How to Fight Production Incidents? An Empirical Study on a Large-scale Cloud Service

---

By Supriyo Ghosh, Manish Shetty, Chetan Bansal, Suman Nath

Presented by Andrey Satarin (@asatarin)
January, 2023

https://asatarin.github.io/talks/2023-01-how-to-fight-incidents/

Outline

- Methodology
- Root causes and mitigation
- What causes delays in response?
- Lessons learnt
- Multi-dimensional analysis
- Conclusions

Methodology

Incidents to study

- 152 incidents from Microsoft Teams
- Analyze root causes, detection and mitigation approaches
- Only incidents with complete postmortem report
- High severity only: 1 incident SEV0, ~30% SEV1, ~70% SEV2

Factors to study

- Root Cause — What issue caused the incident?
- Mitigation Steps — What steps were performed to restore service health?
- Detection Failure — Why did monitoring not detect the incident?
- Mitigation Failure — What challenges delayed incident mitigation?
- Automation Opportunities — What automation can help improve service resilience?
- Lessons for Resiliency — What lessons were learnt about the service’s behavior and improving resiliency?

Threat to validity

- Microsoft already uses some effective tools and techniques to proactively mitigate many types of incidents
- About 35% of incidents were filtered out because did not have complete postmortem
- Microsoft-Teams only incidents

Root causes and mitigation

Root causes

- **Code Bug — 27.0 %**
- Dependency Failure — 16.4 %
- Infrastructure — 15.8 %
- Deployment Error — 13.2 %
- **Config Bug — 12.5 %**
- Database/Network — 10.5 %
- Auth Failure — 4.6 %

Finding #1

- While 40% incidents were root caused to code or configuration bugs, a majority (60%) were caused due to non-code related issues in infrastructure, deployment, and service dependencies.

- 40 % = Code Bug (27.0 %) + Config Bug (12.5 %)

Mitigation steps

- Rollback - 22.4 %
- Infra Change - 21.1 %
- External Fix - 15.8 %
- **Config Fix - 13.2 %**
- Ad-hoc Fix - 11.8 %
- **Code Fix - 7.9 %**
- Transient - 7.9 %

Finding #2

- Although 40% incidents were caused by code/configuration bugs, nearly **80% of incidents were mitigated without a code or configuration fix.**

- 80 % = 100 % - Config Fix (13.2 %) - Code Fix (7.9 %)

Finding #3

- Mitigation via roll back, infrastructure scaling, and traffic failover account for more than 40% of incidents, indicating their popularity for quick mitigation.

- 40 % = Rollback (22.4 %) + Infra Change (21.1 %)

What causes delays in response?

Finding #5

- The time-to-detect code bugs and dependency failures is significantly higher than other root causes, indicating inherent difficulties in monitoring such incidents.

Finding #6

- Manually fixing code and configuration take significantly higher time-to-mitigate, when compared to rolling back changes. This supports the popularity of the latter method for mitigation.

Detection failure

- **Not Failed — 52.0 %**
- Unclear — 11.8 %
- Monitor Bug — 10.5 %
- **No Monitors — 8.6 %**
- **Telemetry Coverage — 8.6 %**
- Cannot Detect — 4.6 %
- External Effect — 4.0 %

Finding #7

- 17 % of incidents either lacked monitors or telemetry coverage, both of which result in significant detection delays.

- 17 % = No Monitors (8.6 %) + Telemetry Coverage (8.6 %)

Mitigation failure category

- Not Failed — 27.6 %
- Unclear — 27.6 %
- Documents-Procedures — 10.5 %
- Deployment Delay — 10.5 %
- Manual Effort — 9.2 %
- Complex Root Cause — 7.2 %
- External Dependency — 7.2 %

Finding #8

- While complex root causes can affect time-to-mitigate, 30% of incidents had mitigation delays even after identifying the root cause due to poor documentation, procedures, and manual deployment steps.

Lessons learnt

Automation opportunities

- Unclear — 32.2 %
- **Manual Test — 25.7 %**
- None — 15.1 %
- Auto Alert/Triage — 15.1 %
- **Config Test — 5.9 %**
- Auto Deployment — 5.9 %

Finding #9

- **Improving testing was a popular choice** for automation opportunities, over monitoring, indicating a need to reduce incidents by identifying issues before they reach production services.

Lesson learnt category

- Unclear — 37.5 %
- Improve Monitoring — 15.8 %
- Behavioral Change — 11.8 %
- External Coordination — 10.5 %
- Improve Testing — 9.9 %
- Documents/Training — 7.9 %
- Auto Mitigation — 6.6 %

Finding #10

- While improving monitoring/testing accounts for majority of the lessons learnt, **a significant ≈20% feedback indicated improved documentation, training**, and practices for better incident management and service resiliency.

- 20 % = Behavioral Change (11.8 %) + Documents/Training (7.9 %)

Multi-dimensional analysis

Finding #11

- 70% of incidents with no monitors were root caused to code bugs, i.e., it is inherently difficult to monitor regressions introduced due to code changes.

- => For code changes, **we should improve testing** rather than relying on monitoring.

Finding #12

- 42% of incidents that cannot be detected by monitoring today, were associated with dependency failures

- => There is a **need to introduce/increase monitoring coverage** and observability across related services.

Finding #13

- 47% of configuration bugs mitigated with a rollback compared to a lesser 21% mitigated with a configuration fix; i.e., A large portion of misconfigurations are due to recent changes

- => They can be identified by rigorous configuration testing.

Finding #14

- 21% of incidents where manual effort delayed mitigation, expected improvements in documentation and training.

- => Just like with source code, we need to design new metrics and methods to monitor documentation quality. Also, automating repeating mitigation tasks can reduce manual effort and on-call fatigue.

Finding #15

- 25% of incidents where **mitigation delay was due to manual deployment** steps, expected automated mitigation steps to manage service infrastructure (like traffic-failover, node reboot, and auto-scaling).

Conclusions

Conclusions

- 152 incident reports studied
- Identified potential automation opportunities
- Multi-dimensional analysis uncovers important insights for improving reliability

Microsoft 365 Status

@MSFT365Status

We've rolled back a network change that we believe is causing impact. We're monitoring the service as the rollback takes effect.

1:26 AM · Jan 25, 2023 · **224.5K** Views

346 Retweets **115** Quote Tweets **930** Likes

https://twitter.com/MSFT365Status/status/1618178407316987905

Today’s outage

> We've **rolled back** a network change
Mitigation strategy — Rollback (22.4 %)

> We've rolled back **a network change**
Root cause — Database/Network (10.5 %)

> We’re monitoring the service as the rollback takes effect

References

References

- Self reference for this talk (slides, video, etc) https://asatarin.github.io/talks/2023-01-how-to-fight-incidents/

- “How to fight production incidents?: an empirical study on a large-scale cloud service” paper https://dl.acm.org/doi/10.1145/3542929.3563482

Contacts

- Follow me on Twitter @asatarin
- Follow me on Mastodon https://discuss.systems/@asatarin
- Profession profile https://www.linkedin.com/in/asatarin/
- Other public talks https://asatarin.github.io/talks/