SRE: Tick Tock - Incident

Something happened, something is happening or something will happen.

Incidents are the most noticeable part of SRE, when the team does it right nobody notices and when they do it badly everyone sees (how do you appreciate what did not happen?).

First, what qualifies as an incident? An incident is any situation that requires a coordinated response to prevent damage (or further damage) to the user experience (customer, developers, engineers = brand/reputation) and/or monetary losses (operational costs, revenue lost).

The monetary impact has a corollary relationship with the amount of time it takes to resolve the incident. The costs go up the longer the incident persists (cost per minute multiplied by the total incident minutes). Example: The app is down; an hour it generates 100K profit, the app is down for 10 hours...there is a loss of at least 1 million (although at some point the costs become exponential instead of linear... but as far as rough estimates go this is fine).

Therefore, an incident has two levers to control the impact: the time and the cost. The cost is controlled by the size – how many customers are affected and the depth – what features are affected (resulting in revenue loss, etc.). The time – is controlled by how long the full remediation takes (MTTR) from detection to response to resolution.

A team can also find themselves in the context of the three response states: reacting (it is happening or has happened), proacting (it is imminent), or preventing (it has not happened). There needs to be a balance between managing incidents and preventing them, if the team prevents more - they get to manage less. If the team is always managing incidents, they will never get out in front (so what do I have to do today to solve or avoid this problem tomorrow?).

What can be done?

***Time** – Speed up the mean time to resolution.*

1. Using AI, basic anomaly detection and predictive analytics to see an incident coming before the impact/cost timer begins. (PROACTIVE)
2. Rehearse. Practice with fire drills to ensure the team acts in a structured manner. (PROACTIVE)
3. Go risk hunting. Implement an SRE Red Team to find flaws before they become incidents. (PREVENTITIVE)

Cost- Shrink the amount of customers and features impacted.

1. Implement sharding, microservices, and modular applications. (PROACTIVE)
2. Use release methodologies such as canary to decrease impacts from changes. (PREVENTITIVE)
3. Create resilient architectures rooted in fault-tolerance, high availability, and disaster recovery to prevent cascading or terminal failures. (PROACTIVE)
4. Establish an SLO for your services that is reflective of the maximum financial loss that is tolerable and create the reflexive error budget.

You will not be able to stop every incident. In the worst possible scenario, you do a postmortem/RCA to ensure you do not make the same mistake twice or at least not in the same way.