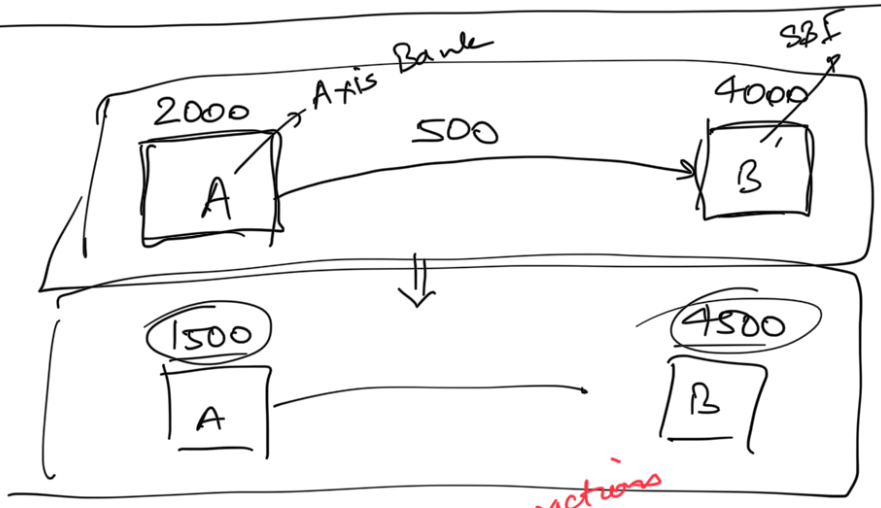
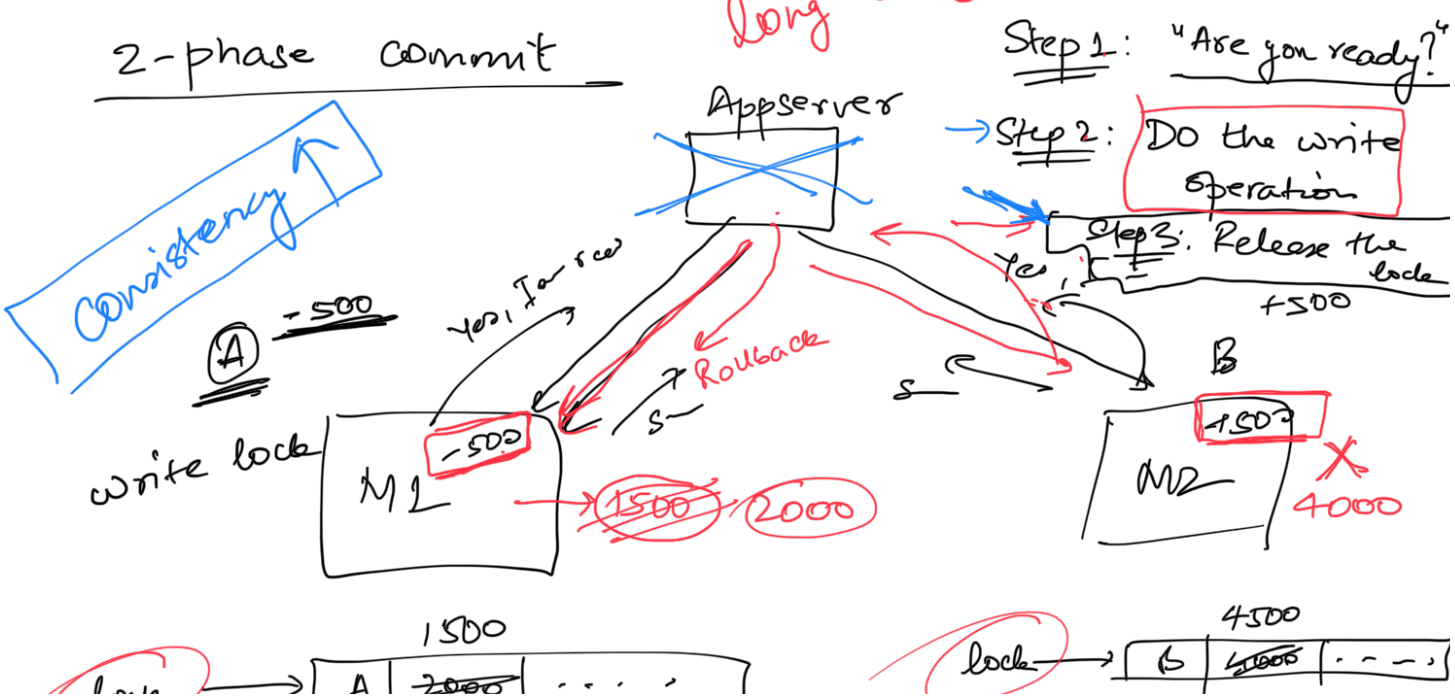
MICROSERVICES DESIGN PATTERN

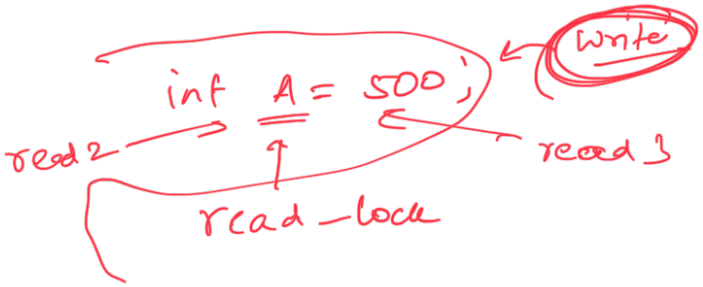
- Transactions
  - 2-phase commit (Anti-Pattern)
  - SAGA → Orchestration, Choreography.
  - CQRS
- Handling outages (Circuit Breaker)
- Observability (Tracing, Monitoring, Time Series DB)

Transactions



2-phase commit



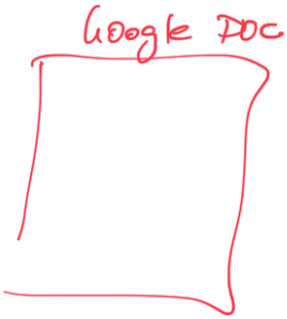


read lock → reads

write lock → read
↓ write
cannot

(A) → write lock

A = 500;

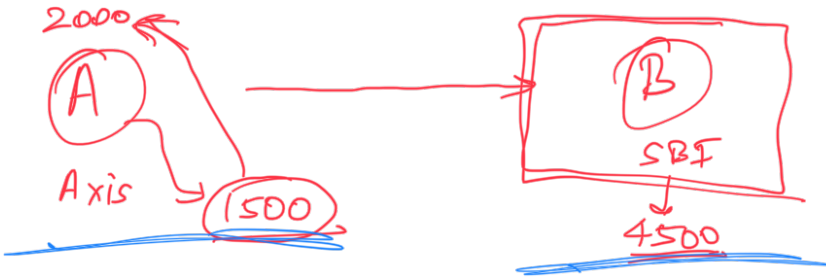


Anti-Pattern

① SLOW → long living transactions.
latency ↑

② Appserver dies → latency ↑

SAGA :



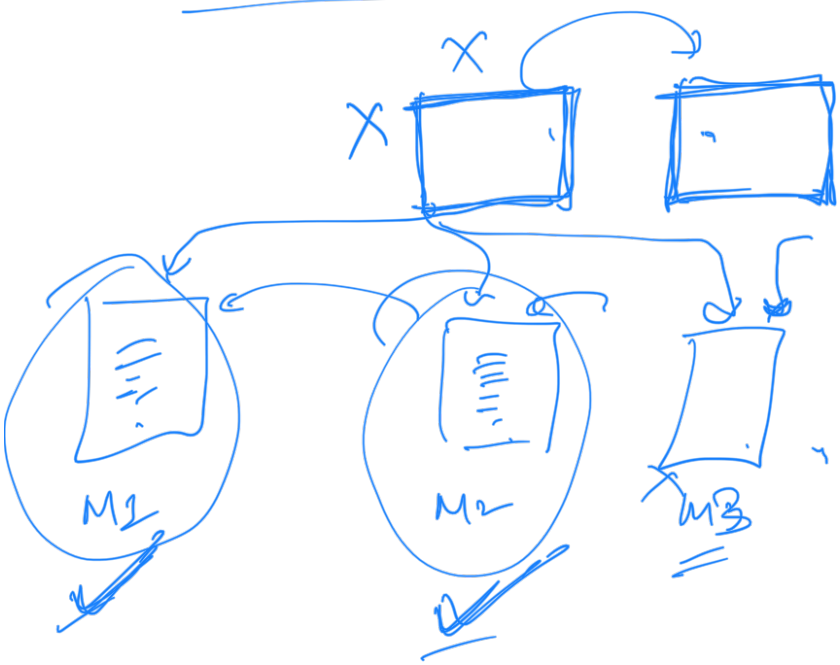
① Steps → 1 step → 1 microservice

② If failure, rollback guaranteed.

All steps should succeed

No steps should succeed

Orchestration



Choreography

M_1 M_2 M_3 M_4

M_X txn started data

M_X txn ended data

M_2 txn data

Choreography

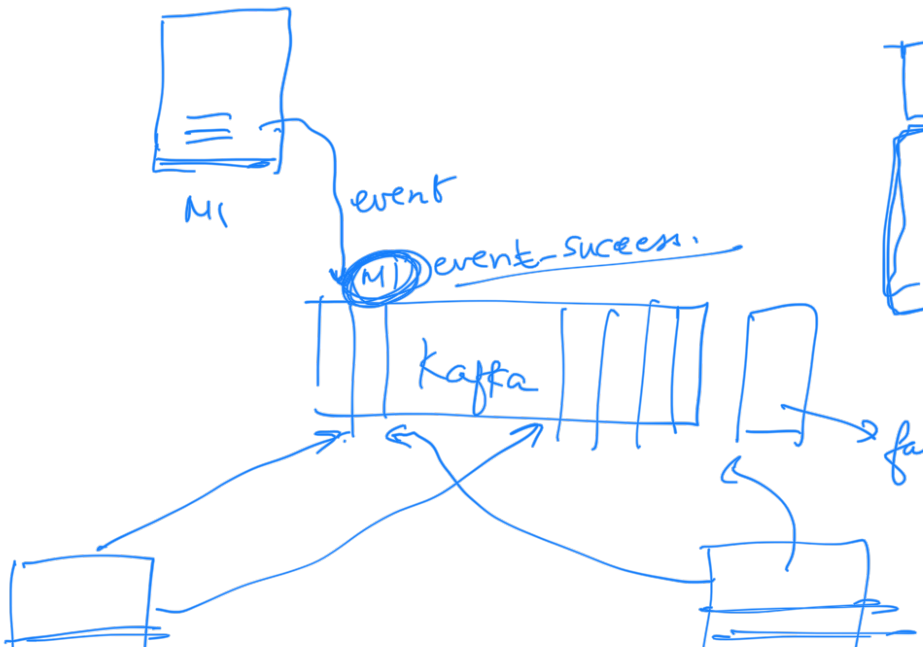
A → Hello → B

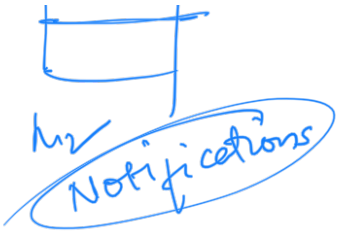
DB_A → Hello

DB_B → Hello

Send ~~sms~~ push notif to B

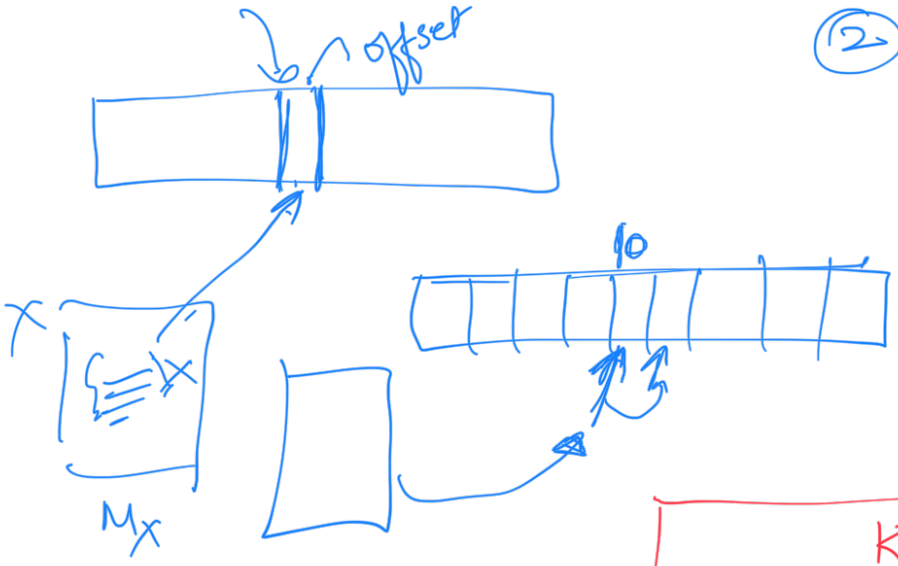
mark A as active online



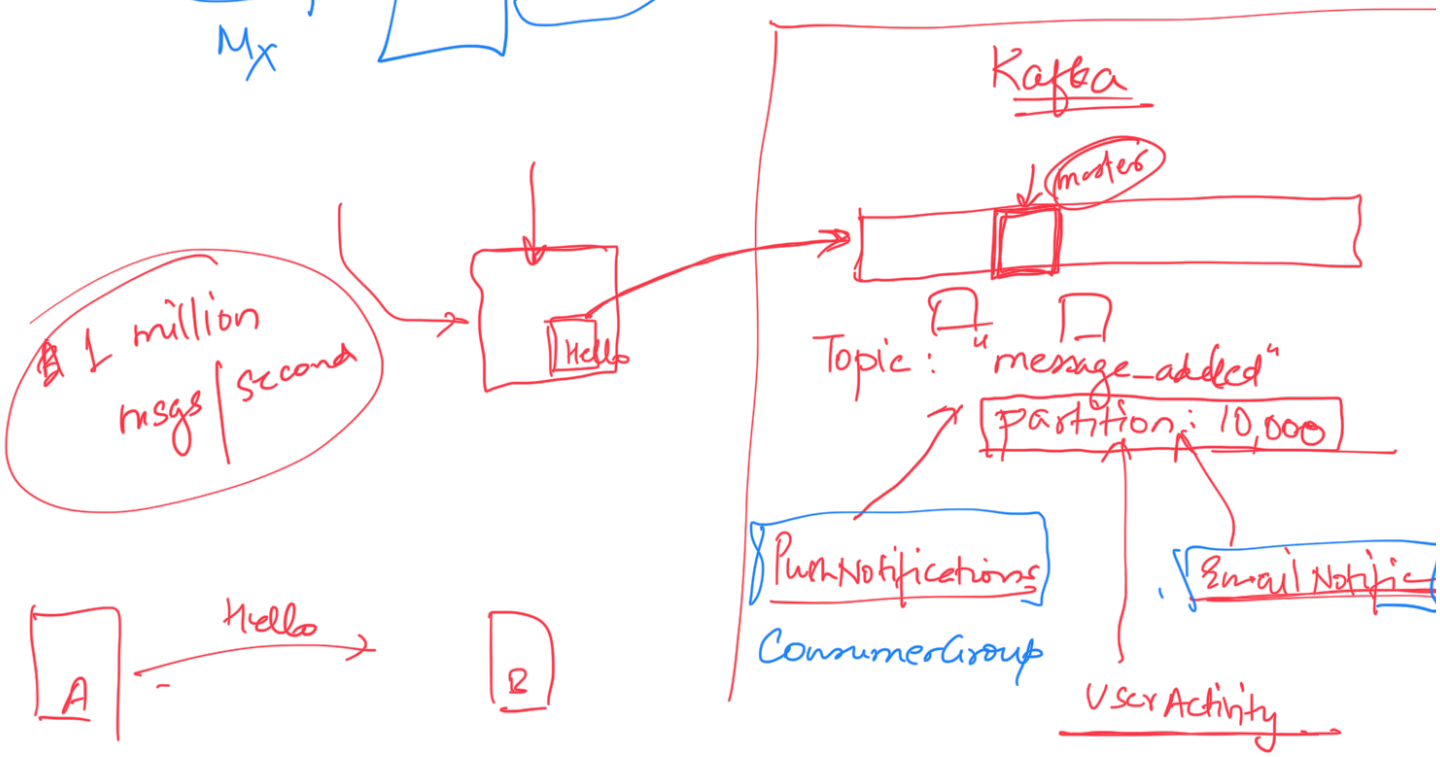


(1) Txns in parallel

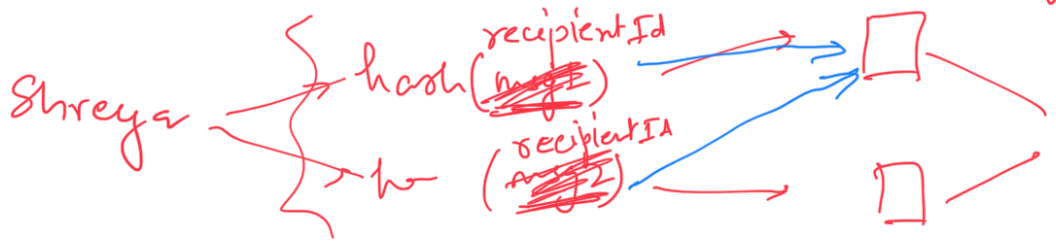
(2) Configure retries / rollback

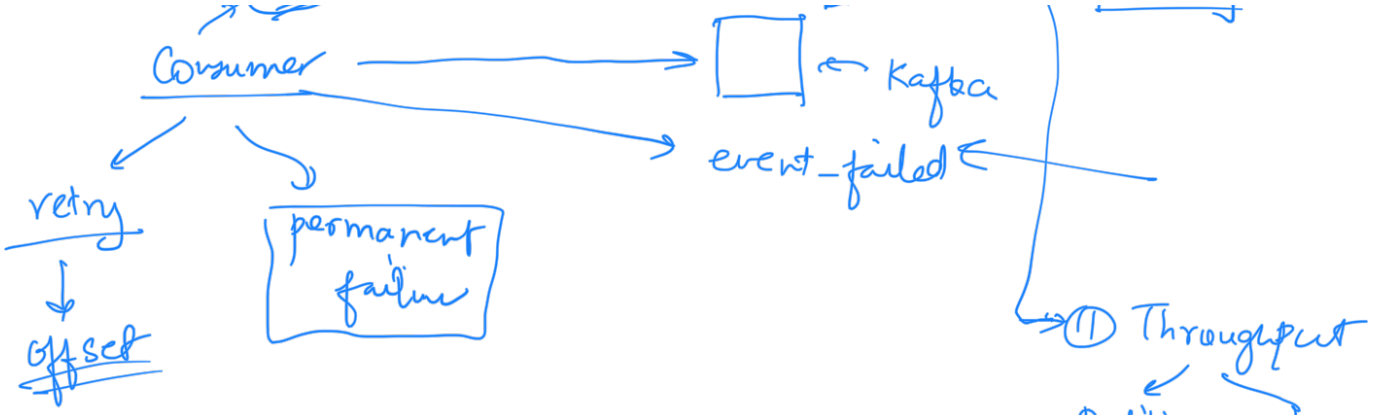


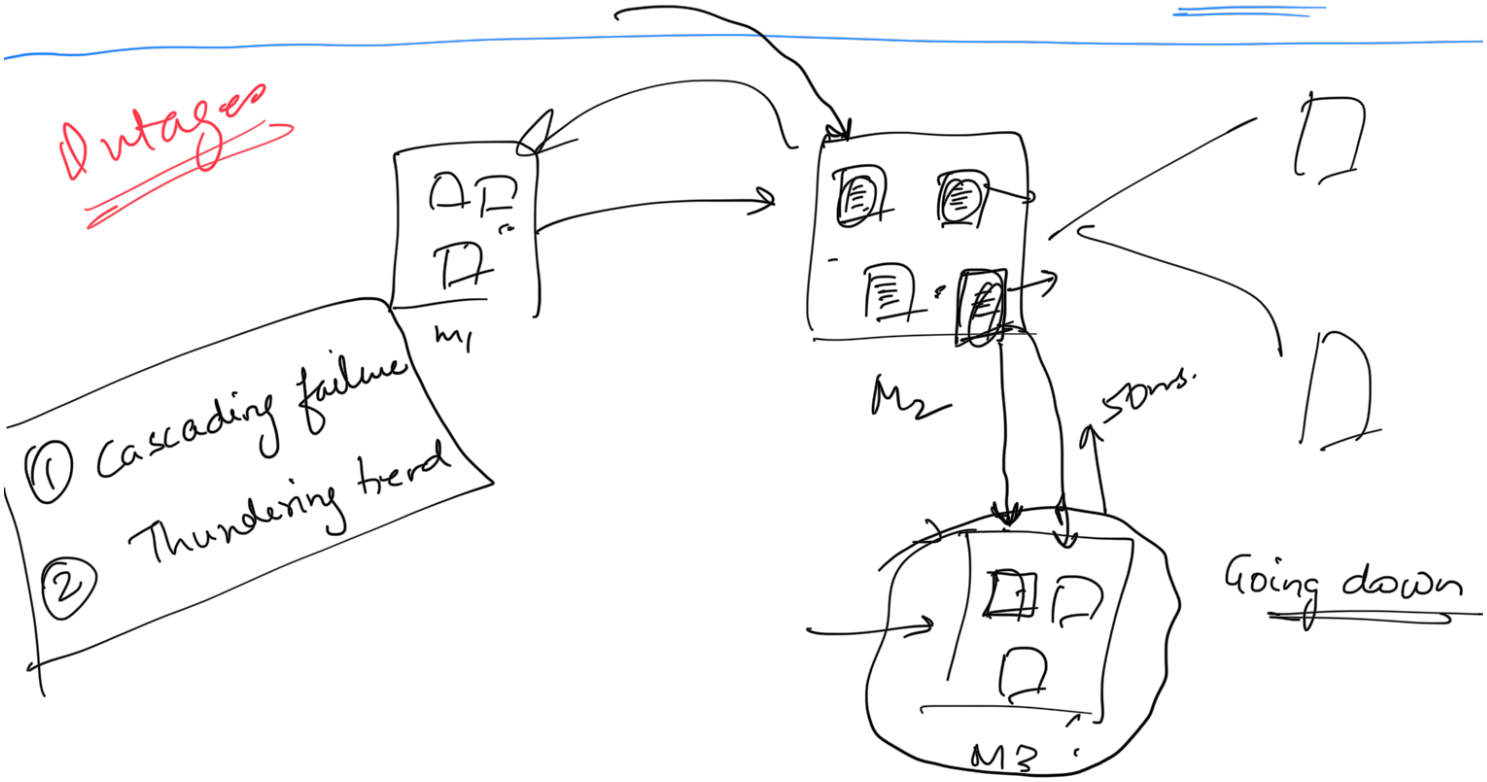
Event Driven Architecture

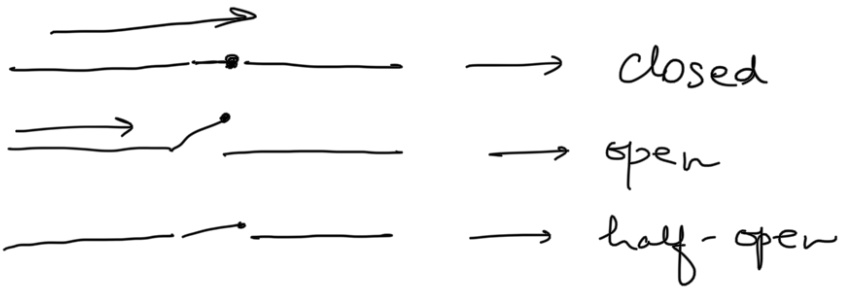


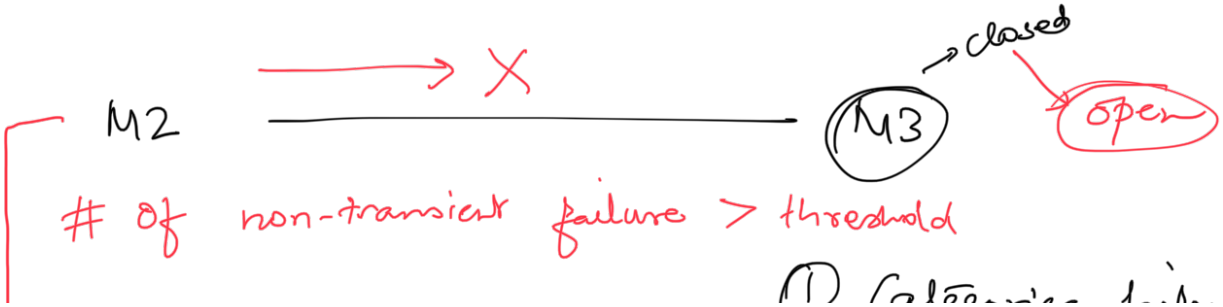
hash(sharding_key) % no._of_partitions → partition_key

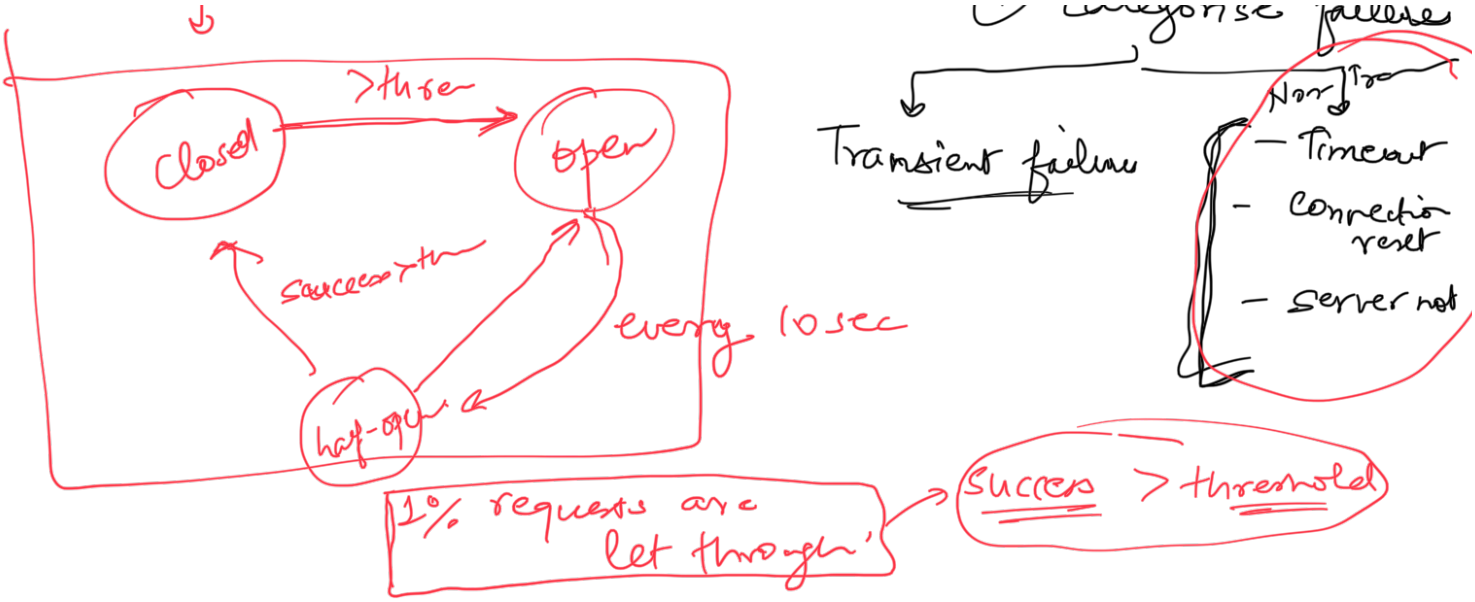



Consumer
Kafka
event_failed
retry
offset
permanent failure
① Throughput
partitions
② Multi-region
③ Retries

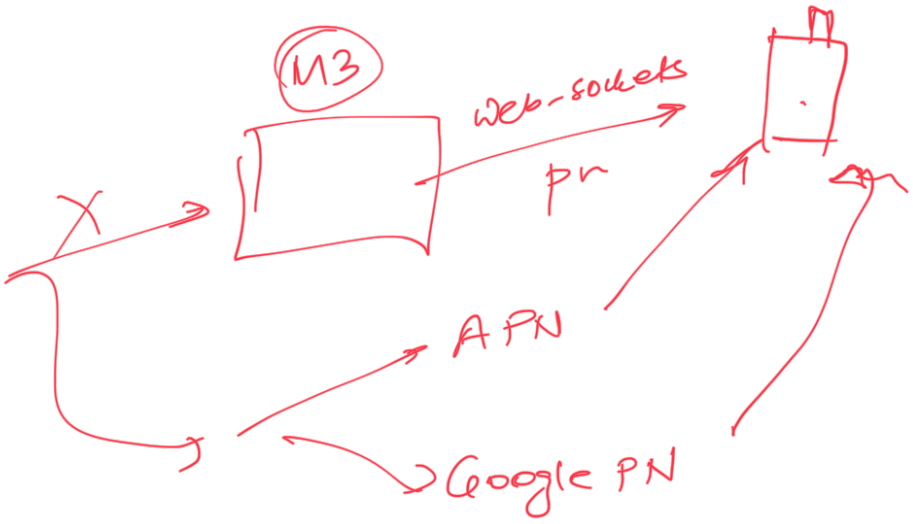

Outages
M1
M2
M3
50ms.
① Cascading failure
② Thundering herd
Going down


closed
open
half-open


M2
X
M3
closed
open
# of non-transient failure > threshold

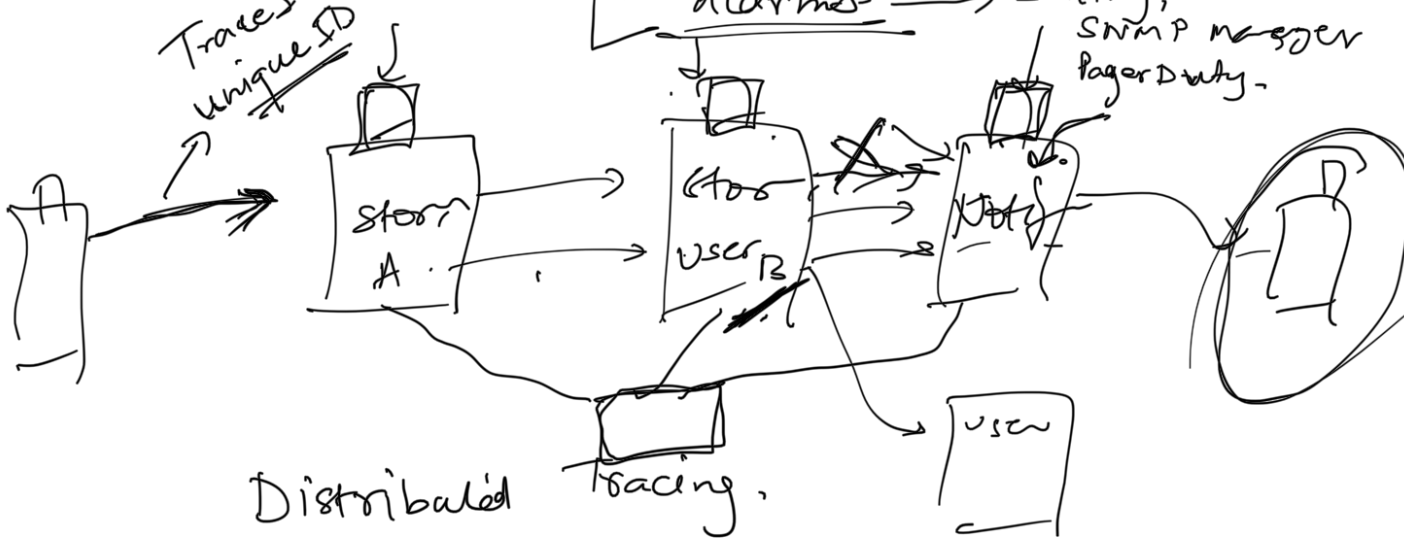



```
def circuit_breaker(API, data):
    state = redis.read("m3_state")
    if state == "closed":
        try:
            m3.API(....)
        catch(e):
            if e is non-transient:
                cnt = redis.read("m3-non-----")
                cnt++;
                if cnt > threshold:
                    redis.put("m3-state", "open").
                redis.put("---", cnt)
    else if state == "open":
        backup_function();
    else:
        if rand()% 100 == 0:
            m3.....
            success > threshold
            redis.put("state"...)
        else
            backup_function()
```



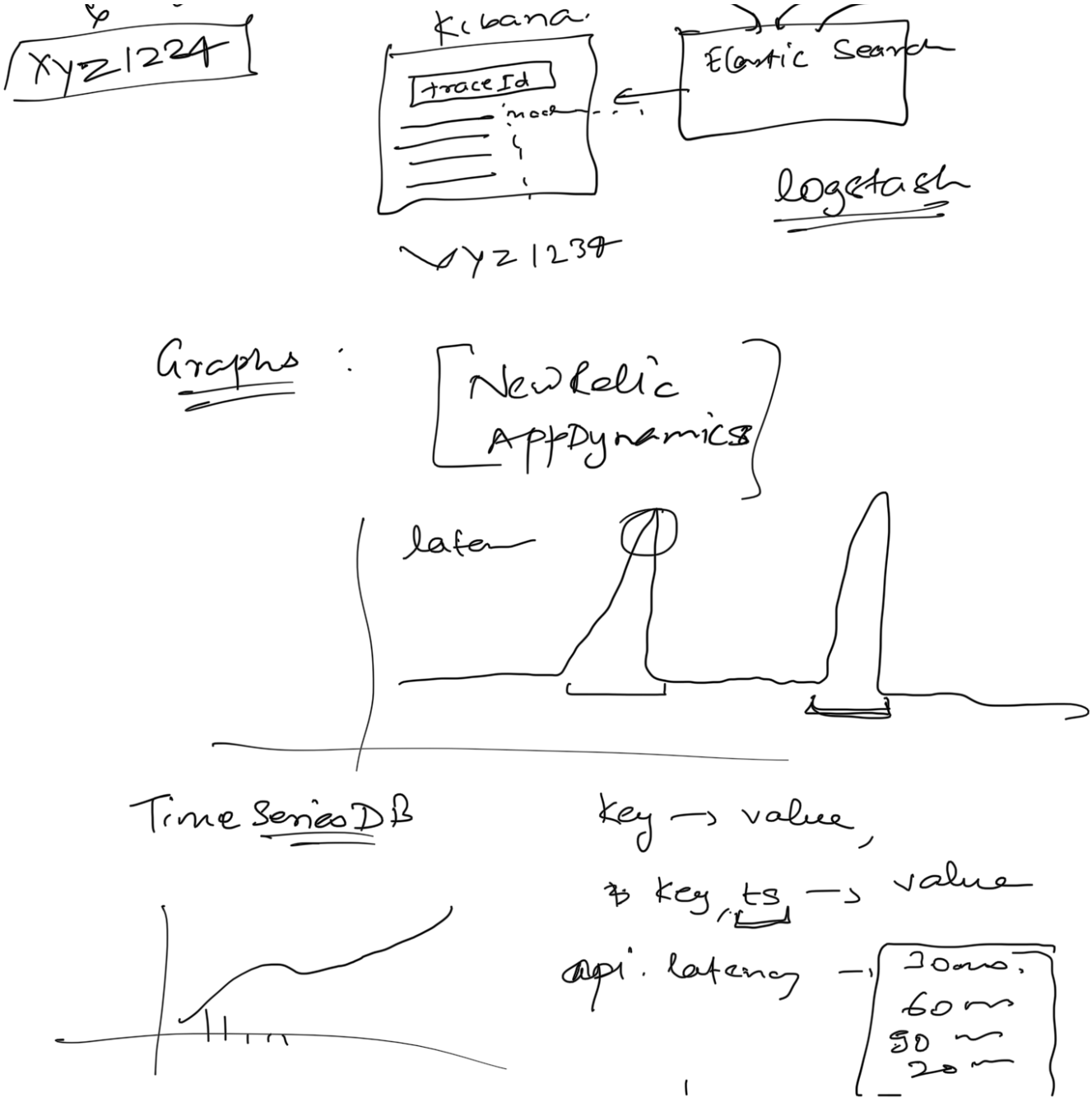
Observability :

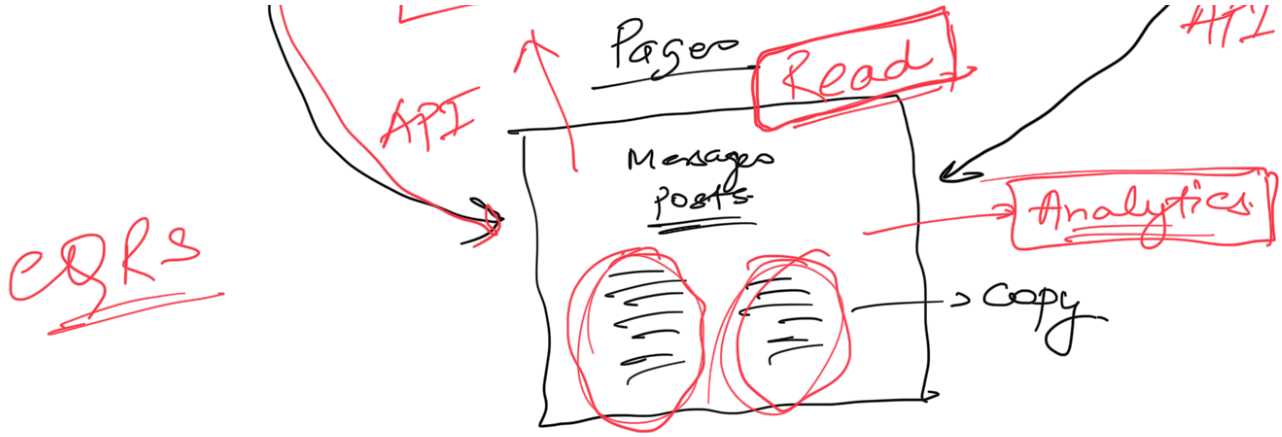
- Diagnosing failure
- Graphs to monitor health
- Alarms → Sentry, SNMP manager PagerDuty.

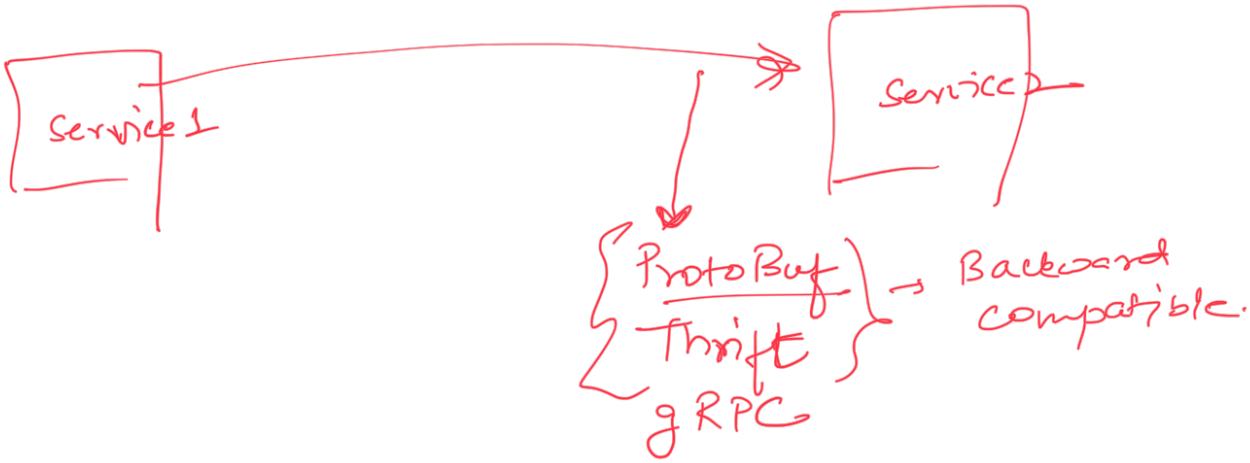


Step1 : TraceID → Going to all requests to microservices

Step2 : Splunk
ELK → ES, Logstash, Kibana


Xyz1224
Kibana
traceId
Elastic Search
logstash
Xyz1234
Graphs :
NewRelic
AppDynamics
latency
Time Series DB
key → value,
key, ts → value
api. latency →
10ms.
60 ms
90 ms
20 ms
CQRS (Command Query Responsibility Segregation)
Facebook
Pages
Messages
99.9% users-
0.1% pages
writes
Posts
Pages
Post.
Pages.


CQRS
API
Pages
Read
API
Message posts
Analytics
copy


Service 1
Service 2
ProtoBuf
Thrift
gRPC
Backward compatible.