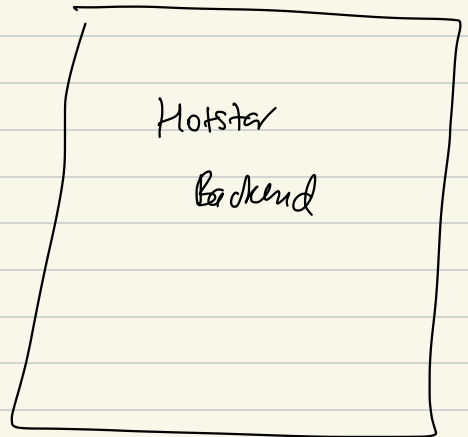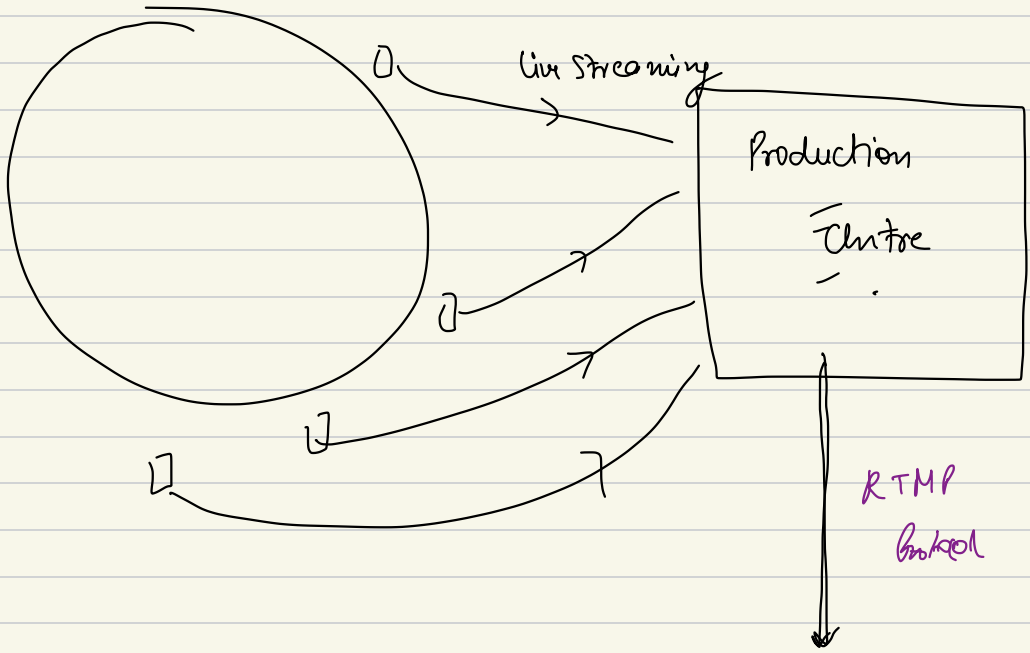# Hotstar System Design
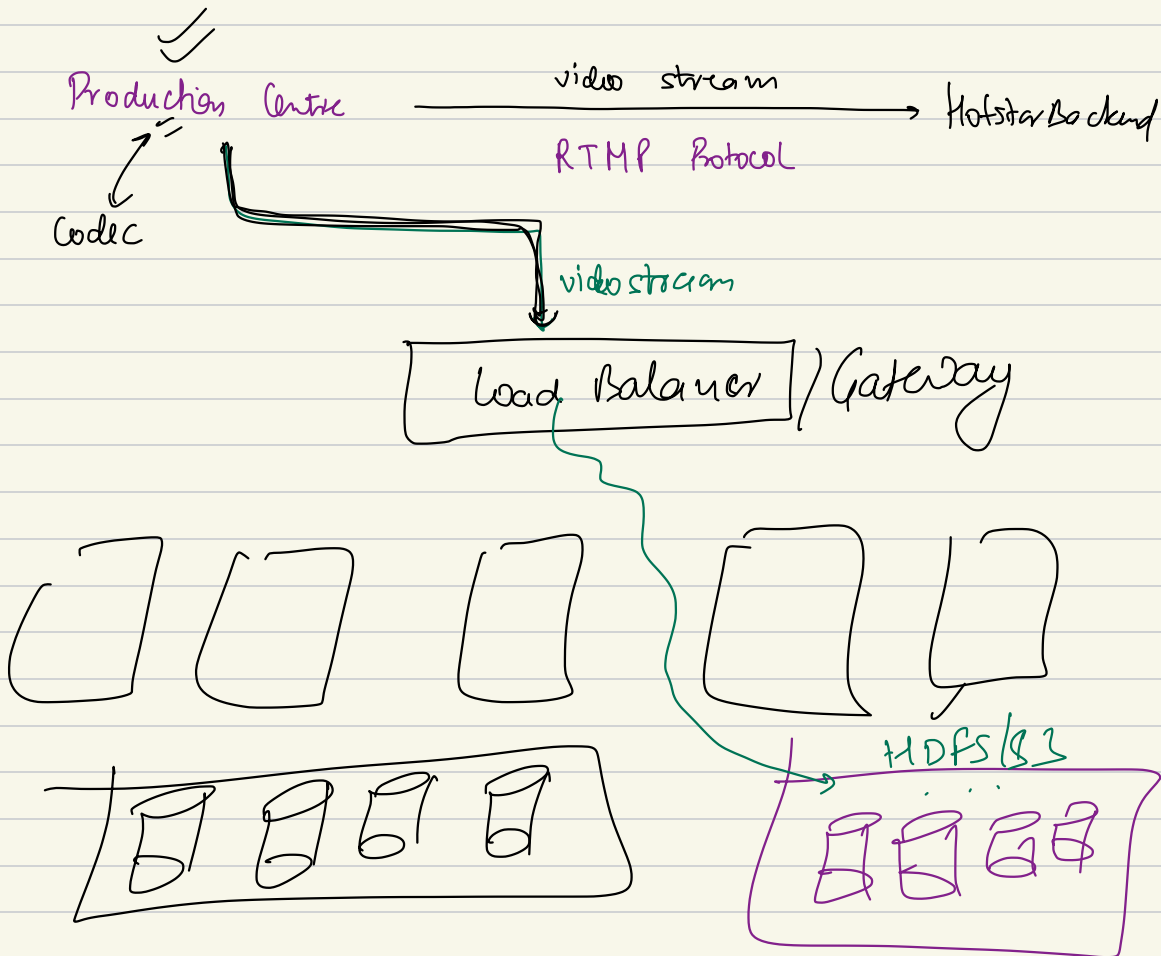
① Content Ingestion

② Discoverability

③ Streaming

Live Streaming
Content

Live Streaming of Content

Live Streaming

Production Centre

RTMP Protocol

Hotstar Backend

① In Live Streaming, the chunk size should be way smaller, let's say 15 - 30 seconds each.

Production Centre

Codec

video stream

RTMP Protocol

Hotstar Backend

video stream

Load Balancer / Gateway

HDFS/S3

Ingestion Pipeline running at backend

8K video → different resolution

→ 240p
→ 480p
→ 720p
→ 1080p
2K

15-30s each

↓ stored ☺

S3 | HDFS

metadata

database ——— populate ——→ Global Cache
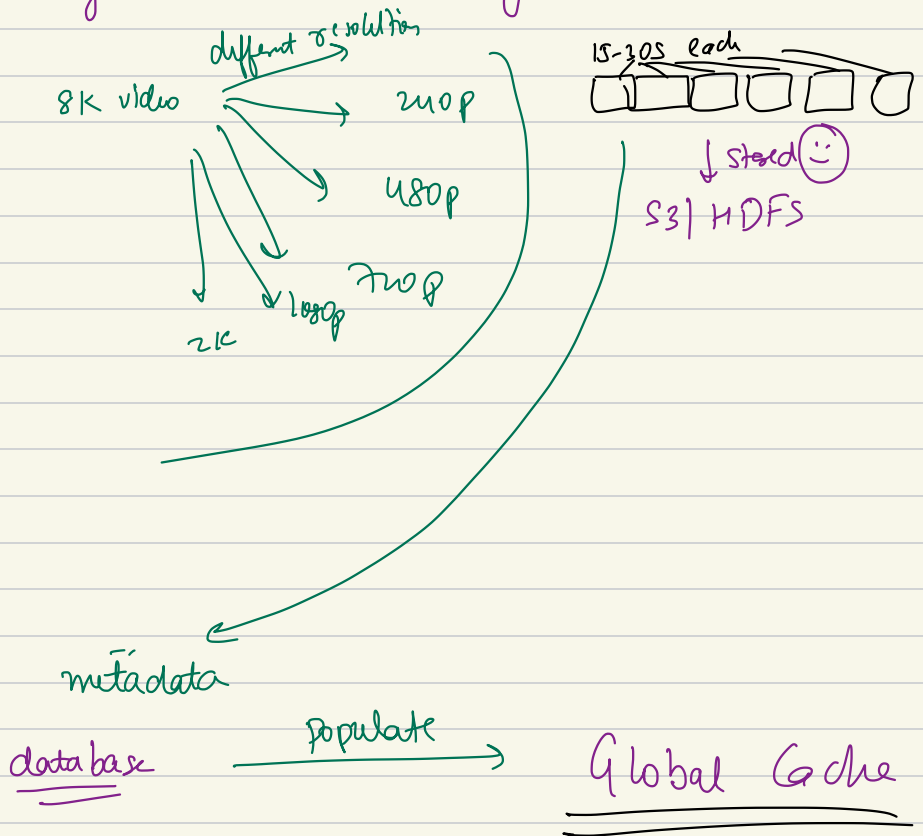
① store video chunks of diff resolution
in backend's S3 / file storage

② populate meta data in (DB) + Global Cache
↳ ex SQL

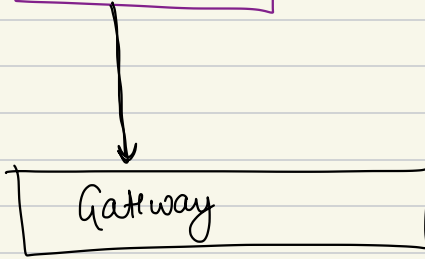③ The chunks, also need to be
stored / sent to CDN ☺

detour

Client

big state data

CDN link

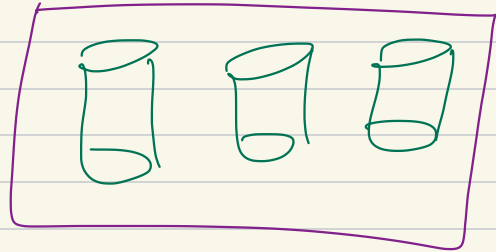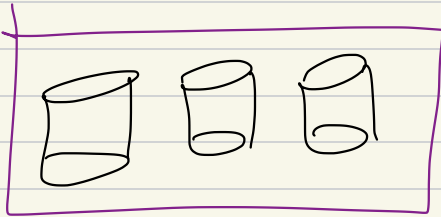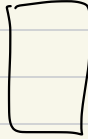Gateway | LB
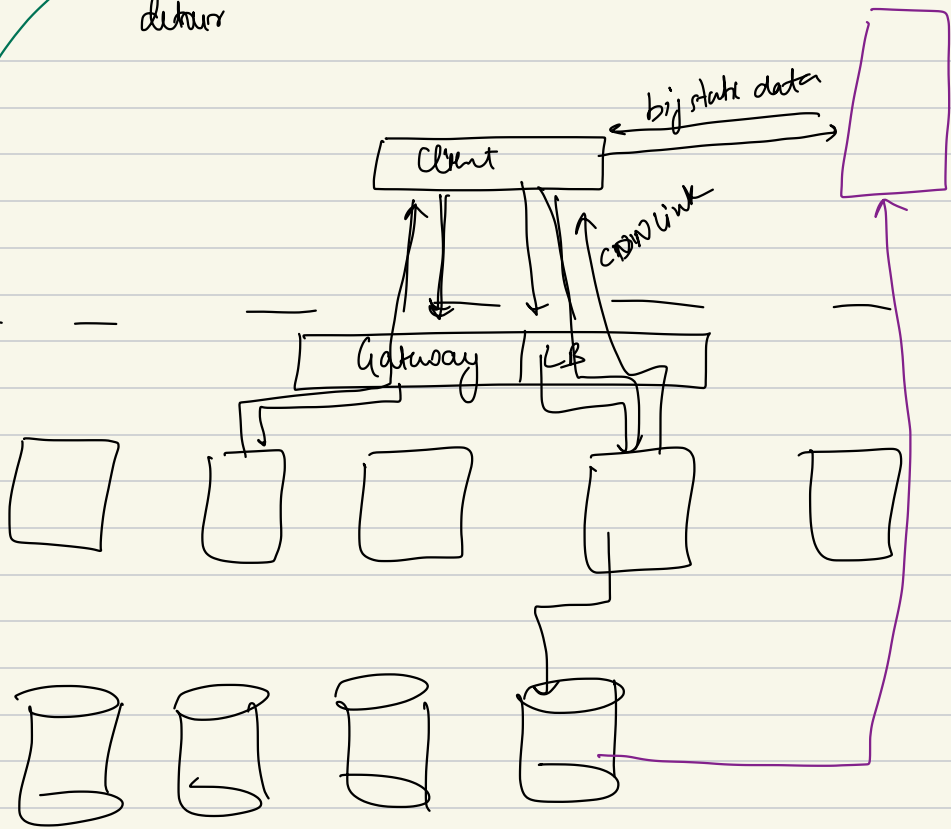
Streaming For Live Feed

CDNS

Client   x   millions 🙂

Gateway
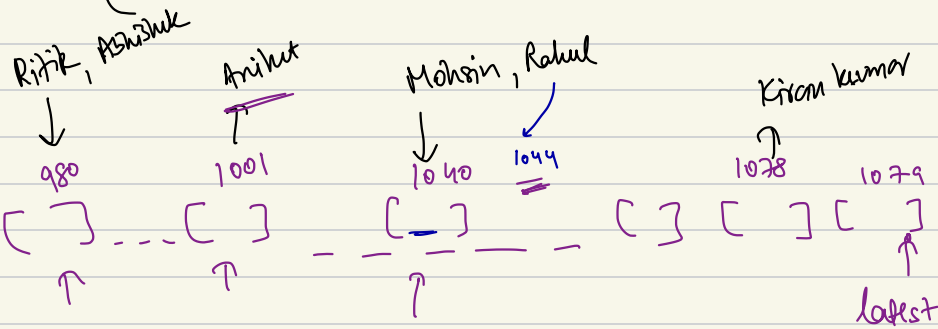
Global Cache

Streaming Platforms like Hotstar, allow users
to watch at their own lagging delta
in other words, Hotstar allows for lagging
clients to watch different segments.

Ritik, Abhishek
↓
980
Aniket
↑
1001
Mohsin, Rahul
↓   1044
1040
Kiran kumar
↑
1028   1079

[ ] ... [ ] _ [_] _ _ [ ] [ ] [ ]
 ↑   ↑   ↑   ↑
                                     latest

get Chunk Metadata (last_chunk_id , Count)
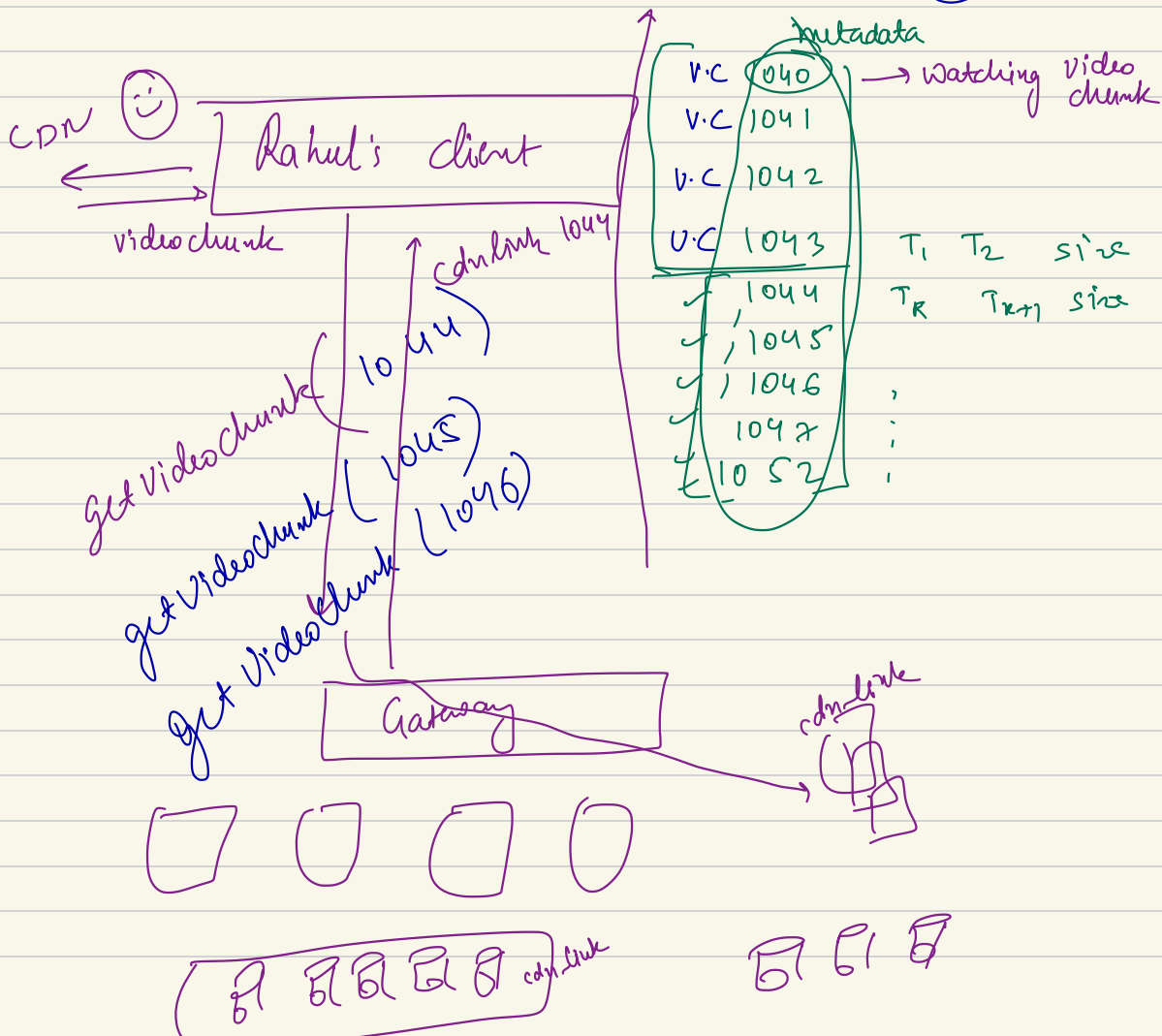
Aniket
get chunk Metadata (1001 ,   20)

Rahul
get chunk Meta data (1044 ,   20)

(2) In case of live streaming

get-chunkMetadata() calls can't happens only once per user, instead it needs to happen again and again throughout the stream 😊

metadata → Watching video chunk

V.C	1040
V.C	1041
V.C	1042
U.C	1043
	1044
	1045
	1046
	1047
	1052

T₁ T₂   size
T_R T_{R+1} size

CDN 😊
Rahul's client
← video chunk

cdn link 1044

Get Video Chunk (1044)
get Video chunk (1045)
get Video chunk (1046)

Gateway

cdn link

cdn link

Q   How do systems like these
     scale up /down drastically??

1,00,000   →   50 Million

① Ingestion → Same

② Discoverability → select different UI
                              based on load

③ Streaming ⟶ Meta data fetch
            ⟶ Video chunk Fetch

Scale up App Servers + Global Cache
              App Servers + Cache +
                    CDNS ↑↑ Scale
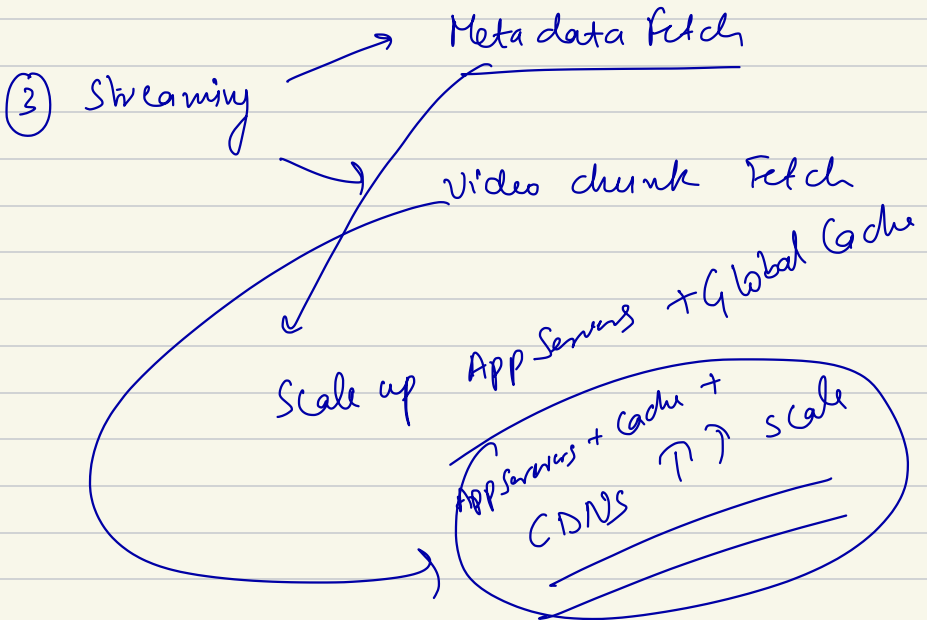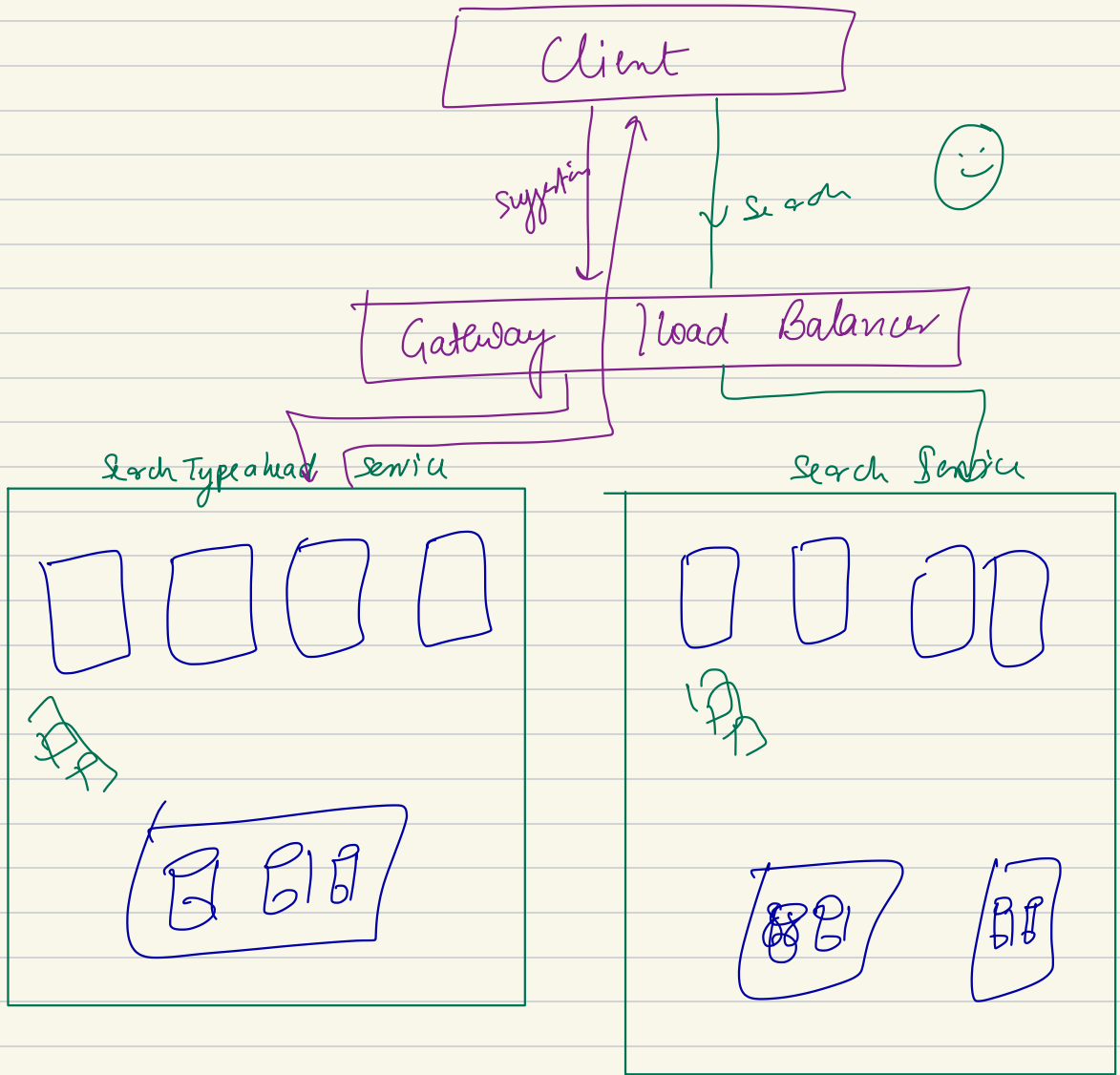
Hotstar vs Live Class
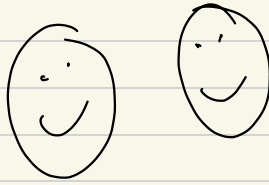
10:45PM— 10:52 PM

Microservices

ZIRP

Zero Interest Rate Phenomena

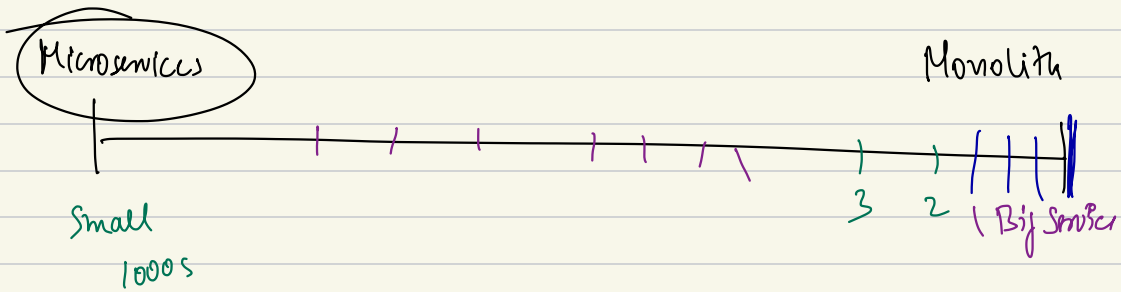Services

Search Typeahed

Client

suggetion → Gateway

Search → Load Balancer

Search Typeahead Service

Search Service

Flipkart

APIs

login · Forgot Password · sign UP

User Authentication

UDB

Seller

Seller

Buyer

Buyer

Inventory

Microservices ————————————————————————————— Monolith

Small

1000 s

3   2 | Big Service
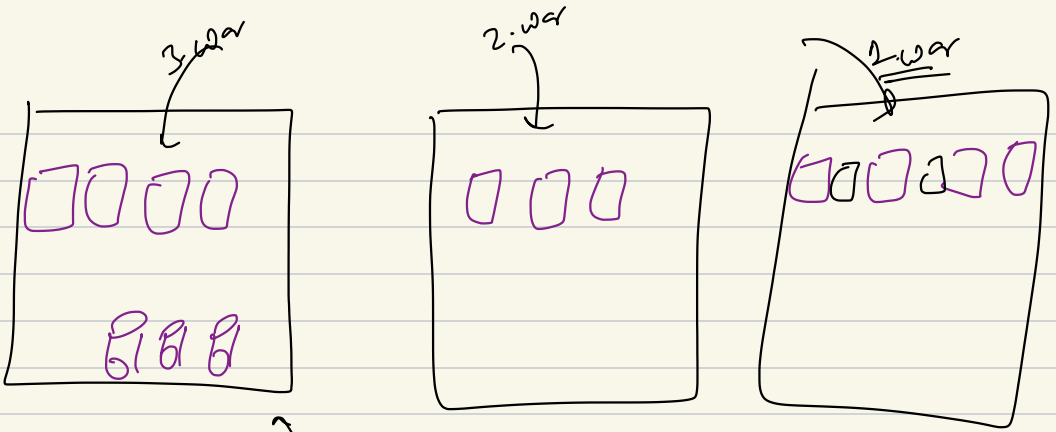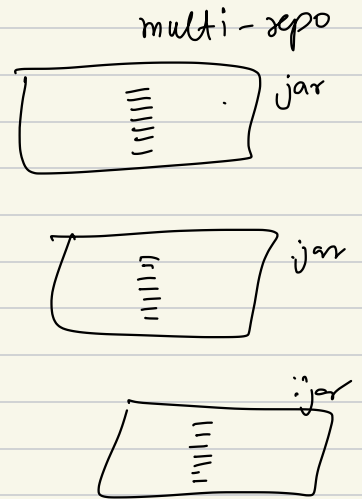
3.war

2.war

2.war
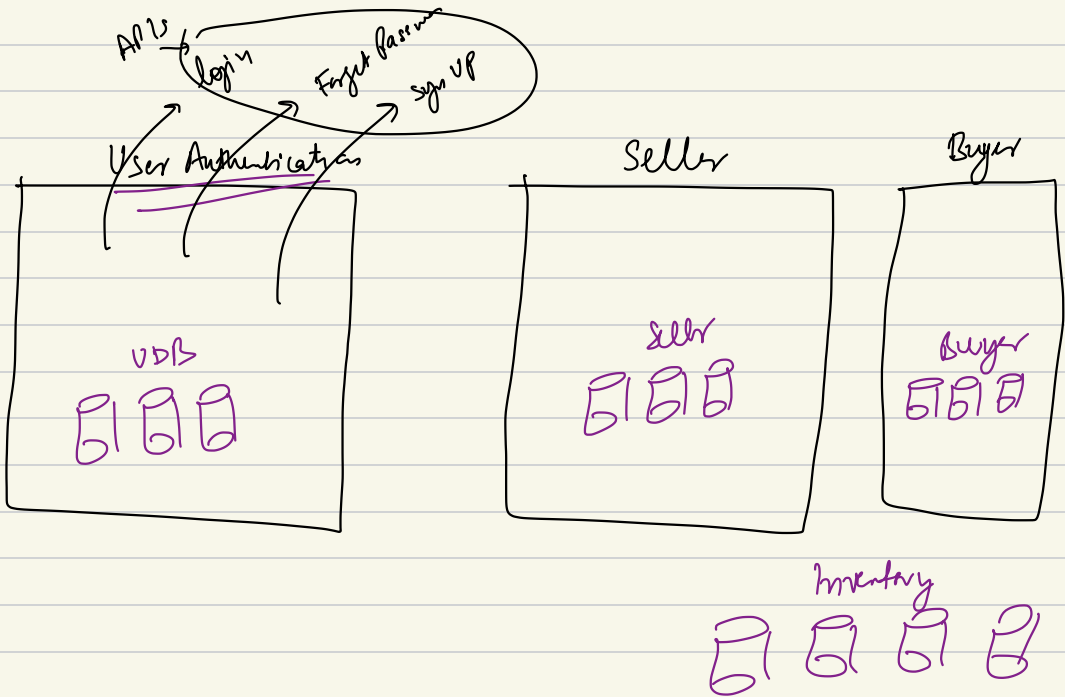
Every service is going to have a different deployable

mono repo

build file

build file

multi-repo

jar

jar

.jar
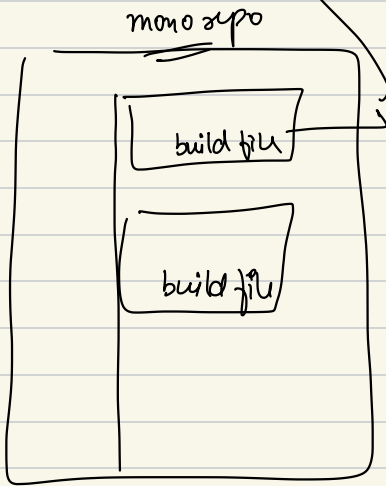
Monoliths
= > disadvantages

Microservices
advantages

① small code change
↓
recreate deployment

② developer onboarding difficult

③ blast radius ↑↑↑↑ / impact radius →

④ Build Time ↑↑↑↑↑

⑤ Deployment Time ↑↑↑↑↑↑

⑥ Kills the flexibility to choose
different tech stacks for
different use cases.

We have discussed so many
advantages of smaller services [microservices]
and inturn disadvantages of bigger services
[monoliths]

:)

---

downside of micro services

① with every new service you have the
overhead of managing a new service

→ deployment pipeline
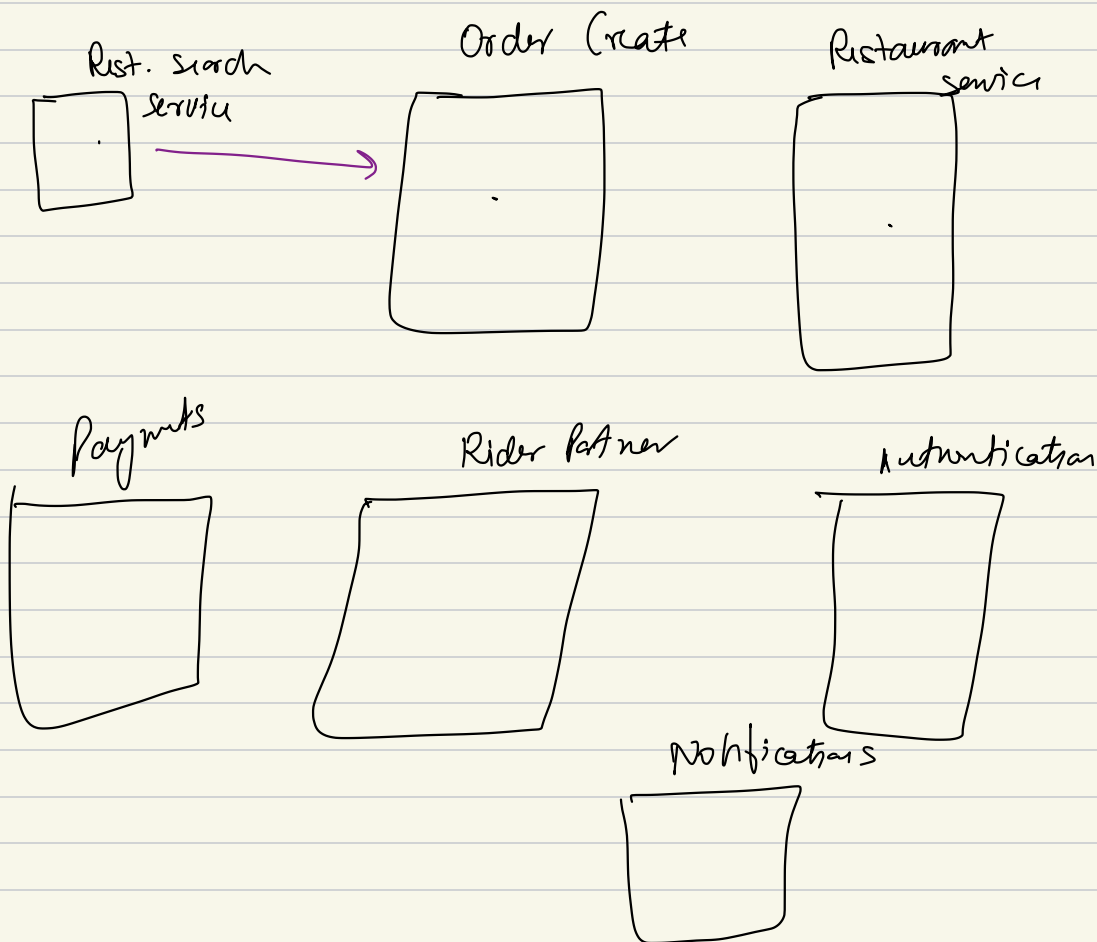
→ Test suite

→ Monitoring

→

Ep.

Swiggy as micro service architecture

Rest. Search
Service

Order Create

Restaurant
Service

Payments

Rider Partner

Authentication

Notifications

Strangler Fig Pattern ☺

Monolith | 1 service

Scale ⬆ ⬆

S2 ≡ authentication

S1

Scale ⬆ ⬆ ⬆ ⬆

S3 ≡ payments

S2 ≡ auth

S1