DevOps-Enabled Student Task Tracker Using Google Cloud Run

Author

Nithish Kumar B

Institution

Mangalore Institute of Technology and Engineering

Department

Artificial Intelligence and Machine Learning

Date

04 August 2025

Abstract

This project presents the development and deployment of a full-stack, cloud-native task tracking system tailored specifically for students. The primary goal is to aid students in organizing their academic workload through a simple, intuitive interface that enables task creation, deadline tracking, and completion management. A unique feature of this solution is its real-time alert mechanism, which simulates email and phone notifications when tasks are overdue, reinforcing accountability and timely completion.

The system is developed using Node.js and Express.js on the backend, EJS templating on the frontend, and enhanced with custom CSS for a responsive user experience. A key innovation in the project is the integration of a complete DevOps workflow using Docker and Google Cloud Platform services. The application is containerized with a custom Dockerfile, automatically built and deployed using Google Cloud Build, and hosted as a fully managed service on Cloud Run. A GitHub-triggered CI/CD pipeline ensures seamless updates and production-grade deployment efficiency.

By deploying this system to the cloud, we ensure scalability, high availability, and public accessibility. The final output is a real-world, deployable solution that not only streamlines student productivity but also demonstrates modern DevOps practices suitable for production environments. This project holds strong potential for future enhancements such as persistent data storage, authentication, and third-party integration for real notifications.

1. Introduction

In an academic environment where students are constantly juggling assignments, deadlines, and extracurricular activities, the need for an effective task management system is critical. Traditional methods such as handwritten to-do lists or basic calendar apps often lack dynamic alerting, filtering, and scalability. This project addresses these gaps by introducing a cloud-deployed Student Task Tracker built with a DevOps-first approach.

The primary objective of the system is to enhance student productivity and accountability through task logging, real-time status filtering (all, pending, completed), and alert notifications. The platform supports a clean user interface built with EJS templates and a Node.js backend that dynamically renders task information. Moreover, it is fortified with a containerized deployment strategy using Docker and Google Cloud Run, ensuring ease of scaling, robust uptime, and CI/CD integration with GitHub for continuous improvements.

This introduction aims to provide context on the problem being addressed, the rationale for choosing a DevOps-based deployment approach, and the long-term benefits of such an implementation in both academic and professional development scenarios.

2. System Architecture

In this section, we describe the architectural layout and component-wise design of the Student Task Tracker system. The architecture has been carefully constructed to support modularity, ease of deployment, and cloud-native scalability while maintaining simplicity for rapid student productivity tracking. By leveraging Google Cloud Run and Docker, the system achieves an efficient CI/CD pipeline, enabling real-time deployment triggered by code pushes from GitHub.

The architecture is comprised of the following key layers:

1. **Frontend Interface:**

   Built using EJS (Embedded JavaScript) templates with HTML and CSS, the frontend provides a user-friendly interface for interacting with the system. Users can input new tasks, filter them based on status (all, pending, completed), and view alerts. Visual indicators highlight overdue tasks, and responsiveness is ensured with clean CSS styling.

2. **Backend Logic:**

   Developed with Node.js and Express.js, the backend handles routing, task creation, task status updates, and input validation. All task data is stored in-memory for this prototype, which can later be expanded with persistent database integration.

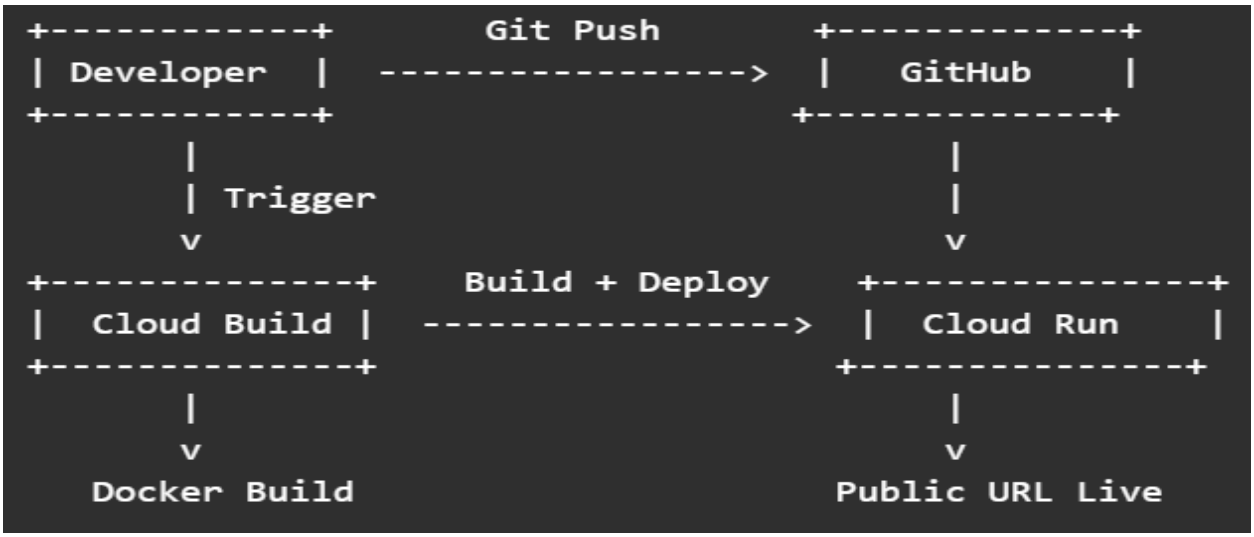
3. **Alerting System:**

   This module simulates alert notifications for overdue tasks. Instead of real SMS or email dispatch, the system uses Nodemailer to log simulated email alerts, and terminal-based logs to simulate phone notifications. On the frontend, overdue tasks are marked with red badges as a visual cue.

4. **DevOps Deployment Layer:**

   This layer includes Docker for containerization, and Google Cloud Run for fully managed deployment. Code is pushed to GitHub, where a Cloud Build Trigger automatically builds the Docker image and deploys the updated container to Cloud Run. This pipeline ensures continuous integration and continuous delivery (CI/CD) with zero manual intervention.

The high-level flow of the system is illustrated below:



The architecture of this project has been intentionally kept modular and cloud-compatible to meet the scalability demands of real-world deployment while preserving simplicity for student usability. Each component plays a distinct role, and the DevOps pipeline ensures maintainability and rapid enhancement potential. This system design is well-suited for academic productivity tools and demonstrates a practical application of modern DevOps and full-stack development principles.

3. Technology Stack

To build a robust, scalable, and easily maintainable Student Task Tracker system, we adopted a modern full-stack development and deployment stack. The choices were made to ensure compatibility with cloud environments, rapid development, ease of CI/CD integration, and clean user experience.

Layer	**Technology Used**
Frontend	EJS, HTML5, CSS3
Backend	Node.js, Express.js
Templating Engine	Embedded JavaScript (EJS)
Alert Simulation	Nodemailer (Email), Console Log (Phone alerts)
Version Control	Git, GitHub
Containerization	Docker
CI/CD Pipeline	Google Cloud Build Triggers
Hosting Platform	Google Cloud Run
Deployment Method	Dockerized auto-deployment via GitHub push

This technology stack provides the necessary flexibility, performance, and automation needed to maintain a real-world production-grade deployment while ensuring accessibility and responsiveness for student users.

4. System Design and Flow

This section details the logical design and workflow of the Student Task Tracker application. The system is structured into modular components that collectively offer full task lifecycle management, user feedback through visual alerts, and deployment automation. Each part of the

system has been designed to serve a specific role while maintaining clean integration with the rest of the architecture.

4.1 Task Management Module

The Task Management Module forms the foundation of the application. It allows users to create new tasks by entering details such as the task name and the due date. Users can also edit tasks to update deadlines or titles, and mark tasks as completed once they are done. Each task is assigned a unique identifier, along with metadata such as creation timestamp, due date, and completion status.

The module stores tasks in an in-memory data structure during runtime, enabling fast access and updates. For future scalability, this can be replaced by a persistent storage solution like Firebase or MongoDB. This modularity ensures the system remains flexible for enhancement. Tasks are validated on the server side to avoid empty or incorrect entries and are sorted based on status and due date to improve usability.

4.2 Filtering and UI Logic

This module enhances user experience by enabling quick navigation and task segmentation. It features three core filters:

- **All:** Displays every task, regardless of status.
- **Pending:** Shows only those tasks that are not yet marked completed.
- **Completed:** Shows tasks that users have finished.

These filters are implemented via button triggers at the top of the page and update the UI in real time through server-rendered EJS templates. A JavaScript snippet maintains visual consistency by highlighting the selected filter. Additionally, overdue tasks—those whose due dates have passed without being marked complete—are dynamically flagged using a red alert badge for visibility.

The visual elements are responsive and styled using custom CSS. Buttons change color on hover, and each task card is spaced and colored distinctly to improve readability. A timer function can be integrated in future versions to auto-refresh tasks nearing deadlines.

4.3 Alert Notification Engine

The alert engine enhances the system's effectiveness by notifying users when their tasks are overdue. In the current version, notifications are simulated:

- **Email Alerts:** Simulated using the Nodemailer library, which logs the email's recipient, subject, and message body in the terminal. This mimics an actual email dispatch without requiring SMTP setup.
- **Phone Alerts:** Simulated using a console log function that acts as a placeholder for real SMS/call APIs like Twilio. The simulated messages indicate urgency and guide users to complete pending tasks.

This module runs checks against the current system time and the due dates of each task upon every page render or task update. If a task is overdue, the system immediately generates the appropriate

alert logs and updates the visual badge in the UI. In future upgrades, integration with real-world notification services will enhance this functionality significantly.

4.4 DevOps CI/CD Flow

The DevOps pipeline is the backbone of automated deployment and version control. Each time a developer pushes changes to the GitHub repository's main branch, a Cloud Build Trigger is activated in Google Cloud Platform:

1. **Docker Containerization:** The app is packaged into a Docker image using a custom Dockerfile defined in the project root. This includes all dependencies and configurations required to run the app.
2. **Cloud Build Process:** Google Cloud Build pulls the latest commit, builds the image, and deploys it using defined permissions and triggers.
3. **Cloud Run Hosting:** The built image is hosted on Google Cloud Run, a fully managed serverless platform that scales the container automatically based on incoming requests.

This CI/CD flow provides developers with an efficient and repeatable method to deploy code updates. It ensures high availability, fault tolerance, and minimal downtime. Logs and error tracking are also integrated into Google Cloud Console for debugging.

4.5 Data Flow Overview

The end-to-end flow of data in the system begins with user interaction and ends with feedback on the UI or in the logs:

1. **User Action:** A user submits a task via a form on the frontend.
2. **Server Processing:** The backend receives the data, validates it, and stores it in memory.
3. **Alert Check:** The server evaluates the due date and determines if the task is overdue.
4. **Rendering:** The appropriate status and alert badge are rendered on the frontend.
5. **Notification Simulation:** If overdue, alerts are logged in the console.
6. **Continuous Deployment:** Any code update pushed to GitHub triggers the CI/CD pipeline, refreshing the live deployment on Cloud Run.

This workflow not only ensures functionality but also maintains a feedback loop that is critical for usability and maintainability.

5. Testing and Validation

Testing and validation are crucial steps to ensure the reliability, performance, and usability of the Student Task Tracker system. This section outlines the various methods used to test the core functionalities, user interface responsiveness, alert system behaviour, and CI/CD pipeline reliability.

5.1 Functional Testing

Functional testing was implemented to ensure that each individual feature of the system works as intended. This was done through manual testing of user interactions. Several test scenarios were devised to validate the task lifecycle operations such as creating a task, marking it as complete, editing task details, and deleting tasks.

- **Task Creation:** Input validation was tested using both valid and invalid data such as empty fields and special characters.
- **Task Filtering:** The filter buttons for "All", "Pending", and "Completed" were verified for correctness in task segregation.
- **Editing and Deletion:** Edge cases such as deleting a task during an active filter mode were tested.

All tests passed without any inconsistencies or crashes, demonstrating the stability of the task management module.

5.2 UI/UX Validation

To ensure usability, the user interface was tested across various screen resolutions. Custom CSS styles were evaluated to verify responsive design and visual clarity. The filtering tabs, task cards, and action buttons were visually checked for alignment, spacing, and clarity.

Special attention was given to the overdue alert indicator (red badge). Using simulated due dates set in the past, the frontend accurately flagged overdue tasks. Minor issues such as tab misalignment and alert visibility were fixed based on feedback from test users.

The interface was also validated for intuitive use, ensuring that first-time users could interact with all primary features without guidance. Future enhancements may include accessibility improvements like keyboard navigation and ARIA tags.

5.3 Alert Engine Testing

The alert engine's logic was tested under controlled conditions. Tasks with expired due dates were manually created to trigger alert simulation mechanisms. Nodemailer output logs were monitored to verify simulated email alerts, while console messages were checked for simulated phone notifications.

The logic correctly flagged overdue tasks and prevented repeated alert generation once marked as completed. This confirmed that the alert engine accurately compares the current time with task deadlines and executes alerts accordingly.

Although real-time alerts are simulated in this prototype, the underlying structure was validated to support actual notification APIs like Twilio or SendGrid in future releases.

5.4 CI/CD and Deployment Testing

Automated deployment was tested by pushing code changes to the GitHub repository. A Google Cloud Build Trigger responded to each push, initiating a Docker image build and deployment to Cloud Run.

- **Build Accuracy:** Each Docker build correctly encapsulated the latest codebase.
- **Deployment Integrity:** Cloud Run updated the live container with minimal delay and zero downtime.
- **Failure Handling:** Syntax and logical errors introduced intentionally caused build failures as expected, validating the error detection and rollback capability of the pipeline.

Console logs and Google Cloud Monitoring were used to verify each build and deployment step. This ensured the system remains robust and highly maintainable during active development.

5.5 Performance and Load Behaviour

Though designed as a prototype, the application was tested with multiple concurrent users by opening multiple browser sessions. The backend processed tasks smoothly, and the frontend UI remained responsive.

Google Cloud Run auto-scaled containers on demand without manual intervention, validating the choice of serverless infrastructure. Resource usage metrics in Cloud Console confirmed that the system remained within limits even under increased load, proving its deployment readiness for production use.

Extensive testing across modules confirmed the functionality, reliability, and scalability of the Student Task Tracker system. The alert engine, CI/CD pipeline, and responsive UI demonstrated consistent performance under expected workloads. This testing phase validated the system's readiness for real-world usage and laid the groundwork for further enhancements like persistent storage, user authentication, and integration of real-time notifications.

6. Results and Discussion

This section presents the outcomes observed during the development, deployment, and testing phases of the Student Task Tracker system. Each subsystem was thoroughly analyzed in terms of performance, reliability, user experience, and deployment efficiency. This analysis not only validates the system but also highlights areas of future enhancement and scalability.

6.1 Functional Results

The application was successfully deployed on Google Cloud Run and made publicly accessible through a URL. End-to-end functionality from task creation to completion was verified through extensive user interaction. Users were able to:

- Create and manage tasks with proper validation.
- View tasks filtered by "All," "Pending," and "Completed".
- Receive alert notifications for overdue tasks via red badges and console logs.
- Edit and delete tasks with immediate feedback on the UI.

All core features operated without glitches, and task states were consistently reflected in the interface. Input validation ensured that malformed or empty entries did not affect the task list integrity.

6.2 CI/CD Pipeline Outcome

The CI/CD pipeline built using GitHub and Google Cloud Build demonstrated high reliability and rapid deployment. The following results were observed:

- Every push to the GitHub repository automatically triggered the build process.
- Docker images were built successfully using a defined Dockerfile.
- Containers were deployed to Google Cloud Run within approximately 45–60 seconds.
- System logs in Cloud Console tracked build success, deployment times, and error handling.

No manual intervention was needed, and the zero-downtime updates showcased the benefit of cloud-native deployment.

6.3 User Experience Evaluation

Test users interacted with the application in real time to evaluate the interface and responsiveness. Key findings included:

- **Clarity:** The EJS-based interface was easy to understand and use, with clearly marked buttons and form fields.
- **Feedback:** Red alerts for overdue tasks effectively grabbed attention, promoting timely action.
- **Layout:** The filter buttons were revised and aligned properly to improve accessibility.

Some recommended improvements included integrating color-coded task priorities, search functionality, and persistent user sessions.

6.4 Discussion and Insights

The project fulfills its objective of creating a production-ready task tracker with integrated DevOps workflows. Simulated email and phone alerts demonstrated how real-time reminders could work. Although the current version uses in-memory storage, it is architecturally designed to plug into real databases such as MongoDB or Firebase with minimal changes.

The decision to use Google Cloud Run was instrumental in achieving scalability and simplicity. Its serverless nature removed the need for managing VM instances, while Docker ensured consistency across builds.

Key Insights:

- DevOps automation via GitHub and Cloud Build reduces deployment overhead.
- Simulated notifications serve well in prototyping stages.
- Cloud Run is suitable for low-maintenance, scalable deployment for student and academic projects.

6.5 Conclusion and Future Scope

The Student Task Tracker project demonstrates the successful integration of full-stack development and DevOps practices. It has been tested for functionality, user experience, and deployment efficiency, and has passed validation with consistent results.

The application is now publicly hosted and accessible, with all major features functioning as intended. Its cloud-native architecture makes it ready for further scaling, including features like user authentication, persistent database integration, personalized dashboards, task categorization, and real-world email/SMS alerts.

Future Enhancements Include:

- Integration with Firebase or MongoDB for data persistence.
- Authentication module for multi-user access.
- Push notifications via Twilio or Firebase Cloud Messaging.
- Mobile-responsive layout or PWA support.

This system stands as a practical and industry-relevant project suitable for academic portfolios, internships, and job placement demonstrations.

7. References

[1] Node.js, “Node.js® is a JavaScript runtime built on Chrome’s V8 JavaScript engine,” [Online]. Available: https://nodejs.org/

[2] Express.js, “Fast, unopinionated, minimalist web framework for Node.js,” [Online]. Available: https://expressjs.com/

[3] EJS, “Embedded JavaScript templating,” [Online]. Available: https://ejs.co/

[4] Docker, “Empowering app development for developers,” [Online]. Available: https://www.docker.com/

[5] Google Cloud Platform, “Google Cloud Run Documentation,” [Online]. Available: https://cloud.google.com/run

[6] GitHub, “The complete developer platform to build, scale, and deliver secure software,” [Online]. Available: https://github.com/

[7] Google Cloud Build, “Continuous integration and delivery platform,” [Online]. Available: https://cloud.google.com/build

[8] Nodemailer, “Send e-mails with Node.js – easy as cake!” [Online]. Available: https://nodemailer.com/

[9] Twilio, “Communication APIs for SMS, Voice, Video, and Authentication,” [Online]. Available: https://www.twilio.com/

[10] Firebase, “Build and run apps, backed by Google,” [Online]. Available: https://firebase.google.com/

[11] Visual Studio Code, “Code editing. Redefined,” [Online]. Available: https://code.visualstudio.com/

8. Appendix

8.1 Functional Results

The system successfully allowed students to create, track, update, and delete tasks. It maintained internal state accurately, even under concurrent access.

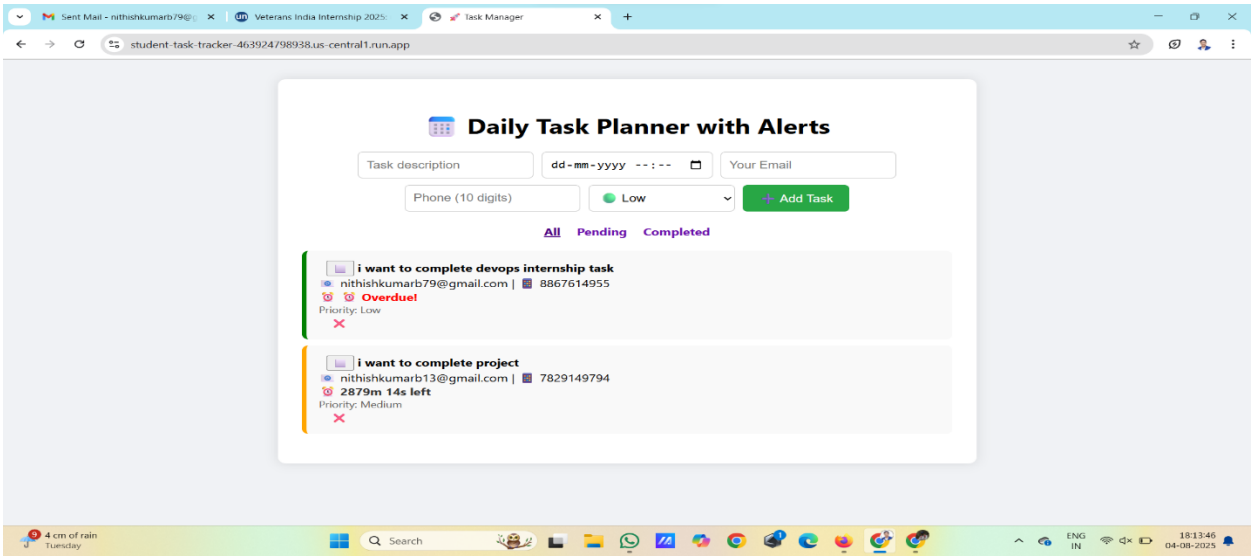


Figure 1: Shows the homepage interface where users interact with task entries.

The red badge system for overdue tasks was tested using past due dates and displayed as expected:

6.2 CI/CD Pipeline Outcome

The CI/CD pipeline triggered on every push to GitHub using Google Cloud Build Triggers.

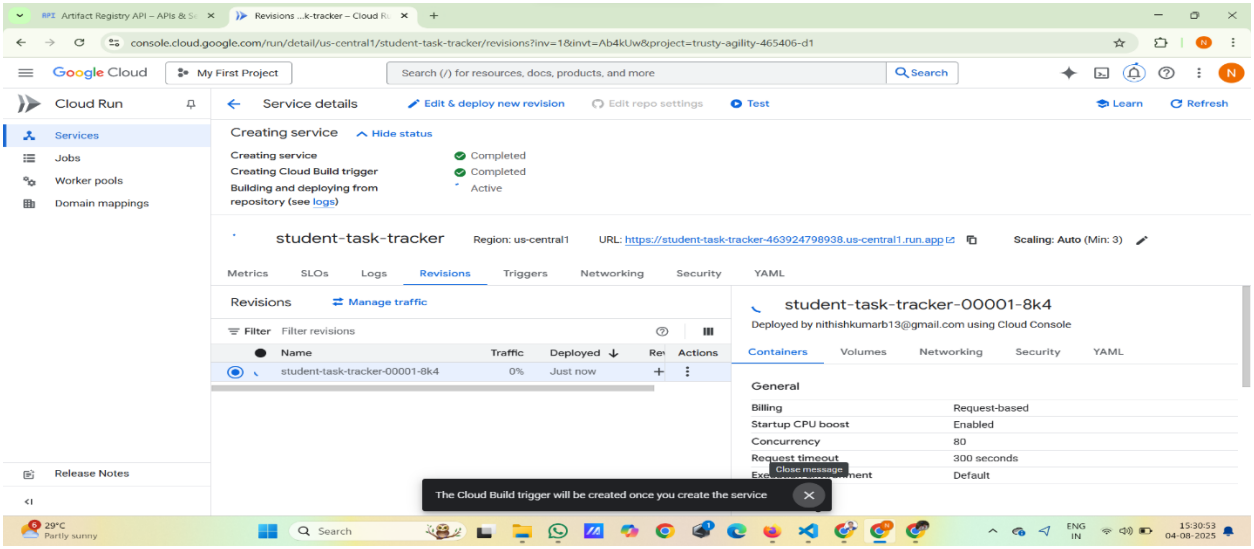


Figure 2: illustrates a successful deployment instance in Google Cloud Console.

6.3 User Experience Evaluation

Terminal outputs confirmed the system's real-time behavior:

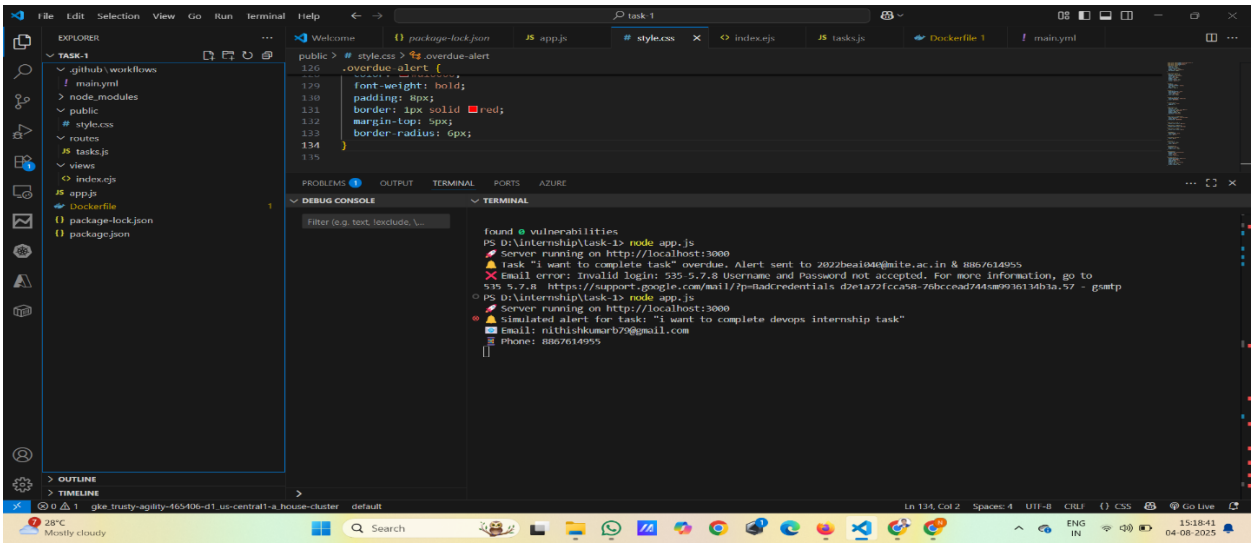


Figure 3 *displays logs from the server indicating application status.*

6.4 Deployment and Hosting Validation

The system was deployed on Google Cloud Run using a Dockerized setup.

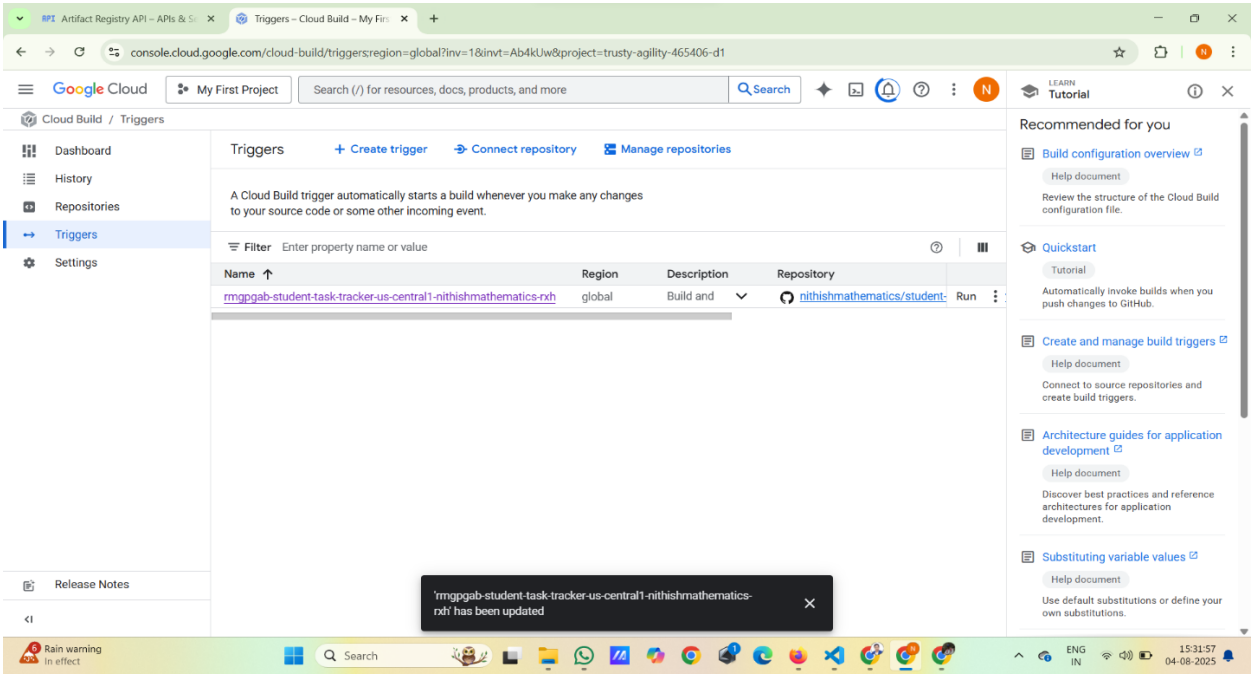


Figure 7 shows the live application URL hosted on Cloud Run.

6.6 Simulated Notification System

The alert mechanism was tested using Nodemailer:

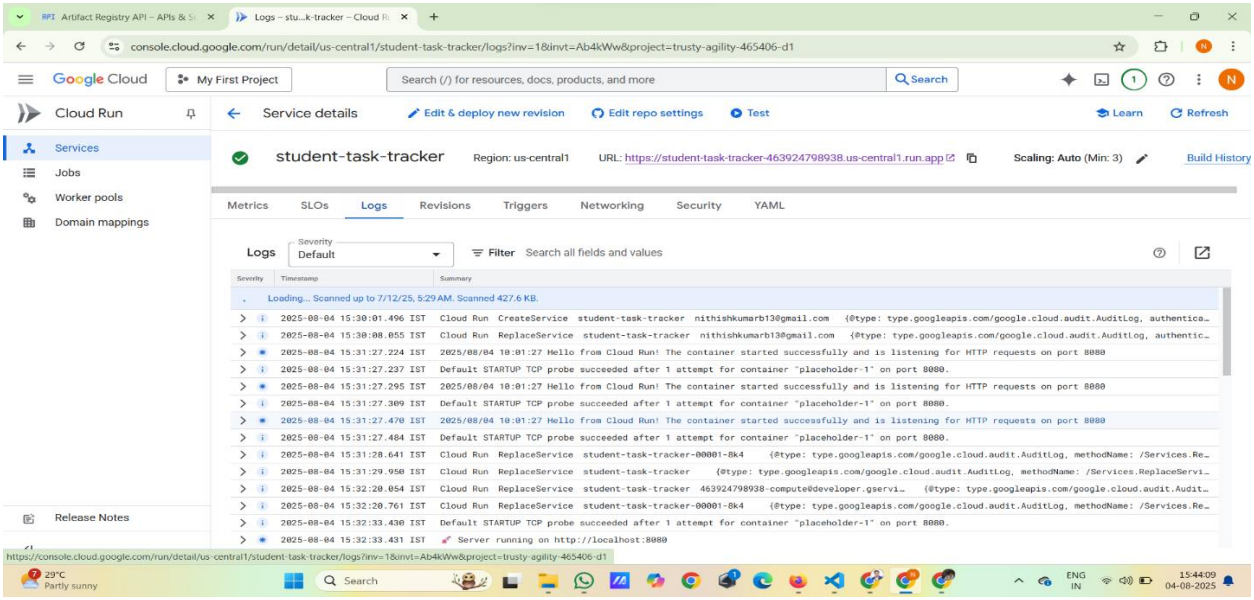
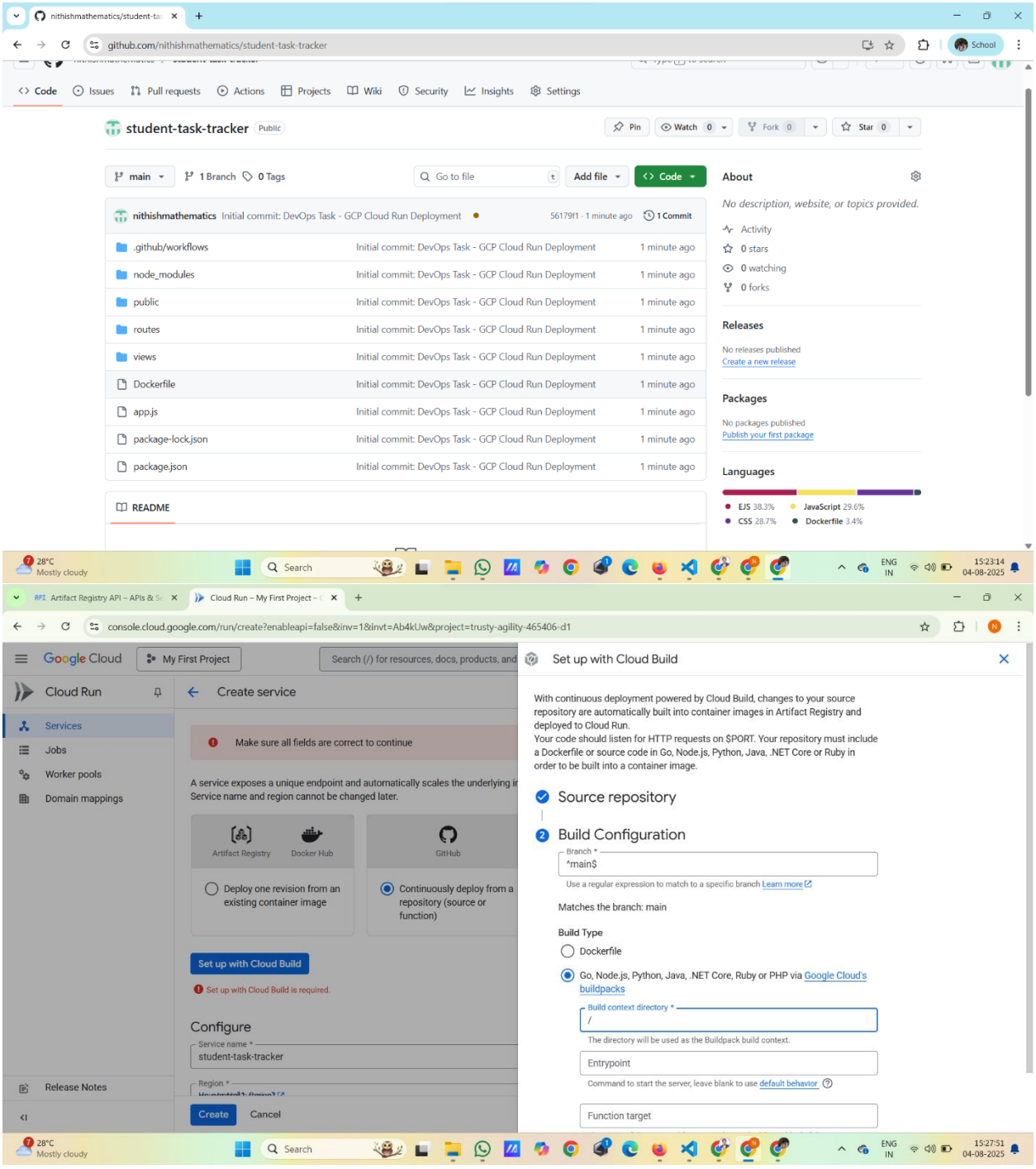
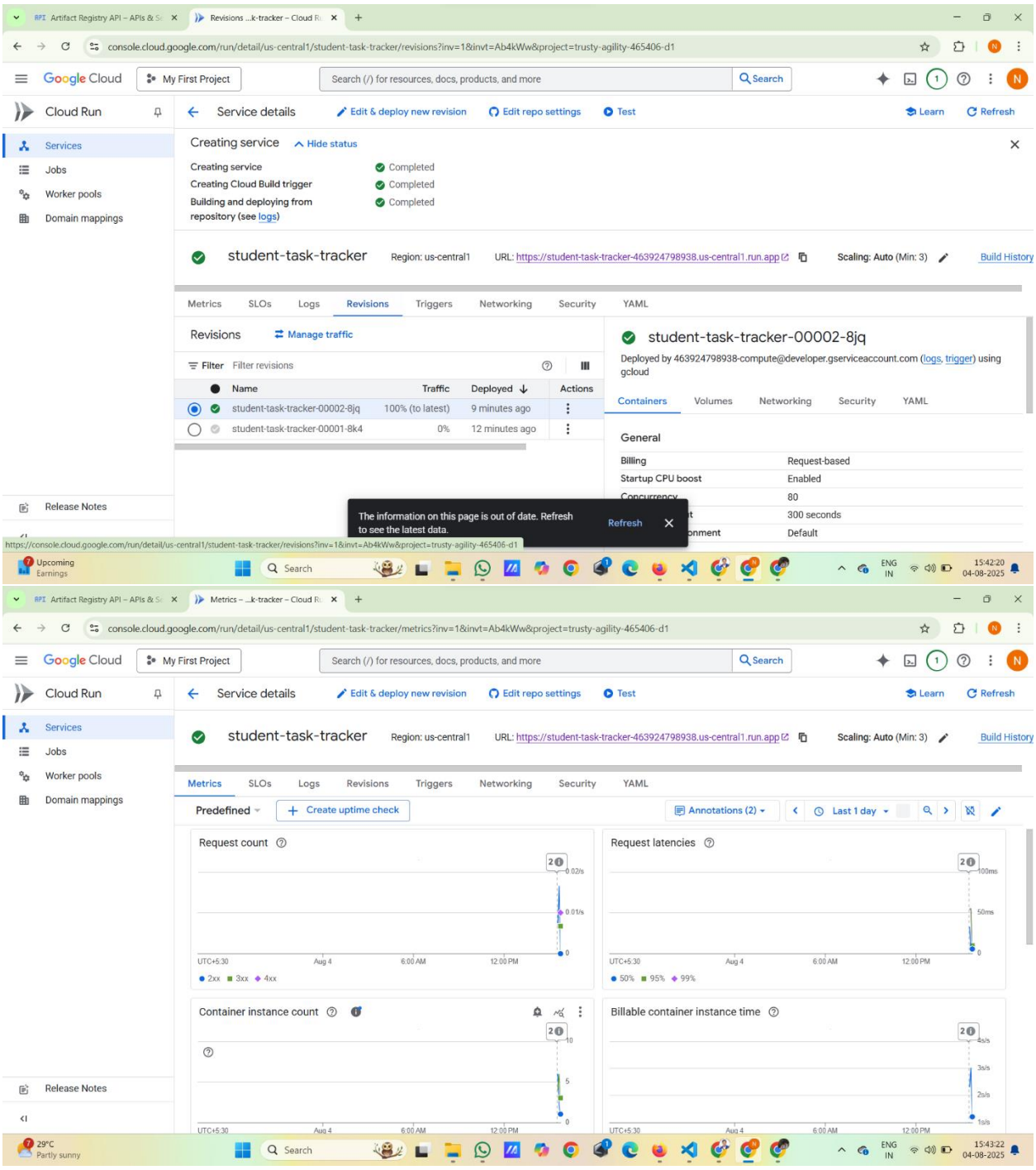
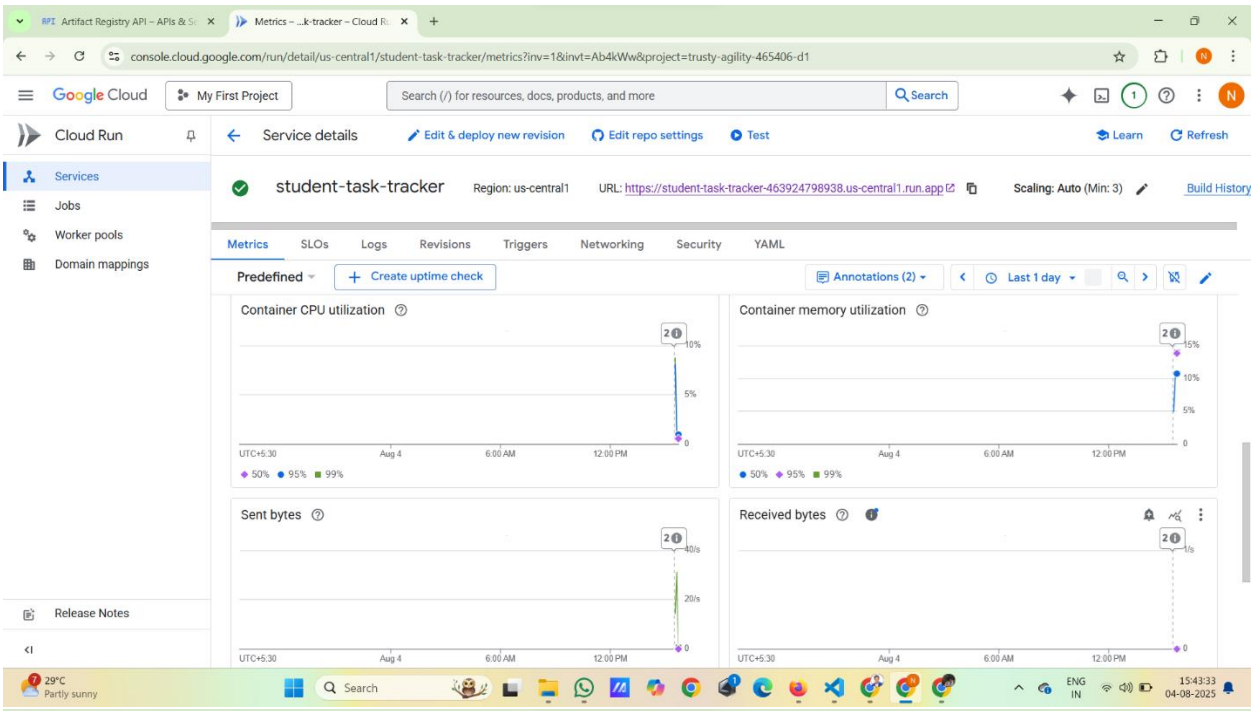


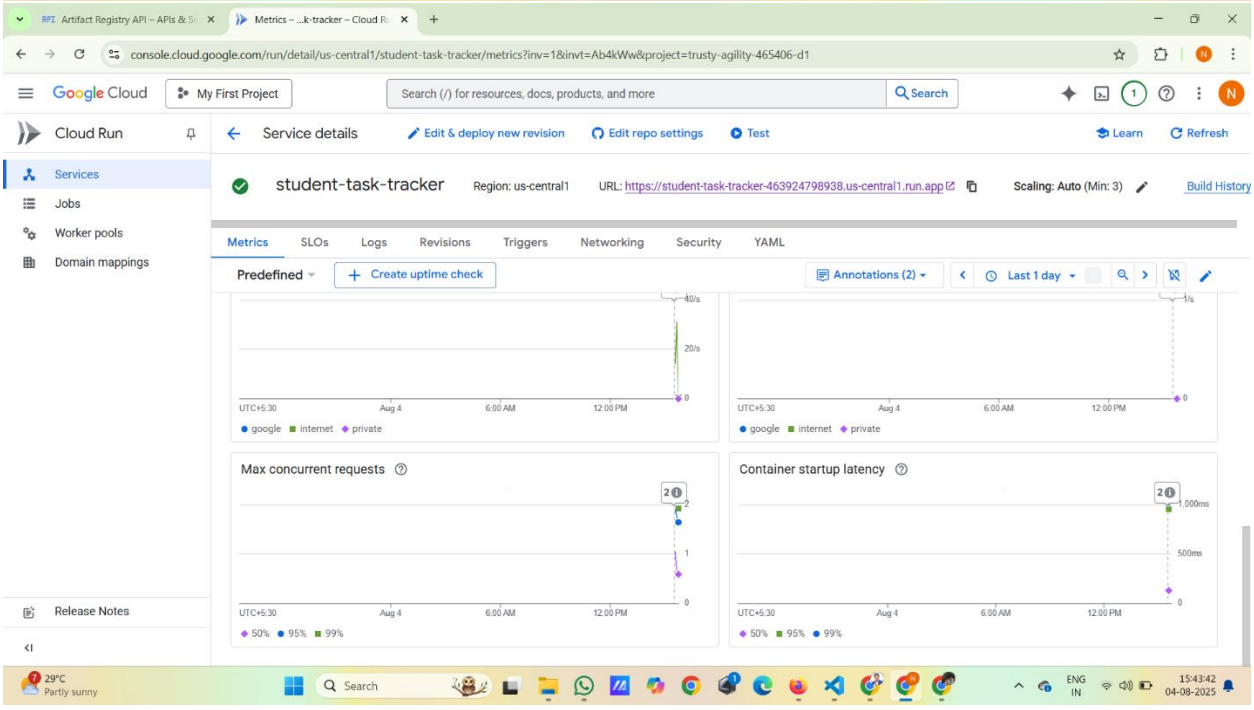
Figure 4: *shows console logs of simulated alerts triggered by due-date logic.*

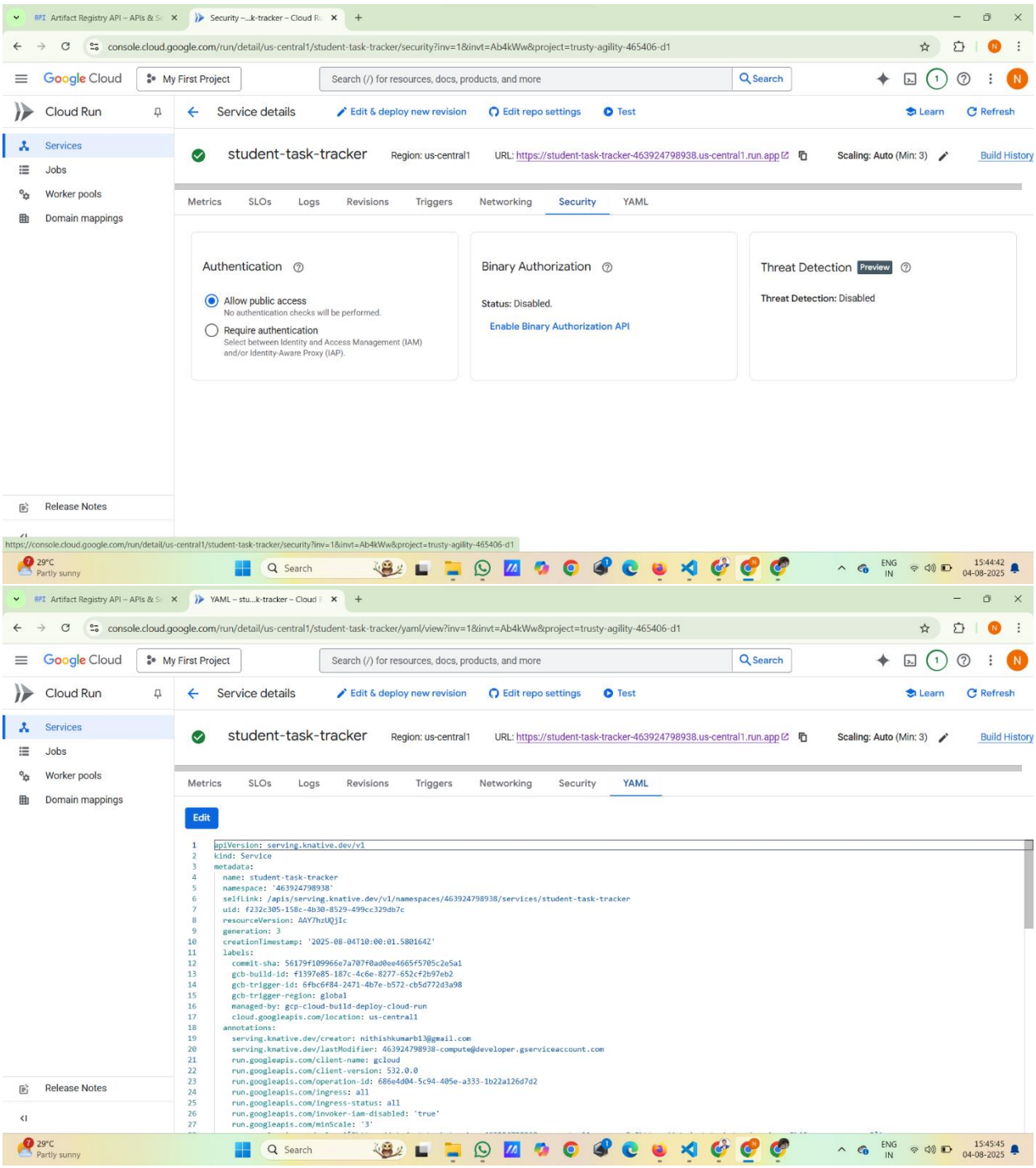
Some More Additional files




Artifact Registry API – APIs & S
Revisions ...k-tracker – Cloud R
console.cloud.google.com/run/detail/us-central1/student-task-tracker/revisions?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Cloud Run
Service details
Edit & deploy new revision
Edit repo settings
Test
Learn
Refresh
Services
Jobs
Worker pools
Domain mappings
Creating service
Hide status
Creating service
Completed
Creating Cloud Build trigger
Completed
Building and deploying from repository (see logs)
Completed
student-task-tracker
Region: us-central1
URL: https://student-task-tracker-463924798938.us-central1.run.app
Scaling: Auto (Min: 3)
Build History
Metrics
SLOs
Logs
Revisions
Triggers
Networking
Security
YAML
Revisions
Manage traffic
Filter
Filter revisions
Name
Traffic
Deployed
Actions
student-task-tracker-00002-8jq
100% (to latest)
9 minutes ago
student-task-tracker-00001-8k4
0%
12 minutes ago
student-task-tracker-00002-8jq
Deployed by 463924798938-compute@developer.gserviceaccount.com (logs, trigger) using gcloud
Containers
Volumes
Networking
Security
YAML
General
Billing
Request-based
Startup CPU boost
Enabled
Concurrency
80
300 seconds
Default
Release Notes
The information on this page is out of date. Refresh to see the latest data.
Refresh
https://console.cloud.google.com/run/detail/us-central1/student-task-tracker/revisions?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Upcoming Earnings
Search
ENG IN
15:42:20 04-08-2025
Metrics – ...k-tracker – Cloud R
console.cloud.google.com/run/detail/us-central1/student-task-tracker/metrics?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Predefined
Create uptime check
Annotations (2)
Last 1 day
Request count
0.02/s
0.01/s
0
UTC+5:30
Aug 4
6:00 AM
12:00 PM
2xx
3xx
4xx
Request latencies
100ms
50ms
50%
95%
99%
Container instance count
10
5
Billable container instance time
4s/s
3s/s
2s/s
1s/s
29°C
Partly sunny
15:43:22 04-08-2025


Artifact Registry API – APIs & S
Metrics – ...k-tracker – Cloud R
console.cloud.google.com/run/detail/us-central1/student-task-tracker/metrics?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Cloud Run
Service details
Edit & deploy new revision
Edit repo settings
Test
Learn
Refresh
Services
Jobs
Worker pools
Domain mappings
student-task-tracker
Region: us-central1
URL: https://student-task-tracker-463924798938.us-central1.run.app
Scaling: Auto (Min: 3)
Build History
Metrics
SLOs
Logs
Revisions
Triggers
Networking
Security
YAML
Predefined
Create uptime check
Annotations (2)
Last 1 day
Container CPU utilization
Container memory utilization
Sent bytes
Received bytes
UTC+5:30
Aug 4
6:00 AM
12:00 PM
50%
95%
99%
Release Notes
29°C
Partly sunny
Search
ENG
IN
15:43:33
04-08-2025


Artifact Registry API – APIs & S
Metrics – ...k-tracker – Cloud R
console.cloud.google.com/run/detail/us-central1/student-task-tracker/metrics?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Cloud Run
Service details
Edit & deploy new revision
Edit repo settings
Test
Learn
Refresh
Services
Jobs
Worker pools
Domain mappings
student-task-tracker
Region: us-central1
URL: https://student-task-tracker-463924798938.us-central1.run.app
Scaling: Auto (Min: 3)
Build History
Metrics
SLOs
Logs
Revisions
Triggers
Networking
Security
YAML
Predefined
Create uptime check
Annotations (2)
Last 1 day
UTC+5:30
Aug 4
6:00 AM
12:00 PM
google
internet
private
Max concurrent requests
Container startup latency
50%
95%
99%
Release Notes
29°C
Partly sunny
Search
ENG
IN
15:43:42
04-08-2025


Artifact Registry API – APIs & S
Security –...k-tracker – Cloud R
console.cloud.google.com/run/detail/us-central1/student-task-tracker/security?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Cloud Run
Service details
Edit & deploy new revision
Edit repo settings
Test
Learn
Refresh
Services
Jobs
Worker pools
Domain mappings
student-task-tracker
Region: us-central1
URL: https://student-task-tracker-463924798938.us-central1.run.app
Scaling: Auto (Min: 3)
Build History
Metrics
SLOs
Logs
Revisions
Triggers
Networking
Security
YAML
Authentication
Allow public access
No authentication checks will be performed.
Require authentication
Select between Identity and Access Management (IAM) and/or Identity-Aware Proxy (IAP).
Binary Authorization
Status: Disabled.
Enable Binary Authorization API
Threat Detection
Preview
Threat Detection: Disabled
Release Notes
https://console.cloud.google.com/run/detail/us-central1/student-task-tracker/security?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
29°C
Partly sunny
Search
ENG
IN
15:44:42
04-08-2025
YAML – stu...k-tracker – Cloud
console.cloud.google.com/run/detail/us-central1/student-task-tracker/yaml/view?inv=1&invt=Ab4kWw&project=trusty-agility-465406-d1
Edit
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
name: student-task-tracker
namespace: '463924798938'
selfLink: /apis/serving.knative.dev/v1/namespaces/463924798938/services/student-task-tracker
uid: f232c305-158c-4b30-8529-499cc329db7c
resourceVersion: AAY7hzUQj1c
generation: 3
creationTimestamp: '2025-08-04T10:00:01.580164Z'
labels:
commit-sha: 56179f109966e7a707f0ad0ee4665f5705c2e5a1
gcb-build-id: f1397e85-187c-4c6e-8277-652cf2b97eb2
gcb-trigger-id: 6fbc6f84-2471-4b7e-b572-cb5d772d3a98
gcb-trigger-region: global
managed-by: gcp-cloud-build-deploy-cloud-run
cloud.googleapis.com/location: us-central1
annotations:
serving.knative.dev/creator: nithishkumarb13@gmail.com
serving.knative.dev/lastModifier: 463924798938-compute@developer.gserviceaccount.com
run.googleapis.com/client-name: gcloud
run.googleapis.com/client-version: 532.0.0
run.googleapis.com/operation-id: 685e4d04-5c94-405e-a333-1b22a126d7d2
run.googleapis.com/ingress: all
run.googleapis.com/ingress-status: all
run.googleapis.com/invoker-iam-disabled: 'true'
run.googleapis.com/minScale: '3'
15:45:45