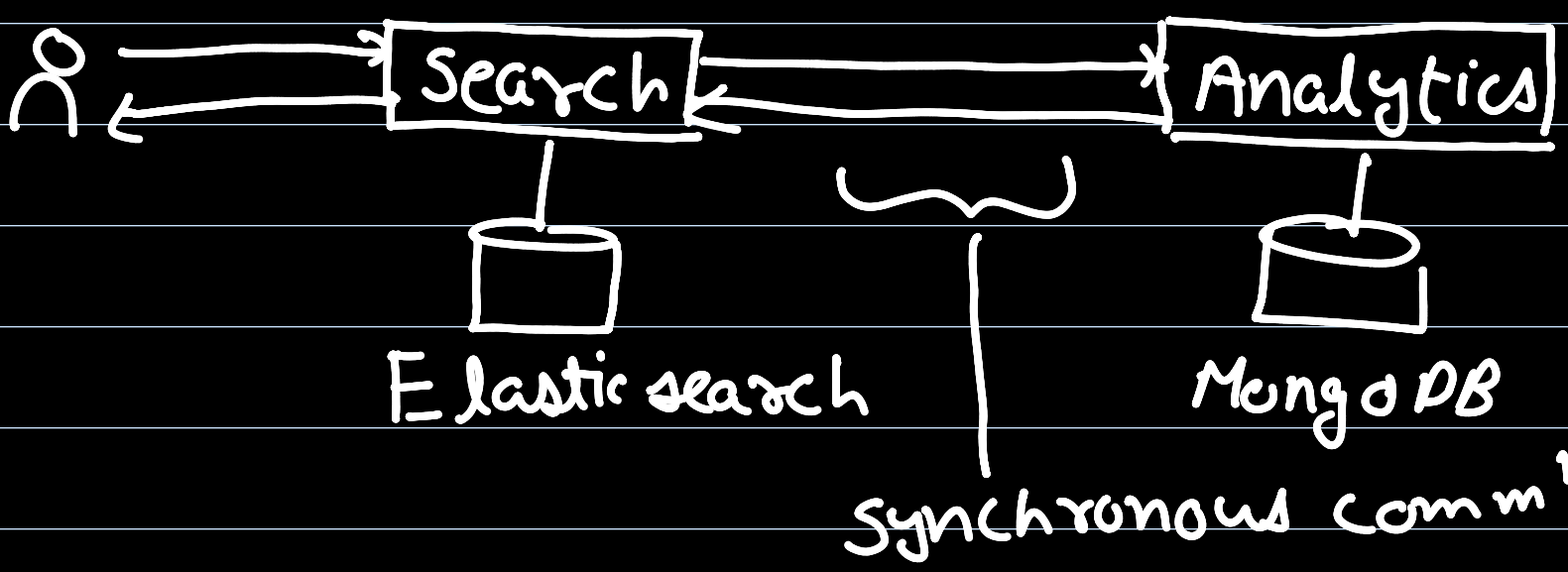
Handling Timeouts

Microservices are great at handling the complex problem & also helps in making new features easy. But there is one problem we need to address:- Timeouts

let's take an example:-



Let's say we have a search service. When the user searches something (blog/product, etc). Then it shares some data with analytics service as well.

Now comes the interesting part, these two are communicating with a synchronous way.

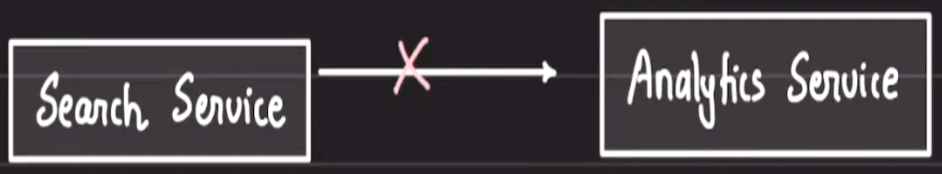
Here comes some cases:-

1 All good:- Everything works fine and result is delivered.

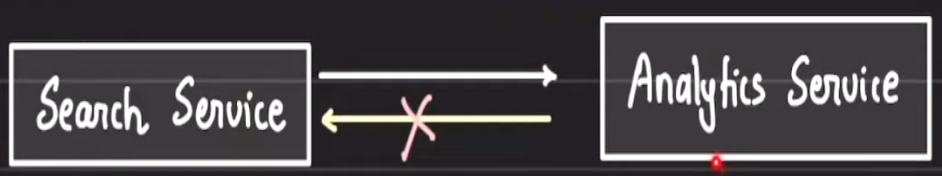
2 Delay occurs:- The search completes it jobs and sends request to the analytics servic

but there is delay. Now this delay is not good for UX. So we need to handle it.

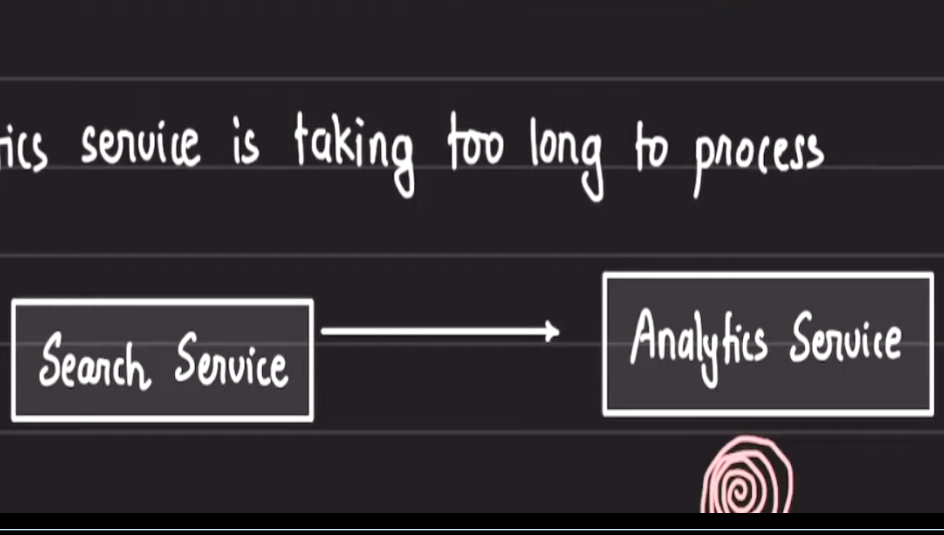
↳ Analytics service never got the request



↳ Response from the Analytics team never reached the Search Service



↳ Analytics service is taking too long to process



Here comes the concept of timeouts. Timeout is basically a wait time for the requesting service we add to move forward.

Rule of Thumb:- Always have Timeouts in your services.

APPROACHES

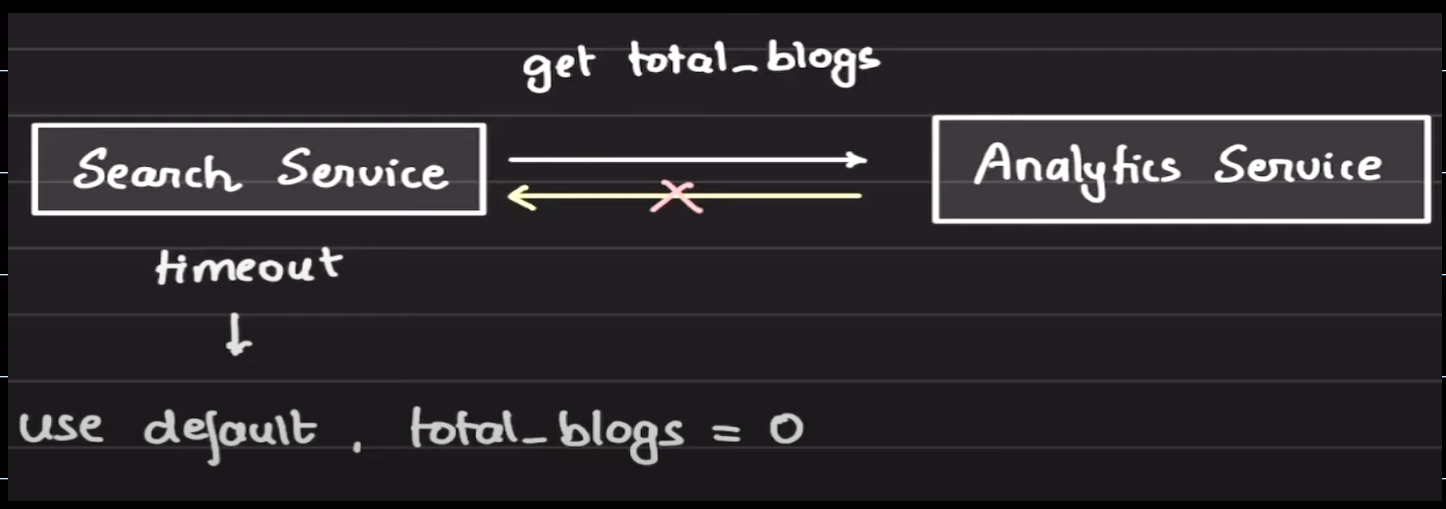
1 Ignore these things (Not Recommended)

In this scenario we assume that everything will run succesfully (but isn't). This leads to unpredictable user experience.

Good practice :- Catch all exceptions and then take an informed decision.

2. Configure and use defaults

Upon timeout use a default value (choice)



Retries are simple when it is a "read request"
But, sometimes situation becomes tricky

↳ request is **non-idempotent** eg: moving money from A to B
↳ request is **expensive** eg: heavy analytics query
↳ other service is **overloaded**
and you add more load with retry

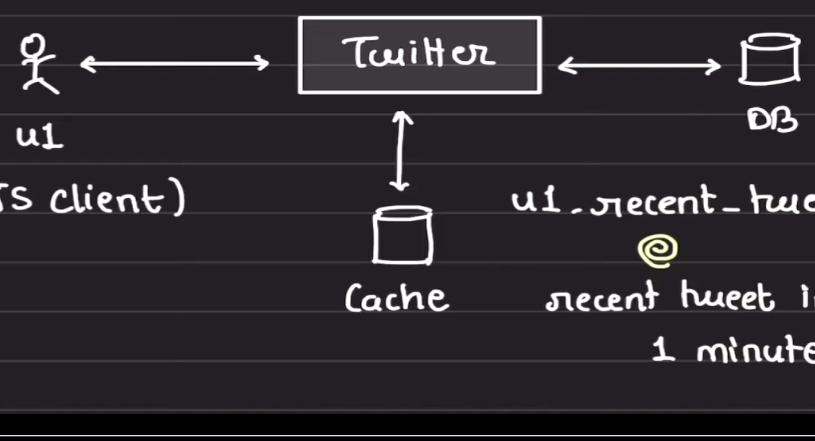
Good to have : Retries with exponential backoffs
[1s, 2s, 4s, 8s, ...]
Make services as idempotent as possible

Approach 4: Retry only if needed

In some cases, we may check for the completion and then decide to retry

In some cases, we may check for the completion and then decide to retry

eg: User tweeting the same post twice accidentally (within one minute)



Approach 5: Rearchitect

Event driven arch.

Remove synchronous communication wherever possible