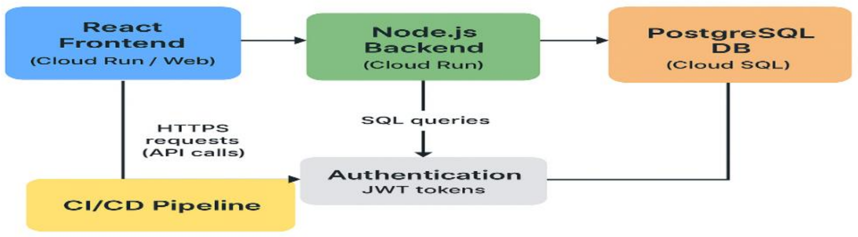
John J. Pretz SWENG861 FALL 2025 Semester Prof: Prof. Santosh Nalubandhu

Social Media Application Deployment to Public Cloud Report

GitHub Repository: https://github.com/JPretz/sweng861-2025-FALL-Social_Media_Application_to_Public_Could
Google Cloud Project ID: sweng861-socialmediapc

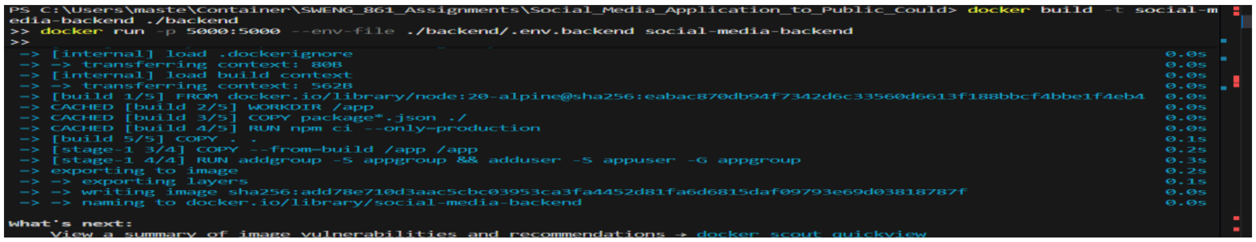
1. Deployment Process

The Social Media Application is deployed using microservices architecture, with the backend REST API and the frontend static site deployed separately on Google Cloud Run. The PostgreSQL database is hosted on Google Cloud SQL, and an optional API Gateway can be used to route requests between the frontend and backend. This architecture ensures separation of concerns, scalability, and easier maintenance. The frontend communicates with the backend API, which in turn interacts with the Cloud SQL database to process data and user requests.



React Frontend (Cloud Run / Web)
Node.js Backend (Cloud Run)
PostgreSQL DB (Cloud SQL)
HTTPS requests (API calls)
SQL queries
Authentication JWT tokens
CI/CD Pipeline

The backend is deployed by first building a Docker image from the backend source code and pushing it to Google Container Registry. It is then deployed to Cloud Run with environment variables that configure the database connection and JWT secret.



```
PS C:\Users\maste\Container\SWENG_861_Assignments\Social_Media_Application_to_Public_Could> docker build -t social-m
edia-backend ./backend
>> docker run -p 5000:5000 --env-file ./backend/.env.backend social-media-backend
>>
 => [internal] load .dockerignore                                                                          0.0s
 => => transferring context: 80B                                                                           0.0s
 => [internal] load build context                                                                          0.0s
 => => transferring context: 562B                                                                          0.0s
 => [build 1/5] FROM docker.io/library/node:20-alpine@sha256:eabac870db94f7342d6c33560d6613f188bbcf4bbe1f4eb4  0.0s
 => CACHED [build 2/5] WORKDIR /app                                                                        0.0s
 => CACHED [build 3/5] COPY package*.json ./                                                               0.0s
 => CACHED [build 4/5] RUN npm ci --only-production                                                        0.0s
 => [build 5/5] COPY . .                                                                                   0.1s
 => [stage-1 3/4] COPY --from-build /app /app                                                              0.2s
 => [stage-1 4/4] RUN addgroup -S appgroup && adduser -S appuser -G appgroup                               0.3s
 => exporting to image                                                                                     0.2s
 => => exporting layers                                                                                    0.1s
 => => writing image sha256:add78e710d3aac5cbc03953ca3fa4452d81fa6d6815daf09793e69d03818787f               0.0s
 => => naming to docker.io/library/social-media-backend                                                    0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

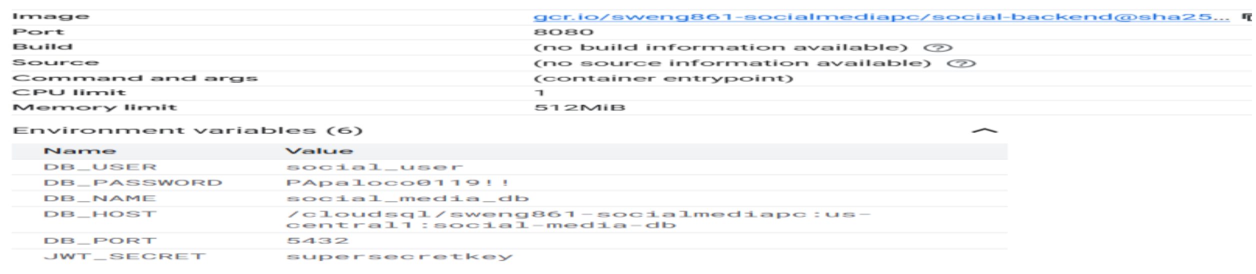
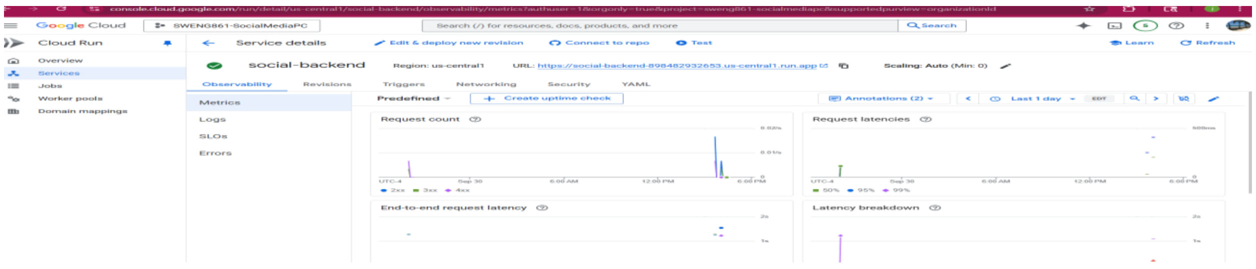


Image	gcr.io/sweng861-socialmediapc/social-backend@sha25...
Port	8080
Build	(no build information available)
Source	(no source information available)
Command and args	(container entrypoint)
CPU limit	1
Memory limit	512MiB

Environment variables (6)

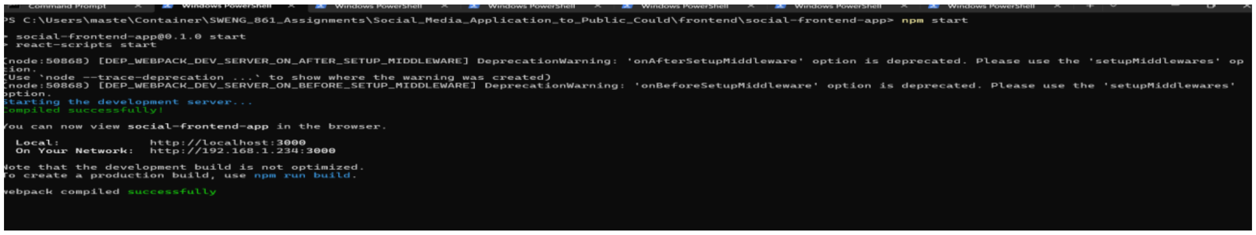
Name	Value
DB_USER	social_user
DB_PASSWORD	PApaloco0119!!
DB_NAME	social_media_db
DB_HOST	/cloudsql/sweng861-socialmediapc:us-central1:social-media-db
DB_PORT	5432
JWT_SECRET	supersecretkey



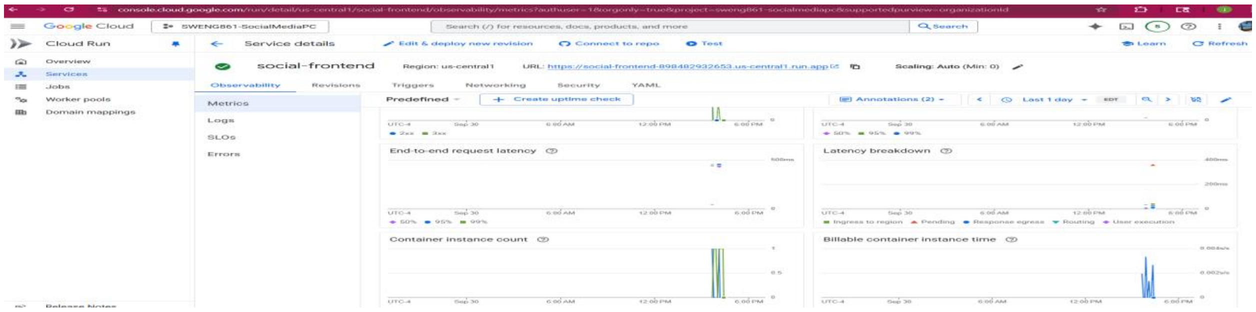
The frontend is built using npm and deployed separately to Cloud Run, allowing it to scale independently.

John J. Pretz SWENG861 FALL 2025 Semester Prof: Prof. Santosh Nalubandhu
Social Media Application Deployment to Public Cloud Report

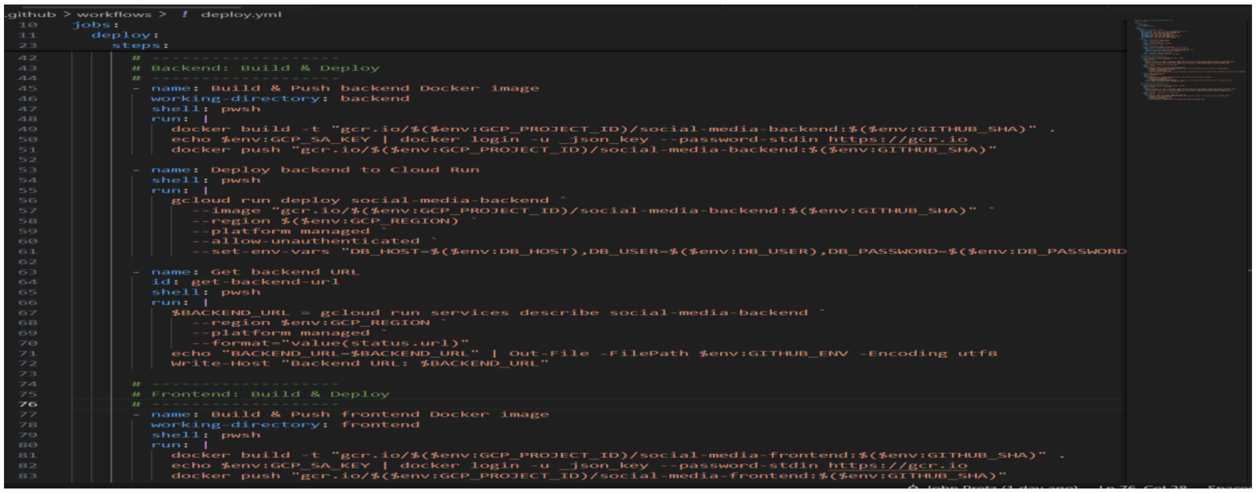
GitHub Repository: https://github.com/JPretz/sweng861-2025-FALL-Social_Media_Application_to_Public_Could
Google Cloud Project ID: sweng861-socialmediapc



```
PS C:\Users\maste\Container\SWENG_861_Assignments\Social_Media_Application_to_Public_Could\frontend\social-frontend-app> npm start

> social-frontend-app@0.1.0 start
> react-scripts start

(node:50868) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' op
tion.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:50868) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares'
option.
Starting the development server...
Compiled successfully!

You can now view social-frontend-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.234:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



A GitHub Actions CI/CD pipeline automates the build, testing, deployment, and monitoring processes, ensuring a streamlined and repeatable deployment process.



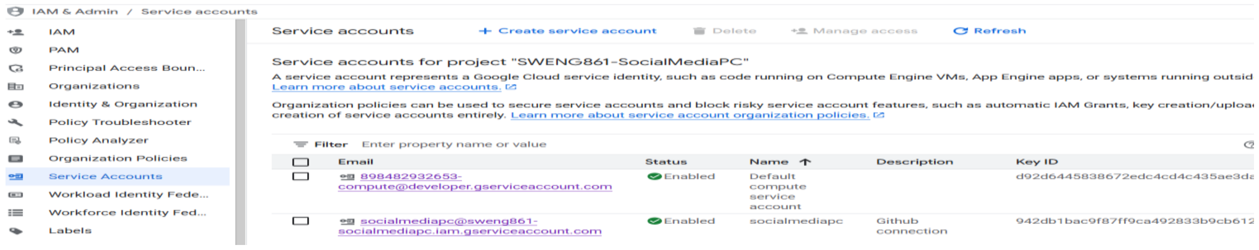
```
.github > workflows > ! deploy.yml
  jobs:
    deploy:
      steps:
        # ------------------
        # Backend: Build & Deploy
        # ------------------
        - name: Build & Push backend Docker image
          working-directory: backend
          shell: pwsh
          run: |
            docker build -t "gcr.io/$($env:GCP_PROJECT_ID)/social-media-backend:$($env:GITHUB_SHA)" .
            echo $env:GCP_SA_KEY | docker login -u _json_key --password-stdin https://gcr.io
            docker push "gcr.io/$($env:GCP_PROJECT_ID)/social-media-backend:$($env:GITHUB_SHA)"

        - name: Deploy backend to Cloud Run
          shell: pwsh
          run: |
            gcloud run deploy social-media-backend `
              --image "gcr.io/$($env:GCP_PROJECT_ID)/social-media-backend:$($env:GITHUB_SHA)" `
              --region $($env:GCP_REGION) `
              --platform managed `
              --allow-unauthenticated `
              --set-env-vars "DB_HOST=$($env:DB_HOST),DB_USER=$($env:DB_USER),DB_PASSWORD=$($env:DB_PASSWORD

        - name: Get backend URL
          id: get-backend-url
          shell: pwsh
          run: |
            $BACKEND_URL = gcloud run services describe social-media-backend `
              --region $env:GCP_REGION `
              --platform managed `
              --format="value(status.url)"
            echo "BACKEND_URL=$BACKEND_URL" | Out-File -FilePath $env:GITHUB_ENV -Encoding utf8
            Write-Host "Backend URL: $BACKEND_URL"

        # ------------------
        # Frontend: Build & Deploy
        # ------------------
        - name: Build & Push frontend Docker image
          working-directory: frontend
          shell: pwsh
          run: |
            docker build -t "gcr.io/$($env:GCP_PROJECT_ID)/social-media-frontend:$($env:GITHUB_SHA)" .
            echo $env:GCP_SA_KEY | docker login -u _json_key --password-stdin https://gcr.io
            docker push "gcr.io/$($env:GCP_PROJECT_ID)/social-media-frontend:$($env:GITHUB_SHA)"
```

2. Security & Scalability Measures

The application implements multiple security measures to protect data and user interactions. Authentication and authorization are handled through JWT tokens, and access to Cloud Run services and Cloud SQL is restricted using Google Cloud IAM roles following the principle of least privilege. Data in transit is encrypted using HTTPS/TLS, and database connections are encrypted as well. These measures ensure that both user credentials and application data remain secure.

John J. Pretz SWENG861 FALL 2025 Semester Prof: Prof. Santosh Nalubandhu
Social Media Application Deployment to Public Cloud Report

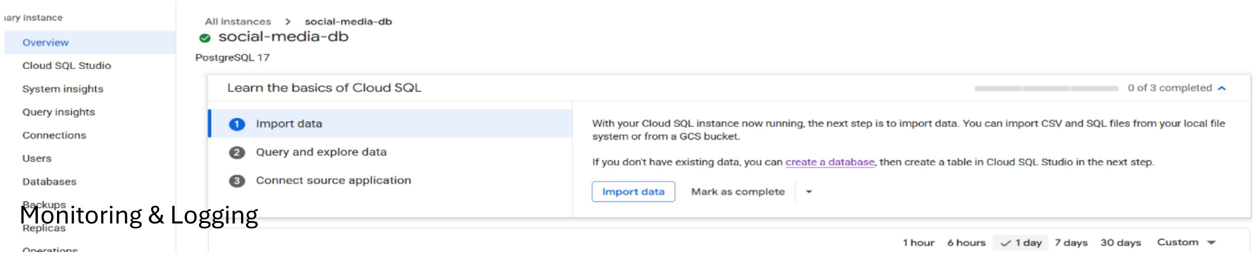
GitHub Repository: https://github.com/JPretz/sweng861-2025-FALL-Social_Media_Application_to_Public_Could
Google Cloud Project ID: sweng861-socialmediapc


IAM & Admin / Service accounts

Service accounts for project "SWENG861-SocialMediaPC"
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outsid
Learn more about service accounts.
Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/uploa
creation of service accounts entirely. Learn more about service account organization policies.

Email	Status	Name	Description	Key ID
898482932653-compute@developer.gserviceaccount.com	Enabled	Default compute service account		d92d6445838672edc4cd4c435ae3da
socialmediapc@sweng861-socialmediapc.iam.gserviceaccount.com	Enabled	socialmediapc	Github connection	942db1bac9f87ff9ca492833b9cb612

Scalability is achieved through Cloud Run's automatic scaling capabilities, allowing backend and frontend services to handle increased traffic. Can be scaled using read replicas and connection pooling to maintain performance under high loads. An API Gateway or load balancer can also be configured to distribute incoming requests efficiently, ensuring consistent performance even during traffic spikes.


All instances > social-media-db
social-media-db
PostgreSQL 17

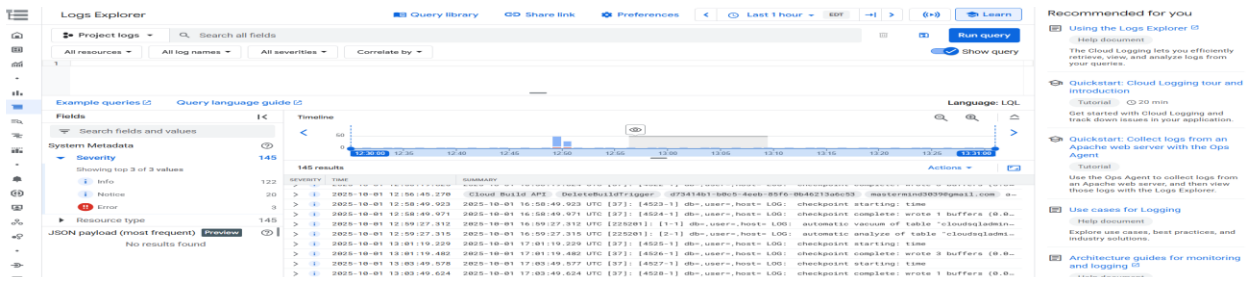
Learn the basics of Cloud SQL — 0 of 3 completed
1 Import data
2 Query and explore data
3 Connect source application

With your Cloud SQL instance now running, the next step is to import data. You can import CSV and SQL files from your local file system or from a GCS bucket.

If you don't have existing data, you can create a database, then create a table in Cloud SQL Studio in the next step.

3. Monitoring & Logging

Monitoring and logging are crucial for maintaining application reliability. Backend and frontend logs are captured using Google Cloud Logging, which includes information on API requests, authentication attempts, and errors.


Logs Explorer
145 results

```
SEVERITY TIME                     SUMMARY
>  i  2025-10-01 12:56:45.270  Cloud Build API  DeleteBuildTrigger  d73414b1-b0c5-4eeb-85f6-0b46213a6c53  mastermind3039@gmail.com  a…
>  i  2025-10-01 12:58:49.923  2025-10-01 16:58:49.923 UTC [37]: [4523-1] db=,user=,host= LOG:  checkpoint starting: time
>  i  2025-10-01 12:58:49.971  2025-10-01 16:58:49.971 UTC [37]: [4524-1] db=,user=,host= LOG:  checkpoint complete: wrote 1 buffers (0.0…
>  i  2025-10-01 12:59:27.312  2025-10-01 16:59:27.312 UTC [225201]: [1-1] db=,user=,host= LOG:  automatic vacuum of table "cloudsqladmin…
>  i  2025-10-01 12:59:27.315  2025-10-01 16:59:27.315 UTC [225201]: [2-1] db=,user=,host= LOG:  automatic analyze of table "cloudsqladmi…
>  i  2025-10-01 13:01:19.229  2025-10-01 17:01:19.229 UTC [37]: [4525-1] db=,user=,host= LOG:  checkpoint starting: time
>  i  2025-10-01 13:01:19.482  2025-10-01 17:01:19.482 UTC [37]: [4526-1] db=,user=,host= LOG:  checkpoint complete: wrote 3 buffers (0.0…
>  i  2025-10-01 13:03:49.578  2025-10-01 17:03:49.577 UTC [37]: [4527-1] db=,user=,host= LOG:  checkpoint starting: time
>  i  2025-10-01 13:03:49.624  2025-10-01 17:03:49.624 UTC [37]: [4528-1] db=,user=,host= LOG:  checkpoint complete: wrote 1 buffers (0.0…
```

Query library
Share link
Preferences
Last 1 hour
Learn
Project logs
Search all fields
Run query
All resources
All log names
All severities
Correlate by
Show query
Example queries
Query language guide
Language: LQL
Fields
Search fields and values
System Metadata
Severity 145
Showing top 3 of 3 values
Info 122
Notice 20
Error 3
Resource type 145
JSON payload (most frequent) Preview
No results found
Timeline
12:30:00
12:35
12:40
12:45
12:50
12:55
13:00
13:05
13:10
13:15
13:20
13:25
13:31:00
Actions
Recommended for you
Using the Logs Explorer
Help document
The Cloud Logging lets you efficiently retrieve, view, and analyze logs from your queries.
Quickstart: Cloud Logging tour and introduction
Tutorial
20 min
Get started with Cloud Logging and track down issues in your application.
Quickstart: Collect logs from an Apache web server with the Ops Agent
Tutorial
Use the Ops Agent to collect logs from an Apache web server, and then view those logs with the Logs Explorer.
Use cases for Logging
Help document
Explore use cases, best practices, and industry solutions.
Architecture guides for monitoring and logging

Google Cloud Monitoring tracks performance metrics, including CPU and memory usage, request latency, and error rates. Alerts are configured to notify administrators in case of failures, high latency, or unusual error patterns. Testing and verification are performed using Postman or curl to ensure all API endpoints respond correctly. Frontend interactions are verified to confirm that login, post creation, commenting, and liking functionality operate as intended, and database updates are confirmed in Cloud SQL.

4. Conclusion: In conclusion, the Social Media Application has been successfully deployed to Google Cloud using Docker, Cloud Run, and Cloud SQL. The deployment process is automated via CI/CD, security is enforced through JWT authentication and IAM access controls, and Cloud Run and Cloud SQL features support scalability. Comprehensive monitoring and logging ensure that the application is reliable and maintainable, making it production-ready and capable of handling increased user traffic efficiently.

John J. Pretz SWENG861 FALL 2025 Semester Prof: Prof. Santosh Nalubandhu

Social Media Application Deployment to Public Cloud Report

GitHub Repository: https://github.com/JPretz/sweng861-2025-FALL-Social_Media_Application_to_Public_Could
Google Cloud Project ID: sweng861-socialmediapc