International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# What Caused the 12th June 2025 Google Cloud Outage and How to Prevent the Next One

**Sarthak Rohilla**

**Abstract**
In June 2025, Google Cloud Platform (GCP) suffered a major outage that disrupted services around the world. Over 50 key services were affected, including Identity and Access Management (IAM), API Gateway, BigQuery, and third-party apps like Spotify, Discord, and Cloudflare Workers. The root cause? A flawed automated quota update that stripped key API permissions. This paper breaks down what went wrong, explains the failure using a smart-home analogy, and outlines practical engineering steps to prevent such incidents in the future.

## 1. Introduction

As more of the world's digital infrastructure moves to cloud giants like Google Cloud, AWS, and Azure, even small missteps can create massive ripple effects. The June 12, 2025 outage at Google Cloud showed how quickly things can go wrong when automation fails.

This article examines the outage from both technical and human perspectives, drawing on official postmortems and third-party analyses. It also offers recommendations to strengthen the resilience of automated systems in cloud environments.

## 2. What Happened

At 10:49 AM PDT on June 12, users across the globe began reporting widespread service failures in apps running on Google Cloud. These included:

- **Google Services**: Gmail, Google Meet, Drive, Cloud Console
- **Core GCP Services**: IAM, API Gateway, BigQuery, Compute Engine, Cloud Run, Cloud Storage
- **Third-Party Platforms**: Spotify, Snapchat, Discord, Cloudflare Workers KV, and others

Initial recovery started within 40 minutes, but full resolution—especially in the US-Central and Europe-West regions—took nearly 7 hours. Complaints surged past 100,000 within the first hour. To make things worse, Google's own service dashboards were sometimes unavailable, making communication harder during the crisis.

## 3. What Caused It

Weeks before the outage, Google introduced a new quota management system called "Service Control." It wasn't fully protected by safety features like feature flags. On June 12, an automated process pushed an incomplete policy—missing key fields—into Google's globally distributed Spanner database.

The Service Control system couldn't handle missing data. The result: a flood of 503 "Service Unavailable" errors from services that couldn't validate quota. That one faulty update effectively told every system "you don't have permission to run," cutting off access to critical resources like authentication, billing, and compute provisioning.

**4. A Smart-Home Analogy**

Picture a smart home where every device—lights, locks, cameras—is managed by one central system. Now imagine that the system suddenly sends the wrong signal: "No devices are allowed to turn on."

What happens?

- The lights stay off
- The doors won't unlock
- The security cameras shut down
- The entire system grinds to a halt

That's essentially what happened with Google Cloud—except the "home" was the digital backbone of the internet. Engineers had to manually revoke the faulty policy and restart affected systems to get things back online.

**5. How Bad Was the Damage?**

**5.1 Global Impact**

Users and companies across North America, Europe, Asia-Pacific, and India lost access to critical systems. IAM failures disrupted login systems, billing verifications, and API requests—even for apps that had nothing wrong on their end.

**5.2 Third-Party Services**

Platforms like Discord, Spotify, and Cloudflare—though independent—were severely affected because they rely on GCP. Cloudflare's Workers KV couldn't respond to requests. Spotify's music playback stopped. Discord's voice channels broke. Each issued their own updates blaming GCP.

**5.3 Visibility Gaps**

Google's own dashboards were hosted on affected systems, putting users in the dark throughout the height of the disaster. External observers like ThousandEyes and Catchpoint were left to piece together the history from the outside

**6. What Can Be Done: Prevention Strategies**

This wasn't a hardware crash or a cyberattack—it was a software configuration error amplified by automation. Here's what could help avoid a repeat:

**6.1 Roll Out Changes Gradually**

Updates should never go global all at once. Instead, treat configuration changes like software deployments—test them in stages:

- **Tier 1**: Internal environments with fake traffic
- **Tier 2**: Pre-production environments with simulated real-world usage
- **Tier 3**: Canary deployments to a small fraction of live traffic

Also use automated rollback systems tied to error thresholds, and avoid syncing changes globally unless they've passed validation in smaller zones.

**6.2 Validate Everything—and Build for Failure**

The quota system should never have accepted a blank config. A few basic checks would've prevented the outage:

- **Schema Checks**: Every config should follow a strict template. Fields like "quota_limit" and "service-_name" must be mandatory.
- **Static + Fuzz Testing**: Run configs through automated testing tools to catch edge cases.

- **Simulated Environments**: Test updates on replicas of real systems to catch latency or logic failures.
- **Anomaly Detection**: Use machine learning to flag unusual changes—like setting a quota to zero in production.

When something still goes wrong, systems should default to safe behavior:

- **Fail-open** (e.g. allow limited compute if quota data is missing)
- **Fail-closed** (e.g. block access if IAM credentials are invalid)

### 6.3 Add Kill Switches and Emergency Overrides

One of the biggest problems? There was no quick way to shut down the broken quota system. It took hours to fix something that should've taken minutes.

Future systems should include:

- **Feature Flags**: Allow teams to disable new systems instantly
- **Circuit Breakers**: Detect repeated failures and isolate faulty components
- **Backup Routes**: Let services skip broken systems when needed

Had Google used an emergency override to disable Service Control, the issue could've been contained much faster.

## 7. Conclusion

The June 2025 Google Cloud outage is a case study in how automation—while powerful—can magnify simple mistakes at lightning speed. A single broken configuration brought down vast swaths of the internet.

Like a smart home that suddenly stops working because of one bad command, the cloud depends on central systems that need strong checks, rollback paths, and human override options.

Building more resilient infrastructure means:

- Testing changes in stages
- Validating configs like you would production code
- Equipping systems with kill switches for fast intervention

These aren't just best practices—they're critical design choices for the next era of cloud computing.

## 8. References

1. Google Cloud Incident Report
2. BleepingComputer: "Google Cloud Outage Explained"
3. ThousandEyes Monitoring Summary
4. Simon Willison: "GCP IAM Failure Explained"
5. TechRadar: "Spotify, Discord, Google Meet Affected"
6. ForgeCode.dev: Lessons from GCP Outage
7. Catchpoint: Observability Strategies
8. Fierce Network: "What Went Wrong at Google in June 2025"