RELIABILITY IN MODERN CLOUD SYSTEMS

Summer 2025

LOGISTICS

ASSIGNMENT 2

- Assignment is 90% graded
- Grades will be pushed to the assn2_grades branch
- Update by tomorrow morning

ASSIGNMENT 3

- Due today at 5pm CEST

ASSIGNMENT 4

- Project selections have been sent out
- 1st check in on Monday during office hours.
- 1st Check-in requirement:
  - Come up with an implementation + evaluation plan
  - List out what you need to implement (with and without Blueprint integration)
  - What experiments do you think you will need to run?
  - We will authorize every group's list during the check in

ROOT CAUSE ANALYSIS DISCUSSION

PAPER SUMMARY

PAPER SUMMARY

- Enables dynamic monitoring and data collection
- Tracepoints + Aspect-oriented code injection allow for dynamicity
- Baggage Propagation carries the necessary information
  - This paper proposed Baggage as an abstraction
- Users can specify queries which get executed during runtime
  - Queries use the happened-before join operator
  - Happened-before join operator allows for grouping and filtering of events based on properties that causally precede them

DISCUSSION THEMES

- What is the best method for doing root cause analysis?
- What are the challenges for extracting critical paths?
- How do we efficiently compare two traces?
  - How do we compare 1 trace to a group of traces?
  - How do we compare two different groups of traces?

DISCUSSION THEMES

- What is the best method for doing root cause analysis?

DISCUSSION THEMES

- What is the best method for doing root cause analysis?

There is no strict best method. All methods provide some value. Automated methods usually provide more value but are inflexible.

DISCUSSION THEMES

- What are the challenges for extracting critical paths?

DISCUSSION THEMES

- What are the challenges for extracting critical paths?

Spans may seem totally ordered by time but that's not necessarily true as clocks at different nodes may not be synchronized

DISCUSSION THEMES

- How do we efficiently compare two traces?
  - How do we compare 1 trace to a group of traces?
  - How do we compare two different groups of traces?

DISCUSSION THEMES

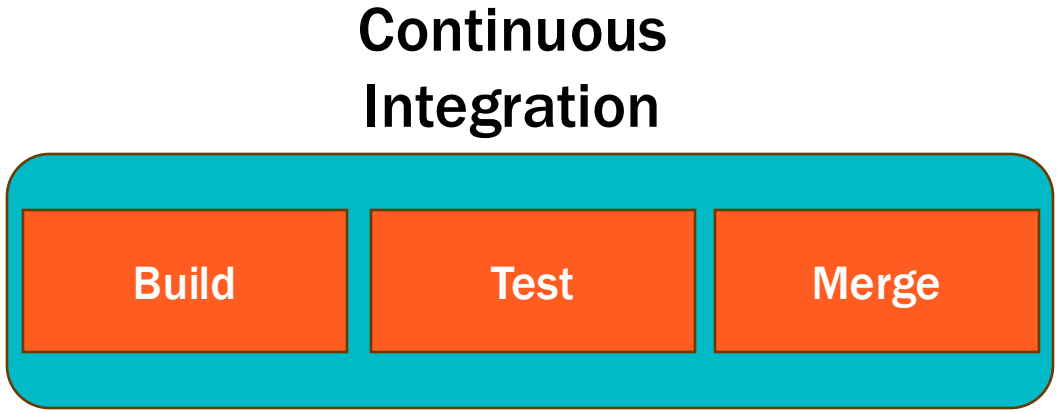
- How do we efficiently compare two traces?
  - How do we compare 1 trace to a group of traces?
  - How do we compare two different groups of traces?

This is currently an open-research question. Comparisons are best effort and easier to do with numerical attributes of traces.

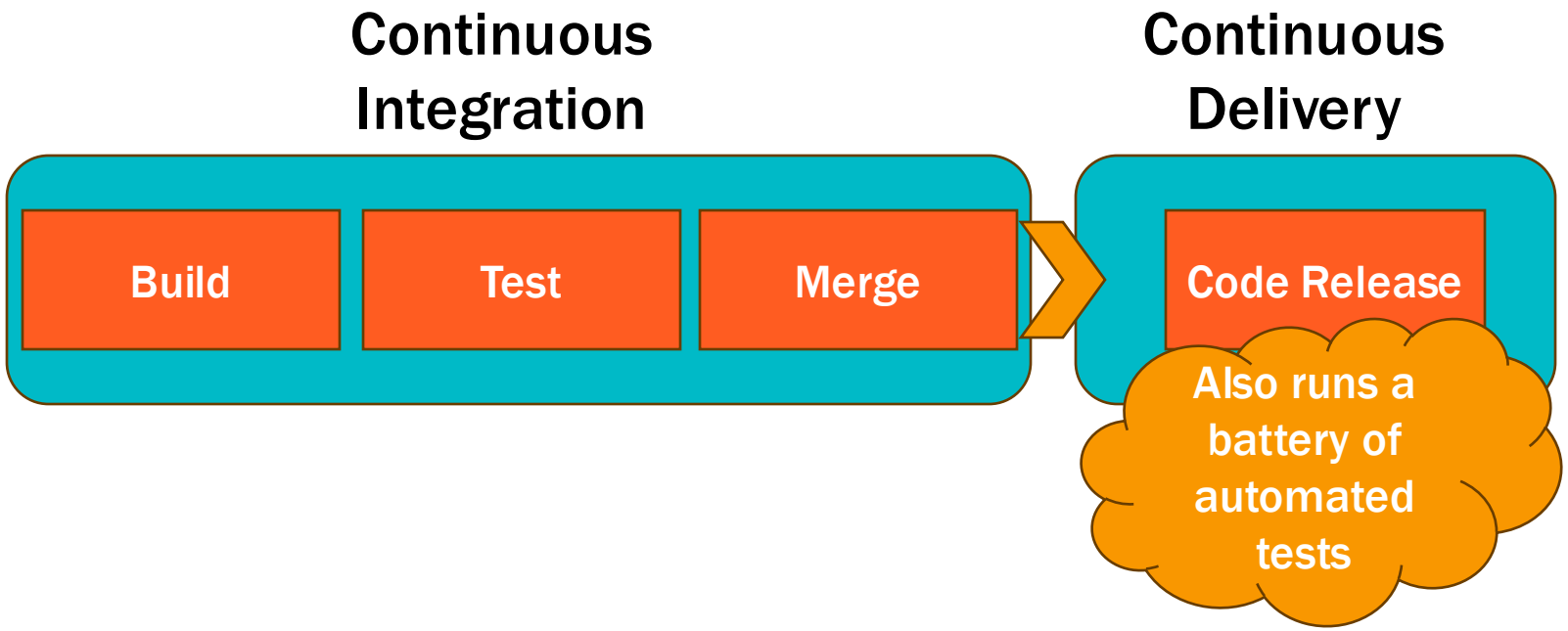
TEST AND VERIFY

A TYPICAL DEVOPS PIPELINE

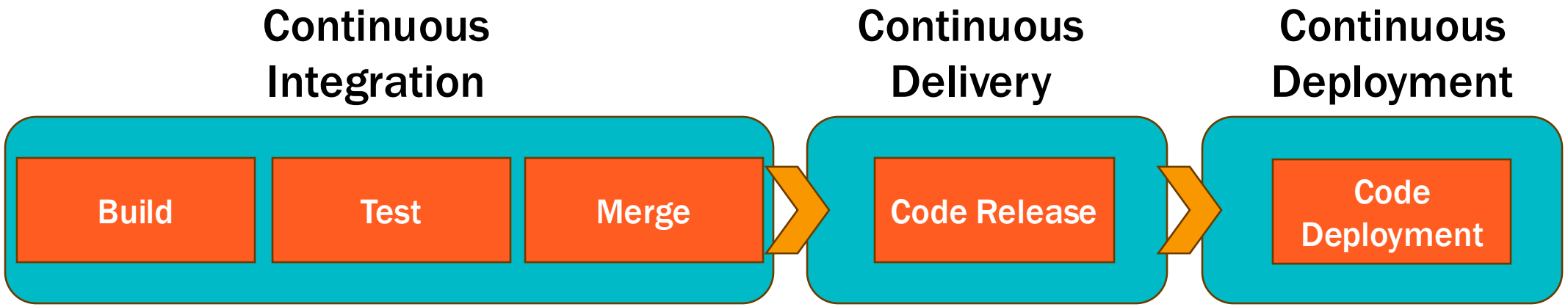
A TYPICAL DEVOPS PIPELINE



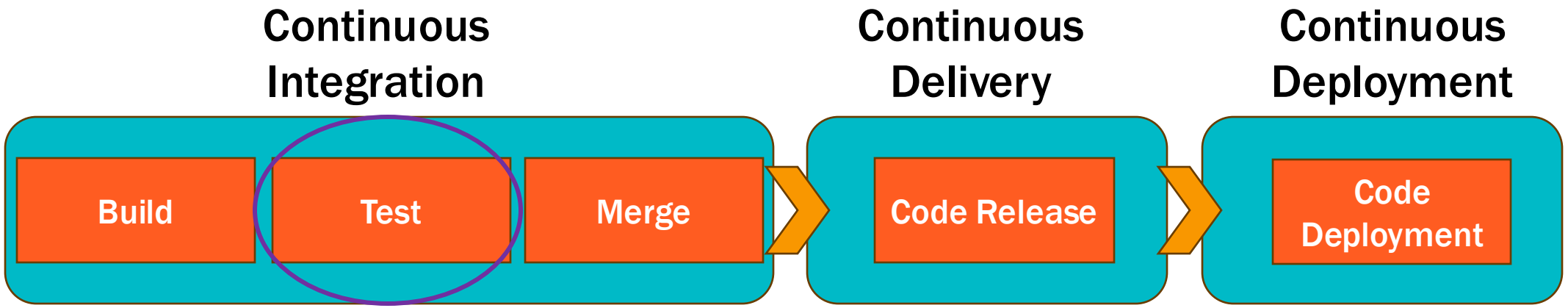
A TYPICAL DEVOPS PIPELINE



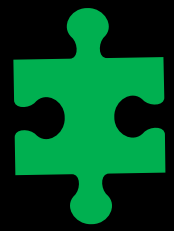
A TYPICAL DEVOPS PIPELINE



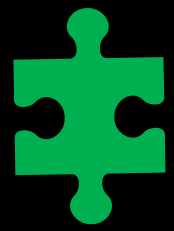
A TYPICAL DEVOPS PIPELINE



UNIT TESTING

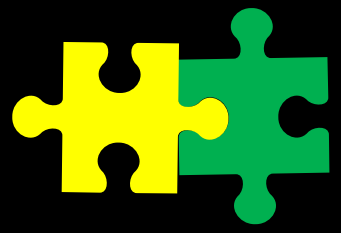


UNIT TESTING

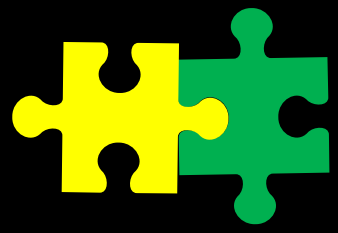


- Tests and Validates each individual component
- Primary focus is functional correctness
  - Component behaves as intended
  - Component handles different situations correctly
  - Checks correct handling of inputs, outputs, and errors
- For components with dependencies
  - Use mocking framework to mock the behavior of the dependency

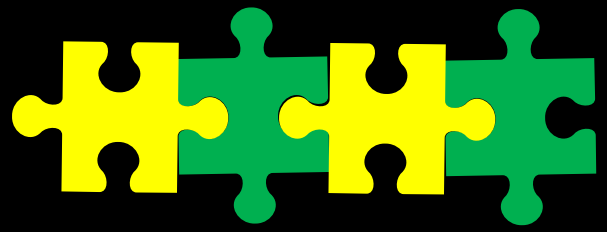
INTEGRATION TESTING



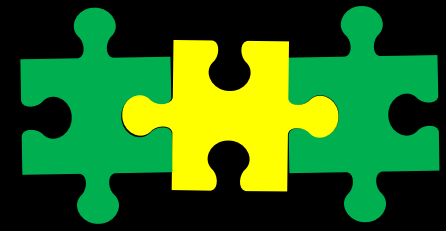
INTEGRATION TESTING

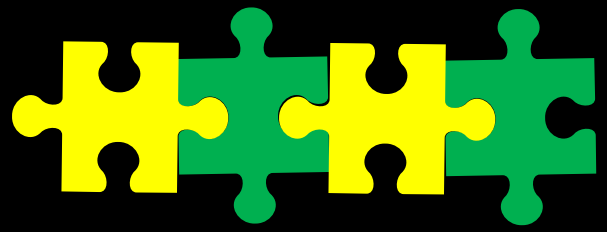


- Tests the behavior of interacting components
- Correctness of the interaction between two components
- Typically done for services/modules that have dependencies
  - Instead of using mock instances for dependencies, use actual instances

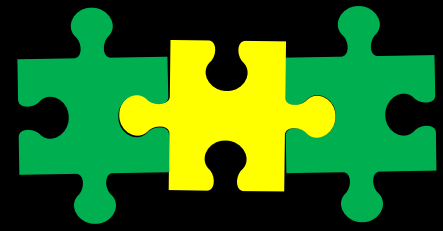


END-TO-END TESTING





END-TO-END TESTING



- Checks whether the application works as intended in whole
  - Make requests to the front-end services
  - Validate the outputs of the requests
- White-Box Testing
  - Test internal state at every stage/step
- Black-Box Testing
  - Testing without any knowledge of the internal code or architecture
  - Not testing or checking the internal state of the system
- Grey Testing
  - BlackBox Testing but with partial knowledge
  - Useful for crafting tests for edge-cases

FORMAL METHODS

Formal methods make use of mathematical techniques for ensuring that the systems are correct

The strongest techniques provide proofs of correctness for a given system specification

Main drawback of some of these techniques is that they require a lot of developer effort

LIGHTWEIGHT FORMAL METHODS

- Fuzz Testing
- Deterministic Simulation
- Property Testing

FUZZ TESTING

- Generate test inputs that can expose bugs in programs
- Depending on the fuzzing technique used, test inputs can be generated in multiple different ways
- Random Fuzzing
  - Generate random inputs and see how the system handles these inputs
  - Also called black-box fuzzing as it doesn't use any feedback from the test executions

PROPERTY TESTING

- Developers specify properties that they want the system to hold

- With property testing, the goal is to ensure there are no violations of these properties during the system execution

TYPES OF PROPERTIES

- Safety Properties: System does not enter any bad state

- Liveness Properties: System eventually does something good

The above properties do not necessarily capture all potential behaviors that a system could exhibit (or properties we would want from the system)

DETERMINISTIC SIMULATION

- Execute a distributed system in a single-threaded simulator
- Control all sources of randomness
- Developers can specify orderings of threads and events they want to explore

HEAVYWEIGHT FORMAL METHODS

- Model Checking
- Symbolic Execution
- Formal Verification

SYMBOLIC EXECUTION

- Executes a program with symbolic inputs
- Builds all pathways the execution could take with respect to the symbolic input
- Combine with fuzzing to generate more interesting inputs that exercise different paths

MODEL CHECKING

- Build an abstract model of the system
- Developers provide safety and liveness properties
- Model checker explores all possible pathways and states defined by the model to find violations of the properties

FORMAL VERIFICATION

- Uses a theorem prover or a constraint solver
- Encode the system behavior in a mathematical form
- Theorem prover/Constraint solver verifies that certain properties hold for the system behavior described in the mathematical form

RELIABILITY TESTING

We are testing that the system will continue to behave normally in the presence of errors and faults

The reliability techniques will behave correctly

FAULT INJECTION

FAULT INJECTION

- Inject different type of faults into the system
- Ensure that the system can handle the faults correctly
- Where/When to inject faults?
  - Fuzzing based fault injection during CI/CD testing
  - E2E tests augmented with Fault Injection

FAULT INJECTION

- Inject different type of faults into the system
- Ensure that the system can handle the faults correctly
- Where/When to inject faults?
  - Fuzzing based fault injection during CI/CD testing
  - E2E tests augmented with Fault Injection
  - IN PRODUCTION!

CHAOS ENGINEERING

CHAOS ENGINEERING

- Started by Netflix (~10 yrs ago)
- System's capability to withstand realistic turbulent conditions
- Doing reliability experiments in production
  - Randomly fail a server in a controlled way
  - See whether the reliability techniques kick in
- Divide into a control group and an experimental group
  - Define some steady state behaviour that is hypothesised to continue
  - Try to find deviations between the behaviour in the 2 groups

Source: https://principlesofchaos.org/

DISCUSSION THEMES

- If you are building a system, which testing techniques are useful?
- If a model checker, for a given model of a system, does not find any property violations, does this mean that there are no property violations in the system implementation?
- How can developers combine formal methods with actual runtime behaviour of implementations?

GOOGLE & GOOGLE CLOUD OUTAGE

THE INCIDENT

⊗ 12 Jun 2025	11:46 PDT	We will provide an update by Thursday, 2025-06-12 12:15 PDT with current details. We apologize to all who are affected by the disruption. **Symptoms:** Multiple GCP products are experiencing varying level of service impacts. **Workaround:** None at this time.

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

THE INCIDENT

12 Jun 2025	11:46 PDT	We will provide an update by Thursday, 2025-06-12 12:15 PDT with current details. We apologize to all who are affected by the disruption. **Symptoms:** Multiple GCP products are experiencing varying level of service impacts. **Workaround:** None at this time.
12 Jun 2025	18:27 PDT	Vertex AI Online Prediction is full recovered as of 18:18 PDT. All the services are fully recovered from the service issue We will publish analysis of this incident once we have completed our internal investigation. We thank you for your patience while we worked on resolving the issue.

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

THE INCIDENT

12 Jun 2025	11:46 PDT	We will provide an update by Thursday, 2025-06-12 12:15 PDT with current details. We apologize to all who are affected by the disruption. **Symptoms:** Multiple GCP products are experiencing varying level of service impacts.

Summary

Google Cloud, Google Workspace and Google Security Operations products experienced increased 503 errors in external API requests, impacting customers.

We deeply apologize for the impact this outage has had. Google Cloud customers and their users trust their businesses to Google, and we will do better. We apologize for the impact this has had not only on our customers' businesses and their users but also on the trust of our systems. We are committed to making improvements to help avoid outages like this moving forward.

12 Jun 2025	18:27 PDT	Vertex AI Online Prediction is full recovered as of 18:18 PDT. All the services are fully recovered from the service issue We will publish analysis of this incident once we have completed our internal investigation. We thank you for your patience while we worked on resolving the issue.

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

AFFECTED SERVICES

Google Cloud Products:

- Identity and Access Management
- Cloud Build
- Cloud Key Management Service
- Google Cloud Storage
- Cloud Monitoring
- Google Cloud Dataproc
- Cloud Security Command Center
- Artifact Registry
- Cloud Workflows
- Cloud Healthcare
- Resource Manager API
- Dataproc Metastore
- Cloud Run
- VMWare engine
- Dataplex
- Migrate to Virtual Machines
- Google BigQuery
- Contact Center AI Platform
- Google Cloud Deploy
- Media CDN
- Colab Enterprise
- Vertex Gemini API
- Cloud Data Fusion
- Cloud Asset Inventory
- Datastream
- Integration Connectors
- Apigee
- Google Cloud NetApp Volumes
- Google Cloud Bigtable
- Looker (Google Cloud core)
- Looker Studio
- Google Cloud Functions
- Cloud Load Balancing
- Traffic Director
- Document AI
- AutoML Translation
- Pub/Sub Lite
- API Gateway
- Agent Assist
- AlloyDB for PostgreSQL
- Cloud Firestore
- Cloud Logging
- Cloud Shell
- Cloud Memorystore
- Cloud Spanner
- Contact Center Insights
- Database Migration Service
- Dialogflow CX
- Dialogflow ES
- Google App Engine
- Google Cloud Composer
- Google Cloud Console
- Google Cloud DNS
- Google Cloud Pub/Sub
- Google Cloud SQL
- Google Compute Engine
- Identity Platform
- Managed Service for Apache Kafka
- Memorystore for Memcached
- Memorystore for Redis
- Memorystore for Redis Cluster
- Persistent Disk
- Personalized Service Health
- Speech-to-Text
- Text-to-Speech
- Vertex AI Search
- Retail API
- Vertex AI Feature Store
- BigQuery Data Transfer Service
- Google Cloud Marketplace
- Cloud NAT
- Hybrid Connectivity
- Cloud Vision
- Network Connectivity Center
- Cloud Workstations

Google Workspace Products:

- AppSheet
- Gmail
- Google Calendar
- Google Drive
- Google Chat
- Google Voice
- Google Docs
- Google Meet
- Google Cloud Search
- Google Tasks

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

KEY INCIDENT TIMINGS

(All Times US/Pacific)

Incident Start: 12 June, 2025 10:49

All regions except us-central1 mitigated: 12 June, 2025 12:48

Incident End: 12 June, 2025 13:49

Duration: 3 hours

Regions/Zones: Global

How did we communicate?

We posted our first incident report to Cloud Service Health about ~1h after the start of the crashes, due to the Cloud Service Health infrastructure being down due to this outage. For some customers, the monitoring infrastructure they had running on Google Cloud was also failing, leaving them without a signal of the incident or an understanding of the impact to their business and/or infrastructure. We will address this going forward.

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

KEY INCIDENT TIMINGS

(All Times US/Pacific)

Incident Start: 12 June, 2025 10:49

Full Recovery took 7 hours!

✓ 12 Jun 2025	18:27 PDT	Vertex AI Online Prediction is full recovered as of 18:18 PDT. All the services are fully recovered from the service issue We will publish analysis of this incident once we have completed our internal investigation. We thank you for your patience while we worked on resolving the issue.

How did we communicate?

We posted our first incident report to Cloud Service Health about ~1h after the start of the crashes, due to the Cloud Service Health infrastructure being down due to this outage. For some customers, the monitoring infrastructure they had running on Google Cloud was also failing, leaving them without a signal of the incident or an understanding of the impact to their business and/or infrastructure. We will address this going forward.

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

KEY INCIDENT TIMINGS

Within 2 minutes, our Site Reliability Engineering team was triaging the incident. Within 10 minutes, the root cause was identified and the red-button (to disable the serving path) was being put in place. The red-button was ready to roll out ~25 minutes from the start of the incident. Within 40 minutes of the incident, the red-button rollout was completed, and we started seeing recovery across regions, starting with the smaller ones first.

- Time to Detect: 2 minutes
- Time to find Root Cause: 10 minutes
- Time to Mitigation: 25 minutes-40 minutes
- Time to full recovery: 7 hours

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

ROOT CAUSE

On May 29, 2025, a new feature was added to Service Control for additional quota policy checks. This code change and binary release went through our region by region rollout, but the code path that failed was never exercised during this rollout due to needing a policy change that would trigger the code. As a safety precaution, this code change came with a red-button to turn off that particular policy serving path. The issue with this change was that it did not have appropriate error handling nor was it feature flag protected. Without the appropriate error handling, the null pointer caused the binary to crash. Feature flags are used to gradually enable the feature region by region per project, starting with internal projects, to enable us to catch issues. If this had been flag protected, the issue would have been caught in staging.

On June 12, 2025 at ~10:45am PDT, a policy change was inserted into the regional Spanner tables that Service Control uses for policies. Given the global nature of quota management, this metadata was replicated globally within seconds. This policy data contained unintended blank fields. Service Control, then regionally exercised quota checks on policies in each regional datastore. This pulled in blank fields for this respective policy change and exercised the code path that hit the null pointer causing the binaries to go into a crash loop. This occurred globally given each regional deployment.

What kind of failure is this?

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

ROOT CAUSE

On May 29, 2025, a new feature was added to Service Control for additional quota policy checks. This code change and binary release went through our region by region rollout, but the code path that failed was never exercised during this rollout due to needing a policy change that would trigger the code. As a safety precaution, this code change came with a red-button to turn off that particular policy serving path. The issue with this change was that it did not have appropriate error handling nor was it feature flag protected. Without the appropriate error handling, the null pointer caused the binary to crash. Feature flags are used to gradually enable the feature region by region per project, starting with internal projects, to enable us to catch issues. If this had been flag protected, the issue would have been caught in staging.

On June 12, 2025 at ~10:45am PDT, a policy change was inserted into the regional Spanner tables that Service Control uses for policies. Given the global nature of quota management, this metadata was replicated globally within seconds. This policy data contained unintended blank fields. Service Control, then regionally exercised quota checks on policies in each regional datastore. This pulled in blank fields for this respective policy change and exercised the code path that hit the null pointer causing the binaries to go into a crash loop. This occurred globally given each regional deployment.

Cross-System Interaction Failure!!!

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

ROOT CAUSE

On May 29, 2025, a new feature was added to Service Control for additional quota policy checks. This code change and binary rel
by region rollout, but the code path that failed was never exercised during this rollout due to needing a policy change that would
precaution, this code change came with a red-button to turn off that particular policy serving path. The issue with this change wa
appropriate error handling nor was it feature flag protected. Without the appropriate error handling, the null pointer caused the
are used to gradually enable the feature region by region per project, starting with internal projects, to enable us to catch issues.
the issue would have been caught in staging.

On June 12, 2025 at ~10:45am PDT, a policy change was inserted into the regional Spanner tables that Service Control uses for po
of quota management, this metadata was replicated globally within seconds. This policy data contained unintended blank fields.
exercised quota checks on policies in each regional datastore. This pulled in blank fields for this respective policy change and exe
null pointer causing the binaries to go into a crash loop. This occurred globally given each regional deployment.

Cross-System Interaction Failure!!!

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

ROOT CAUSE

Within some of our larger regions, such as us-central-1, as Service Control tasks restarted, it created a herd effect on the underlying infrastructure it depends on (i.e. that Spanner table), overloading the infrastructure. Service Control did not have the appropriate randomized exponential backoff implemented to avoid this. It took up to ~2h 40 mins to fully resolve in us-central-1 as we throttled task creation to minimize the impact on the underlying infrastructure and routed traffic to multi-regional databases to reduce the load. At that point, Service Control and API serving was fully recovered across all regions. Corresponding Google and Google Cloud products started recovering with some taking longer depending upon their architecture.

What kind of failure is this?

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

ROOT CAUSE

Within some of our larger regions, such as us-central-1, as Service Control tasks restarted, it created a herd effect on the underlying infrastructure it depends on (i.e. that Spanner table), overloading the infrastructure. Service Control did not have the appropriate randomized exponential backoff implemented to avoid this. It took up to ~2h 40 mins to fully resolve in us-central-1 as we throttled task creation to minimize the impact on the underlying infrastructure and routed traffic to multi-regional databases to reduce the load. At that point, Service Control and API serving was fully recovered across all regions. Corresponding Google and Google Cloud products started recovering with some taking longer depending upon their architecture.

Metastability Failure!

Source: https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW

DISCUSSION THEMES

- If you are building a system, which testing techniques are useful?
- If a model checker, for a given model of a system, does not find any property violations, does this mean that there are no property violations in the system implementation?
- How can developers combine formal methods with actual runtime behaviour of implementations?