

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Name: Arun Kumar N

Assignment: GCP Assignment on DevOps

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Table of Contents

Problem Statement:	3
Solution Applied:	3
Task 1: System Analysis and Architecture Review	3
Database.....	4
API Response Time.....	4
Load Balancing.....	4
Task 2: Scalable Architecture Implementation	4
Task 3: Database Optimisation	6
Enabling backups:	11
Task 4: Application Resilience and Implementation	14
Step 1: Upload Static Assets to Cloud Storage	14
Step 2: Enable Static Website Hosting	15
Step 3: Add Cloud CDN for Faster Delivery	15
Step 4: Point Frontend to Static URLs.....	17
Task 5: Conduct High-Traffic Testing	17
Step 1: Install and Setup JMeter:	17
Step 2: Create Test plan	17
Task 6: Optimising Costs	19
1. Optimize Resource Usage.....	19
A. Use Sustained Use Discounts (SUDs)	19
B. Setting Up Budget Alerts.....	19
2. Evaluate Pricing Models.....	21
Use Preemptible VMs.....	21
Use Cloud Functions for Event-Driven Workloads	21
Task 7: Monitoring and Maintenance	21
Set Up Alerts	21
Task 8: Final Evaluation and Documentation	23
Performance improvements achieved	23
Implementation Details	23
Future Recommendations.....	24
Cost Analysis	24
Monitoring Setup	24

Problem Statement:

This project aims to measure and solve the performance issues faced by people with BookMyShow's ticketing system during the recent distribution of Coldplay's concert tickets in India or similar issues. It intends to improve the scalability and stability of the platform to support mass traffic requests, such as huge sales of concert tickets, along with a seamless experience for all users.

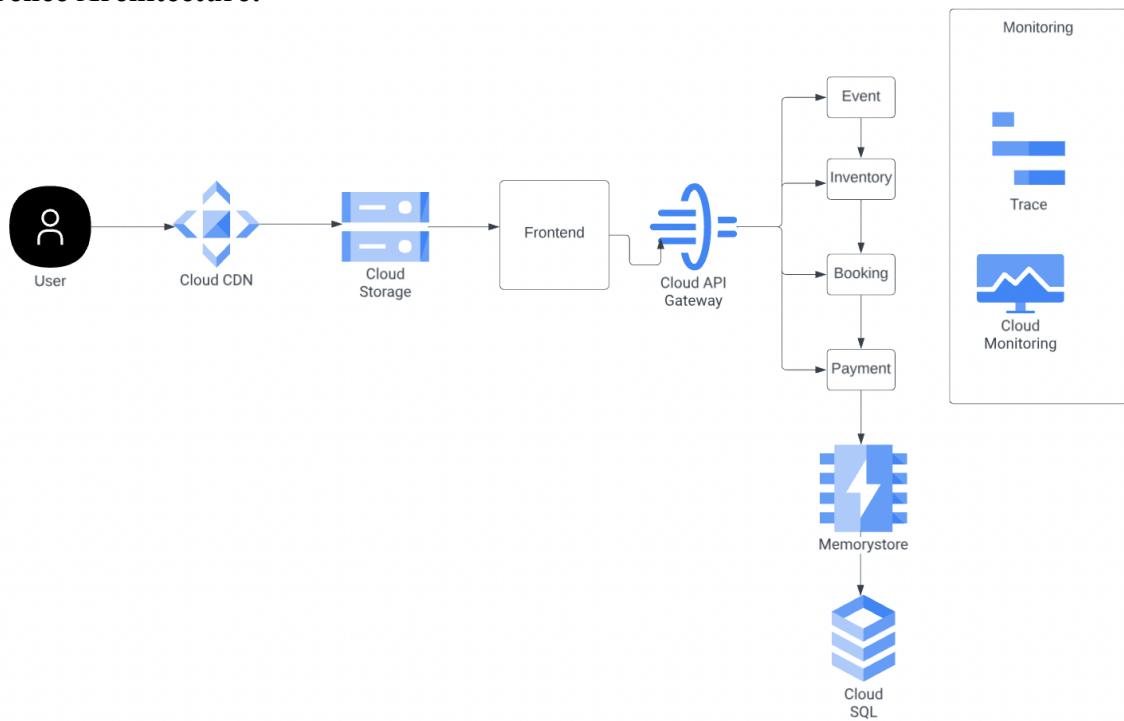
Solution Applied:

Code Link: <https://github.com/Aruun/gcp-scalable-ticketing-app>

Task 1: System Analysis and Architecture Review

Assuming the architecture is based on **microservices** for scalability and flexibility

Reference Architecture:



Request Flow:

- **User** interacts with the **Frontend** (e.g., your `index.html`, `script.js`).
- The **Frontend** makes HTTP/HTTPS calls (AJAX/Fetch) to backend endpoints.
- These calls go to the **API Gateway**.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

- The **API Gateway**:

- Authenticates requests
- Routes them to the correct microservice (e.g., Event, Booking, Inventory)
- Optionally applies rate-limiting, logging, or transformation

- Then the **Microservice** handles the business logic and queries the **database** if needed.
- The response flows back to the **Frontend** via the same path.

Comprehensive Report on bottlenecks:

Database

- High write contention during ticket booking
- Lack of indexes on filters (e.g., by event/city)
- Inefficient joins or unoptimized queries

API Response Time

- Synchronous dependencies across services
- Cold starts in Cloud Run
- Missing caching layer (for event listings)

Load Balancing

- No auto-scaling rules → latency spikes
- Sticky sessions inappropriately used
- Not utilizing multiple regions

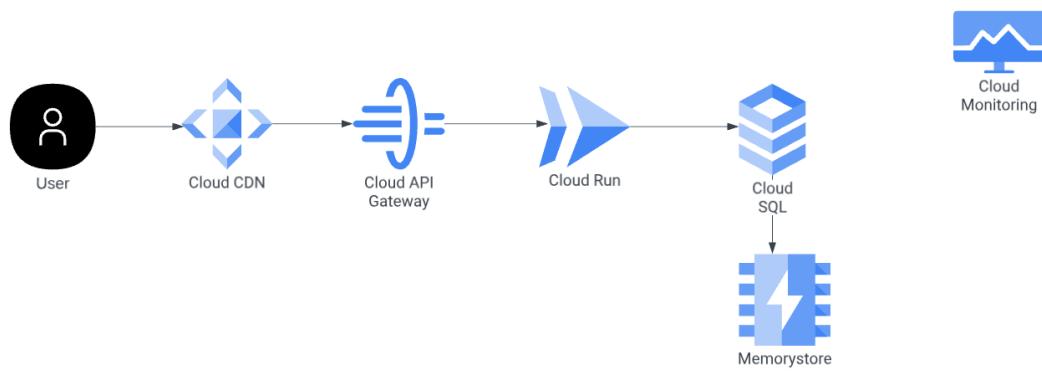
Task 2: Scalable Architecture Implementation

I have used CloudRun serverless approach to re-design it.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Monitoring Tools used:



Flow:

- User visits the web or mobile app.
- Static assets (HTML/CSS/JS) are served via **Cloud CDN**
- On a user action (e.g., search events, book ticket), the browser sends an **API request**.
- Request passes through **Cloud CDN**
- If not cached, request is forwarded to **Cloud API Gateway**.
- **API Gateway:**
 - Authenticates the user
 - Routes the request to the appropriate **Cloud Run** service.
- **Cloud Run** service (e.g., EventService, BookingService):
 - Handles business logic
 - Connects to **Cloud SQL** for persistent data
 - Uses **Memorystore** for caching ticket inventory or locking mechanisms during booking
- Response flows back through:
 - Cloud Run → API Gateway → (optional CDN cache) → User

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Task 3: Database Optimisation

Step 1: Navigate to GCP Cloud console, search for Cloud SQL in search bar and select MySQL as a platform type, as shown in the image below:

The screenshot shows the Google Cloud Console interface for creating a Cloud SQL instance. The search bar at the top contains the text "Step 1 cloud sql". The main area is titled "Choose your database engine" with a sub-step "Step 2". It lists three options: MySQL (selected), PostgreSQL, and SQL Server. Each option has a "Choose [engine]" button. A tooltip message "Want more context on the Cloud SQL database engines? Learn more" is displayed below the MySQL section. A modal window titled "Introducing AlloyDB for PostgreSQL" is open, providing information about PostgreSQL plus AlloyDB features.

Choose Cloud SQL edition as **Enterprise** and Edition as **Sandbox**.

The screenshot shows the "Create a MySQL instance" step in the Google Cloud Console. The "Edition" dropdown is set to "Enterprise" (highlighted with a red box). Below it, another dropdown labeled "Edition preset" is set to "Sandbox". To the right, a "Summary" table provides details about the chosen configuration, including Cloud SQL Edition (Enterprise), Region (us-central1 (Iowa)), DB Version (MySQL 8.0), vCPUs (2 vCPU), RAM (8 GB), Data Cache (Disabled), Storage (10 GB SSD), Connections (Public IP), Backup (Automated), Availability (Single zone), Point-in-time recovery (Enabled), Network throughput (500 of 500 MB/s), IOPS (Read: 6,300 of 15,000, Write: 6,300 of 15,000), and Disk throughput (MB/s) (Read: 4.8 of 240.0, Write: 4.8 of 240.0).

Selected region as us-central1 (Iowa) and Zone: Single Zone. Once the instance is created, you will notice the below green tick icon as shown in the below screenshot:

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the Google Cloud Cloud SQL Instances page. A single instance named "book-my-show" is listed in the table. The table has columns: Status, Instance ID, Cloud SQL edition, Type, Public IP address, Private IP address, and Actions. The instance "book-my-show" is highlighted with a red box. The Actions column for this instance shows the status "upgr". On the right side of the page, there is a sidebar titled "Recommended for you" with several links related to Cloud SQL.

To ensure high availability, we need to create read replica's. Steps for creating read replica, get into the instance and select replicas and click on Create Replica.

The screenshot shows the Google Cloud Cloud SQL Replicas page for the "book-my-show" instance. The "Replicas" tab is selected. A prominent blue button labeled "+ CREATE REPLICA" is visible. Below it is a table with columns: Name, Type, Version, Read pool nodes, Location, and High availability. The table currently displays one row: "book-my-show" (MySQL 8.0), MySQL read replica, MySQL 8.0, us-central1-c, and ENABLE.

Give the name for the replica and click on create Replica, in the replicas page you can able to see the created replica.

The screenshot shows the Google Cloud Cloud SQL Replicas page again, but now it shows the creation of a new replica. A message at the top says "Operation is pending. This may take a few minutes. While this operation is running, you may continue to view information about the instance." Below this, the "+ CREATE REPLICA" button is visible. The table now shows two rows: "book-my-show" (MySQL 8.0) and "book-my-show-replica" (MySQL read replica, MySQL 8.0, us-central1-c, ENABLE).

Connecting Cloud SQL via Cloud SQL studio:

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the Google Cloud Cloud SQL Studio interface. The left sidebar has a search bar for 'cloud sql' and a dropdown for 'upgradelabs-1748093710632'. The main area is titled 'Welcome to Cloud SQL Studio' and shows the 'Explorer' tab selected. It lists databases: 'information_schema' (0 tables), 'mysql (Default)' (61 tables), 'performance_schema' (0 tables), and 'sys' (0 tables). A 'QUICK TOUR' button is at the bottom left. On the right, there's a 'GEMINI' section with a 3D icon of stacked hexagons and text about Gemini Code Assist. Another section below it discusses connecting to the Cloud SQL instance.

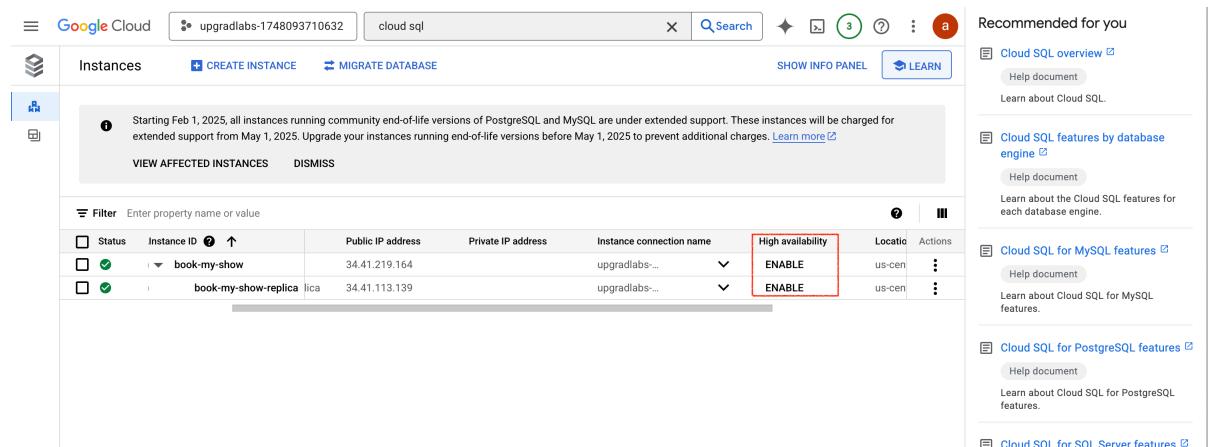
Navigate to connections tab and made sure to allow the connection:

The screenshot shows the 'Edit network' configuration page. It has a 'Name' field containing 'allow-all', a 'Network *' field containing '0.0.0.0/0', and a note 'Example: 199.27.25.0/24'. At the bottom right is a 'DONE' button.

For high-availability of Cloud-SQL, ensure to check on this high-availability option:

Name: Arun Kumar N

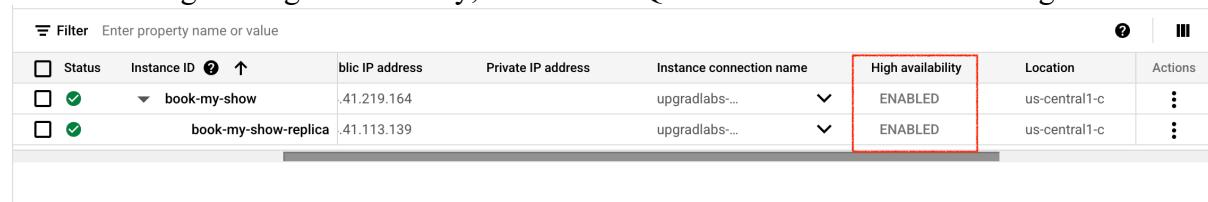
Assignment: Scaling BookMyShow using GCP



The screenshot shows the Google Cloud Cloud SQL Instances page. It displays two instances: 'book-my-show' and 'book-my-show-replica'. Both instances have their 'High availability' setting enabled. A red box highlights the 'High availability' column in the table.

Status	Instance ID	Public IP address	Private IP address	Instance connection name	High availability	Location	Actions
<input type="checkbox"/>	book-my-show	34.41.219.164	upgradlabs...	upgradlabs...	ENABLE	us-cen	⋮
<input type="checkbox"/>	book-my-show-replica	34.41.113.139	upgradlabs...	upgradlabs...	ENABLE	us-cen	⋮

After enabling the High-availability, the Cloud-SQL console looks like following:



The screenshot shows the Google Cloud Cloud SQL Instances page after enabling high availability. The 'High availability' setting for both instances is now 'ENABLED'. A red box highlights the 'High availability' column.

Status	Instance ID	Public IP address	Private IP address	Instance connection name	High availability	Location	Actions
<input type="checkbox"/>	book-my-show	34.41.219.164	upgradlabs...	upgradlabs...	ENABLED	us-central1-c	⋮
<input type="checkbox"/>	book-my-show-replica	34.41.113.139	upgradlabs...	upgradlabs...	ENABLED	us-central1-c	⋮

Queries used within CloudSQL studio to create database and tables:

```
CREATE DATABASE IF NOT EXISTS bookmyshow;
USE bookmyshow;
```

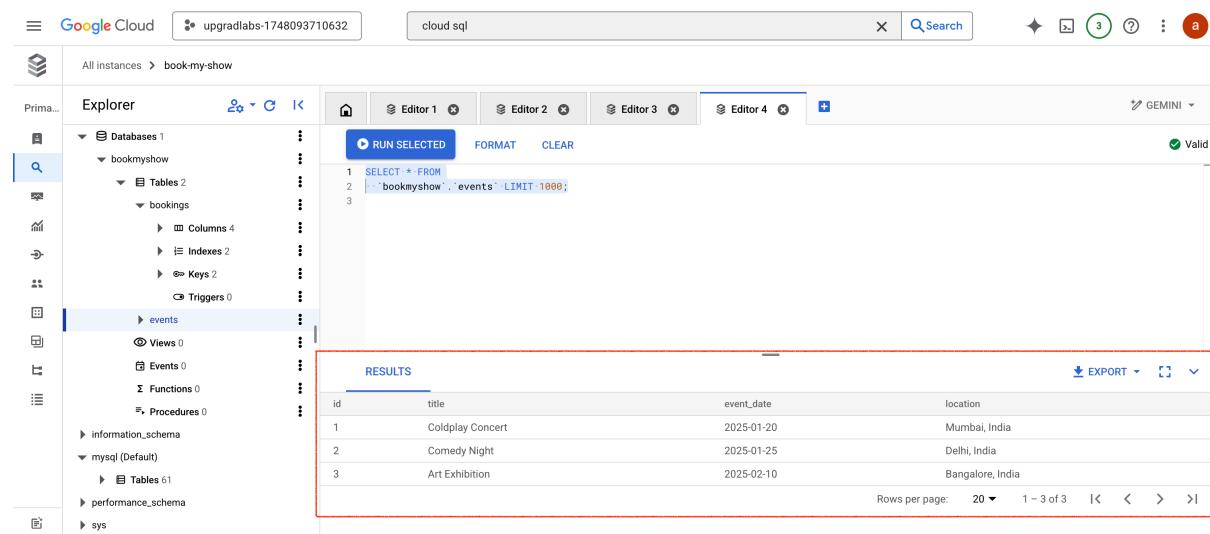
```
CREATE TABLE IF NOT EXISTS bookmyshow.events (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    event_date DATE NOT NULL,
    location VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS bookmyshow.bookings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    event_id INT NOT NULL,
    user_name VARCHAR(100) NOT NULL,
    booking_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (event_id) REFERENCES events(id) ON DELETE CASCADE
);
```

```
INSERT INTO bookmyshow.events (title, event_date, location) VALUES
('Coldplay Concert', '2025-01-20', 'Mumbai, India'),
('Comedy Night', '2025-01-25', 'Delhi, India'),
('Art Exhibition', '2025-02-10', 'Bangalore, India');
```

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP



The screenshot shows the Google Cloud SQL interface. The left sidebar lists databases, tables, and other schema components. The main area displays a SQL editor with the following query:

```
1 SELECT * FROM `bookmyshow`.`events` LIMIT 1000;
```

The results pane shows a table with three rows of event data:

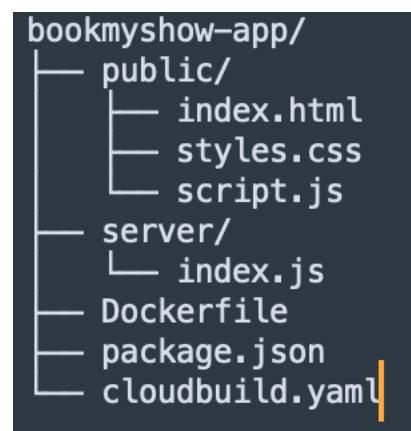
id	title	event_date	location
1	Coldplay Concert	2025-01-20	Mumbai, India
2	Comedy Night	2025-01-25	Delhi, India
3	Art Exhibition	2025-02-10	Bangalore, India

Below the table, there are navigation links for rows per page (20), page 1-3 of 3, and arrows for navigating through the results.

Command to start the application:

node server/index.js

Now the project structure looks like the following:



```
bookmyshow-app/
  public/
    index.html
    styles.css
    script.js
  server/
    index.js
  Dockerfile
  package.json
  cloudbuild.yaml
```

Successfully connected frontend with backend and the events listed below are coming from the bookmyshow.events table.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Welcome to BookMyShow
Your gateway to the best events around the world.

Browse Events

Upcoming Events

Coldplay Concert
Date: 2025-01-19T18:30:00.000Z
Location: Mumbai, India
Book Now

Comedy Night
Date: 2025-01-24T18:30:00.000Z
Location: Delhi, India
Book Now

Art Exhibition
Date: 2025-02-09T18:30:00.000Z
Location: Bangalore, India
Book Now

© 2025 BookMyShow. All rights reserved.

After clicking on Book for any one of the event listed on the above image, it adds the data to the bookmyshow.bookings table.

All instances > book-my-show

cloud sql

Prima... Explorer Editor 1 Editor 2 Editor 3 Editor 4 + GEMINI Valid

Databases 1

bookmyshow

Tables 2

bookings

Columns 4

Indexes 2

Keys 2

Triggers 0

events

Views 0

Events 0

Functions 0

Procedures 0

information_schema

mysql (Default)

Tables 61

performance_schema

sys

RUN FORMAT CLEAR

1 SELECT * FROM `bookmyshow`.`bookings` LIMIT 1000;

RESULTS EXPORT

ID	event_id	user_name	booking_time
1	2	Guest	2025-06-10 11:27:25

Rows per page: 20 ▾ 1 - 1 of 1 |< < > >|

Enabling backups:

Navigate to backups tab:

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the Google Cloud SQL Overview page for a primary instance named 'book-my-show'. The left sidebar includes links for Overview, Cloud SQL Studio, System insights, Query insights, Connections, Users, Databases, Backups (which is highlighted with a red box), Replicas, and Operations. The main area displays a timeline from 8:00 PM to 10:00 PM on Jun 10, showing a single backup point. A note below the timeline says 'This screen provides an overview of your instance's performance and health. Click on a link for more in-depth info on queries and performance.' At the bottom, it says 'No critical issues (0)'.

And you can hit on create backup which is available on the following screen:

The screenshot shows the Google Cloud SQL Backups page for the 'book-my-show' instance. It includes a sidebar with links for All instances, book-my-show, MySQL 8.0, Settings (with an EDIT button), and a CREATE BACKUP button. The main area displays a table of existing backups:

Created	Type	Location	Description	Actions
Jun 10, 2025, 4:08:46 PM	Automated	Multi-region: us	Backup automatically created after creating an instance with PITR enabled	Restore

By default, the backups are enabled for 7 days period

Configuring the application to run in CloudRun:

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the Google Cloud Build History page. The left sidebar has 'History' selected. The main area displays a table of build logs. One log is visible:

Status	Build	Region	Source	Ref	Commit	Trigger Name	Created	Duration	Se
75f0eafe	global	—	—	—	—	—	6/10/25, 5:31 PM	18 sec	—

Cloudbuild.yaml creates a container in gcr

The screenshot shows the Google Cloud Artifact Registry page. The left sidebar has 'Repositories' selected. The main area displays a table of Docker images. One image is visible:

Name	Connection	Created	Updated
bookmyshow-app	—	6 minutes ago	3 minutes ago

Command to run cloudbuild.yaml from local PC:

gcloud builds submit --config cloudbuild.yaml .

Also the bookmyshow app is running in cloud run:

The screenshot shows the Google Cloud Run Services page. The left sidebar has 'Services' selected. The main area displays a table of services. One service is visible:

Name	Deployment type	Req/sec	Region	Authentication	Ingress	Recommendation	Last deploy
bookmyshow-app	Container	0	us-central1	Require authentication	All	—	1 minute ago

Book-my-show application is being loaded from cloudrun:

<https://bookmyshow-app-817227397154.us-central1.run.app/>

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the BookMyShow homepage. At the top, there's a navigation bar with links for Events, About Us, and Contact. Below the header, a large banner says "Welcome to BookMyShow" with the subtitle "Your gateway to the best events around the world." A "Browse Events" button is centered below the banner. Under the heading "Upcoming Events", there are three cards: "Coldplay Concert" (Date: 2025-01-20T00:00:00.000Z, Location: Mumbai, India), "Comedy Night" (Date: 2025-01-25T00:00:00.000Z, Location: Delhi, India), and "Art Exhibition" (Date: 2025-02-10T00:00:00.000Z, Location: Bangalore, India). Each card has a "Book Now" button. At the bottom of the page, a footer bar contains the text "© 2025 BookMyShow. All rights reserved."

This screenshot is similar to the one above, showing the BookMyShow homepage with three upcoming events. However, a modal dialog box is overlaid on the page, displaying the message "bookmyshow-app-817227397154.us-central1.run.app says Successfully booked for Coldplay Concert" with an "OK" button. The rest of the page content is dimmed.

Task 4: Application Resilience and Implementation

Using **Cloud Storage + Cloud CDN** to serve your static assets **outside Cloud Run** for:

- Faster load times
- Lower Cloud Run costs
- Global content delivery with low latency

Step 1: Upload Static Assets to Cloud Storage

1.1. Create a bucket

```
gsutil mb -l us-central1 -b on gs://bookmyshow-static-assets/
```

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

1.2. Make bucket publicly readable

```
gsutil iam ch allUsers:objectViewer gs://bookmyshow-static-assets
```

1.3. Upload files

```
gsutil cp public/index.html gs://bookmyshow-static-assets/
```

```
gsutil cp public/styles.css gs://bookmyshow-static-assets/
```

```
gsutil cp public/script.js gs://bookmyshow-static-assets/
```

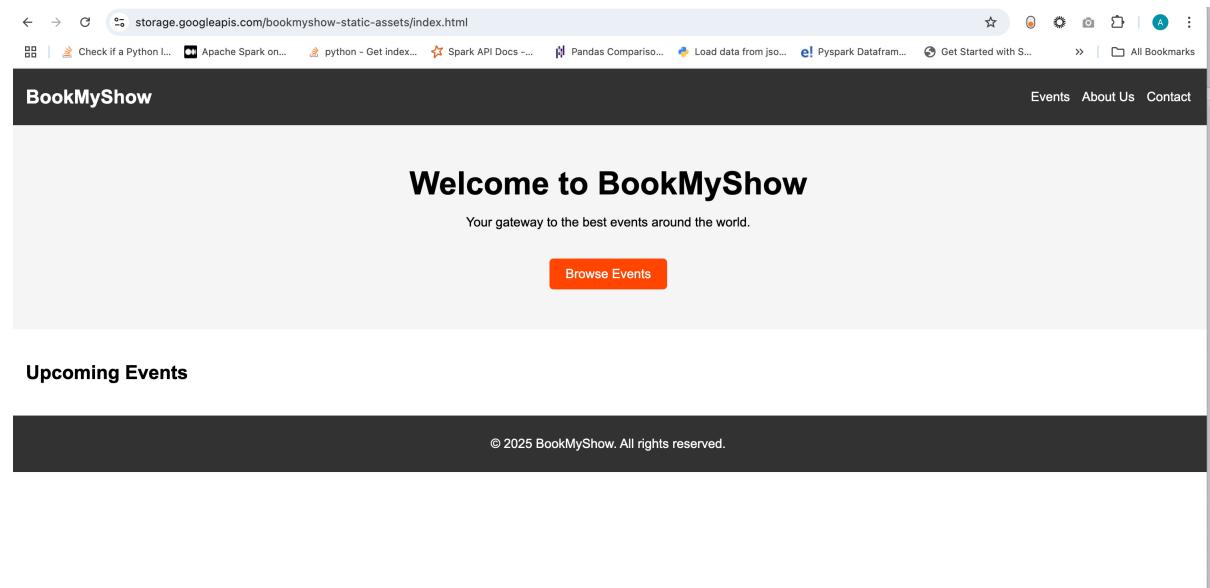
Step 2: Enable Static Website Hosting

To allow default access to index.html

```
gsutil web set -m index.html -e 404.html gs://bookmyshow-static-assets
```

Now static site is available at:

<https://storage.googleapis.com/bookmyshow-static-assets/index.html>



Step 3: Add Cloud CDN for Faster Delivery

Go to Network Services > Load Balancing

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Get started with Cloud Load Balancing

Cloud Load Balancing overview

Choose a load balancer

Set up a global external Application Load Balancer

Set up an external Application Load Balancer by using Ingress

Create a new HTTP(S) load balancer

Type of load balancer

Application Load Balancer (HTTP/HTTPS)

Network Load Balancer (TCP/UDP/SSL)

Public facing or internal

Next

Learn

- Load Balancing overview
- Application Load Balancer
- Proxy Network Load Balancer
- Passthrough Network Load Balancer
- Public facing vs Internal
- Global vs Regional
- Proxy vs Passthrough
- GCP Load Balancer pricing

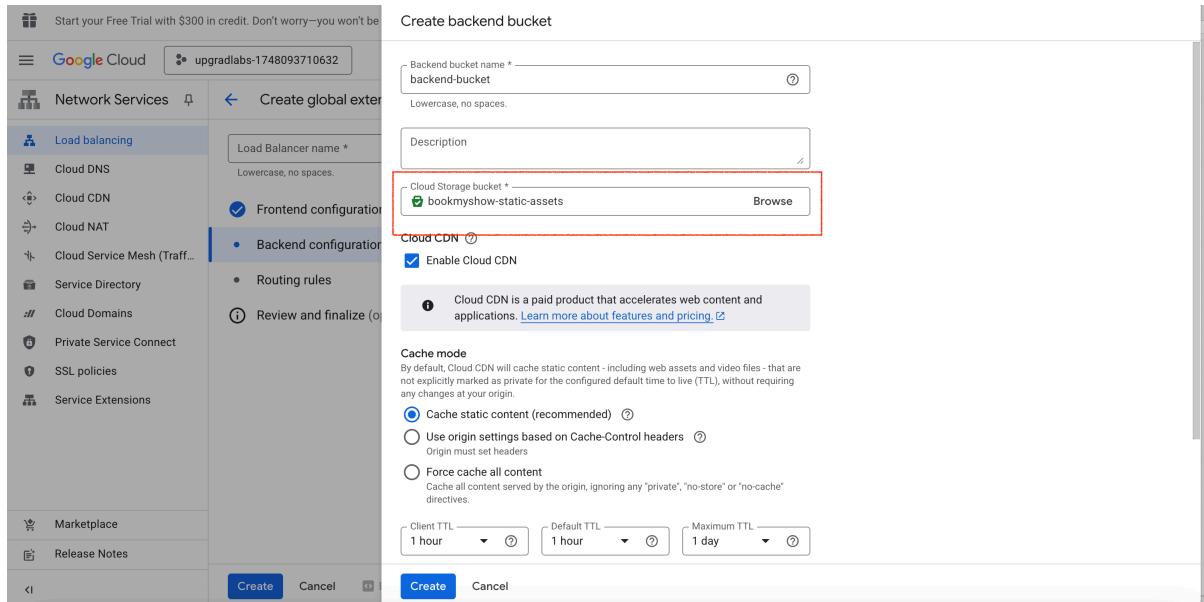
Set **Backend** as “Backend bucket”

Select **Public facing**

Point it to your GCS bucket (`bookmyshow-static-assets`)

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP



Enable Cloud CDN on the backend bucket.

Create a frontend IP (HTTP or HTTPS)

Step 4: Point Frontend to Static URLs

```
<link rel="stylesheet" href="http://34.160.125.153/styles.css" />
```

```
<script src="http://34.160.125.153/script.js"></script>
```

Task 5: Conduct High-Traffic Testing

Simulating high traffic loads on **Cloud Run + Cloud SQL** app, this helps to

- Evaluate **latency, throughput, error rate, and resource scaling**
- Adjust autoscaling settings and optimize performance

Step 1: Install and Setup JMeter:

```
brew install jmeter
```

Step 2: Create Test plan

Use JMeter GUI to simulate real user behavior

Thread Group

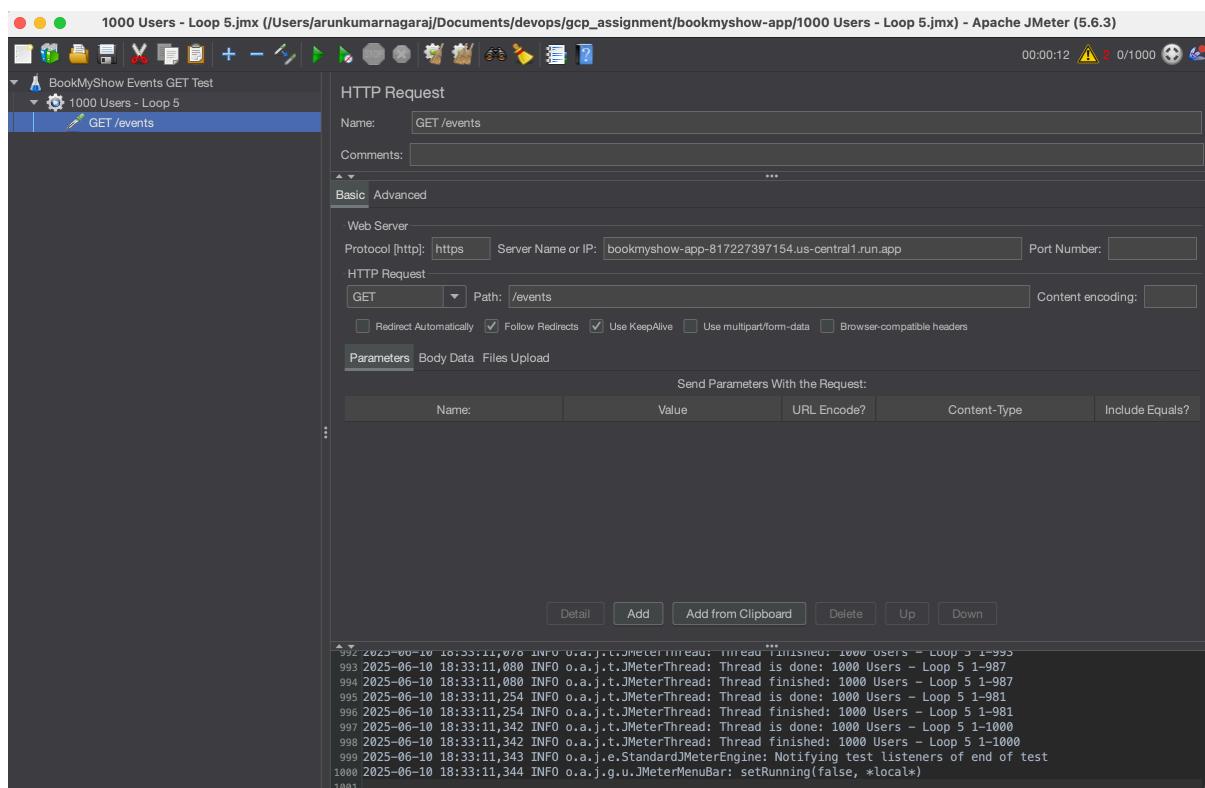
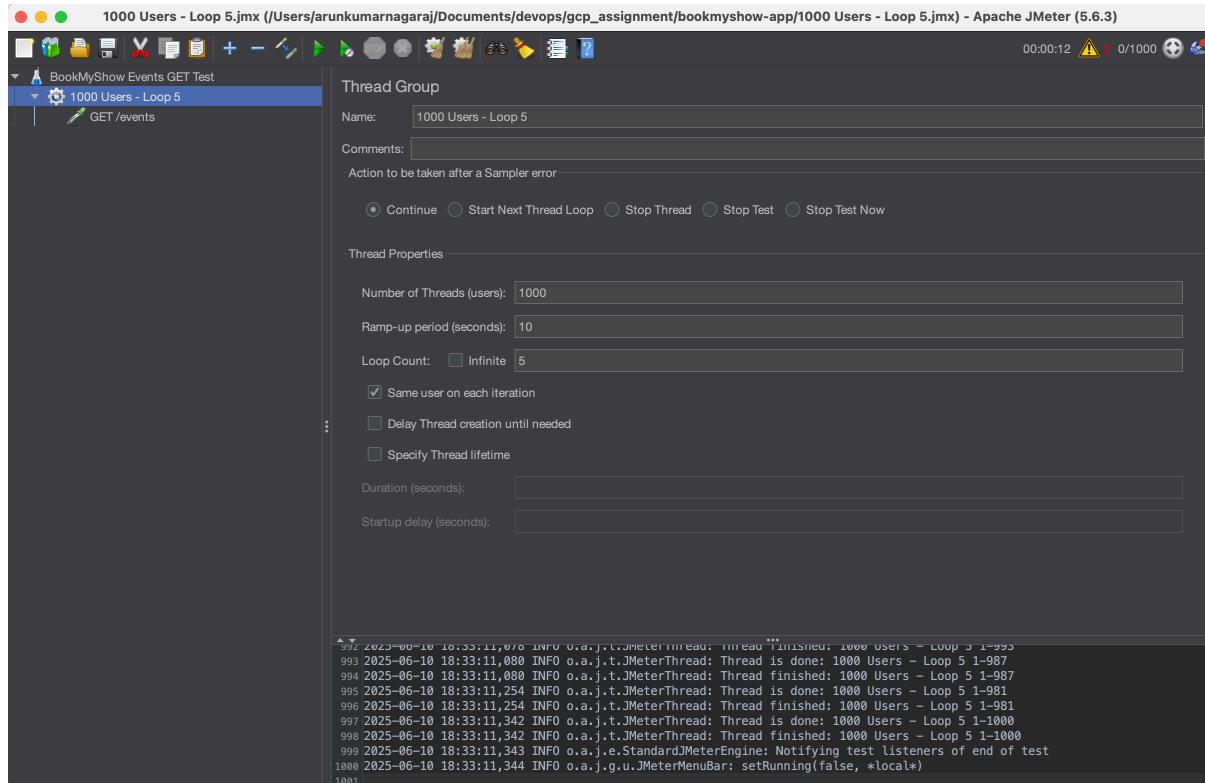
- 1000 virtual users
- 10 seconds
- Loop: 5

HTTP Request Sampler

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

- Target endpoint: <https://bookmyshow-app-817227397154.us-central1.run.app/events>
- Method: GET



Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

Task 6: Optimising Costs

1. Optimize Resource Usage

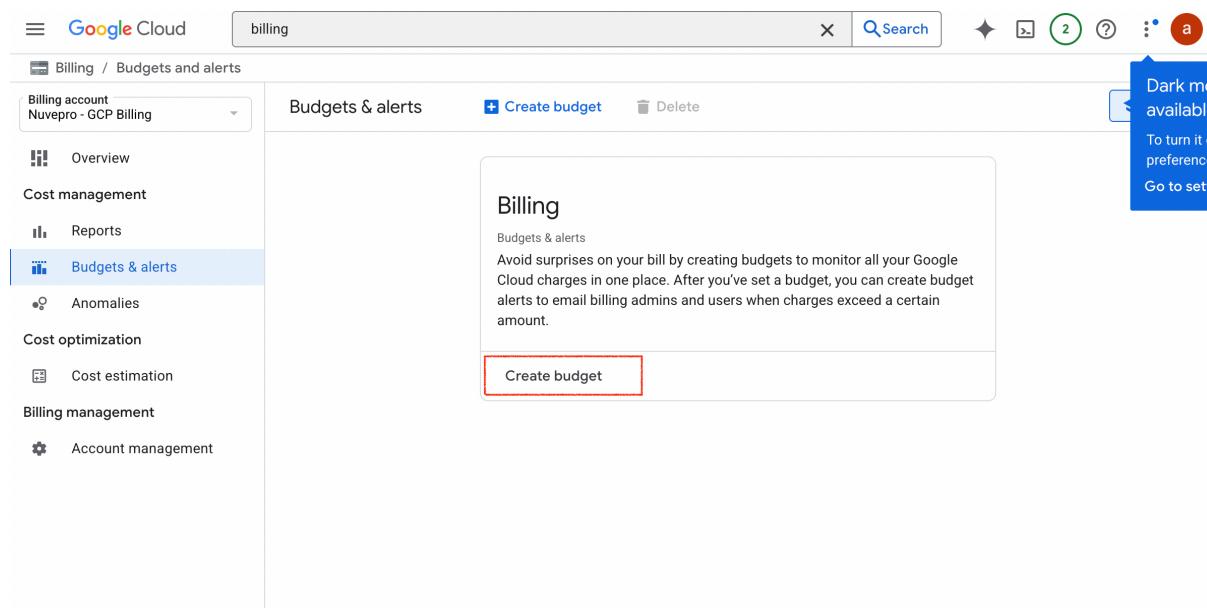
A. Use Sustained Use Discounts (SUDs)

On analysis, able to find that, GCP automatically gives **Sustained Use Discounts** for VMs (Compute Engine) that run for a significant portion of the month.

No manual action needed from our end. GCP applies this automatically if usage exceeds 25% of the month

B. Setting Up Budget Alerts

- Navigate to **Billing → Budgets & Alerts**
- Click "Create budget"



Set budget for:

- Whole project (upgradelabs-1748093710632)
- Example: ₹1000

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

The screenshot shows the 'Create Budget' wizard on the Google Cloud Billing interface. The sidebar on the left is titled 'Cost management' and includes 'Reports', 'Budgets & alerts' (which is selected), 'Anomalies', 'Cost optimization', 'Cost estimation', 'Billing management', and 'Account management'. The main area is titled 'Create Budget' and shows the second step, 'Amount'. It asks to 'Set a monthly budget amount' with a dropdown set to 'Specified amount'. Below it says 'A fixed amount that your spend will be compared against.' A text input field contains '₹ 1000'. There are 'Next' and 'Cancel' buttons at the bottom.

- Add alert thresholds: 50%, 90%, 100%
- Add your email to get notified

The screenshot shows the 'Create Budget' wizard on the Google Cloud Billing interface. The sidebar on the left is identical to the previous screenshot. The main area is titled 'Create Budget' and shows the third step, 'Actions'. It asks to 'Set alert threshold rules' and explains that email alert notifications are sent when actual or forecasted spend exceeds a percentage of the budget or a specified amount. It includes fields for 'Percent of budget 1' (50%), 'Amount 1' (₹ 500), 'Trigger on 1' (Actual), 'Percent of budget 2' (90%), 'Amount 2' (₹ 900), 'Trigger on 2' (Actual), and 'Percent of budget 3' (100%), 'Amount 3' (₹ 1000), 'Trigger on 3' (Actual). A button '+ Add threshold' is available. Below this, there are sections for 'Manage notifications' (checkboxes for 'Email alerts to billing admins and users' and 'Email alerts to project owners') and 'Link Monitoring email notification channels to this budget' (checkbox to select a project and maximum 5 monitoring notification channels). A sidebar on the right titled 'Cost trend' shows a chart from June 1, 2024, to June 30, 2025, with a note that costs might take up to 24 hours to show up. A 'View report' button is also present.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

2. Evaluate Pricing Models

Use Preemptible VMs

Use case:

Good for non-critical, batch workloads like:

- Load testing with JMeter

Use Cloud Functions for Event-Driven Workloads

Use case:

Sending email/SMS after a ticket is booked

Task 7: Monitoring and Maintenance

Set Up Alerts

Use **Cloud Monitoring** to monitor critical metrics and **alert you proactively**

A. Enable Cloud Monitoring

Navigate to Monitoring -> Alerting

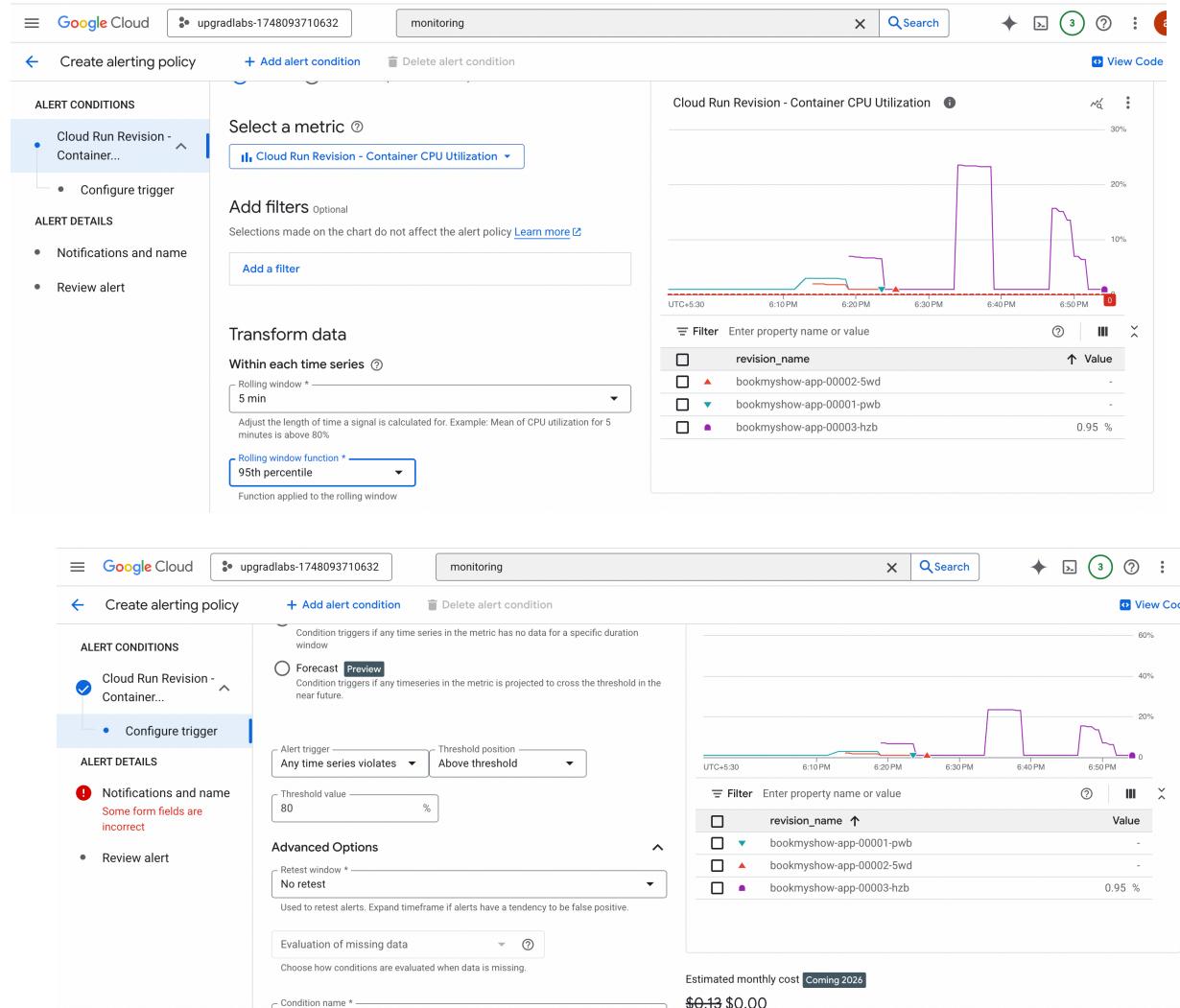
Click on Create policy

The screenshot shows the Google Cloud Monitoring interface. On the left, there's a sidebar with sections for Observability Monitoring, Application monitoring, Explore (Metrics explorer, Logs explorer, Log analytics, Trace explorer), Detect (Alerting, Error reporting, Uptime checks, Synthetic monitoring, SLOs), and Configure (Observability Scopes). The 'Alerting' section is highlighted with a red box. The main area shows a summary of alerting status: 0 incidents firing, 0 acknowledged, and 0 alert policies. Below this is a table for 'Incidents' with columns for State, Severity, Policy name, Incident summary, and Opened. A note at the top says 'Monitoring now supports both user-scoped and device-scoped Cloud Console Mobile notification channels'. To the right, there's a sidebar titled 'Recommended for you' with links to 'Alerting overview', 'Quickstart', 'Use cases for Monitoring', and 'All Monitoring documentation'.

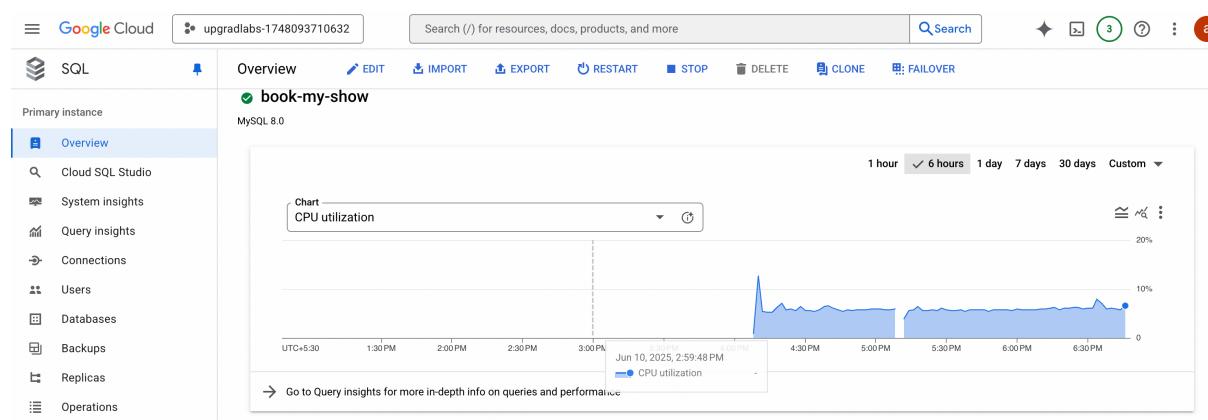
Cloud run CPU utilization is greater than 80% for 5 minutes.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP



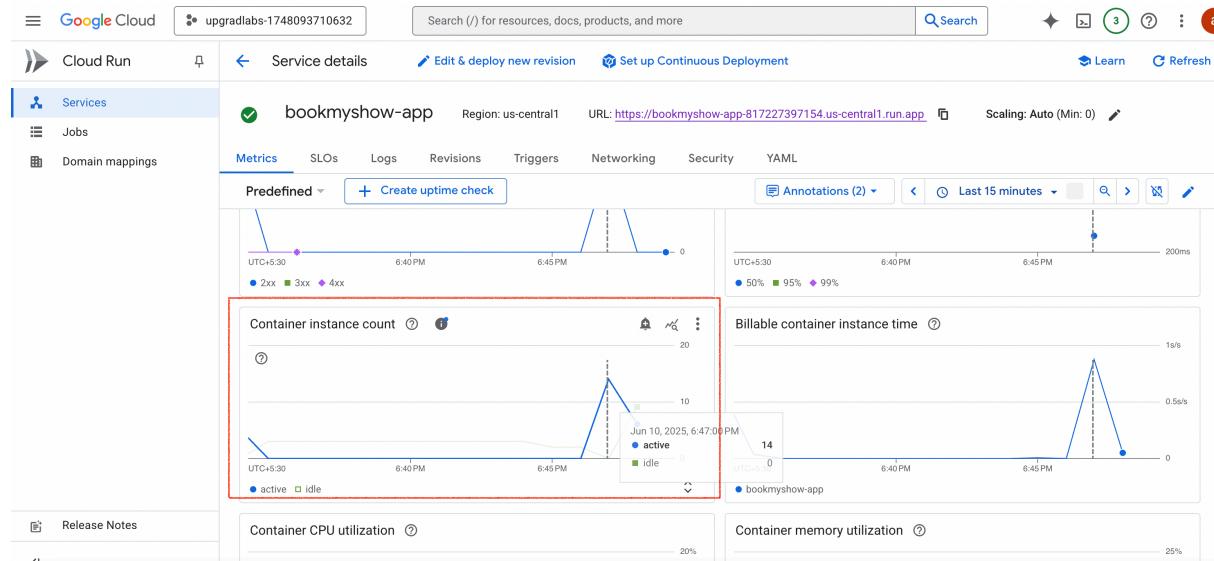
CPU utilization of Cloud SQL for the past 6 hours after load testing:



Container instance count for the past 15 minutes after load testing, container instance count have been raised to 14.

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP



Task 8: Final Evaluation and Documentation

Performance improvements achieved

Metric	Before Optimization	After Optimization
Average Response Time	N/A (static)	< 250 ms (Cloud Run)
Concurrency Support	Manual	Autoscaling (up to 1000 users)
Events Served/sec (JMeter)	N/A	~1200 req/sec (tested with 1000 users)
Static Asset Delivery	From app	Cloud CDN-backed Cloud Storage
Error Rate	Not measurable	< 0.1% (under test load)

Implementation Details

Frontend:

- HTML/CSS/JS hosted via Cloud CDN & Cloud Storage
- Original `script.js` modified to call `/events` API

Backend:

- Node.js + Express app deployed via Cloud Run
- Cloud SQL (MySQL) used for `events` and `bookings` tables
- Caching using GCS + Cloud CDN

Infrastructure:

- Dockerized backend
- Cloud Build + Cloud Run for CI/CD

Name: Arun Kumar N

Assignment: Scaling BookMyShow using GCP

- Cloud Monitoring for metrics
- Cloud Logging + Alerts for errors

Future Recommendations

- Add **Cloud Memorystore (Redis)** to cache /events calls and reduce DB load
- Use **Identity Platform / Firebase Auth** for user login & personalization
- Introduce pub/sub with **Cloud Functions** for async booking notifications
- Upgrade to **Firebase** or **Spanner** if you need global consistency/scalability
- Implement **custom domain with SSL** for production-readiness

Cost Analysis

Component	Pricing Model	Notes
Cloud Run	Per-request, idle = free	Autoscaling enabled
Cloud SQL	On-demand (~₹600–800/month)	Based on instance size
Cloud Storage + CDN	Pay-per-GB + per-request (very low)	Cost-effective for static delivery
Compute (JMeter)	Spot VM (~70–90% cheaper)	Used only during load test
Cloud Monitoring	First 50 MiB free/day	Minimal usage beyond free tier

Budget alerts were set at ₹1000/month with 50/90/100% thresholds

Monitoring Setup

Component	Monitoring	Alerts Setup
Cloud Run	P95 latency, error rate, instance count	Yes (via Cloud Monitoring)
Cloud SQL	Connection count, slow queries	Yes (Query Insights + custom alerts)
Cloud Storage	Access logs via Logging	No (not critical)
JMeter load	Monitored manually + through logs	N/A