# Summary:

* Amazon DocumentDB will be used as our final database, migrating from MongoDB. It provides scalability, reliability, and compatibility with MongoDB applications, making it suitable for our project.
* Research on Amazon RDS and Amazon DynamoDB shows both are viable alternatives for specific use cases, offering distinct advantages in relational and NoSQL environments.
* Using Docker for the database setup ensures consistency, portability, and flexibility in development and production.

1. **Types of AWS Databases:**

AWS provides several types of managed databases, each suited for different use cases.

**Amazon DocumentDB (Preferred Database):**

* Fully managed NoSQL document database service designed for modern apps.
* Compatible with MongoDB APIs, enabling seamless migration.
* Automatically scales up storage as needed, reducing maintenance.
* Multi-AZ support for high availability and durability.
* Best suited for semi-structured data and applications already using MongoDB.

**Amazon RDS (Relational Database Service):**

* Fully managed service for relational databases like MySQL, PostgreSQL, and more.
* Features automatic backups, replication, and monitoring.
* Suitable for structured, transactional data.
* New customers enjoy 12 months of Free Tier benefits:
  + 750 hours of Single-AZ instance usage.
  + 20 GB of SSD storage and backups.

**Amazon DynamoDB:**

* Serverless NoSQL database for applications requiring low-latency, high-performance data access.
* Automatically scales to handle millions of requests per second.
* Free Tier includes:
  + 25 GB of storage.
  + 25 read/write capacity units.
* Ideal for key-value and document-based workloads but not optimized for relational data.

1. **Why would we use AWS for our Project?**

* **Scalability:** As our dashboard might see changes in the traffic loads, AWS databases are highly scalable, which means they can handle varying amounts of data and traffic without performance degradation.
* **Reliability and Availability:** AWS provides features like multi-region replication, automatic backups, and failover, ensuring high availability of our database even in case of failures.
* **Security:** AWS offers encryption at rest and in transit, IAM (Identity and Access Management), and VPC (Virtual Private Cloud) isolation to protect our data.
* **Future-proofing**: AWS constantly evolves, adding new features to stay competitive.

1. **Why Consider Amazon RDS or DynamoDB?**

**Amazon RDS:**

* Best for applications needing strong relational data consistency.
* Simplifies management with automatic updates and monitoring.
* Ideal for transactional data and structured queries.

**Amazon DynamoDB:**

* Perfect for applications with unpredictable traffic spikes.
* Eliminates the need for database schema design.
* Supports high-speed operations for key-value or document-based access.

1. **Migrating from MongoDB to Amazon DocumentDB:**

* **Migration Steps:**
  1. Export current MongoDB collections using mongodump.
  2. Create a new Amazon DocumentDB cluster and configure networking.
  3. Use mongorestore to import data into DocumentDB.
  4. Update application configurations to point to the DocumentDB connection endpoint.
* **Benefits:**
  1. Improved performance for read-heavy workloads.
  2. Managed scaling and maintenance.

1. **Is AWS Easily Accessible?**

Yes, AWS is accessible through:

* **AWS Management Console:** A web-based interface to manage and monitor our services.
* **AWS CLI (Command Line Interface):** For developers who prefer working with the terminal.
* **AWS SDKs:** AWS provides Java SDKs that you can use to interact programmatically with AWS services directly from your Spring Boot app.

1. **Is It Future-Proof?**

Yes, AWS is a market leader and constantly evolves, introducing new features and services to ensure we stay up-to-date with technological advancements.

1. **Can We Give Our Project to the Client Later?**

Yes, we can. AWS offers robust support and documentation, making it easier to transfer control to the client. We can set up the infrastructure on AWS, and the client can manage it via the AWS Console, with us providing necessary documentation for setup, security, and access.

1. **Setting Up a Database Using Docker:**

Docker provides flexibility in setting up databases for local development and production.

Steps to Set Up Amazon DocumentDB Locally Using Docker:

**1. Hosting the Database in Docker on AWS**

We can containerize our MySQL database and run it in a Docker container. Here are the main options for deploying this container in AWS:

* 1. **Create a Docker Container:**
     + Use the official MongoDB image to simulate DocumentDB for local testing:

```docker run --name mongo-container -d -p 27017:27017 mongo```

* Alternatively, use AWS-provided DocumentDB-compatible containers for enhanced compatibility.
  1. **Persist Data:**
     + Mount a volume to ensure data is retained:

``` docker run --name mongo-container -d -p 27017:27017 -v ~/mongo-data:/data/db mongo```

* 1. **Access and Test:**
     + Use MongoDB clients like mongosh or an application to connect to localhost:27017.
  2. **Deploy in AWS:**
     + Use Amazon ECS or EKS to manage containerized instances for production.

1. **Why Use Docker for the Database?**

* **Consistency Across Environments:** The Docker container ensures the same MySQL configuration works locally and in production.
* **Flexibility in Hosting:** We can move the container to any environment (e.g., local, AWS EC2, AWS ECS).
* **Portability:** Makes it easier to switch hosting solutions if needed.
* **Customization:** We can tweak the MySQL configuration directly in the Docker container if necessary.

1. **Challenges and Considerations**

* **Using Docker:**
  + **Networking:** Ensure database containers are accessible to the application using correct port configurations and AWS Security Groups.
  + **Persistence:** Attach volumes (e.g., EBS or EFS) to avoid data loss.
  + **Scalability:** Scaling a containerized database is manual compared to AWS-managed services.
  + **Monitoring:** Set up tools like AWS CloudWatch to monitor performance and logs.
* **Using Amazon DocumentDB:**
  + **Cost:** Pricing is based on instance size, storage, and IO operations.
  + **Migration Effort:** Requires careful planning to ensure data consistency.

1. **Comparison:** Dockerized Database vs. AWS DocumentDB

|  |  |  |
| --- | --- | --- |
| **Feature** | **Docker on AWS EC2/ECS/EKS** | **AWS DocumentDB** |
| Setup Complexity | Higher (manual setup required) | Lower (fully managed) |
| Scalability | Manual scaling | Automatic |
| Cost | Potentially lower | Higher for managed convenience |
| Customization | Fully customizable | Limited to DocumentDB configs |
| Maintenance | User-managed | AWS-managed |

1. **Is using docker Future-Proof?**

Yes, Docker is future-proof due to:

* **Portability**: Containers can be moved across environments.
* **Flexibility**: Supports various configurations and hosting options.

However, **managed services like Amazon DocumentDB** provide better long-term scalability and reduced maintenance overhead.