Web of Influence Research

Web Server Integration and Cloud Deployment: Testing and Findings

Krystal Ng

Table of Contents

1. MySQL Testing Evaluation

Avien

Service Overview: https://console.aiven.io/account/a54ba5702514/project/nkrystal-0b61/services/mysql-woi/overview

- Requires a certificate to be added to the repository for access, which limits seamless remote access across multiple machines.
- Pros: Allows whitelisting of all IP addresses for easier access.

Freesqldatabase.com

- Unable to establish a connection because IP whitelisting was not supported.
- As a result, the service was not usable for this project.

FreeDB

- Similar issues to FreeSQLDatabase; access from different IPs was not permitted, and IP whitelisting was not supported.

Amazon wed services (AWS)

- Required a payment method to continue using the free-tier instance.
- Not ideal for a cost-free development setup.
- Pros: Good option for future expansion due to scalability and reliability.

PlanetScale

- Service required an active paid plan to remain operational.
- Free trial access expired, making it unsuitable for long-term testing without payment.

2. Front-End

GitHub Pages

- Enables hosting directly from a GitHub repository, allowing the project to be shared via a public URL.
- Converts static files in the repository into a static site, making it easy to deploy and maintain.
- Supports custom build processes either locally or on an external server before deployment.

Application Frontends

Any web frontend (React, Angular, Vue, etc.) or backend framework (Flask, Django, Node.js) can work with Cloud SQL.

Connection Options:

- Public IP (with SSL): Secure external access.
- Private IP (within GCP VPC): For internal communication within GCP.
- Cloud SQL Auth Proxy: Simplest option for local development.

Hosting Options:

- Google Firebase Hosting
    - Ideal for React or other static sites.
    - Simple CLI deployment (firebase deploy).
    - Integrates seamlessly if already using GCP.
- Google App Engine
    - Supports both frontend and backend (Flask, Node.js, etc.).
    - Free tier available for small-scale traffic.
- Google Cloud Storage (Static Websites)
    - Host static files (e.g. React build) in a storage bucket.
    - Generous free-tier bandwidth.
- Render (Free Tier)
    - Supports static frontends and backend APIs.
    - Strong free-tier offering for development projects.

- AWS Amplify
    - Similar to Firebase Hosting.
    - Good for serverless hosting and easy integration with AWS ecosystem.

3. Back-End

Render.com

- Cloud application platform that enables building, deploying, and scaling applications.
- Used to run the API backend of the application, which was successfully deployed using this service.
- Provides a straightforward deployment process with integration directly from the repository.
- Links:
    - Dashboard made for testing: https://dashboard.render.com/
    - Deployment status for testing: https://dashboard.render.com/web/srv-d1vkfoer433s73fp0qa0/deploys/dep-d21ba7nfte5s73fdnls0
    - Live API for testing: https://webofinfluenceresearch.onrender.com/

4. Research

Cloud Application Services for Future Use

Google Cloud Platform (GCP)

- Pricing Model: Subscription-based (fixed monthly or yearly fee for access to services).
- Key Features:
    - Virtual machines and internal networking.
    - Computing, data storage, and data analytics.
    - Machine learning and AI tools.
    - Comprehensive management and monitoring tools.

Amazon Web Services (AWS)

- Pricing Model: Pay-as-you-go.
- Key Features:
    - Wide range of cloud services for computing, storage, and networking.
    - Scalable infrastructure for small to enterprise-level projects.
    - Strong ecosystem for integrations and developer tools.

4.1 Hosting API for a MySQL Database with Costs

You can host a MySQL database either using a virtual machine or a managed cloud database service. The two most popular options are Amazon RDS and Google Cloud SQL.

Cost Estimates

- Amazon RDS (db.t3.micro, 20 GB SSD):
  ~ $20–25 USD/month (on-demand)
  https://aws.amazon.com/getting-started/hands-on/create-mysql-db/?p=ft&c=db&refid=4b29d1be-0c31-4263-8d6f-b7bd712a7a71
- Google Cloud SQL (db-f1-micro, 20 GB SSD, 20 GB backups):
  ~ $17–18 USD/month
  https://cloud.google.com/free

Cost Summary Table

Service	Instance Type	Storage	Backup Storage	Estimated Cost (Monthly)
Amazon RDS	db.t3.micro	20 GB SSD	Included	~ $20–25 USD
Google Cloud SQL	db.f1.micro	20 GB SSD	20 GB	~ $17–18 USD

4.2 Amazon RDS vs Google Cloud SQL

Google Cloud SQL

- Beginner-friendly: Start a MySQL instance with just a few clicks in the console.
- Simple setup with fewer configuration options compared to AWS.
- Automatic backups, scaling, and failover are straightforward to enable.
- Easy integration with other GCP services.
- Clear pricing model: Compute + Storage + Backups.

Amazon RDS

- More configuration options for advanced uses.
- Steeper learning curve due to higher complexity.
- Greater flexibility for custom networking and fine-grained control.
- Pricing includes more variables (e.g., IOPS, networking).

Comparison Summary Table

Feature	Google Cloud SQL	Amazon RDS
Ease of Setup	Very beginner-friendly, few clicks	More complex, requires more configuration
Configuration Options	Limited but simplified	Extensive and flexible
Automatic Backups	Easy to enable	Available but requires more setup
Scaling	Simple vertical and horizontal scaling	Advanced scaling options
Networking Control	Basic, integrated with GCP VPC	Fine-grained networking configuration
Integration	Seamless with GCP services	Strong AWS ecosystem integration

Pricing Model	Clear (Compute + Storage + Backups)	Variable (Compute, Storage, IOPS, etc.)
Learning Curve	Low	Higher due to complexity
Best For	Dev/test instances and simple setups	Advanced production environments

✅ Recommendation

- If your priority is ease of setup and management (especially for dev/test environments): Google Cloud SQL is the better choice.
- If you need maximum control, custom networking, or advanced scaling: AWS RDS is more powerful but requires more expertise.

5. Snowflake

Architecture

Snowflake is a fully managed cloud data platform that separates storage, compute, and services for flexible scaling and simplified management.

Snowflake is not a traditional VM or DB hosting solution—it's a managed data warehouse-as-a-service designed for analytics rather than transactional APIs.

Layer	Description
Cloud Storage Layer	Stores all data in a columnar format on AWS S3, Azure Blob, or Google Cloud Storage. Managed entirely by Snowflake.
Compute Layer (Virtual Warehouses)	Independent compute clusters for running queries. Multiple warehouses can access the same data without contention.
Cloud Services Layer	Manages authentication, metadata, query optimization, and transaction handling. No infrastructure management needed.
API/Client Access Layer	Applications, BI tools, and APIs connect securely to Snowflake through standard drivers (ODBC/JDBC, REST API).

Access

- Web UI: Manage databases, create warehouses, run queries, and monitor performance.
- SQL Clients & BI Tools: Connect using JDBC/ODBC drivers (e.g., Tableau, Power BI).
- Snowflake Connector for Python/Node.js: Programmatic access for developers.
- REST API: For automation and custom integrations.
- External Functions: Call external APIs from Snowflake queries for data enrichment.

Hosting Workflow

Data Ingestion

- Load structured data (CSV, JSON, Parquet) from AWS S3, Google Cloud Storage, or Azure Blob.

- Use Snowpipe for continuous data loading.

Storage

- Snowflake manages compressed, columnar storage automatically.

Compute

- Create one or more virtual warehouses for processing queries.
- Scale warehouses independently (e.g., small for dev, large for analytics jobs).

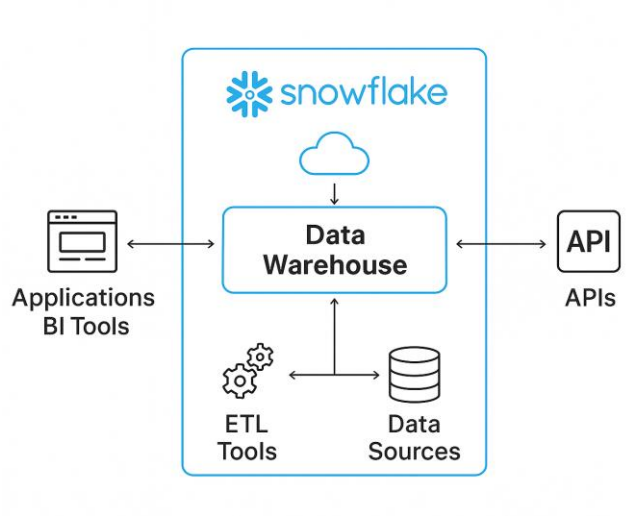
API/Integration

- Expose data to apps via Snowflake SQL endpoints or external REST APIs.
- Optionally integrate with tools like Fivetran, dbt, or Airflow for pipelines.

Evaluation – Pros & Cons

Pros	Cons
Fully managed (no infrastructure)	Not ideal for OLTP or transactional apps
Automatic scaling (compute & storage)	Pricing can increase with heavy compute
Supports multiple clouds (AWS, GCP, Azure)	Vendor lock-in on Snowflake's platform
Easy integration with analytics tools	Limited custom network-level control
Separation of storage and compute	Requires SQL knowledge for most use cases
Time travel & fail-safe for data recovery	API is mainly for analytics, not high-frequency REST workloads

Diagram – How it works



API → Application Programming Interface
A set of rules and tools that allow different software applications to communicate with each other.

BI → Business Intelligence
Tools and processes used to analyze data and present actionable information to help organizations make informed decisions.

ETL → Extract, Transform, Load
A process in data integration where data is extracted from various sources, transformed into the desired format, and loaded into a data warehouse or other system.

The project requires both a front end and a back end, but Snowflake is only used for data storage, would need to structure it so that each layer handles a specific role.

The front end is responsible for the user interface, allowing users to interact with the application, while the back end serves as the bridge between the front end and Snowflake.

The back end manages business logic, authentication, and API endpoints, and it connects to Snowflake using a driver or connector to run SQL queries.

Snowflake, in this setup, functions purely as a data warehouse, optimized for analytics rather than direct user interaction or transactional operations. The backend retrieves data from Snowflake, formats it, and sends it back to the front end, where it can be displayed visually.

This separation of responsibilities ensures a clean, scalable architecture where Snowflake handles storage, the backend manages logic, and the frontend focuses on delivering a smooth user experience.

Recommendations

✅ Best Use Cases for Snowflake

- Data warehousing and analytics
- Business intelligence dashboards
- Machine learning model training (via integrations)
- Cross-cloud data sharing and collaboration

❌ Not Ideal For

- Traditional app database hosting (e.g., MySQL-like API with frequent small transactions)
- APIs requiring low-latency OLTP workloads