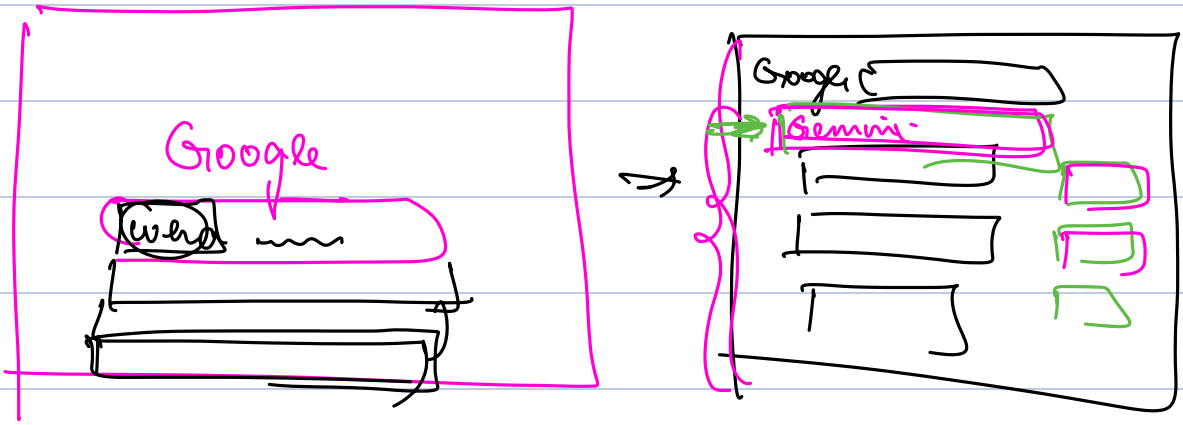
Intro to Microservices

(will start at 9:10)

- What are MS v/s Monoliths
- Pros and Cons of MS and Monoliths
- API Gateways
- SOA v/s Microservices

2 Case Studies

Google Search



get Suggestions (prefix)

Search (term)

fetch Ads (item)

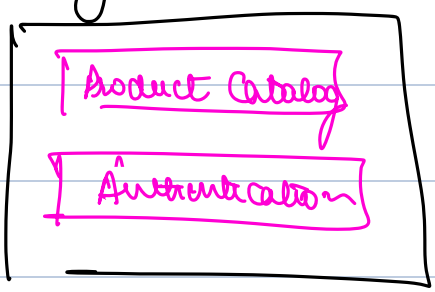
get AI Summary (item)

Should f^n of all of these be a part of same codebase or diff codebase.

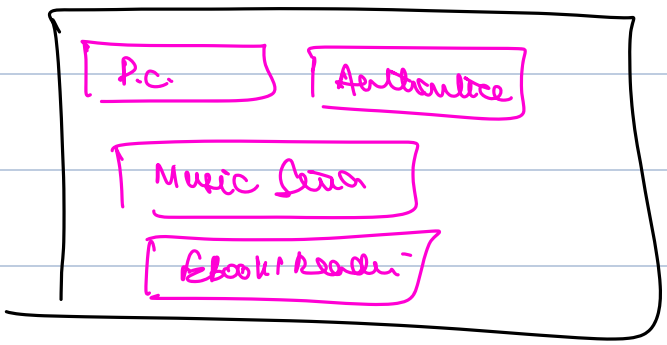
MONOLITHIC — Depends — MICROSERVICE

Flipkart Case Study

2008 — 200 orders/day



2012 → Music Purchasing Service
→ EBook Reader

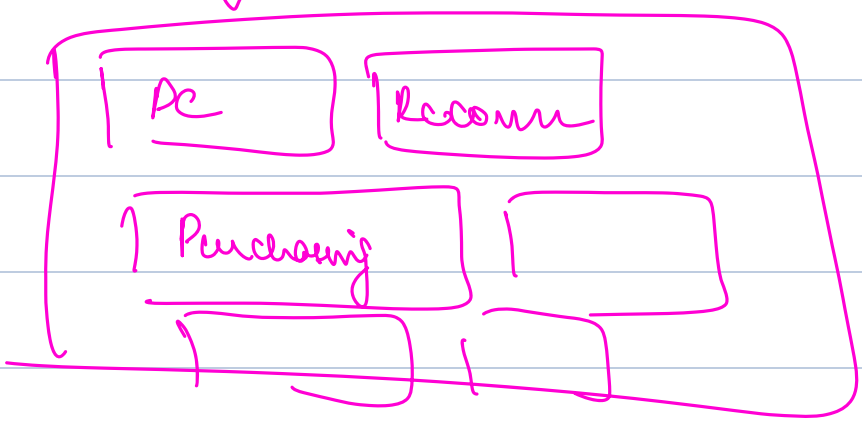


2016

↳ COS Spam detection system

↳ Recommendation

↳ Mailing



If ~~everything~~ is a part of 'Same application'

↳ everything needs to have same lang.

⇒ Tech Debt

↳ Bound to technologies of time when code started.

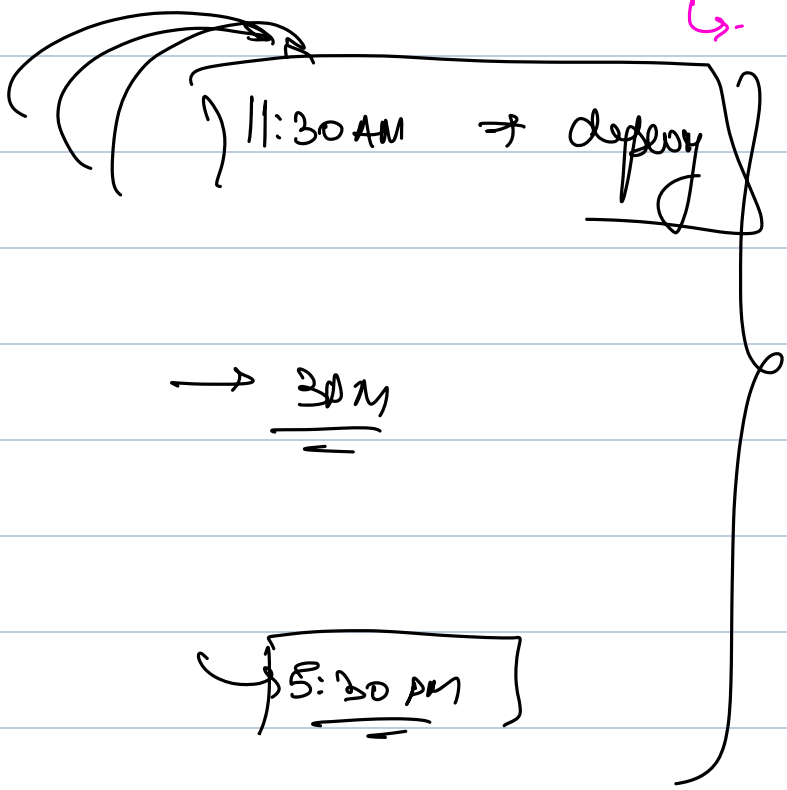
↳ Same code may not be fit for every use case.

↳ Bad Dev Productivity

↳ lot of time to compile.

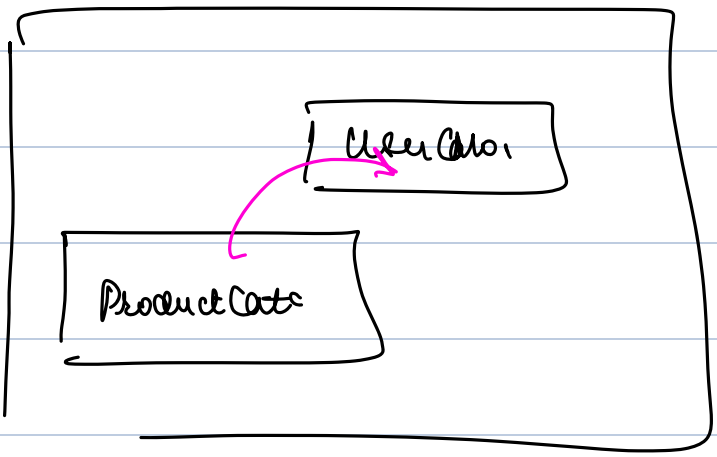
↳ Dev onboarding can be overwhelm

↳ Org can become slow.



Pros

1. Easier to manage team skills.
2. Only Method Calls B/W Modules ⇒ Fast



3. Testing becomes easy

Ruby On Rails

Monolithic architecture

↳ architecture where all functionalities of a company exist as one application.

Micro services Architecture

↳ 2015

Jeff Bezos ⇒ 2 Pizza Team Rule

↳ teams should be of small size.

1.) "small" & "independent" teams

2) every team should be a service provider to someone

↳ another team

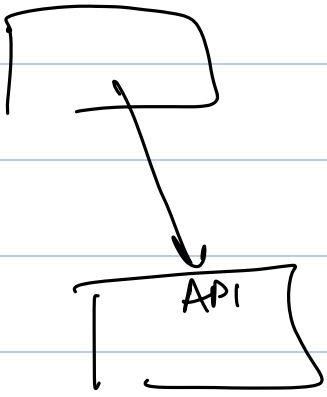
↳ end users

3) every team should be like a Business Startup.

MS Architecture

Better than having everything as part of 1 app^{n}, divide your app^{n} into diff app^{s} where each app^{n}.

1) Completely independent.
   - Separate DB

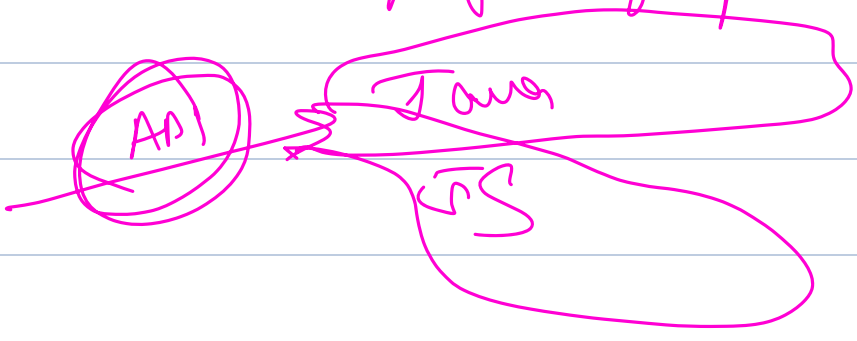


↳ only APIs are exposed.

Why? (PROS)

1.) No tech debt

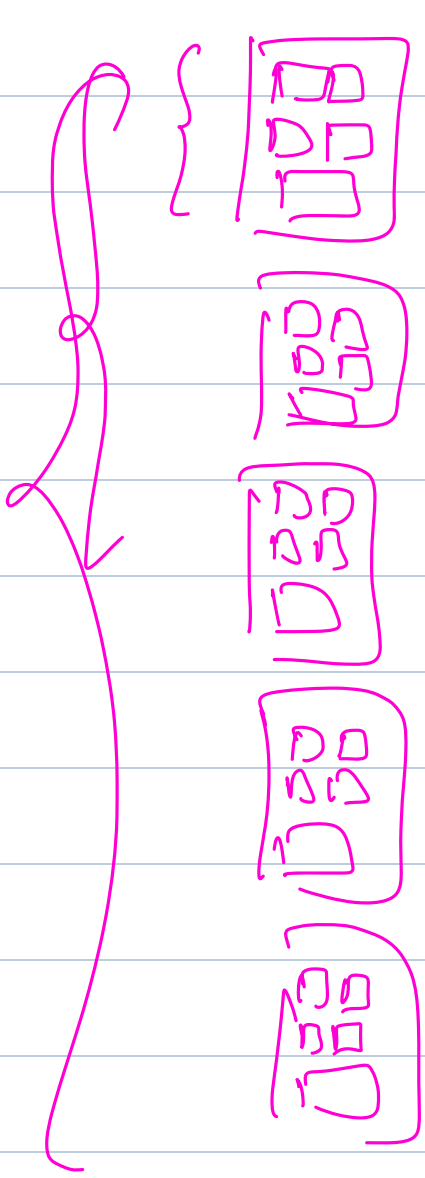
2.) Easier to keep yourself up to date



3.) Easier onboarding: only need to teach about your codebase (MS your team owns)

4.) Good Dev Productivity

5.) Selective Scaling



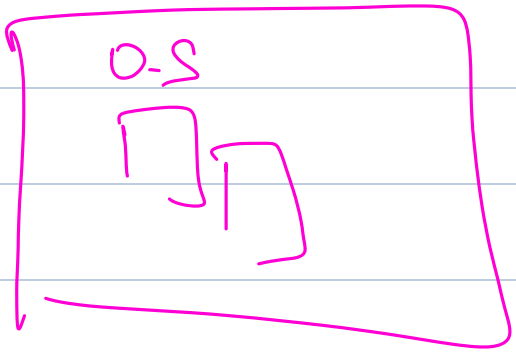
1.) PC 1000

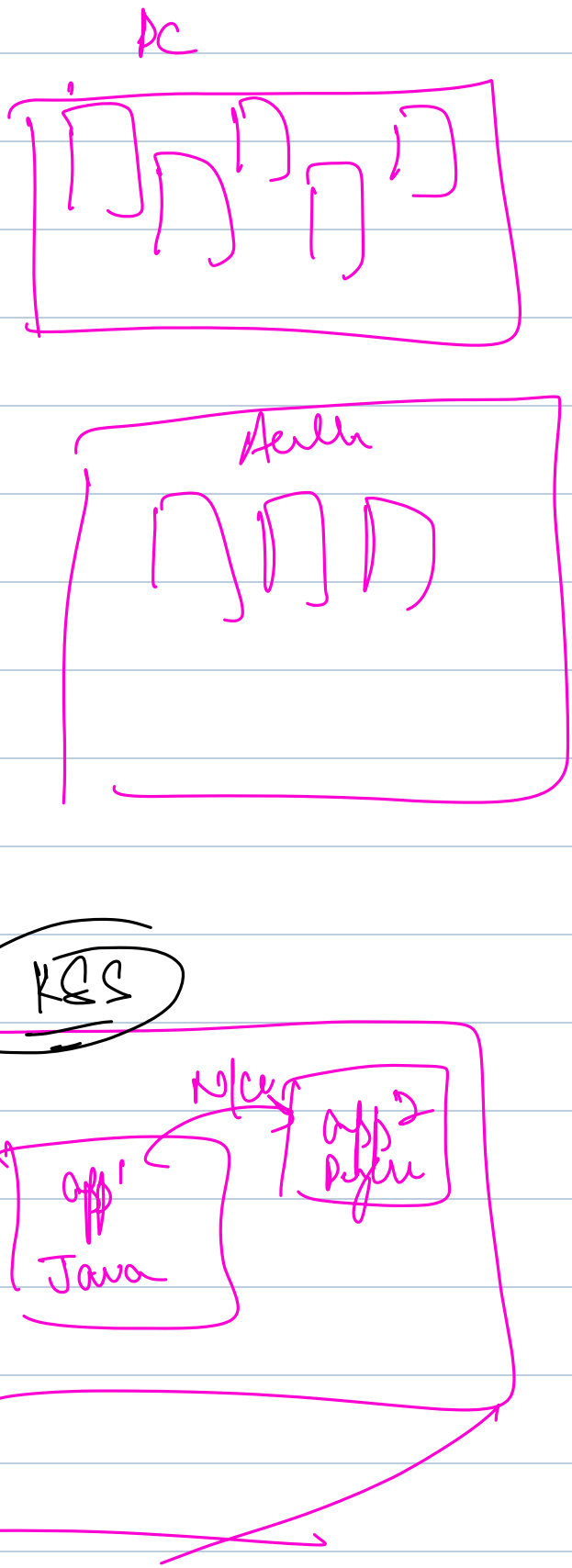
2.) Auth^{n} 100

3.) Adv 10

As let's say OS needs more RAM and PC needs more storage now every level will have more storage + more RAM

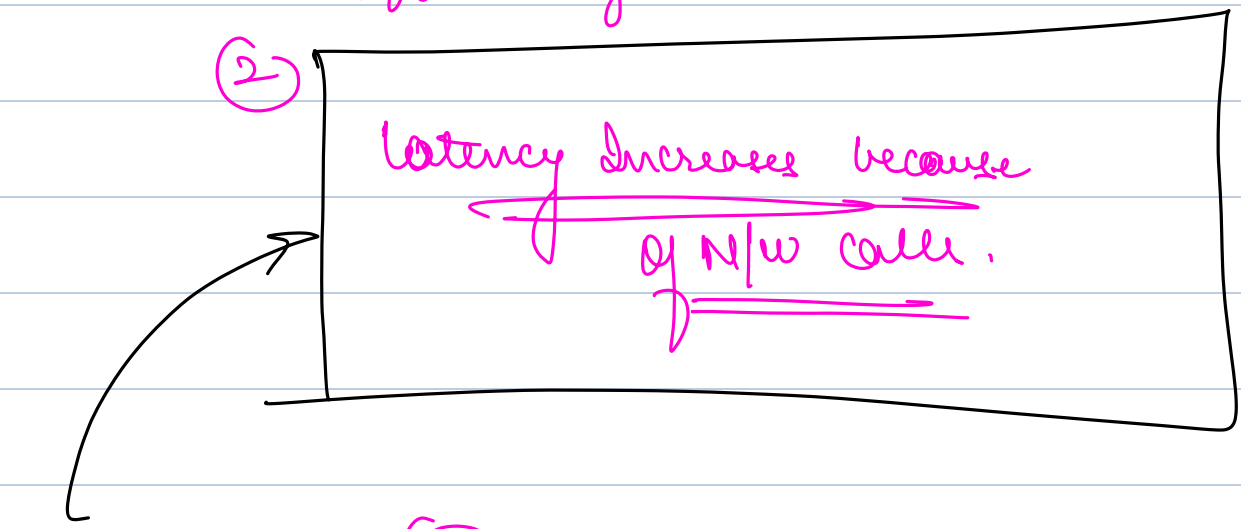
(extra cost)





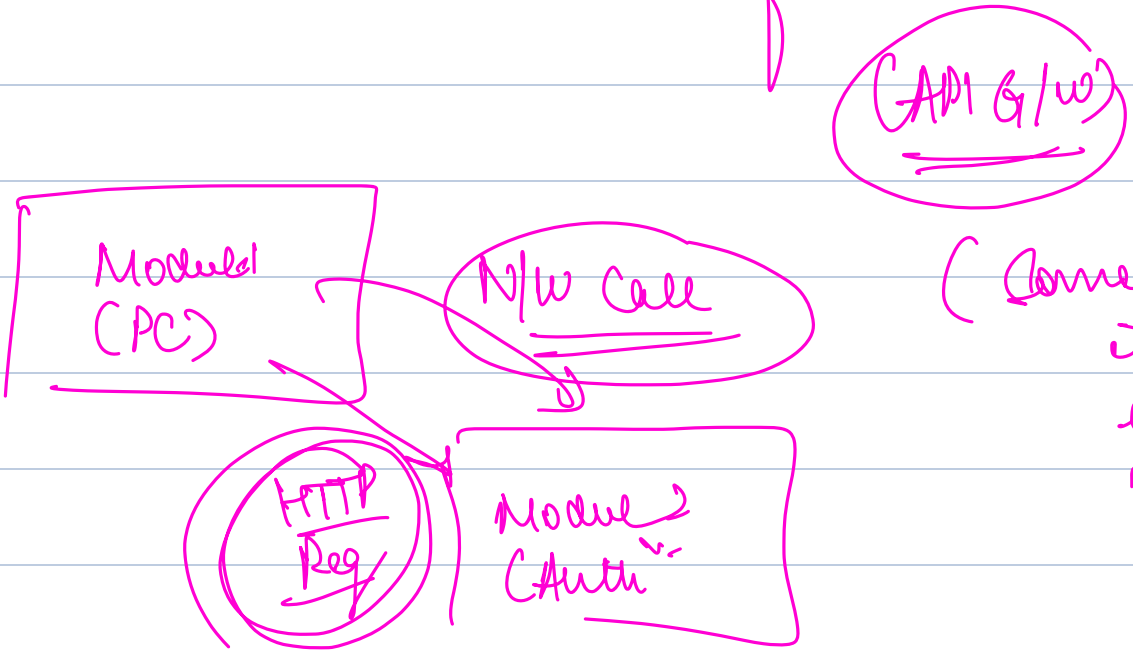
Cons of MS

1. Some teams with special knowledge.
2. Latency Increases because of N/W calls.



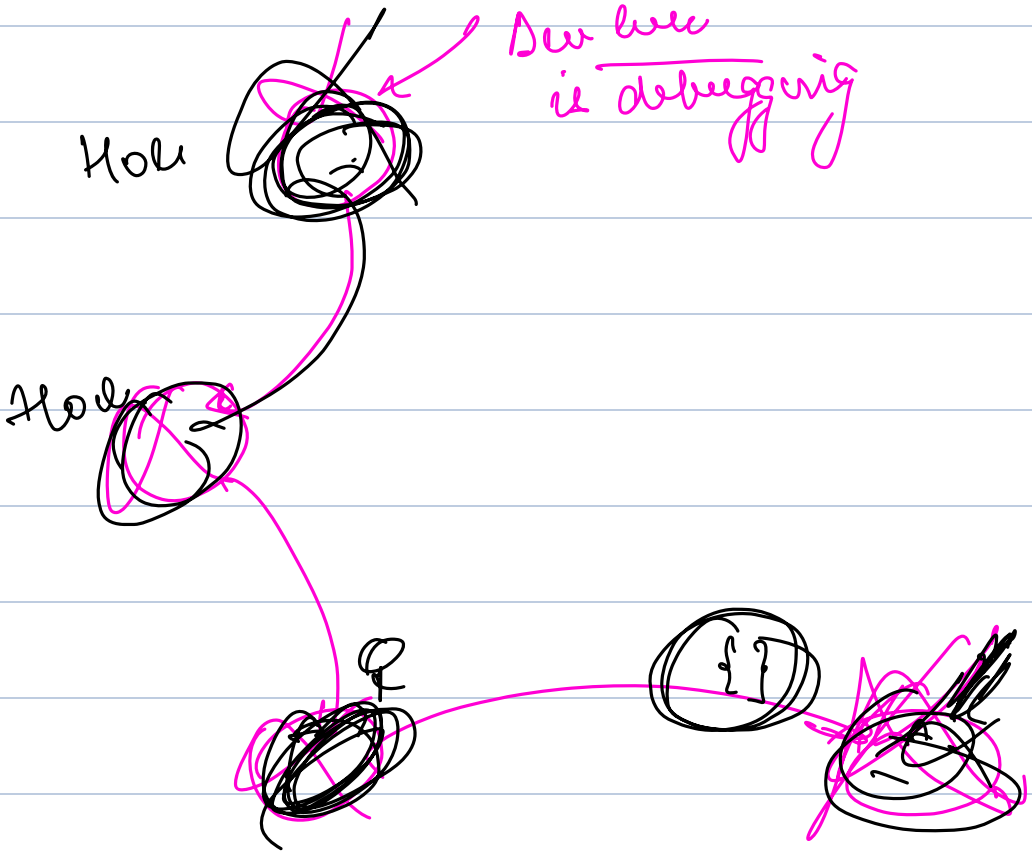
Reason why Prime Video moved away from MS

3. Testing and Debugging becomes diff.
4. Add add^{n} infrastructure for MS

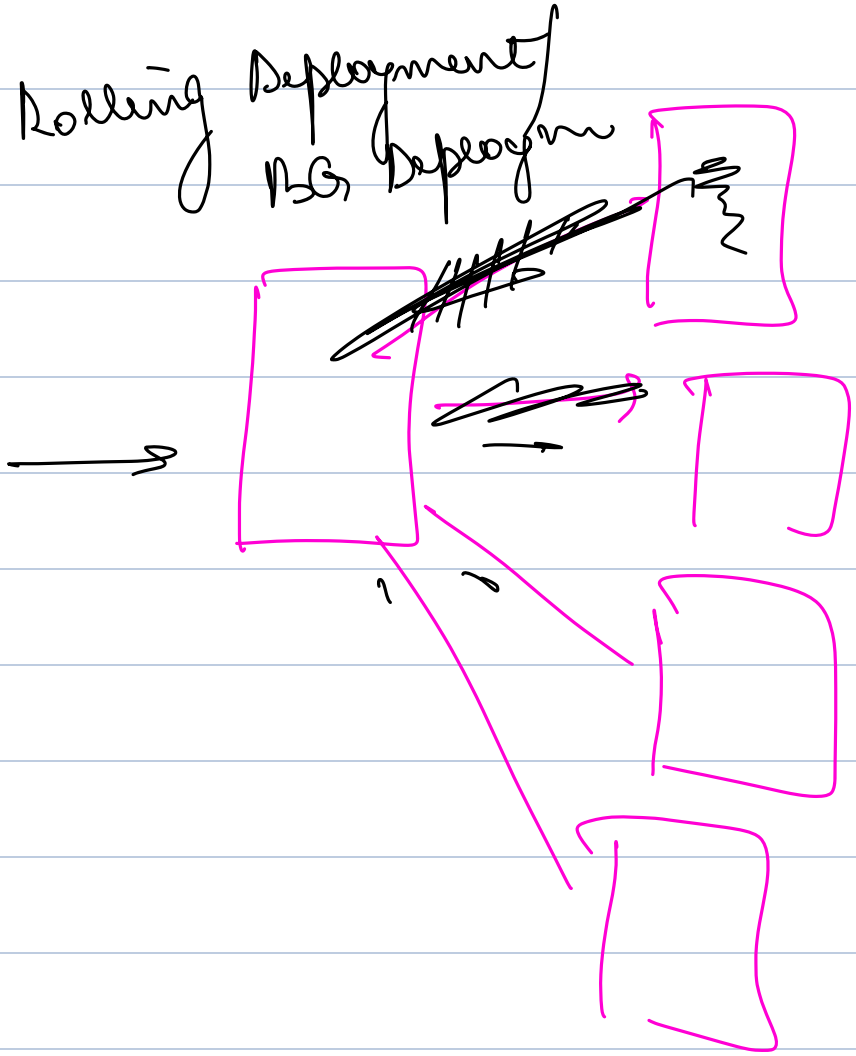


(Sometimes it might end up in cons)

5. Potential Inconsistence.


Dev box
is debugging
Node
Node
Q
IT

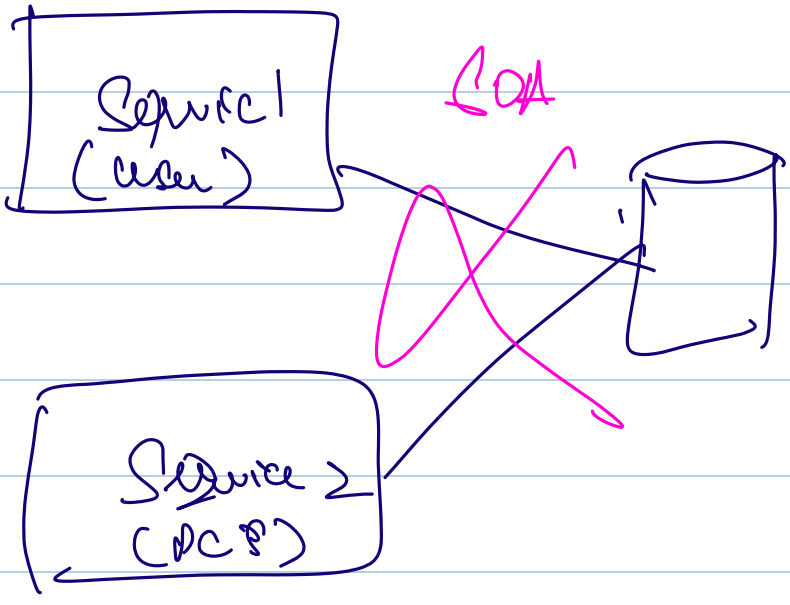
Rolling Deployment
BG Deployment

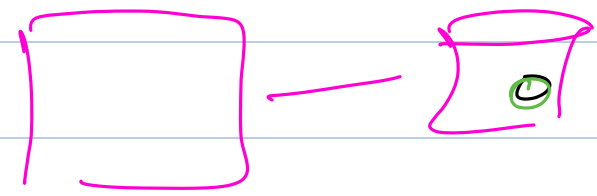


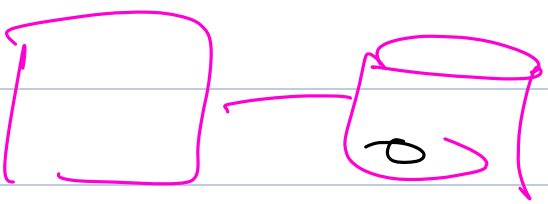
MS v/s SOA
(Service oriented architecture)

→ Sharing of infra (DB)

→ Dep compare
→ comp independent infra

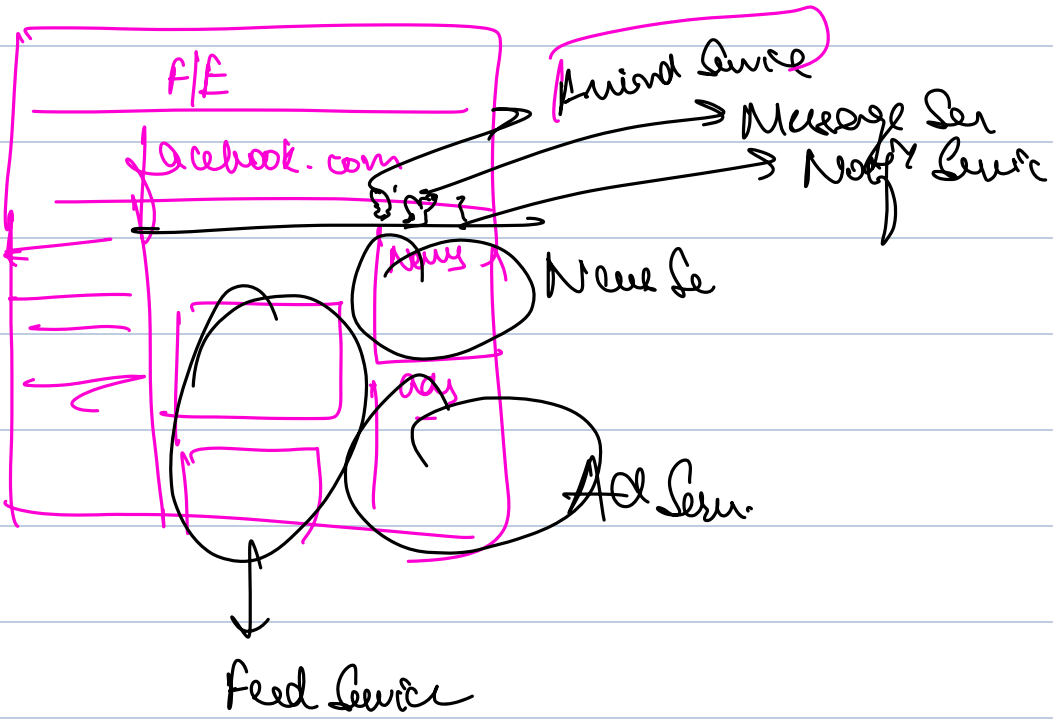






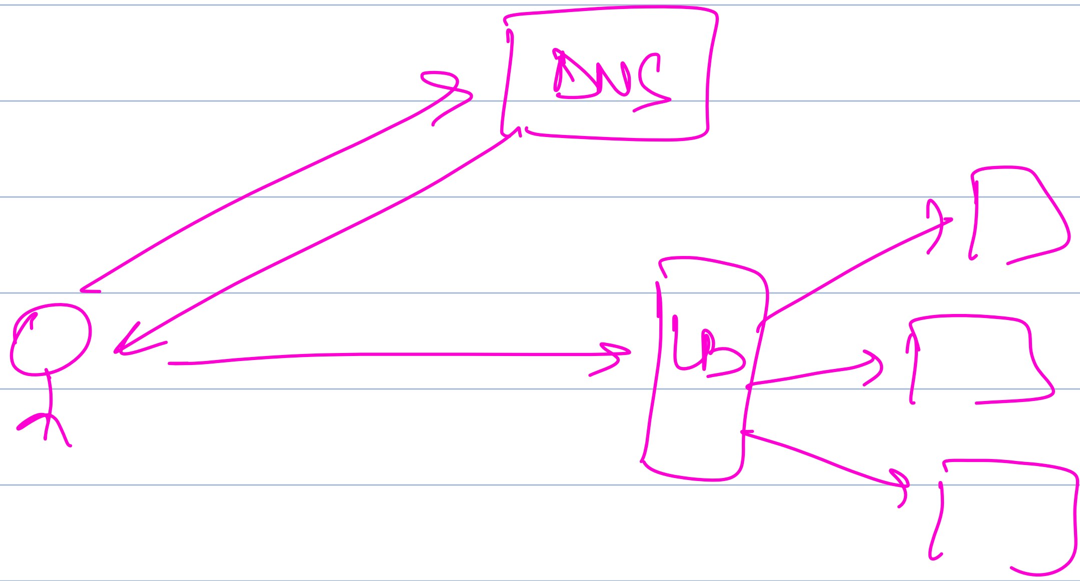
(Break till 10:15)

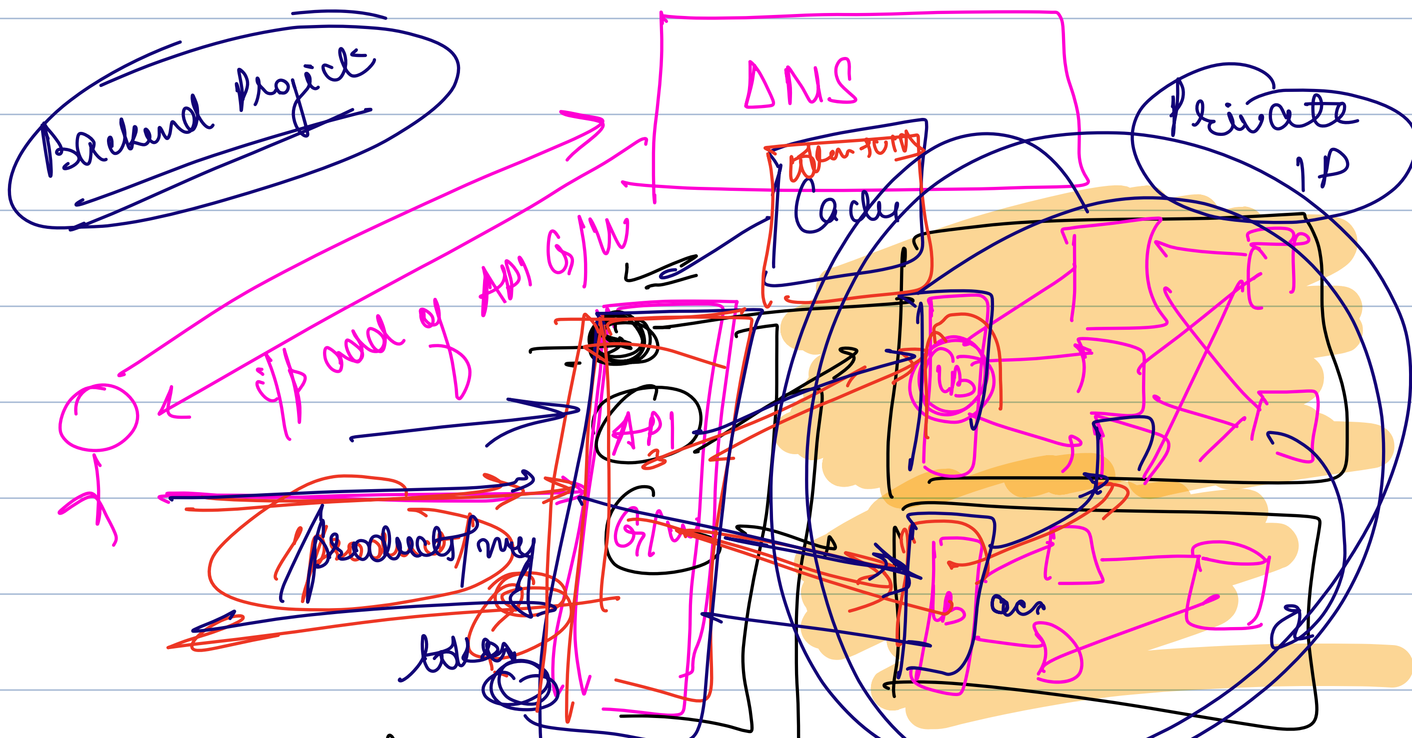
API Gateway



FriendService.fb.com/ ——→
notify

fb.com/ — / ——





Send API req to the correct place

config

x/y → Auth Ser

products/ → Proc

products/my → AUTC

/products/my → AUTH

API G/W

1. Route req to correct service
2. Manages auth validation / filling headers at its end.
3. Rate limiting

GetMyProducts (auth token)

---

should I always use MS

→ NO

Depends.

→ Always start with monolith unless strong reason → product market fit

→ PMF and you start to see issues of monolithic

- slow deploy times
- tech debt (exceptional cases)
- diff onboarding

→ Move to MS thoughtfully

↳ only 1/2 Service to start with

↳ as per need.

Current

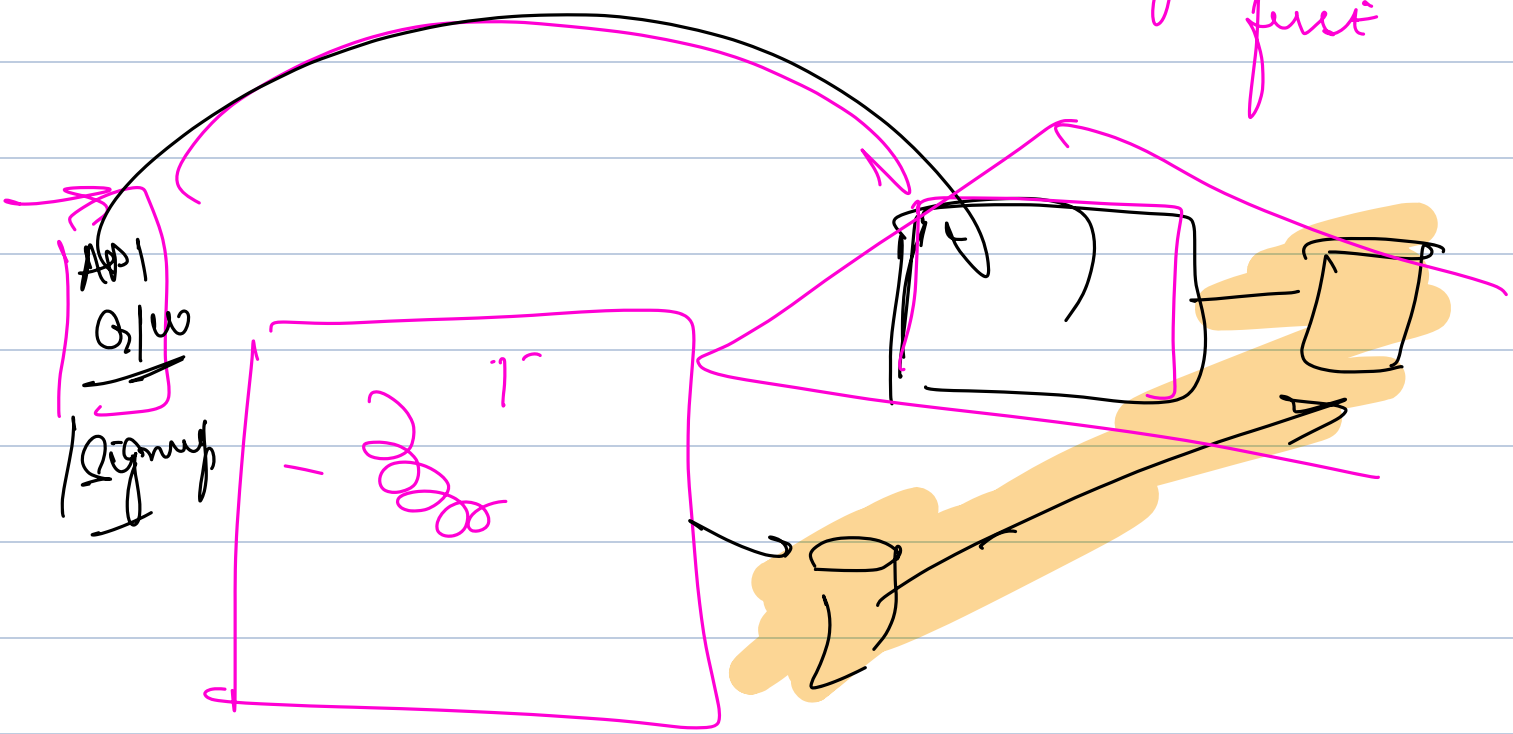
2015 – 2022 ⇒ Monolith

2022 → Auth

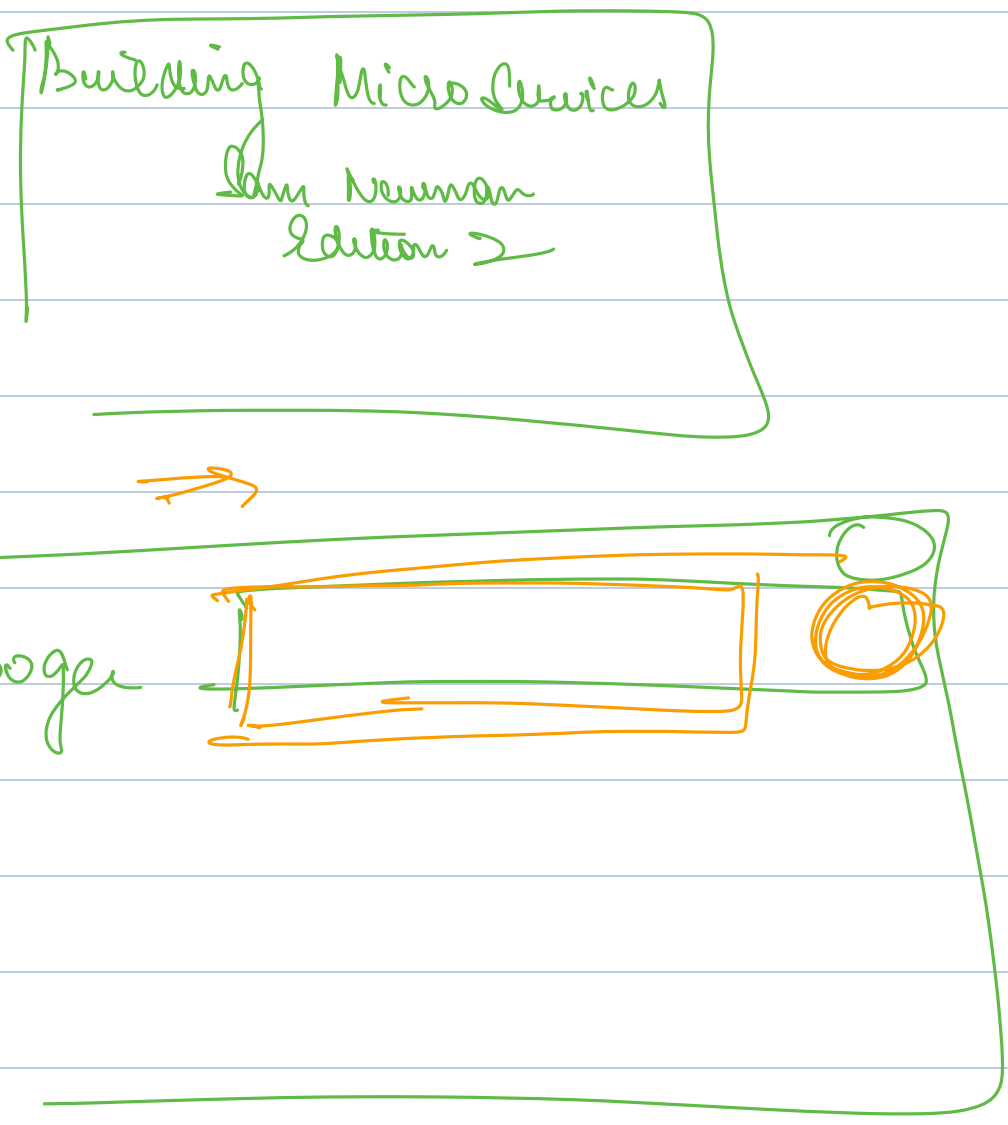
99%

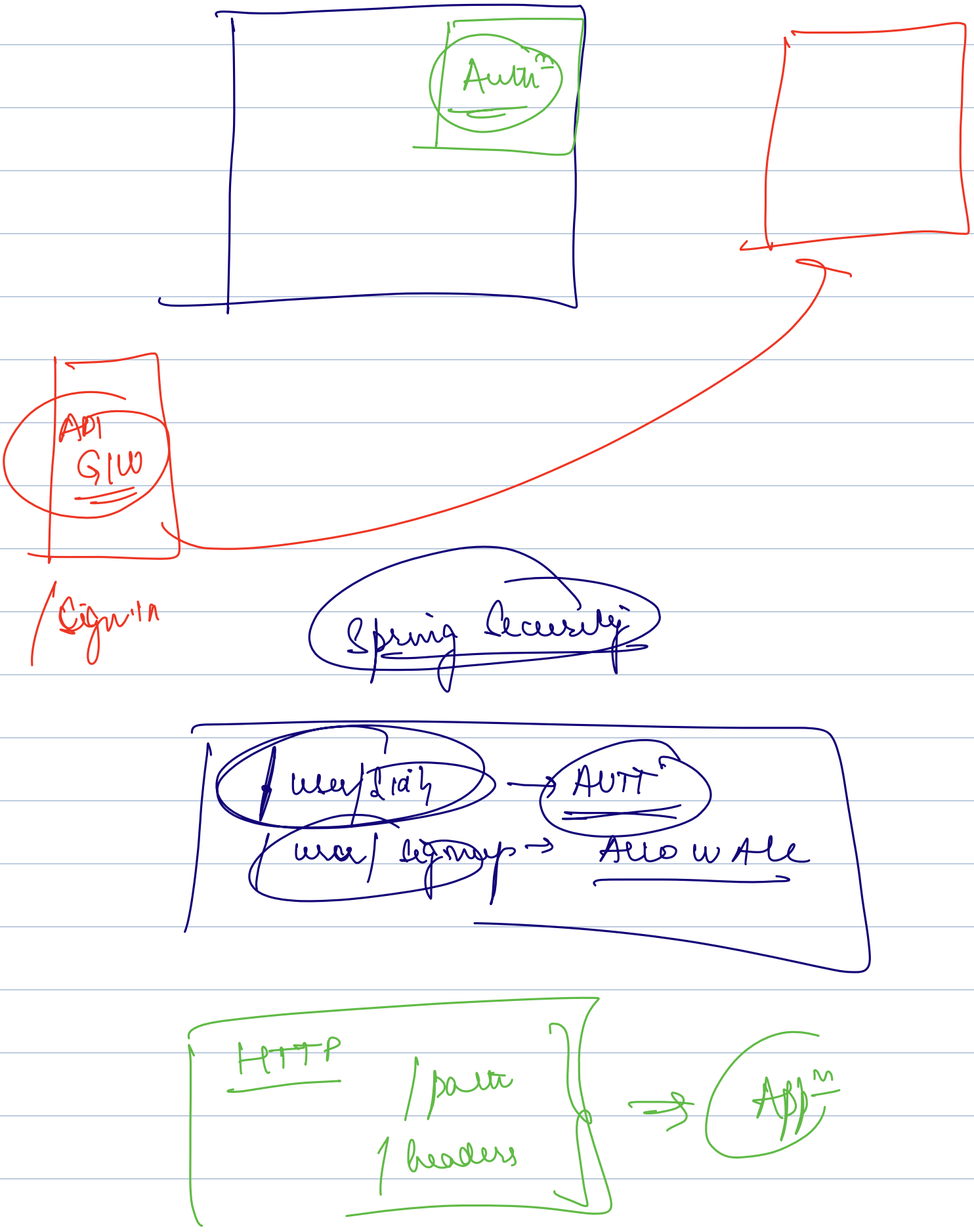
↳ Auth Service get moved out first

2023 → Assessment




Ruby
on
Rails
Quota
problem
solving


Building Micro Services
Sam Newman
Edition 2
Google


Auth
API
GW
/signin
Spring Security
/user/login
AUTH
/user/signup
Allowable
HTTP
/path
/headers
App

Kong | API GEF

Agora