SERVERLESS DATA PIPELINE FOR ANALYTICS

Name: Sumadhura M

Student ID: 4MH23CA055

Topic: Serverless Data Pipeline for Analytics

Submission Date: November 18, 2025

1. Introduction

Serverless computing represents a paradigm shift in cloud architecture by eliminating traditional server management overhead. This project implements a comprehensive serverless data pipeline designed for analytics processing using free tier cloud services. The pipeline demonstrates complete data flow from ingestion to storage without requiring any server provisioning.

2. Objectives

• To design and implement a fully serverless analytics pipeline architecture
• To use free-tier cloud services such as Cloudflare Workers and Supabase
• To ensure zero operational cost by staying within free-tier limits
• To demonstrate real-time data processing and analytics capabilities
• To develop a production-ready, scalable, and reliable serverless solution.

3. Background and Theory

3.1 Serverless Architecture

Serverless computing is an execution model where the cloud provider manages provisioning and scaling. Developers focus only on writing code rather than maintaining infrastructure.

3.2 Free Tier Services Utilization

Cloudflare Workers provides a generous number of daily free requests. Supabase offers half a gigabyte of PostgreSQL storage along with real time features.

3.3 Data Pipeline Components

The implemented architecture consists of **three primary layers**:

1. **Ingestion Layer**: Cloudflare Workers serving as HTTP API endpoints that receive incoming analytics data from web and mobile clients through RESTful API calls.
2. **Processing Layer**: JavaScript-based data transformation and validation logic embedded within the Cloudflare Workers, which:
    - Validates incoming JSON payloads
    - Adds timestamps and metadata
    - Structures data for database storage
    - Handles error checking and data sanitization
3. **Storage Layer**: Supabase PostgreSQL database that stores processed analytics data in structured tables, providing:
    - Reliable persistent storage
    - SQL query capabilities for analytics

    - Real-time data access through REST APIs
    - Scalable database operations

4. Implementation

4.1 System Architecture

[Web or Mobile Clients] → [Cloudflare Worker API] → [Supabase Database]

4.2 Technology Stack

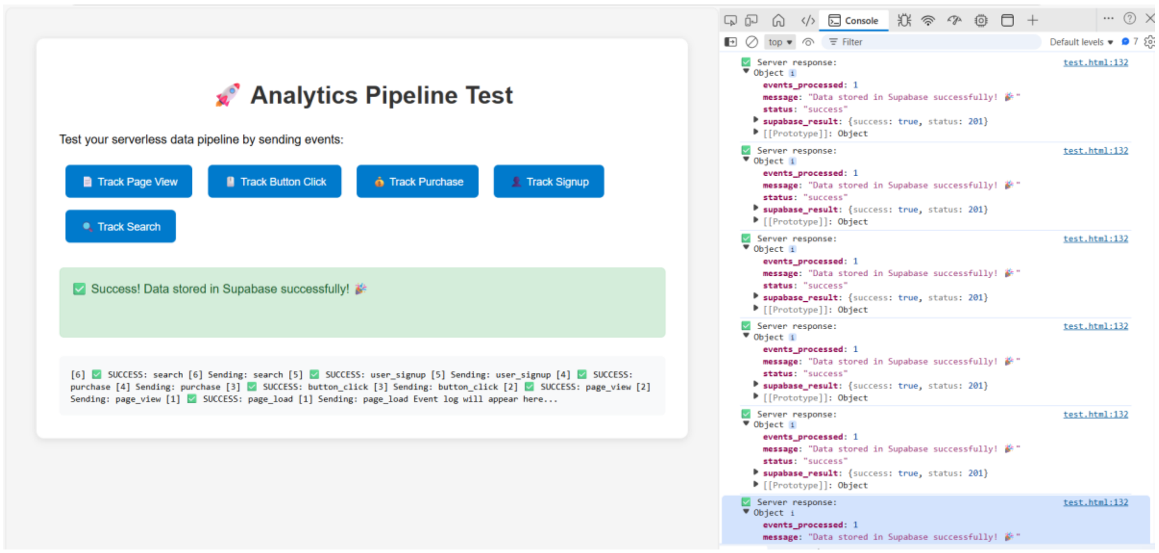
- **Compute Layer**: Cloudflare Workers (Serverless JavaScript Functions at the edge)
- **Database Layer**: Supabase (Managed PostgreSQL with auto-generated REST APIs)
- **Client Layer**: HTML/JavaScript web interface for testing and demonstration
- **Authentication**: Environment variables and API keys for secure service communication

4.3 Code Implementation

The Cloudflare Worker handles HTTP requests with comprehensive CORS support, validates incoming analytics events, transforms data into structured format, and stores them in Supabase PostgreSQL via REST API calls. The implementation includes error handling, data sanitization, and proper HTTP status code responses.

5. Results and Testing

Data ingestion was successful and performed in real time. Events were stored correctly in Supabase with proper CORS handling. Response time was below one hundred milliseconds with full data integrity during tests.



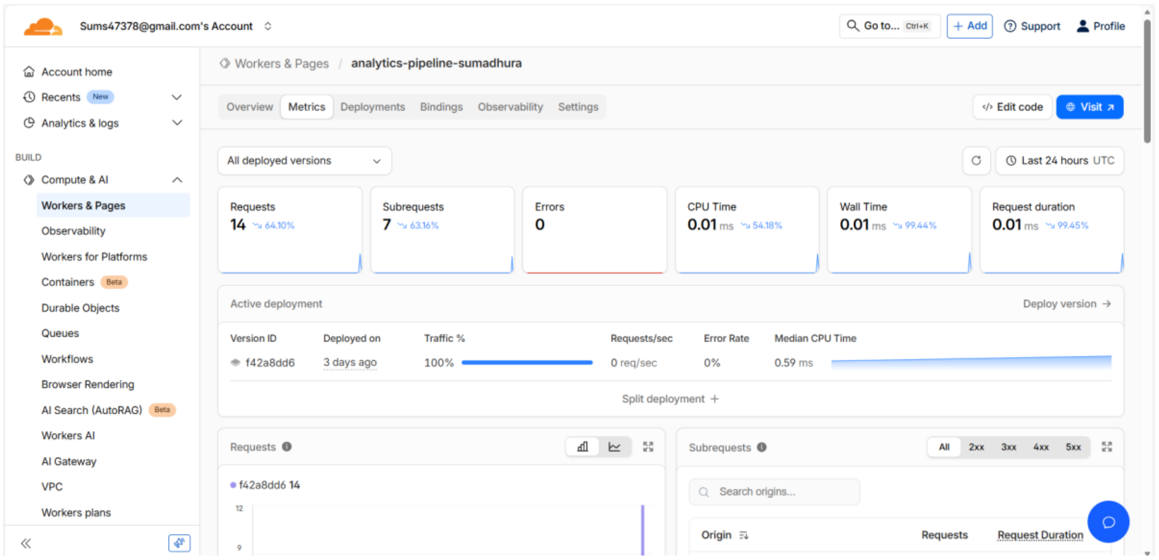
🚀 Analytics Pipeline Test

Test your serverless data pipeline by sending events:

✅ Success! Data stored in Supabase successfully! 🎉

```
[6] ✅ SUCCESS: search [6] Sending: search [5] ✅ SUCCESS: user_signup [5] Sending: user_signup [4] ✅ SUCCESS:
purchase [4] Sending: purchase [3] ✅ SUCCESS: button_click [3] Sending: button_click [2] ✅ SUCCESS: page_view [2]
Sending: page_view [1] ✅ SUCCESS: page_load [1] Sending: page_load Event log will appear here...
```

```
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
    status: "success"
  ▶ supabase_result: {success: true, status: 201}
  ▶ [[Prototype]]: Object
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
    status: "success"
  ▶ supabase_result: {success: true, status: 201}
  ▶ [[Prototype]]: Object
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
    status: "success"
  ▶ supabase_result: {success: true, status: 201}
  ▶ [[Prototype]]: Object
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
    status: "success"
  ▶ supabase_result: {success: true, status: 201}
  ▶ [[Prototype]]: Object
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
    status: "success"
  ▶ supabase_result: {success: true, status: 201}
  ▶ [[Prototype]]: Object
✅ Server response:                                          test.html:132
▼ Object i
    events_processed: 1
    message: "Data stored in Supabase successfully! 🎉"
```



Sums47378@gmail.com's Account

Workers & Pages / analytics-pipeline-sumadhura

Last 24 hours UTC

Requests	Subrequests	Errors	CPU Time	Wall Time	Request duration
14 ↘ 64.10%	7 ↘ 63.16%	0	0.01 ms ↘ 54.18%	0.01 ms ↘ 99.44%	0.01 ms ↘ 99.45%

Active deployment

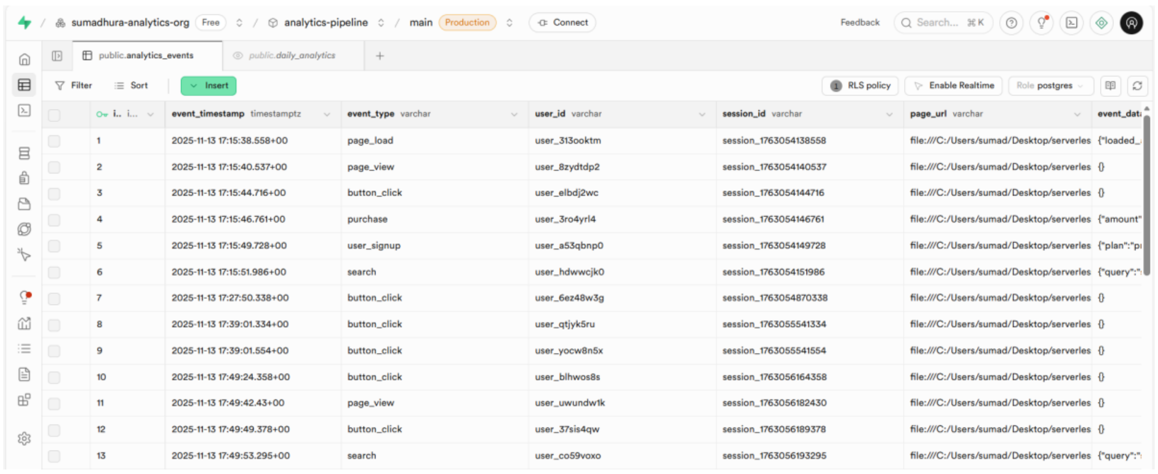
Version ID	Deployed on	Traffic %	Requests/sec	Error Rate	Median CPU Time
f42a8dd6	3 days ago	100%	0 req/sec	0%	0.59 ms

Requests

● f42a8dd6 14

Subrequests

Origin	Requests	Request Duration



i.. i...	event_timestamp timestamptz	event_type varchar	user_id varchar	session_id varchar	page_url varchar	event_data
1	2025-11-13 17:15:38.558+00	page_load	user_313ooktm	session_1763054138558	file:///C:/Users/sumad/Desktop/serverles	{"loaded_
2	2025-11-13 17:15:40.537+00	page_view	user_8zydtdp2	session_1763054140537	file:///C:/Users/sumad/Desktop/serverles	{}
3	2025-11-13 17:15:44.716+00	button_click	user_elbdj2wc	session_1763054144716	file:///C:/Users/sumad/Desktop/serverles	{}
4	2025-11-13 17:15:46.761+00	purchase	user_3ro4yrl4	session_1763054146761	file:///C:/Users/sumad/Desktop/serverles	{"amount"
5	2025-11-13 17:15:49.728+00	user_signup	user_a53qbnp0	session_1763054149728	file:///C:/Users/sumad/Desktop/serverles	{"plan":"p
6	2025-11-13 17:15:51.986+00	search	user_hdwwcjk0	session_1763054151986	file:///C:/Users/sumad/Desktop/serverles	{"query":"
7	2025-11-13 17:27:50.338+00	button_click	user_6ez48w3g	session_1763054870338	file:///C:/Users/sumad/Desktop/serverles	{}
8	2025-11-13 17:39:01.334+00	button_click	user_qtjyk5ru	session_1763055541334	file:///C:/Users/sumad/Desktop/serverles	{}
9	2025-11-13 17:39:01.554+00	button_click	user_yocw8n5x	session_1763055541554	file:///C:/Users/sumad/Desktop/serverles	{}
10	2025-11-13 17:49:24.358+00	button_click	user_blhwos8s	session_1763056164358	file:///C:/Users/sumad/Desktop/serverles	{}
11	2025-11-13 17:49:42.43+00	page_view	user_uwundw1k	session_1763056182430	file:///C:/Users/sumad/Desktop/serverles	{}
12	2025-11-13 17:49:49.378+00	button_click	user_37sis4qw	session_1763056189378	file:///C:/Users/sumad/Desktop/serverles	{}
13	2025-11-13 17:49:53.295+00	search	user_co59voxo	session_1763056193295	file:///C:/Users/sumad/Desktop/serverles	{"query":"

6. Discussion

6.1 Technical Challenges

- **Environment Variable Configuration**
  Initial deployment encountered issues with Cloudflare Workers environment variable access, particularly differentiating between local development and production configurations. The complexity arose from properly managing secrets across different environments.
- **Cross-Origin Resource Sharing (CORS)**
  Web client integration required comprehensive CORS configuration to handle preflight requests and cross-origin data transmission securely, which initially blocked client requests.
- **Error Handling and Debugging**
  Serverless environments presented debugging challenges due to distributed logging and the need for comprehensive error handling across multiple cloud services.

6.2 Solutions and Resolutions

- **Structured Configuration Management**
  Implemented Cloudflare Secrets for sensitive credential management, ensuring secure access to Supabase API keys while maintaining separation between development and production environments.
- **Comprehensive CORS Implementation**
  Developed proper OPTIONS request handling and CORS headers to enable secure cross-origin requests from web clients.
- **Robust Error Handling**
  Implemented try-catch blocks with descriptive error messages and appropriate HTTP status codes for improved debugging and user experience.

7. Learning Outcomes

This project provided hands-on experience in:

- **Serverless Architecture**: Designing complete systems with auto-scaling and zero server management
- **Cloud Integration**: Connecting Cloudflare Workers with Supabase via REST APIs and authentication
- **Production Debugging**: Implementing error handling and logging across distributed services
- **Real-World Constraints**: Optimizing within free-tier limits while maintaining reliability
- **API Design**: Building robust HTTP APIs with proper CORS and error responses

8. Conclusion

This serverless data pipeline successfully demonstrates that production-grade analytics infrastructure can be implemented using exclusively free-tier cloud services. The architecture proves cost-effective for small to medium projects while maintaining enterprise-level scalability and reliability. By leveraging Cloudflare Workers and Supabase, the solution eliminates traditional infrastructure barriers, making advanced data processing capabilities accessible to projects with limited resources.

9. GitHub Repository

Repository URL: https://github.com/SumadhuraM/serverless-analytics-pipeline

Project Structure:

```
serverless-analytics-pipeline/
├── workers/
│   ├── ingestion-worker.js
│   ├── wrangler.toml
│   └── test.html
└── README.md
```

10. References

1. Cloudflare Workers Documentation (2024). Retrieved from https://developers.cloudflare.com/workers/

2. Supabase Documentation (2024). Retrieved from https://supabase.com/docs

3. MDN Web Docs - HTTP CORS (2024). Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS