#ASLI ENGINEERING

# GitHub Outage Downtime due to an Edge Case

SWIPE

BY

ARPIT BHAYANI

# Dissecting GitHub Outage

## Edge case and a poor SQL query

August 2021. GitHub experienced major outage that affected their core services

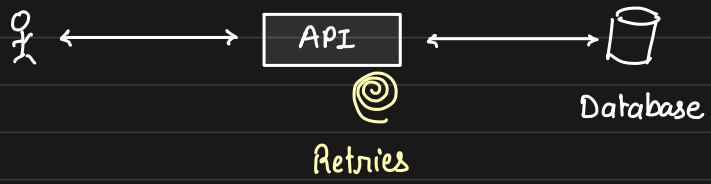
What happened?

The outage was due to an edge case that generated poor SQL query

↓

One of their MySQL masters degraded

+ automatic retry and queueing made it worse




We add retries in order to cover transient issues but it can also become something that adds to load during an outage

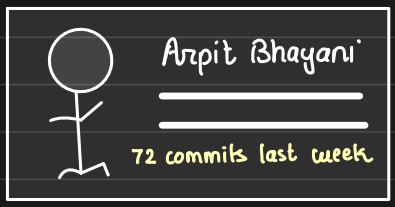
So, how could an "edge case" "degrade" the master?

Note: This is just an example, nothing to do with GitHub
Just trying to explain how an edge case can put huge load on DB

Say, we have commits table

commits

id, msg, author_id, created_at ← epoch ms stored as integer




```
SELECT count(id) FROM commits
  WHERE author_id = 123
  AND created_at > start_date
```

Say, the service is written in Golang

and we get start_date in the query parameter

↓

as epoch seconds

So, from the query parameter, we have to convert it to integer and then pass it to our sql query

https://api.github.com/stats/commits?start_date=<epoch_ts>

Golang based code to do this could look something like this

```
int start_date;

start_date, __ := strconv.Atoi(req.q.start_date)

res. = query.execute("
    SELECT count(id) FROM commits
     WHERE author_id = 123
     AND created_at > " + start_date)
```

Edge case: what if start_date is not passed
what if non-integer value is passed as start_date
↓
some misconfiguration on "calendar" widget

strconv.Atoi() → throws an error
but we did not handle it
So, 'start_date' being integer will get the
default value which is 0

the default value is set

```
int start_date;
```

↓

default value = 0

The query that gets fired is

```
SELECT count(id) FROM commits
WHERE author_id = 123
AND created_at > 0;
```

This would literally count all the commits made by the user ... ever!
and would take a long of time to execute
and put some load on DB

Given this is a common query, it will be invoked frequently
and will put massive load on DB!!
causing degradation and eventually an outage

So, how to protect ourselves from such situation?

So, how to avoid such situations

↗ Be pessimistic

1. Sanitize your input before query execution

2. Apply hard guard rails. even in edge case if goes through say 1000 rows

```
SELECT count(id) FROM commits
 WHERE author_id = 123
 AND created_at > 0
LIMIT 1000;
```