

Cloud Platform Development (B8IS124) — CA1 Report

Student: Michael Egburonu

Student ID: 10603840

Date: 02 January 2026

GitHub Repository: <https://github.com/mikeegg/CloudCA1>

Live URL: <https://cloudca1-272999851790.europe-west1.run.app/>

Project Title: Cloud Run & Firestore Application with Cloud Build CI/CD

Executive Summary

This project deploys a containerised Node.js web application to Google Cloud Platform (GCP) using a fully automated CI/CD pipeline. A GitHub push to the **main** branch triggers Cloud Build to build and push a Docker image to Artifact Registry and deploy a new revision to Cloud Run. The application persists submitted data to Firestore, and credentials are handled securely using Secret Manager (runtime injection; not committed to source control).

1. Application Overview (Report: 35 marks)

Purpose. Provide a simple web interface that accepts user-submitted data and persists it to a managed database (Firestore) to demonstrate cloud deployment, orchestration, and automation.

Target users. Lecturer/marker and classmates reviewing a deployed cloud application. The application logic is intentionally minimal because the assessment focuses on deployment and automation rather than application feature complexity.

Application behaviour. A GET request serves the user interface. A POST request to **/submit** processes the submission and writes a document to Firestore.

Method	Path	Description
GET	/	Serves the HTML user interface
POST	/submit	Accepts form data and writes to Firestore

2. Architecture Diagram

Figure 1 shows the cloud-native architecture. Source code changes are pushed to GitHub, which triggers Cloud Build. Cloud Build builds and publishes a Docker image to Artifact Registry and deploys a new revision to Cloud Run. Cloud Run executes the service and persists data to Firestore. Secret Manager supplies credentials securely at runtime.

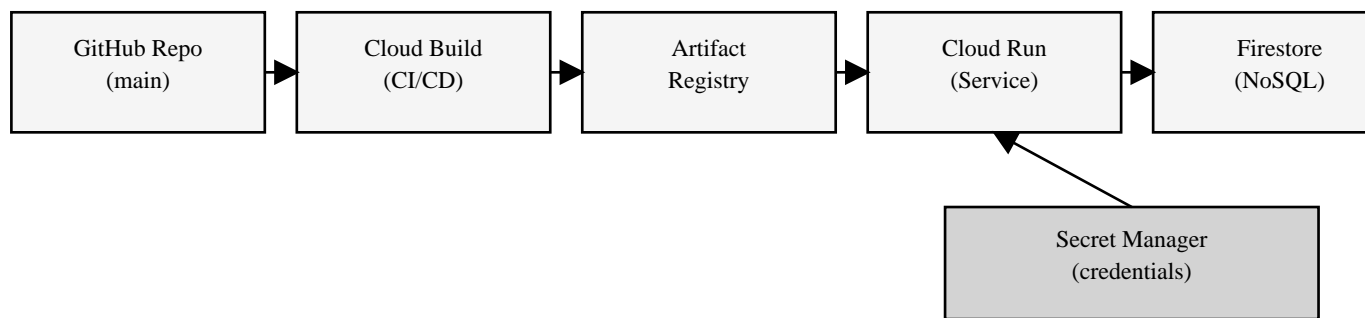


Figure 1: GitHub → Cloud Build → Artifact Registry → Cloud Run → Firestore; Secret Manager injects runtime credentials.

3. GCP Services Description & Justification

Cloud Run (Compute)

Used to host the application as a serverless container. It avoids VM management, supports HTTPS by default, and scales automatically (including scale-to-zero for cost efficiency).

Cloud Build (CI/CD)

Automates build and deployment on each push to main. Provides repeatability, reduces manual errors, and maintains an auditable history of deployments.

Artifact Registry (Images)

Stores versioned Docker images produced by Cloud Build. Improves traceability and supports rollback to known-good images if required.

Firestore (NoSQL Database)

Managed schema-less data store for application submissions. Fits serverless patterns and scales without database administration.

Secret Manager (Secrets)

Stores sensitive configuration securely. Secrets are injected at runtime rather than stored in Git or baked into images.

4. Deployment Documentation

The deployment is fully automated. A Cloud Build Trigger is connected to the GitHub repository and runs on pushes to the **main** branch. The pipeline behaviour is defined in **cloudbuild.yaml** at the repository root.

4.1 Cloud Build Trigger

Trigger settings: GitHub repository connected in Cloud Build → Triggers; event = push to branch **main**; build config = **cloudbuild.yaml**. Successful build history is retained in Cloud Build → History (used as deployment evidence).

4.2 cloudbuild.yaml Pipeline Steps

Stage	What happens	Why it matters
Build	Docker builds an image from the Dockerfile and tags it with \$BUILD_ID	End users can see the SHA and traceable versions.
Push	Docker pushes the tagged image to Artifact Registry	Cloud Run deploys from a registry image, not local files
Deploy	gcloud run deploy updates the Cloud Run service with the built image and deploys it	Artifacts built via CI are deployed from GitHub push

4.3 Infrastructure Configuration

• **Dockerfile:** Uses node:20-alpine, installs production dependencies, copies application code, exposes port 8080 and runs npm start. • **Cloud Run settings:** Deployed to europe-west1 with public access enabled for marking. • **Secrets:** Managed using Secret Manager and provided at runtime (no secrets committed).

5. Cost Calculation

The assessment requires costs to be estimated with the Google Cloud Pricing Calculator and exported as an appendix. Under moderate usage assumptions (e.g., ~1,000 users/day, modest request size, and ~10GB storage), the expected monthly cost is minimal because Cloud Run and Firestore remain within free tiers for typical coursework usage.

Service	Moderate usage assumption	Estimated monthly cost (EUR)
Cloud Run	1,000 requests/day; short execution; scale-to-zero	€0.00 to €1.00
Firestore	Low volume reads/writes; <10GB storage	€0.00 (free tier)
Artifact Registry	Low storage (<0.5GB) and limited pulls	€0.00 (free tier)
Cloud Build	Builds on commits; coursework volume	€0.00 (free tier)

Appendix A (Required): Export the Google Cloud Pricing Calculator estimate and insert it in Appendix A.

6. AI Tools Usage Declaration (Mandatory)

The assessment permits AI-assisted editing. The following declaration must reflect the student’s actual usage.

AI Tool	How it was used	Extent of reliance
ChatGPT	Helped improve clarity/structure of the report and assisted with explanations	Editing with prompts and students reviewing the output
GitHub Copilot	Provided code suggestions during debugging and configuration	Configuration suggestions reviewed before use.

Appendices

Appendix A — Google Cloud Pricing Calculator Export (Required)

Insert the exported estimate (PDF/image) from the Google Cloud Pricing Calculator here.

Appendix B — Deployment Evidence Screenshots (Recommended)

Insert screenshots proving:

- Cloud Build → History (successful build and logs)
- Cloud Build → Trigger configuration (connected to GitHub; triggers on main)
- Artifact Registry → Docker image tag
- Cloud Run → Service overview and revisions
- Browser screenshot showing the live public URL