Research A Backend Server to Store Data that the App Access

By Jan Baraniecki

Index: PG

1. Goals Of This Document

The goal of this document is to provide a comparison between three back end server options to support our phone app project. We are looking at Firebase Realtime Database, AWS Services(DynamoDB), and Google Cloud Firestone.

2. Terms and Definitions

Amazon Web Services (AWS):

1. **EC2 (Elastic Compute Cloud):** AWS service that provides resizable compute capacity in the cloud. EC2 instances are virtual servers that can be configured to run applications.
2. **S3 (Simple Storage Service):** A scalable object storage service used to store and retrieve data. It's often used for data backup, static website hosting, and data archiving.
3. **DynamoDB:** A NoSQL database service that offers high performance and seamless scalability. It's suitable for applications that require flexible, low-latency data storage.
4. **RDS (Relational Database Service):** A managed service for running relational databases such as MySQL, PostgreSQL, or SQL Server in the cloud.
5. **Aurora:** A MySQL and PostgreSQL-compatible relational database built for the cloud, known for its high performance, availability, and scalability.
6. **Lambda:** A serverless computing service that lets you run code without provisioning or managing servers. You can use it for event-driven functions.
7. **IAM (Identity and Access Management):** AWS service for managing users, groups, and permissions to control access to AWS resources.
8. **VPC (Virtual Private Cloud):** A private network within AWS that allows you to isolate resources and control network settings.

9. **Elastic Load Balancer (ELB):** Distributes incoming traffic across multiple EC2 instances to ensure high availability and fault tolerance.
10. **Auto Scaling:** Automatically adjusts the number of EC2 instances to maintain application performance and availability.

Google Cloud Firestore:

1. **Document:** In Firestore, data is organized into documents, which are JSON-like objects that contain key-value pairs.
2. **Collection:** A collection is a group of related documents. It's similar to a table in a relational database.
3. **Indexing:** Firestore automatically indexes fields in your documents to support efficient queries.
4. **Security Rules:** Firestore uses security rules to define who can read and write data in the database.
5. **Real-Time Updates:** Firestore provides real-time data synchronization, meaning that changes to data are automatically pushed to connected clients.
6. **NoSQL:** Firestore is a NoSQL database, which means it is schema-less and offers more flexibility in data storage.

Firebase Realtime Database:

1. **JSON Data Structure:** Firebase Realtime Database uses a JSON data model, where data is organized into a tree-like structure with nodes and key-value pairs.
2. **Data Synchronization:** Data in the Firebase Realtime Database is automatically synchronized across clients in real-time.
3. **References:** To access and manipulate data, you create references to specific parts of the database.
4. **Authentication:** Firebase offers user authentication services, allowing you to secure your data and define access rules.
5. **Rules:** Similar to Firestore's security rules, Firebase has security rules that define who can read and write data.
6. **Websockets:** Firebase uses WebSockets for real-time data synchronization, enabling efficient data updates.
7. **Offline Persistence:** Firebase supports offline data access and synchronization, allowing your app to work even when offline.
8. **Scalability:** While Firebase Realtime Database can scale to accommodate many users, its scaling model is simpler compared to Firestore.

3. Comparisons

Firebase Realtime Database:

Strengths:

- **Real-Time Updates:** Keeps data synced across users in real-time.
- **Easy for Beginners:** Simple and user-friendly for developers with limited backend experience.
- **Serverless:** No need to manage servers; it's fully handled.
- **Scalable:** Works well for small to medium-sized apps.

Weaknesses:

- **Limited Queries:** Not great for complex queries.
- **Data Structure Planning:** Requires careful data structure design.
- **NoSQL Only:** May not be suitable for highly structured data.

AWS Services (DynamoDB, RDS, Aurora):

Strengths:

- **Versatile:** Offers various database options (NoSQL and relational) for different data needs.
- **Highly Scalable:** Handles large data volumes and traffic.
- **Customizable:** You can fine-tune configurations to meet specific requirements.
- **Secure:** Strong security features.

Weaknesses:

- **Complex Setup:** Can be challenging for newcomers due to its complexity.
- **Costly If Not Managed:** Costs can rise as your app grows.
- **Requires Management:** You need to oversee and maintain the databases.

Google Cloud Firestore:

Strengths:

- **User-Friendly:** Designed to be easy for developers.
- **Scalable:** Works for small to large apps and scales easily.
- **Serverless:** Managed by Google, allowing you to focus on your app.
- **Structured Data:** Supports organized data storage with complex queries.

Weaknesses:

- **Costly at Scale:** As your app grows, costs may increase.
- **Vendor Lock-in:** Ties you to Google Cloud.
- **Query Costs:** Charges for reads, writes, and queries, which can affect expenses as your app expands.

4. Considerations

We should consider, due to no budget, free options that can host data. The app can scrape news or announcements from our own free webpages or we can use a form of rss to update our app too.

Using google drive or dropbox might be a good option to host files for the app to grab too.

Unfortunately host options will scale up on our heavy uses, but if possible we should try and keep everything under hood rather than using these techniques mentioned above.