



**#ASLI ENGINEERING**

# Importance of localizing failures



**BY**

**ARPIT BHAYANI**

# Dissecting GitHub Outage

## Localized Failures

What happened?

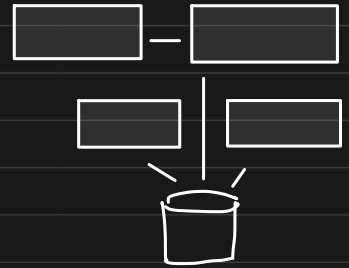
Failure or delay in queued jobs for GitHub Actions

↳ happened because of some infrastructure error in SQL layer

Database failure affected

Authentication and communication

between different microservices for GitHub Actions



What this tells us?

- GitHub actions has multiple microservices and they communicate through a shared DB  
eg: one would pick the job and update DB  
other would execute and update the DB

Shared database  
across multiple  
microservices !!

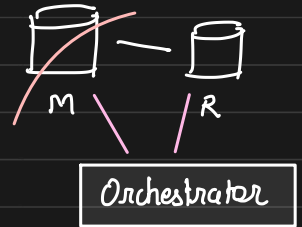
- Zero trust b/w microservices

The database/service also drove authentication ensuring unauthorized requests are not made even b/w services

But, if a database is down, shouldn't there be an automatic failover?

Yes. Usually the databases are configured with auto failover

ie. if master goes down, a replica is promoted



But this did not happen.

Because telemetry did not show that DB was down!

↳ Orchestrator also needs a source to find master is down and failover is needed

Hence it took a long time to determine root cause and mitigate

Once the root cause was identified, they would have

↳ done a manual failover

↳ rebooted the machine

\* Refer other Outage dissections to learn about possible ways and hacks of mitigation!

## long-term fixes

1. Change the automation and ensure it understands this failure

↳ Better failure detection

↳ Better automated failover

} Minimizing impact

2. localize failures

Microservices should be loosely coupled

Outage in one component/service cannot take down entire sub-system. Hence Github team would invest in ensuring

loose coupling, so that outage in one does not affect others



let them be loosely coupled,  
and the communication is  
ASYNCR or through  
tolerant APIs