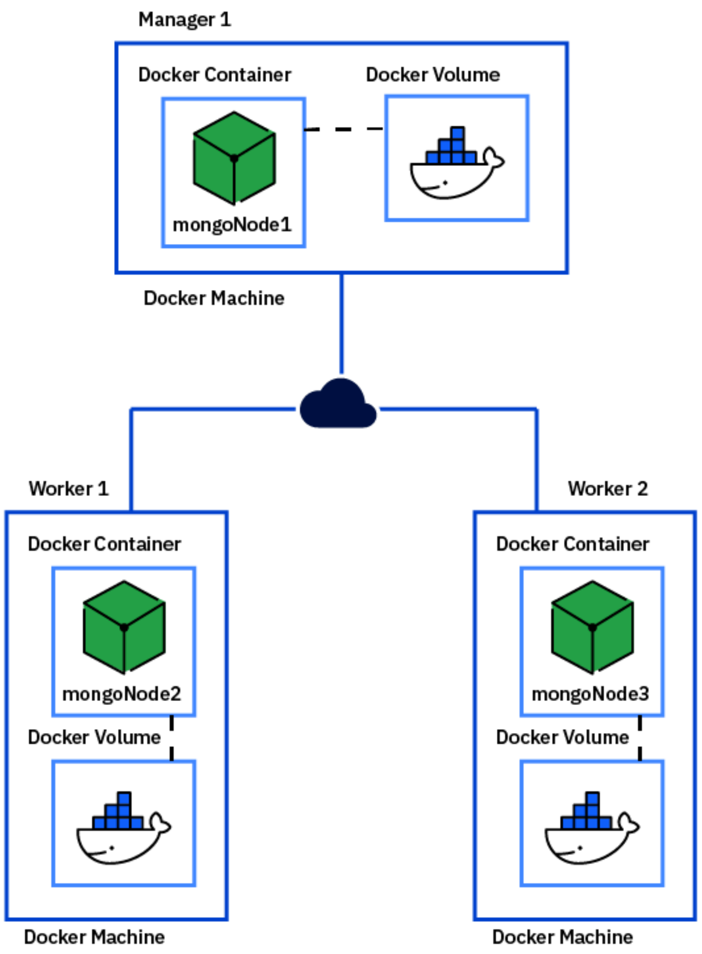
Challenges

Limited control: Organizations have less control over underlying infrastructure, which might be a concern for some organizations.

Data security: Trusting a third-party provider with sensitive data raises security and privacy considerations.

Containerized deployment

Containerization involves packaging the database and its dependencies into containers for consistent deployment across various environments.



Example

A tech company adopts Docker containers to deploy its microservices-based application, with each microservice encapsulating a specific function and a separate container running a MongoDB database.

Advantages

Portability: Containers ensure consistent operation across development, testing, and production environments.

Resource efficiency: Lightweight containers consume fewer resources compared to traditional virtual machines.

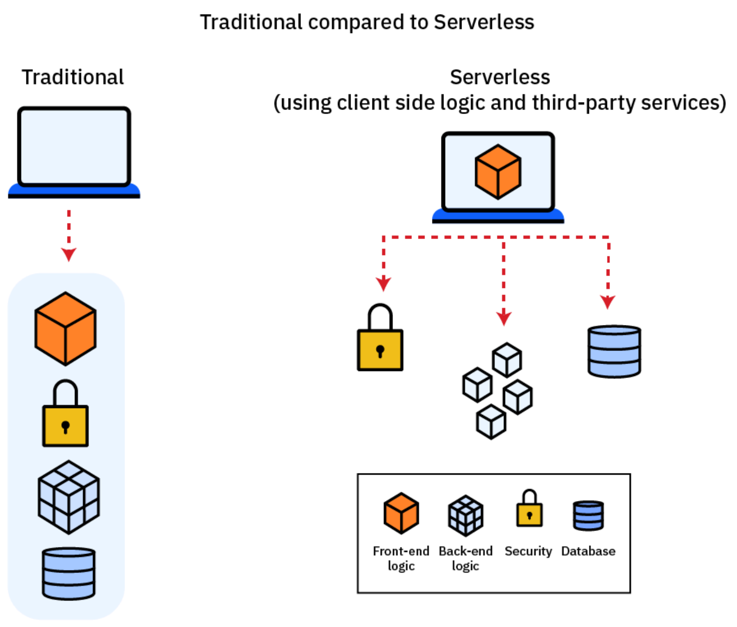
Challenges

Orchestration complexity: Production-scale deployment requires knowledge of container orchestration tools like Kubernetes.

Learning curve: Teams might need to familiarize themselves with containerization concepts and tools.

Serverless deployment

Serverless architecture provisions and scales database resources automatically based on demand, with organizations paying for actual usage.


Traditional compared to Serverless
Traditional
Serverless
(using client side logic and third-party services)
Front-end logic
Back-end logic
Security
Database

Example

A mobile app developer utilizes Google Cloud Firestore as a serverless NoSQL database for storing user data, where the database scales automatically based on demand, and the developer only pays for the data storage and operations used by the app.

Advantages

No manual provisioning: No need for manual provisioning or maintenance of server resources.

Cost savings: Efficient resource utilization leads to cost savings as organizations only pay for the resources consumed.

Challenges

Limited control: Organizations have less control over underlying infrastructure, which might concern some organizations.

Applicability: Serverless might not be suitable for all types of databases or workloads due to architectural constraints.

Recap

In this reading, you learned that:

- The choice of a database deployment option depends on organizational priorities, budget constraints, scalability needs, and how the company manages the nature of the data. It's essential to carefully evaluate these options in alignment with specific use cases to ensure optimal performance and efficiency.
- On-premises deployment involves hosting the entire database infrastructure within the organization's physical location or a dedicated data center, providing full control and compliance with regulatory requirements but with high upfront costs and limited scalability.
- Cloud deployment uses cloud service providers to host and manage databases. Cloud deployment offers scalability, cost-effectiveness, and automation. However, cloud deployments have connectivity dependencies and can be prone to security concerns.
- Hybrid deployment combines on-premises and cloud solutions, allowing organizations to distribute data and workloads based on specific needs, providing flexibility and enhanced disaster recovery capabilities; however, effective integration between on-premises and cloud environments requires increased administrative complexity and costs.
- Database as a Service (DBaaS) provides a fully managed database solution where the company's choice service provider handles administrative tasks. When using DBaaS, organizations focus on application development but usually have limited control over underlying infrastructure and data security.
- Containerized deployment involves packaging the database and its dependencies into containers for consistent deployment across various environments, which can ensure consistent operations and resource efficiency—however, if containerization practices are not already in place, the company will need time for employees to gain knowledge of container orchestration tools and implement them.
- Serverless deployment provisions and scales database resources automatically based on demand, with organizations paying for actual usage, eliminating manual provisioning and maintenance of server resources, and leading to cost savings. However, companies usually have limited control over the underlying infrastructure, and serverless deployments might not be technically suitable for all databases or workloads.

Congratulations! You have completed this reading and are ready for the next topic.

Author(s)

- Richa Arora

Glossary: Basics of NoSQL

Welcome! This alphabetized glossary contains many of the terms you'll find within this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are important for you to recognize when working in the industry, participating in user groups, and participating in other certificate programs.

Term	Definition
ACID	This term is an acronym for Atomicity, Consistency, Isolation, and Durability, which is a set of properties that guarantee reliable processing of database transactions in traditional relational databases.
Atomic	In the context of database transactions, atomic means that an operation is indivisible and either completes fully or is completely rolled back. It ensures that the database remains in a consistent state.
BASE	An alternative to ACID. Stands for basically available, soft state, eventually consistent. BASE allows for greater system availability and scalability, sacrificing strict consistency in favor of performance.
Bigtable	A NoSQL database system developed by Google, designed for handling large amounts of data and providing high performance, scalability, and fault tolerance.
Caching	The temporary storage of frequently accessed data in high-speed memory reduces the need to fetch the data from the primary storage, which can significantly improve response times.
Cluster	A group of interconnected servers or nodes that work together to store and manage data in a NoSQL database, providing high availability and fault tolerance.
Column database	A NoSQL database model that stores data in column families rather than tables, making it suitable for storing and querying vast amounts of data with high scalability. Examples include Apache Cassandra and HBase.
CRUD	CRUD is an acronym for create, read, update, and delete, which are the basic operations for the basic operations for interacting with and manipulating data in a database.
DBaaS	This acronym stands for database as a service, a cloud-based service that provides managed database hosting, maintenance, and scalability, allowing users to focus on application development without managing the database infrastructure.
Document	A NoSQL database model that stores data in semi-structured documents, often in formats like JSON or BSON. These documents can vary in structure and are typically grouped within collections.
Graph database	A NoSQL database model optimized for storing and querying data with complex relationships, represented as nodes and edges. Examples include Neo4j and OrientDB.
Horizontal scaling	The process of adding more machines or nodes to a NoSQL database to improve its performance and capacity. This is typically achieved through techniques like sharding.
Indexing	The creation of data structures that improve query performance by allowing the database to quickly locate specific records based on certain fields or columns.
JSON	JSON is an acronym for JavaScript Object Notation, a lightweight data-interchange format used in NoSQL databases and other data systems. JSON is human-readable and easy for machines to parse.
Key-value	A NoSQL database model that stores data as key-value pairs. It's a simple and efficient way to store and retrieve data where each key is associated with a value.
Normalized	A database design practice where data is organized to minimize redundancy and maintain data integrity by breaking it into separate tables and forming relationships between them.
NoSQL	NoSQL stands for "not only SQL." A type of database that provides storage and retrieval of data that is modeled in ways other than the traditional relational tabular databases.
Sharding	Refers to the practice of partitioning a database into smaller, more manageable pieces called shards to distribute data across multiple servers. Sharding helps with horizontal scaling.
TTL	Stands for "Time to Live," which is a setting in NoSQL databases that determines how long a piece of data should be retained before it's automatically removed from the database.
XML	Stands for Extensible Markup Language, another data interchange format used in some NoSQL databases. It's also human-readable and can represent structured data.

Skills Network

`