MICROSERVICES - 3

1. Event sourcing.
2. Design notification microservice.

---

1. 3 microservices → typed notes.
2. Computer Networks videos.
3. Internals of Indexing (multiple column indexing).
4. Table →

usecase	non-fn	Database choices

→ 19th November

Event sourcing

Orders

id	item_desc	source	dest	status
101	.	.	.	.

model — User.rb

controller

view

Frameworks

Ruby on Rails → PaperTrail.

papertrail → name, username

~~initialized~~

pending

payment_pending

payment_done

shipping_started

shipped

delivered

Cancelled

Activity / Events

item_type	item_id	~~updated_at~~	json{ }	...
Orders	101			

Events

id	type	old_v	...
101	order		

{ id: 101
item: x
...
status: pending }

Notification systems

MVP:

1. Send notifications.
2. Mediums should be pluggable/extendible.
3. Rate limiting
   - ↳ user/sender
   - Size of notifications
   - Authentication
4. Prioritisation
5. users → notification pref.

Step 1: MVP
- Functional
- Non-functional

Step 2: Est of scale
- ↳ Sharding
- read vs write capacity est.

Step 3: API design

Step 4: Overall design.

Non-functional

1. High availability
2. P0 notifications → delivery latency is low

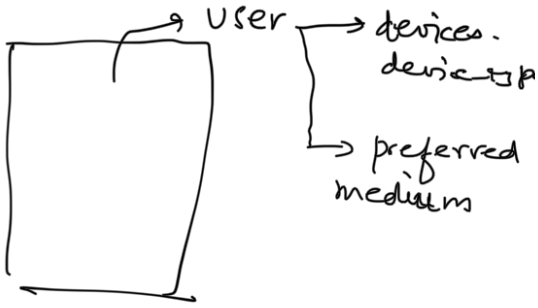
OTP
News Feed
Chats - ..
Event
- Post
-
-

Est. of scale

1. Sharding → ~~NO~~ not needed for undelivered notific
   - ↳ historical notificatn → sharding YES
2. Write heavy

Pub Sub → read + write heavy

① Notification de-duplication
② context notification → API
③ user, notification, mediums, priority.



① Send Notification (notification-id, userId, text, List<String> medium, priority, type) → timestamp

② get status (notification-id)

③ update User Medium (userid, List<String> mediums)

"user"

```
list {
    text: "....."
    medium: "---"
}
```

① Notifications async. → Client doesn't wait.

② Checks → valid notification
duplicate notifi
rate limited.

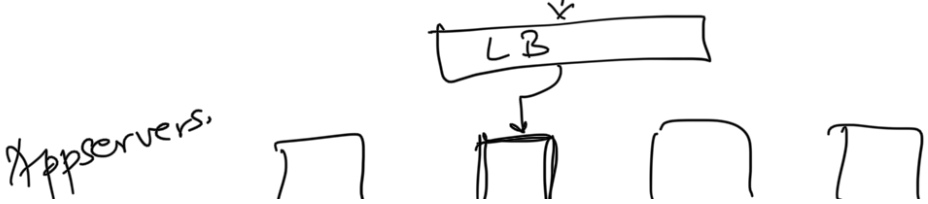
③ Mediums → which

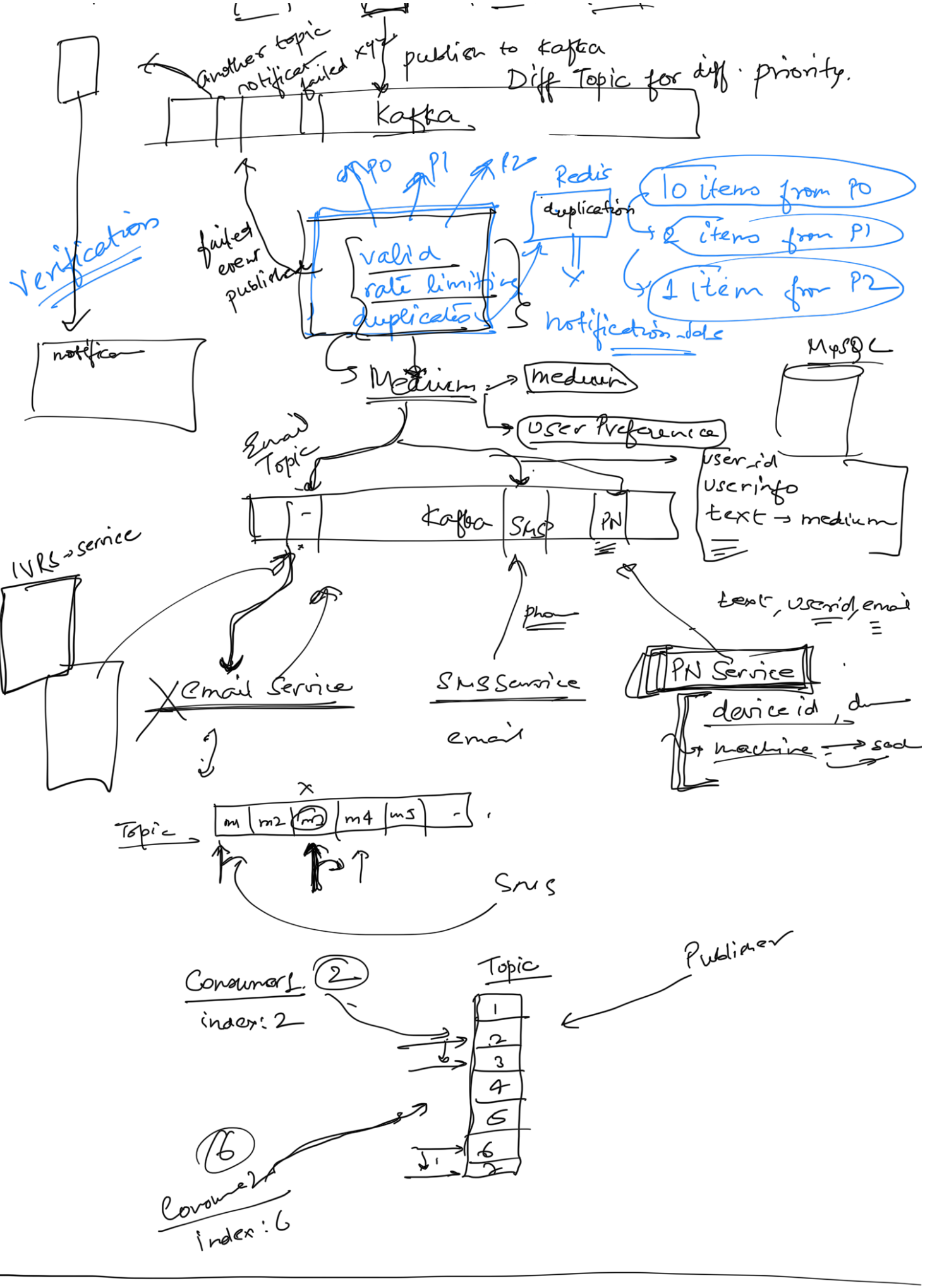
④ Async processing ~~from~~ by mediums

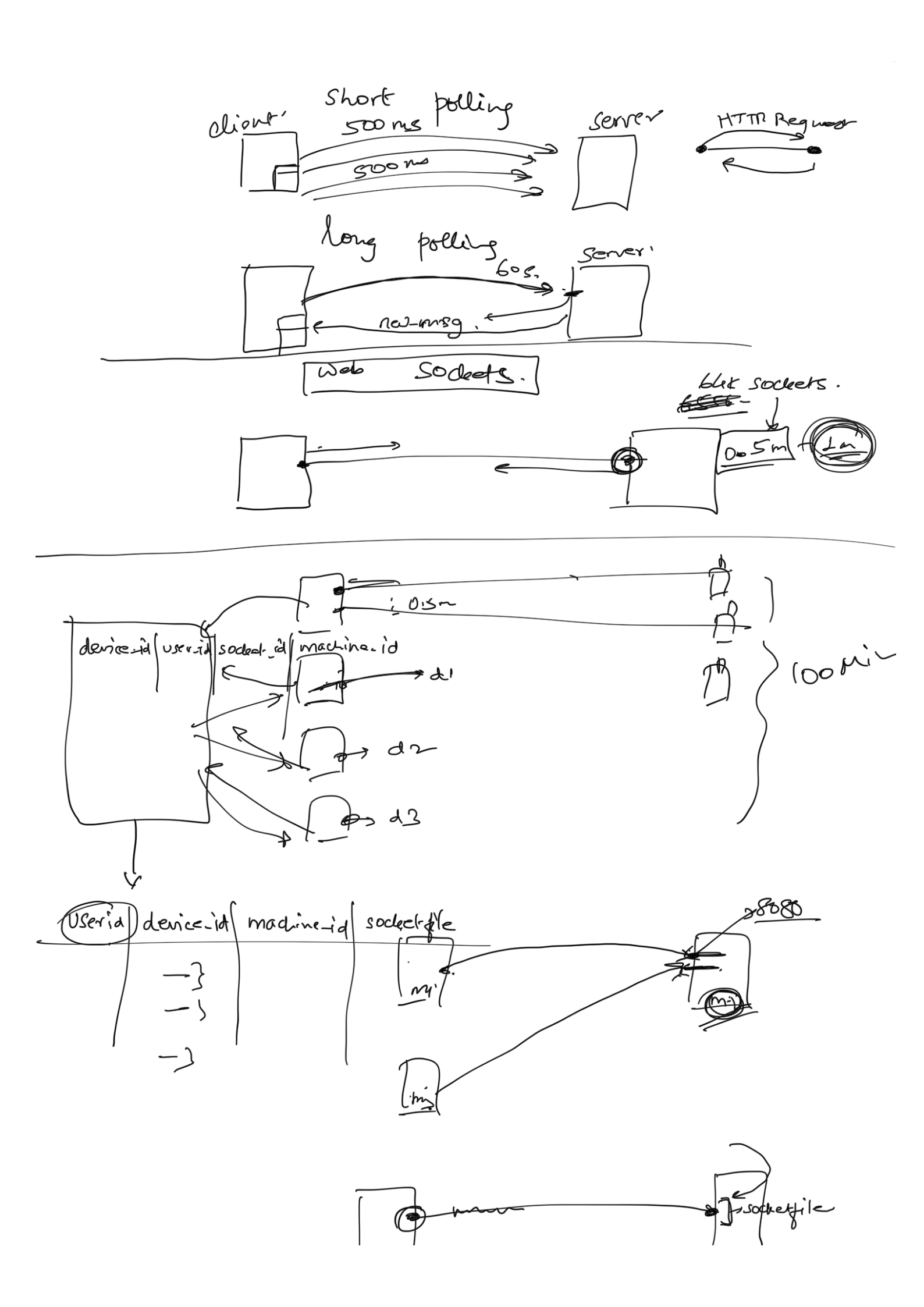
⑤ Storage recent_notification → status

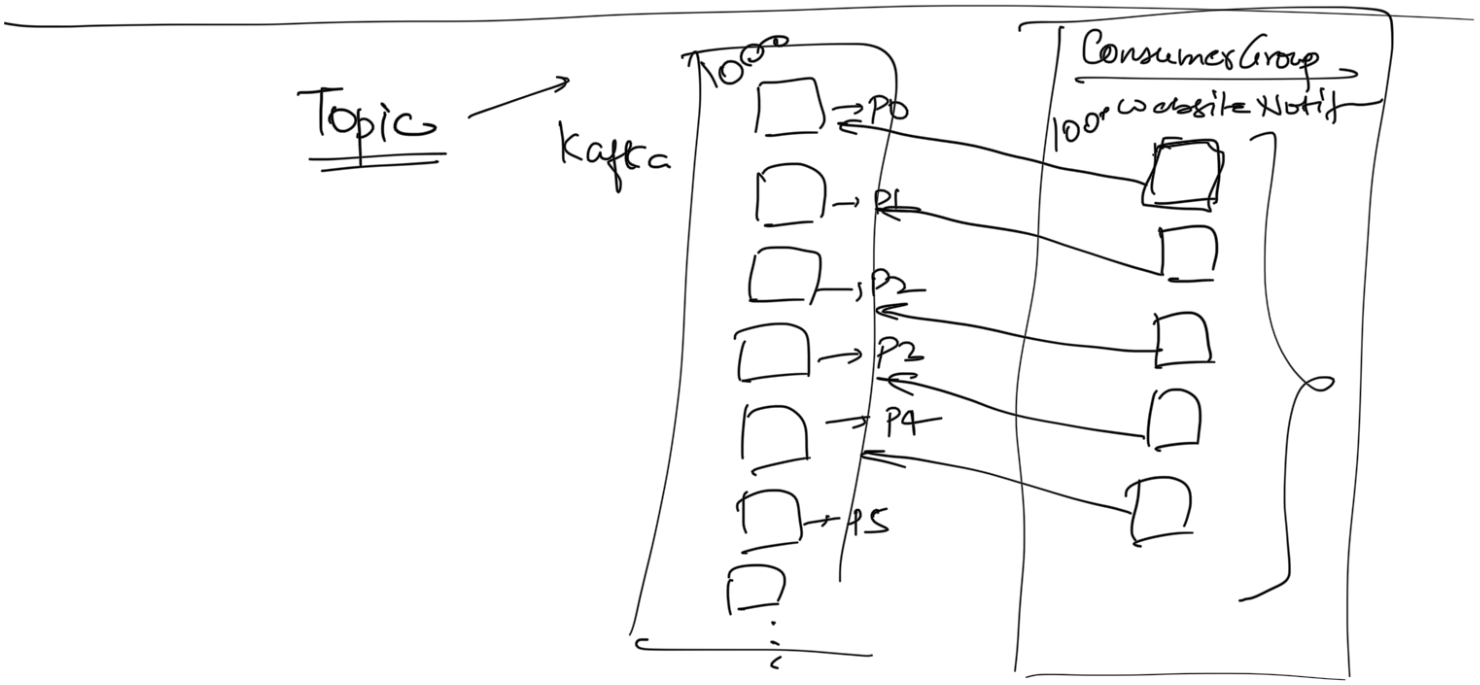
⑥ Publish some event to let other services know notifica

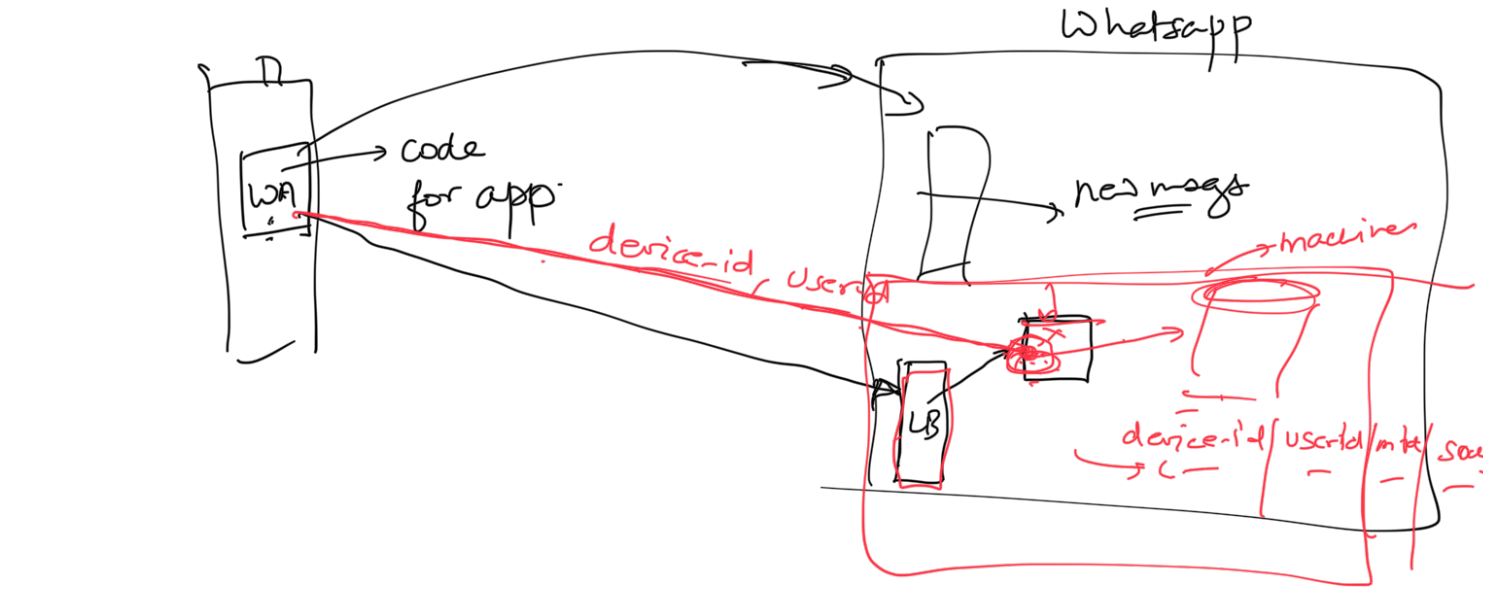
⑦ Delete old notifica from recent_notifica on daily basis.

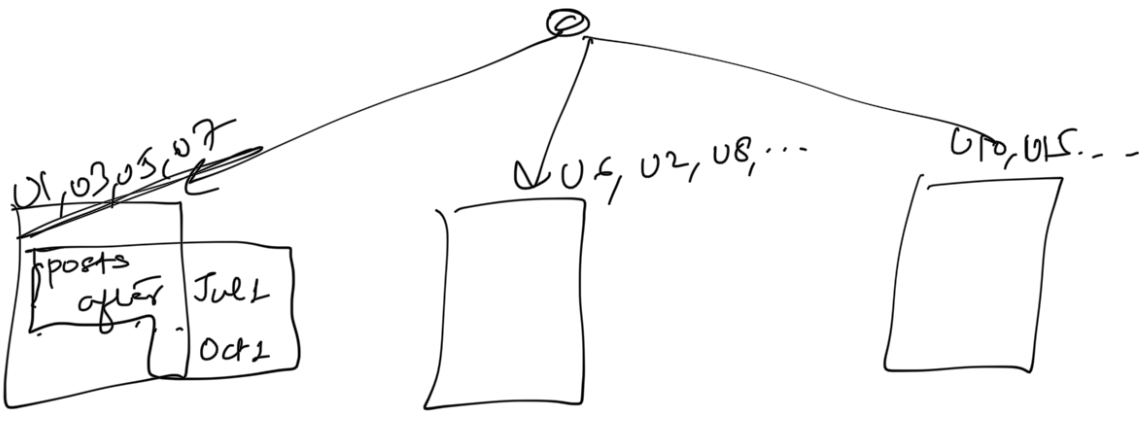



another topic
notification failed
publish to kafka
Diff Topic for diff. priority.
Kafka
P0
P1
P2
Redis
duplication
10 items from P0
1 item from P1
1 item from P2
notification_data
verification
failed event publish
valid
rate limiting
duplication
notifica
Medium
mediun
User Preference
MySQL
user_id
userinfo
text → medium
Email Topic
Kafka
SMS
PN
IVRS → service
Email Service
SMS Service
email
PN Service
device id
machine → sad
text, userid, email
Topic
m1
m2
m3
m4
m5
SMS
Consumer 1
index: 2
Topic
Publisher
Consumer 2
index: 6


Short polling
client
500 ms
500 ms
server
HTTP Request
long polling
60s.
server
new-msg.
Web Sockets.
Web sockets.
0.5m
1m
device_id
user_id
socket_id
machine_id
d1
d2
d3
0.5m
100 Mil
Userid
device_id
machine_id
socketfile
m1
m2
m3
8080
socketfile


Topic
kafka
100
P0
P1
P2
P3
P4
P5
Consumer Group
100 website Notif


Whatsapp
code for app
device-id, User
new msgs
inactive
LB


U1, U3, U5, U7
U4, U2, U8, ...
U10, U15...
sports
after
Jul 1
Oct 1

Partitioning ⇄ sharding