Agenda

1. DNS
2. Load Balancing
3. Sharding
   - Consistent Hashing

Case Study

Del.icio.us

→ I/P

→ Static V/S Public

DNS

Overview

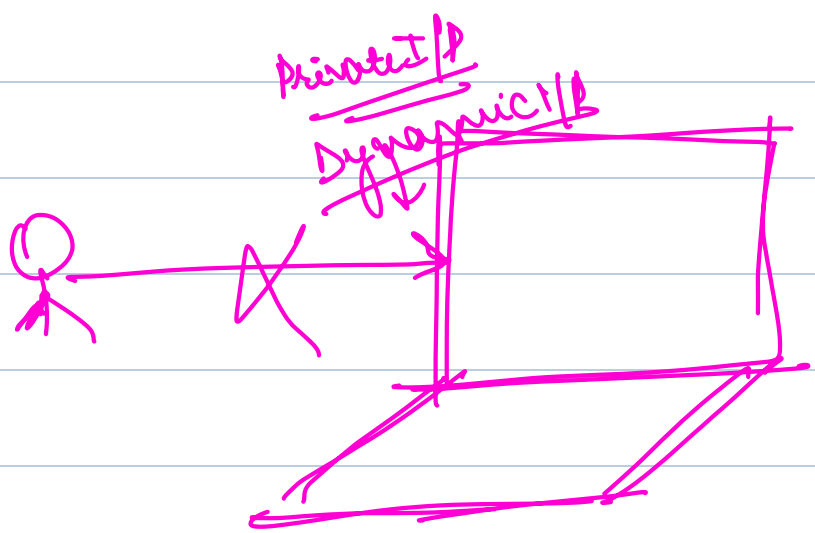
Del.icio.us

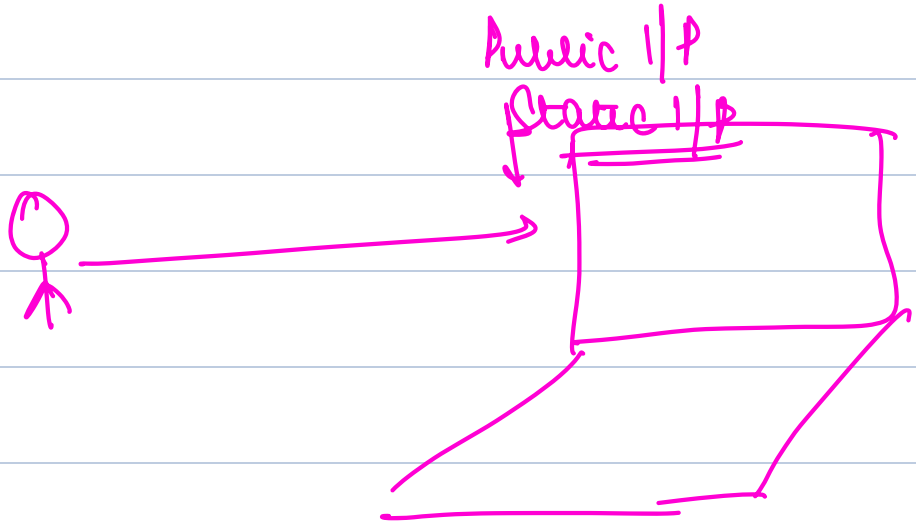
Bookmark Storage Website

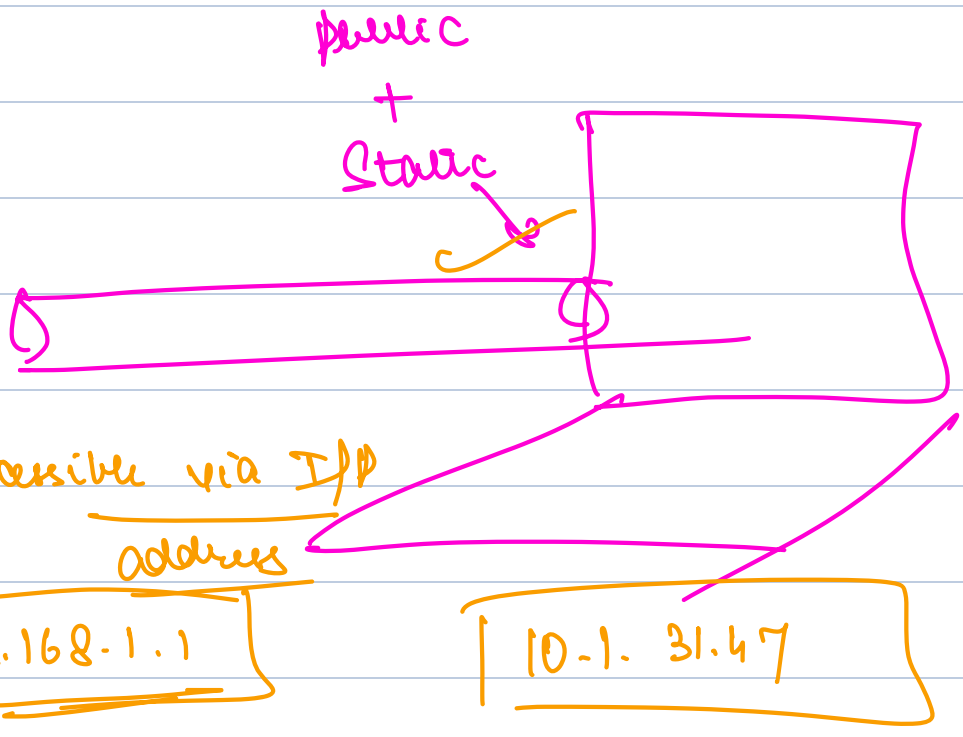
1.) Save Bookmark(url, title)

2.) fetch BOOKMARKS()

3.) login()





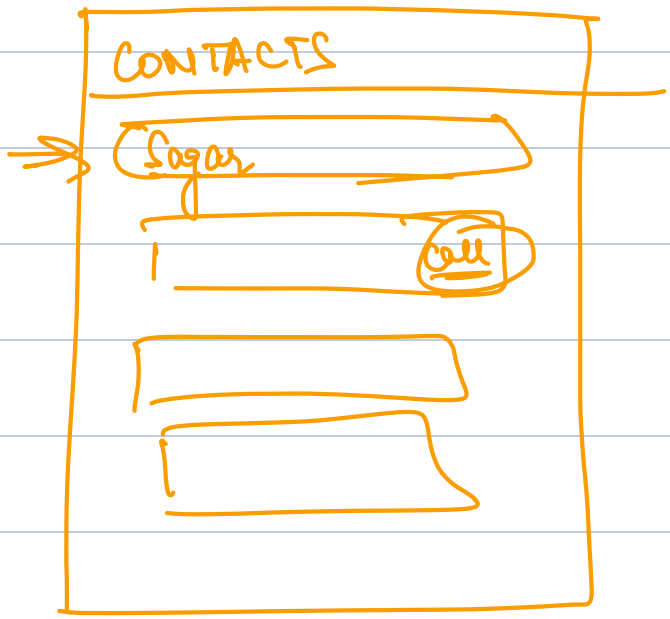


google.com	11.47.72.81
fb.com	10.1.31.47
instagram.com	192.168.1.1

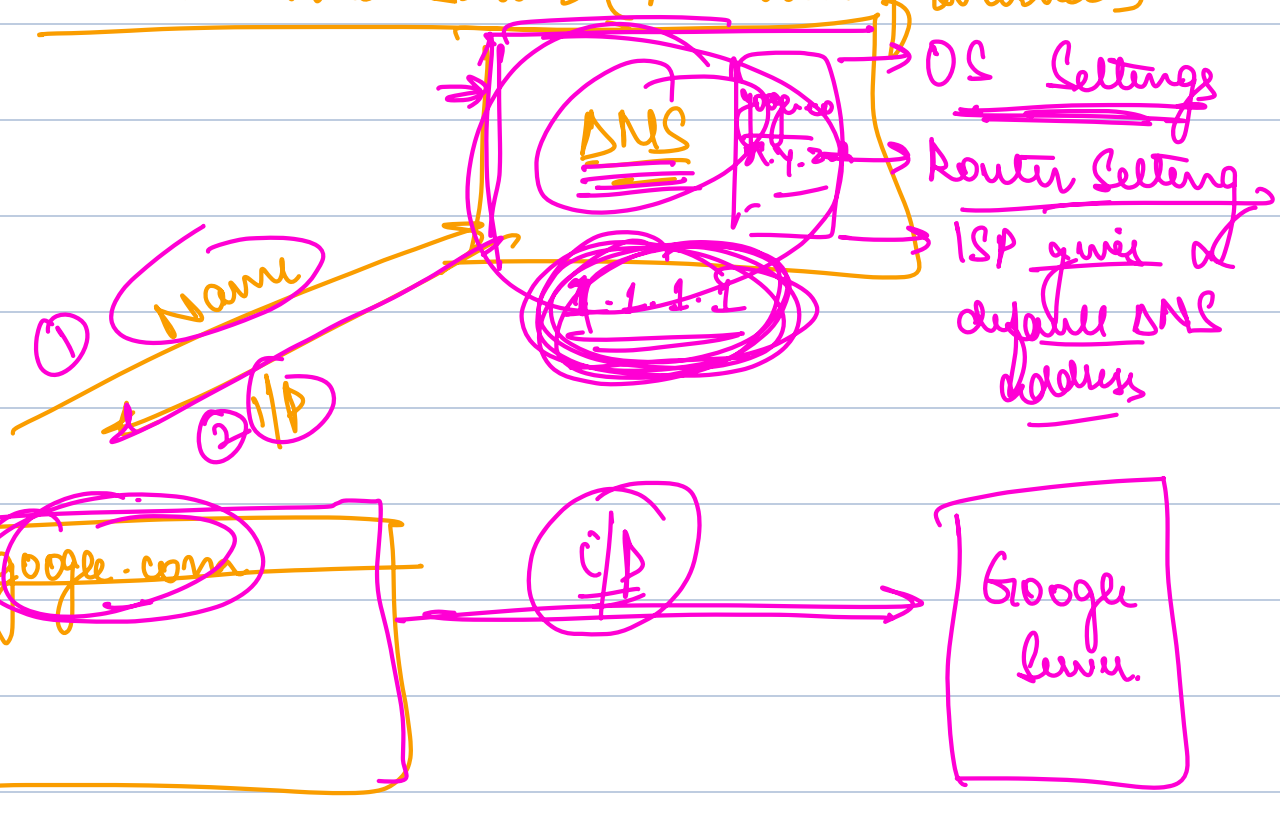
Phone numbers

9996203771

9aa50869



DNS: Domain Name Server (Phonebook of Internet)



1.1.1.1 → Cloudflare

8.8.8.8 → Google.

8.8.8.8

1.1.1.1

How DNS Works

Buy a domain from Broker

- GoDaddy
- Name Cheap
- ParkBun
- Route 53

naman.dev

↳ free Name Server.

↳ Map

↳ Subdomain : IP

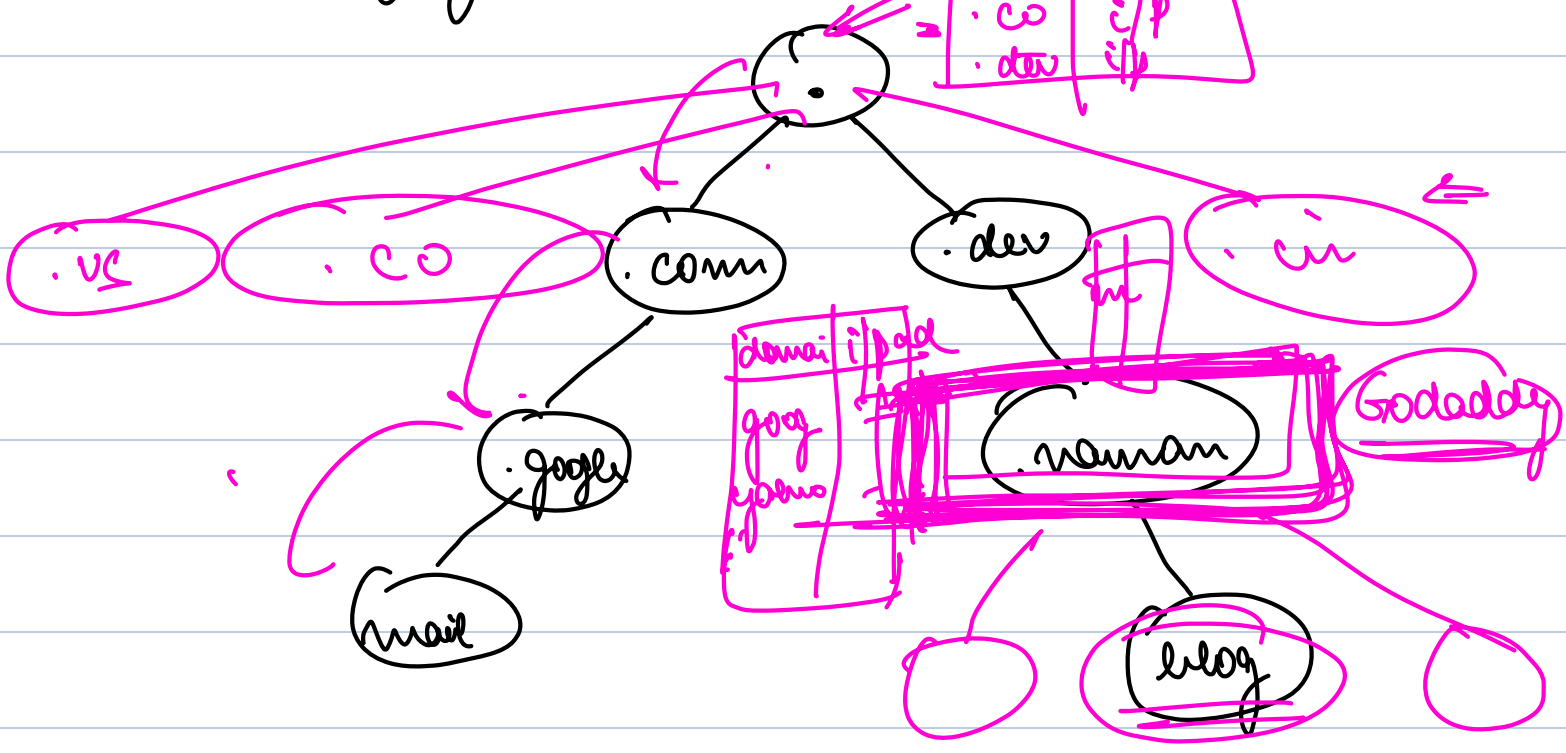
Porkum Dashboard

naman.dev	11.21.51.71
blog.naman.dev	7.8.9.10

14 Root Name Servers

blog.naman.dev
naman.dev
mail.google.com

TLD	IP of NS of TLD
.com	IP
.in	IP
.co	IP
.dev	IP



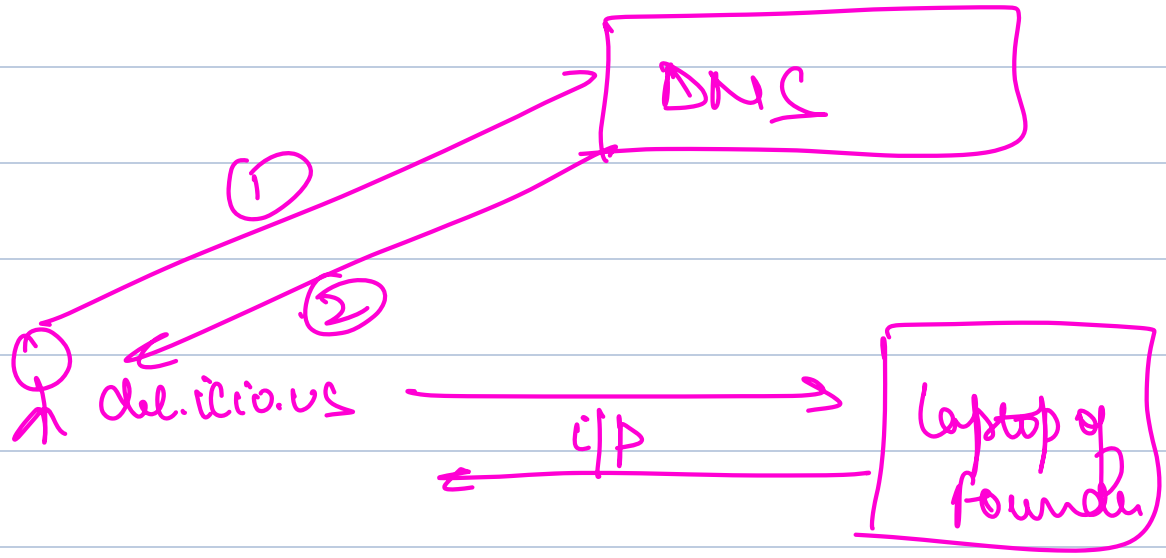
domain	IP of
google	

· guru

1. Public + Static
2. Bought domain Name
   - ↳ free NS
   - ↳ some other NS
3. In NS they put I/P add of dev.io.us

DNS

domain Name	I/P
blog.maman.dev	10.11.12.13



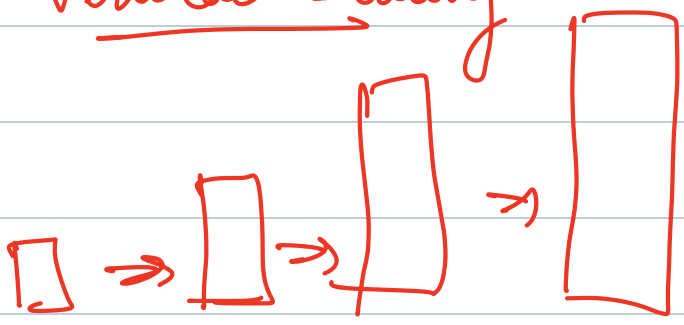
will 1 laptop be able to handle work

→ NO

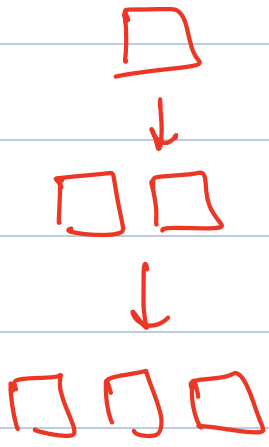
1. get a better laptop

2. get More laptops

Vertical Scaling



Horizontal Scaling



Vertical Scaling

way to enable your infra to handle higher load by increasing the capacity of 1 Machine

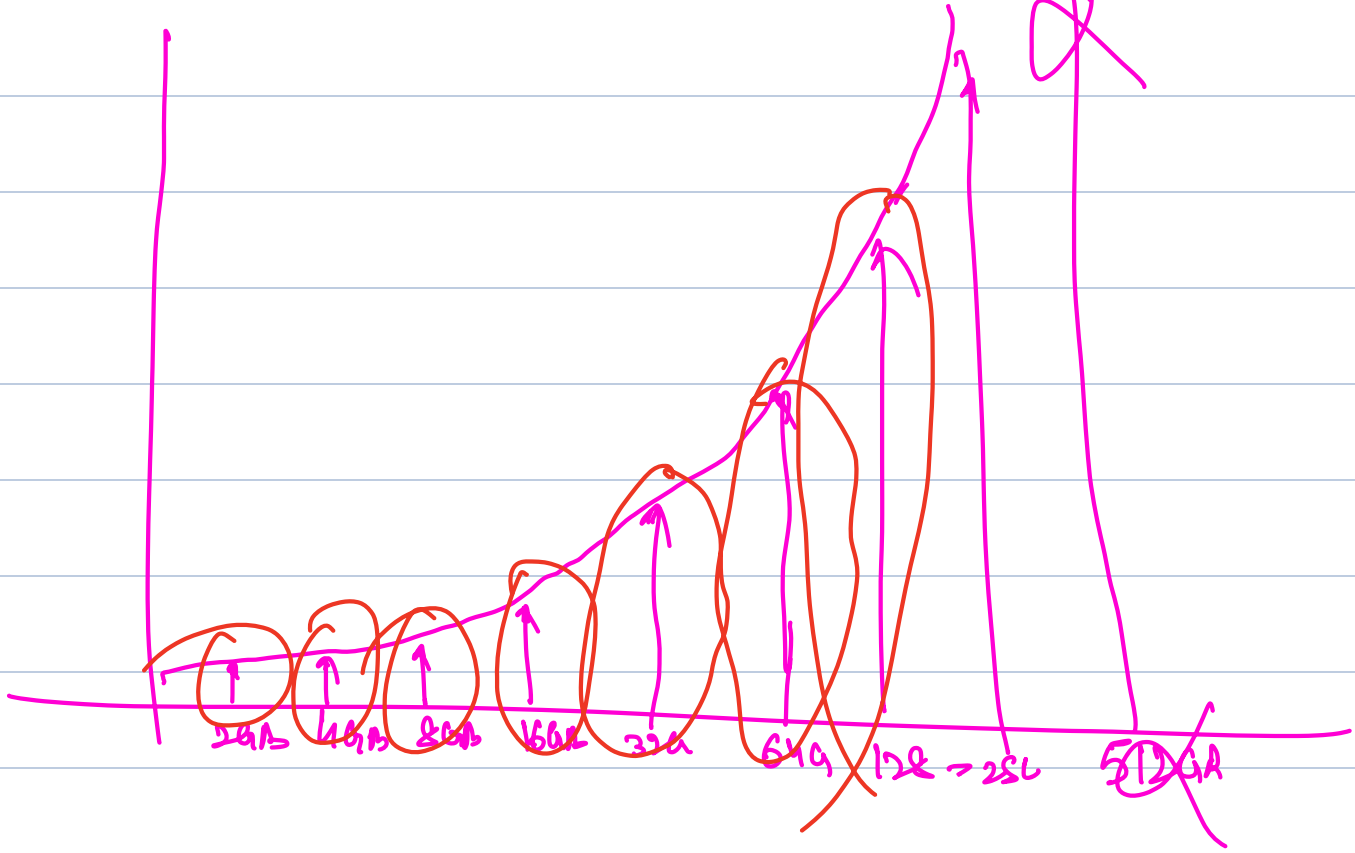
2GB RAM ⇒ 4GB RAM ⇒ 8GB RAM

Pro:

1. Simpler setup

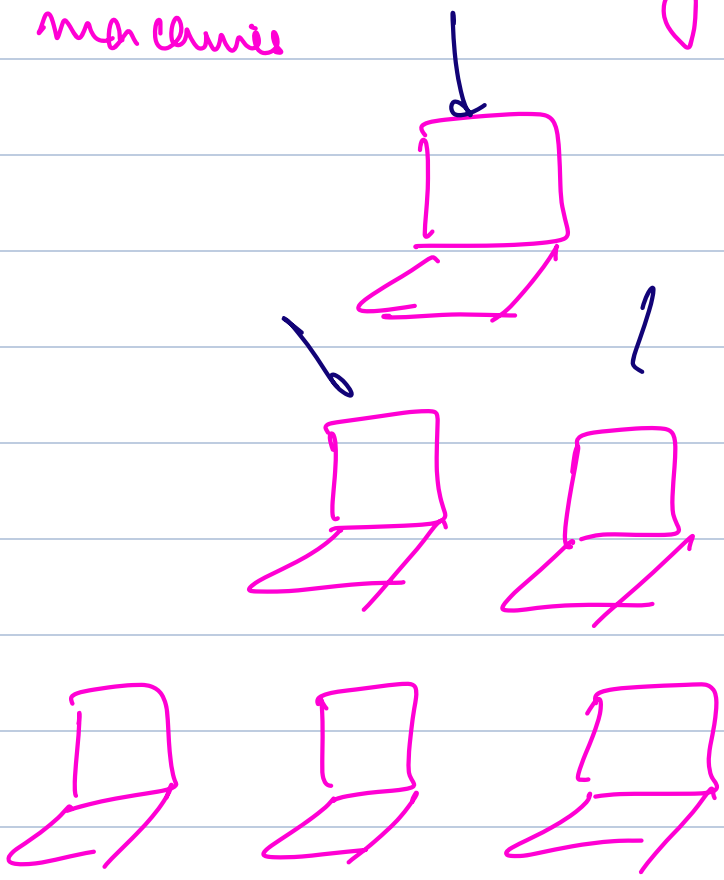
Cons:

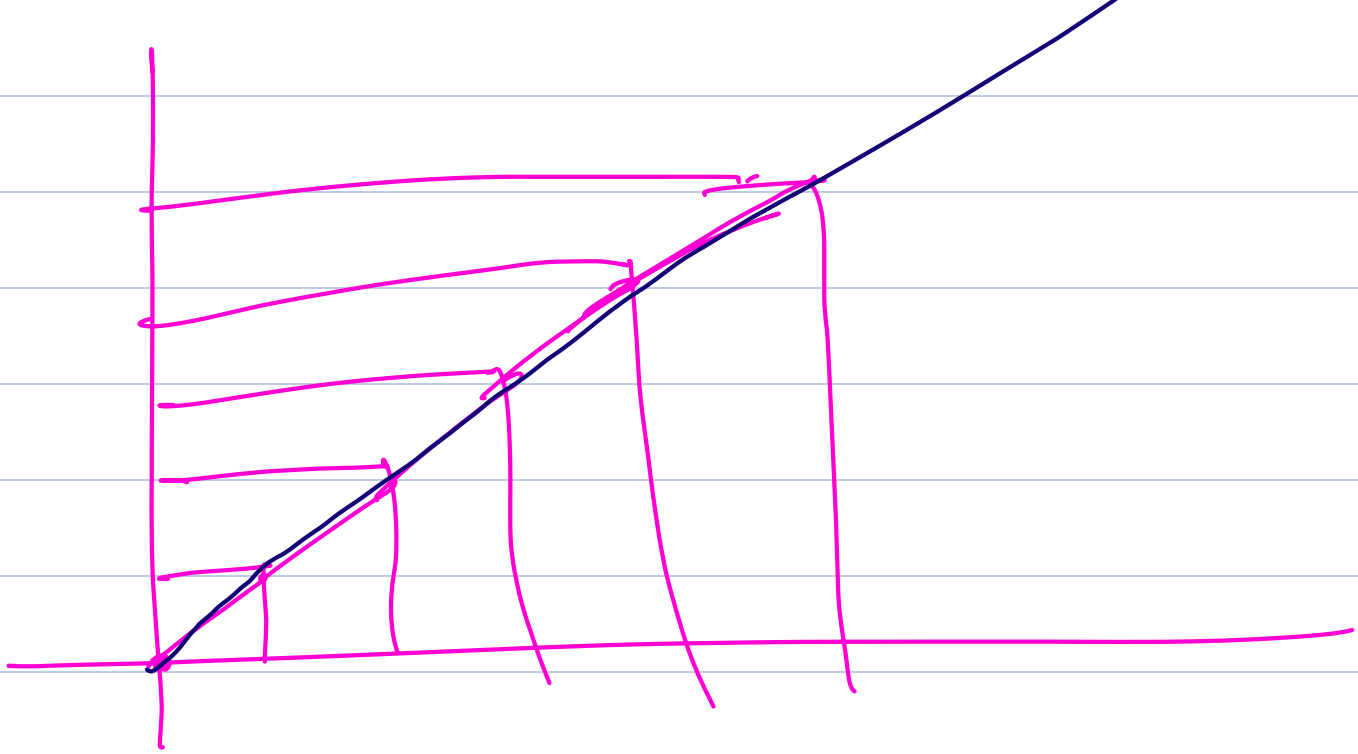
1. Single Point of Failure
2. There is a practical limit
   - ↳ can't scale infinitely
   - ↳ can't buy a machine bigger than a limit
3. Cost
4. updating app/OS ⇒ Downtime



Horizontal Scaling

→ method to enable a system to handle more users by getting more machines





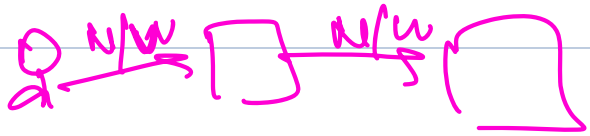
Pro

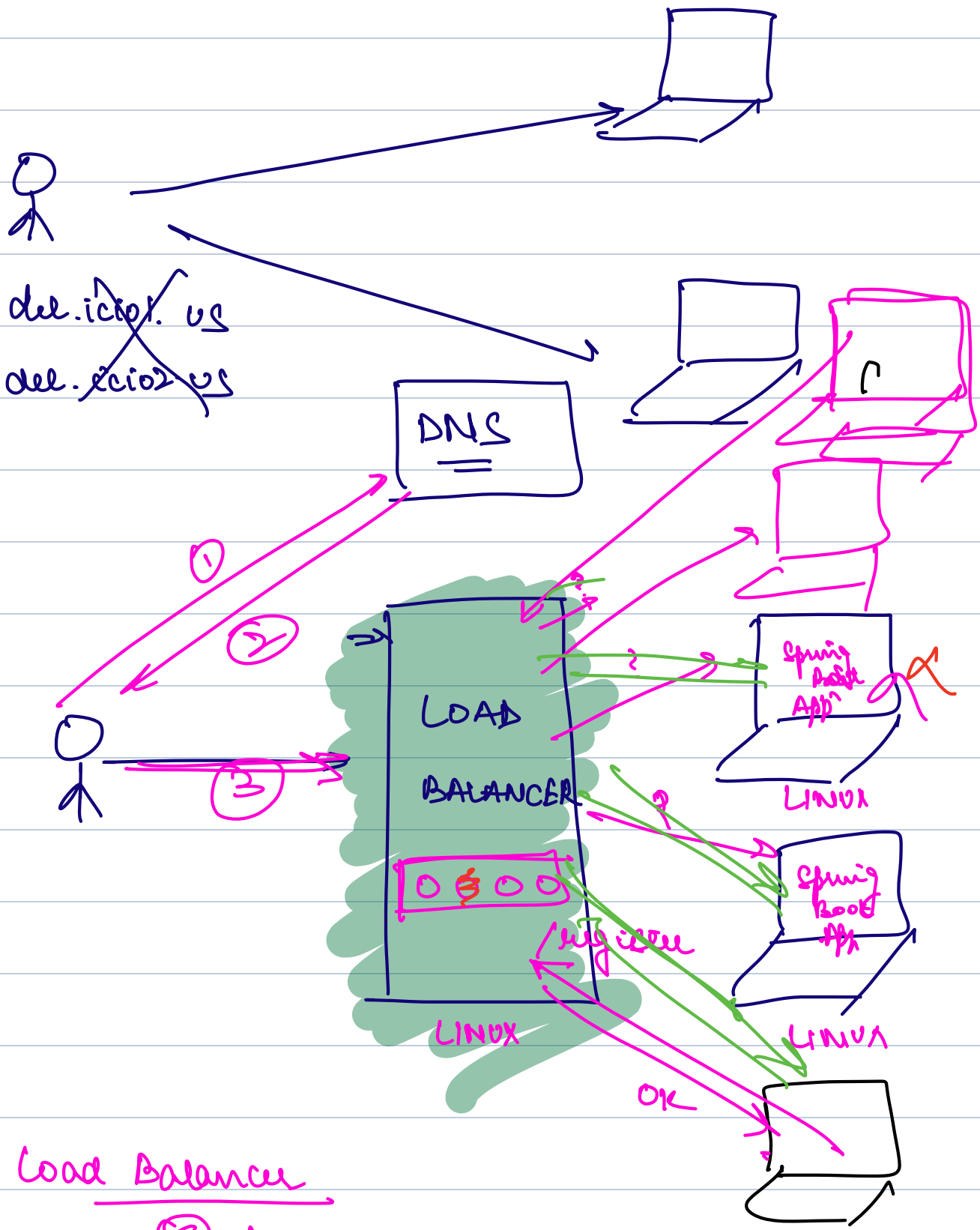
1.) Infinitely Scalable.
2.) Costs increase linearly
3.) Geographical better experience
4.) A lot of redundancy

Con

1.) It makes infra complex

2.) All issue with distributed system.
   a) Data is consistent amongst all
   b) Higher latencies





Load Balancer

① How does it get to know about a server

② How does it forward a req to

server

(3) How does a LB know that a server has died.

→ How does a LB get to know a server

code of server

→ at startup server sends a req to load balancer to reg itself.

How does LB know server is alive or not

(1) Health Check

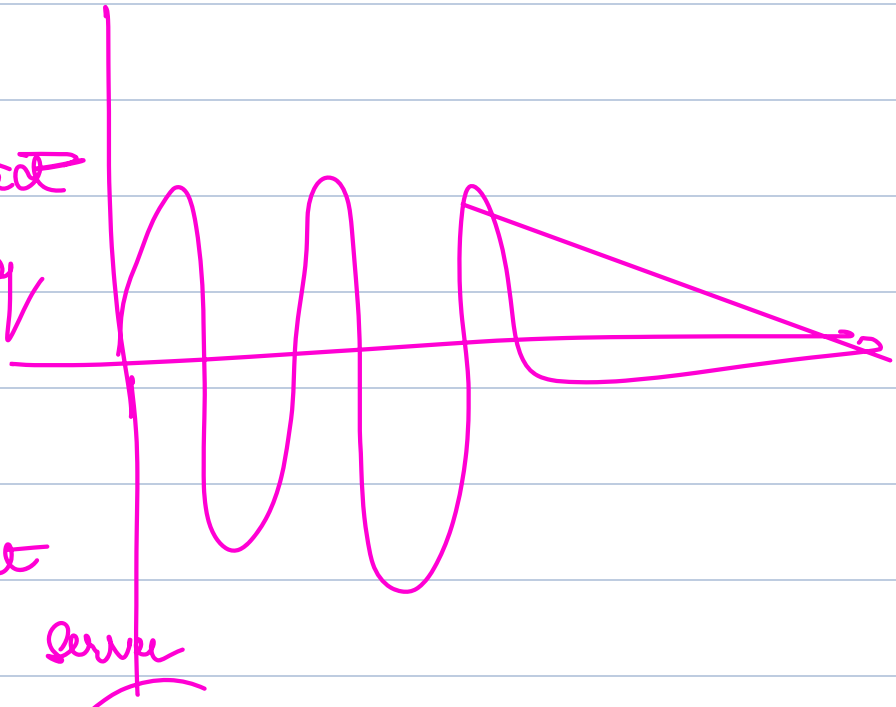
→ LB keeps pinging the server every X sec.

→ If server doesn't respond for 3 consecutive instances, LB assumes server has died and removes it from its list

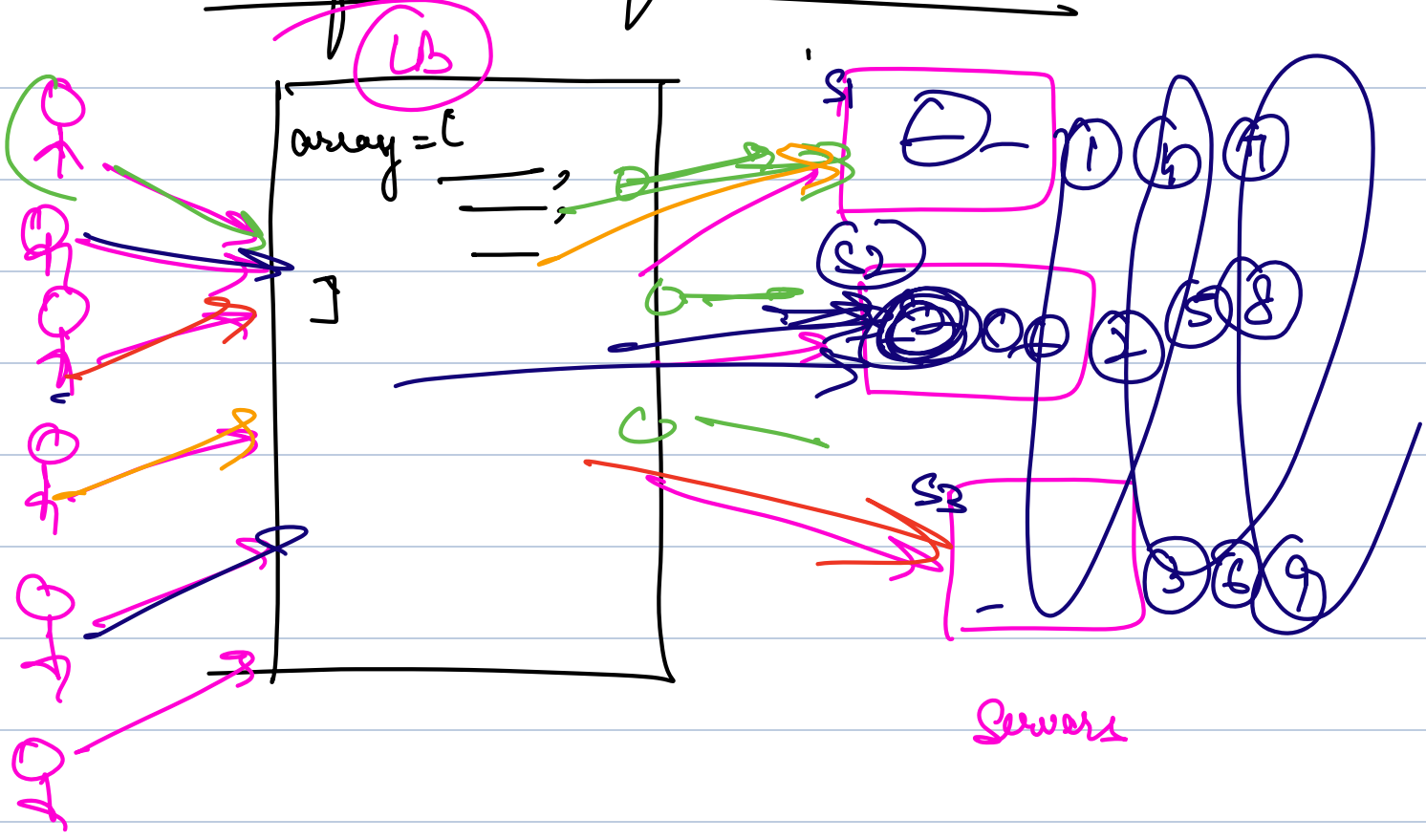
② HeartBeat

Server is expected to send a req to LB every x sec.

If LB doesn't get any req from server for 3*x seconds, LB assumes you are dead and removes it



How LB forwards req to server



Round Robin Algo

array = [, , , ,]

prev = 0 1 2 3 4

prev += 1

+= 1

+= 1

+= 1

+= 1

+= 1 ⇒

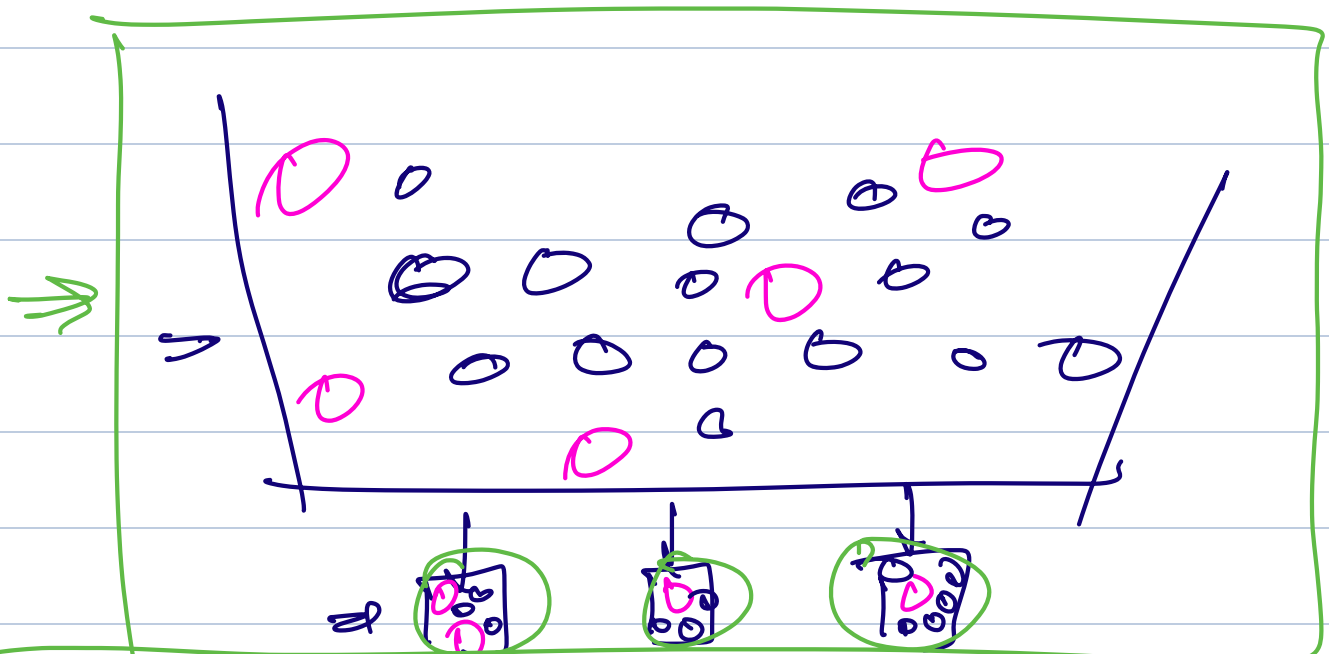
⇒

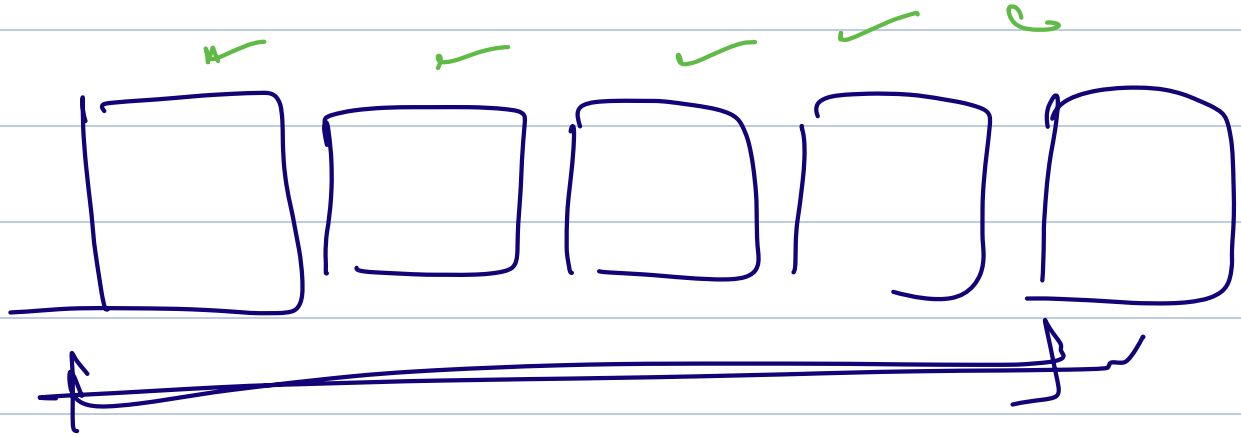
prev += 1

prev % = server_count

forwardTo(array[prev])

⇒ if all servers are similar spec





1 4 7 10

2 5 8 11

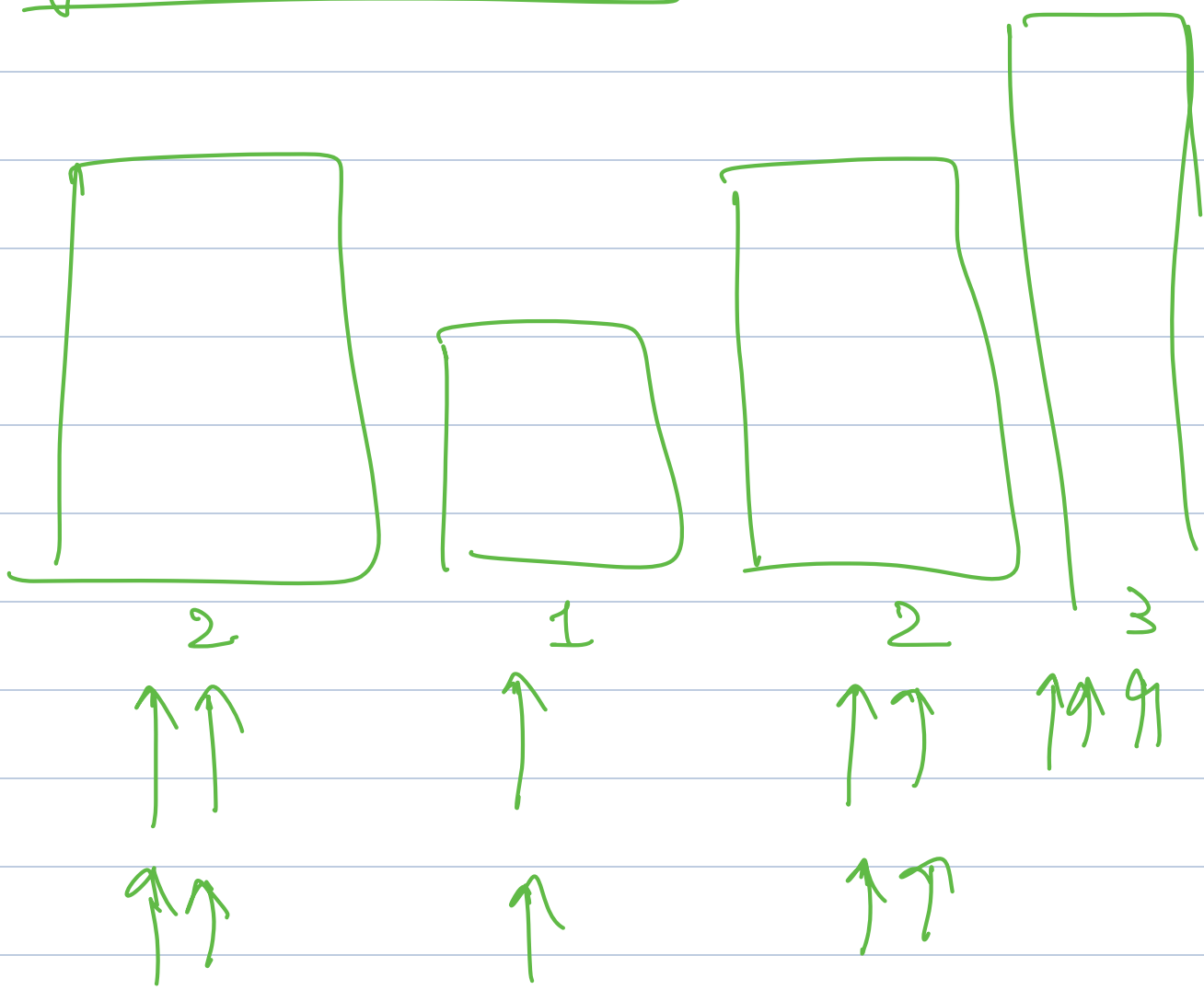
3 6 9 12

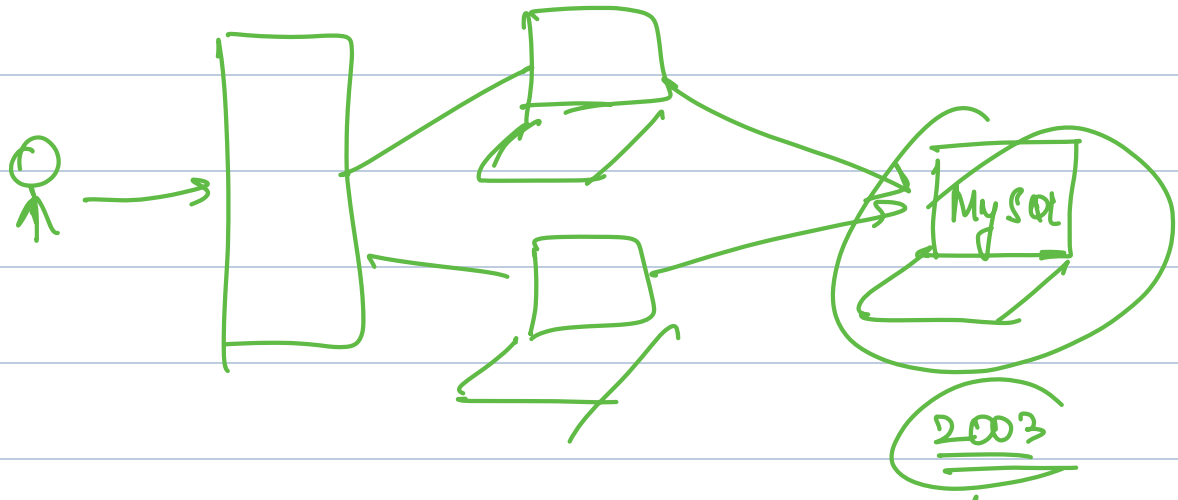
13.

KISS

Keep it Simple Stupid

Weighted Round Robin





2003

40 GB size of Disk

bookmarks

id (8B)	url (30B)	title (30B)	user-id (8B)

≈ 100B/Bookm

$$\frac{40 \times 10^9}{100}$$

$\rightarrow 4 \times 10^8 \rightarrow$ 400 M Bookmarks

≈ 200 M Bookmarks

100 M users

≈ 20 bookm

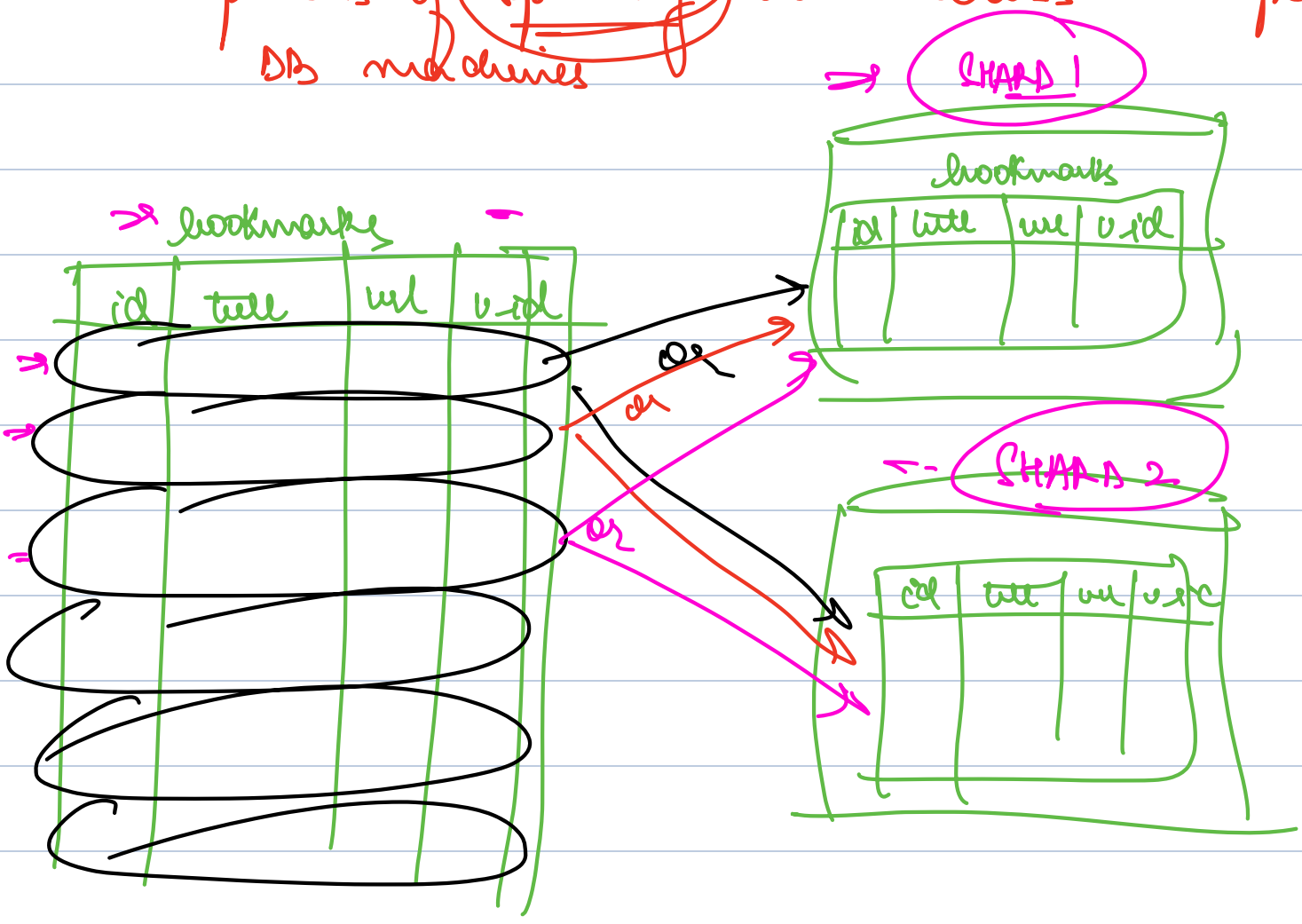
10 Machines to store data

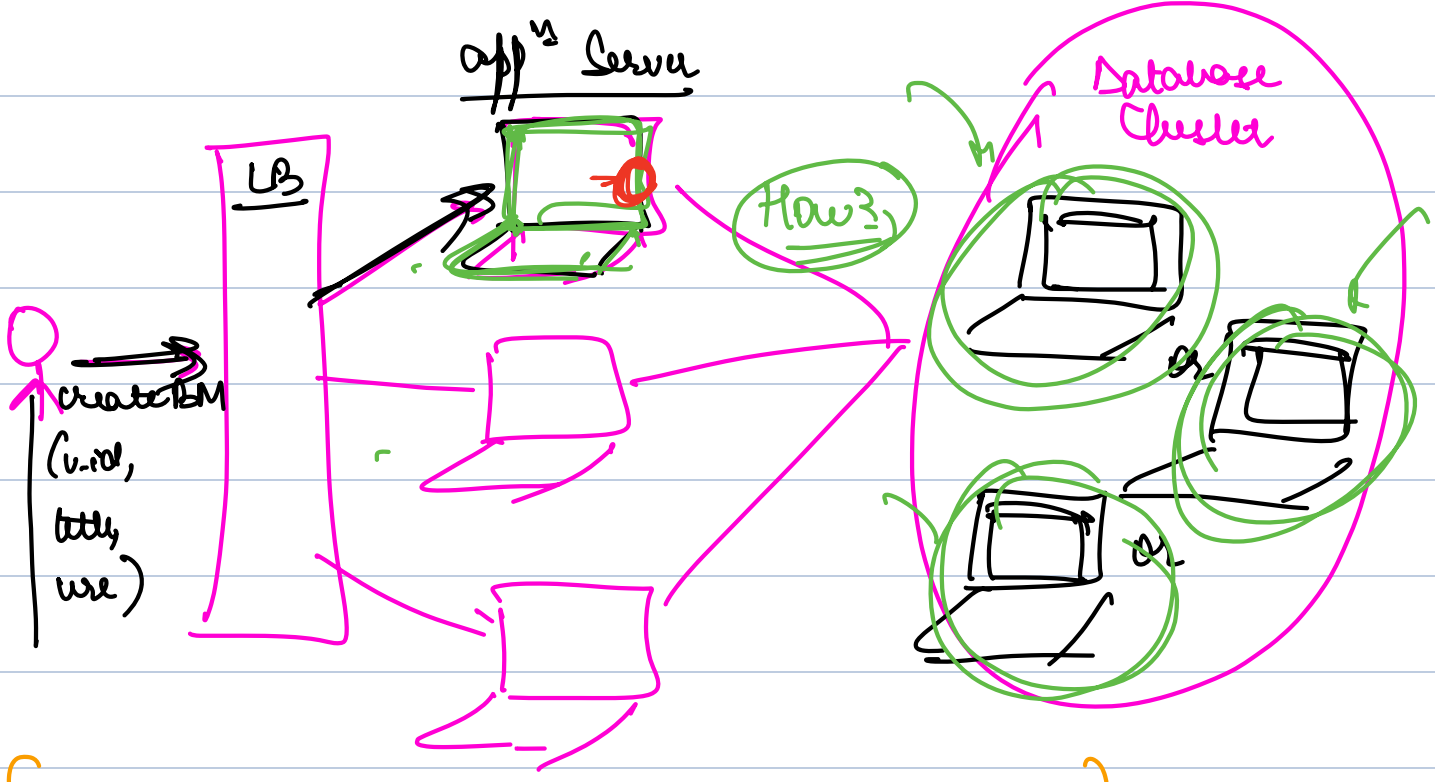
⇒ Now, data isn't fitting in 1 Machine

⇒ we will have to split data amongst multiple machines,

SHARDING

process of splitting (dividing) data across multiple DB machines

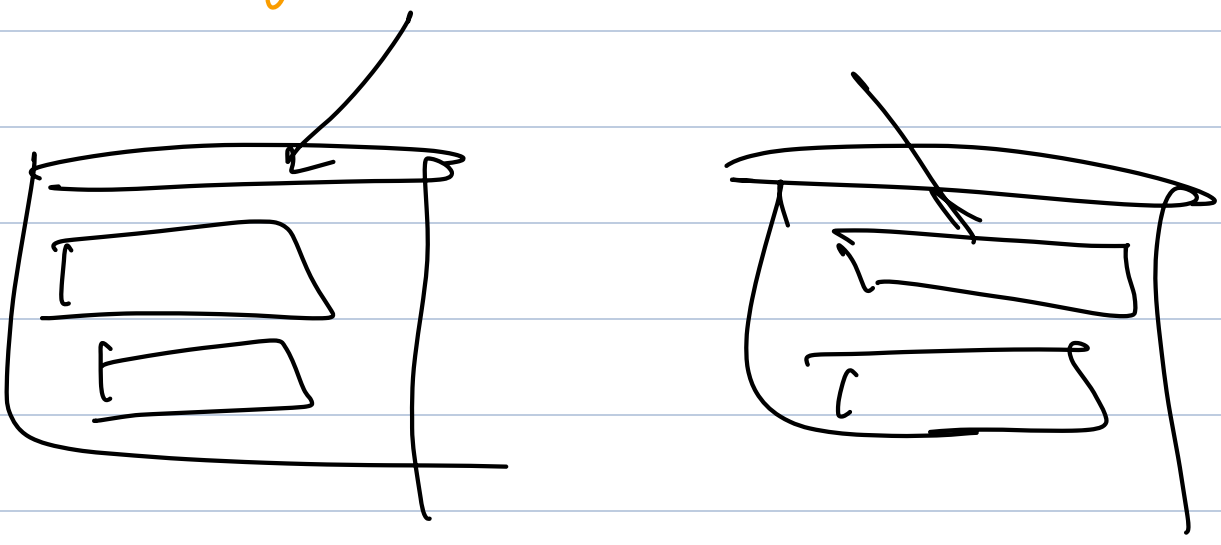




How you divide data should be basis what makes my queries fast

saveBookmark(u_id, url, title) 1

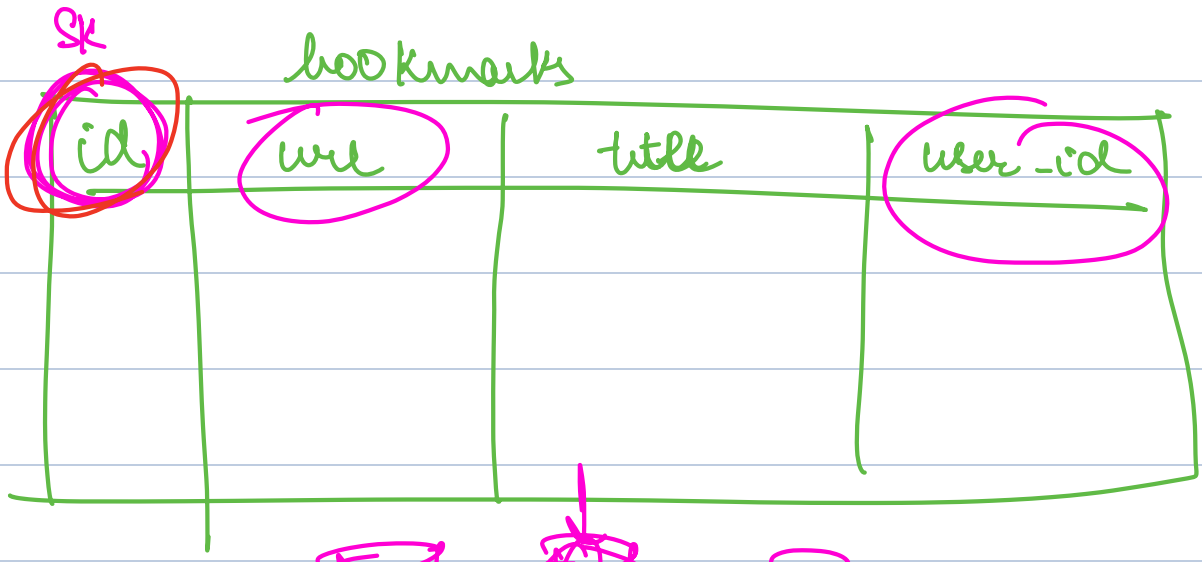
getBookmarks(u_id) ALL



Random → X

SHARDING KEY

attribute of the table based on which you decide where a particular entity should go



Machine ⇐ getShard(url)

Next Class

select count(*) from bookm where url = ( )

⇐ getShard(url)

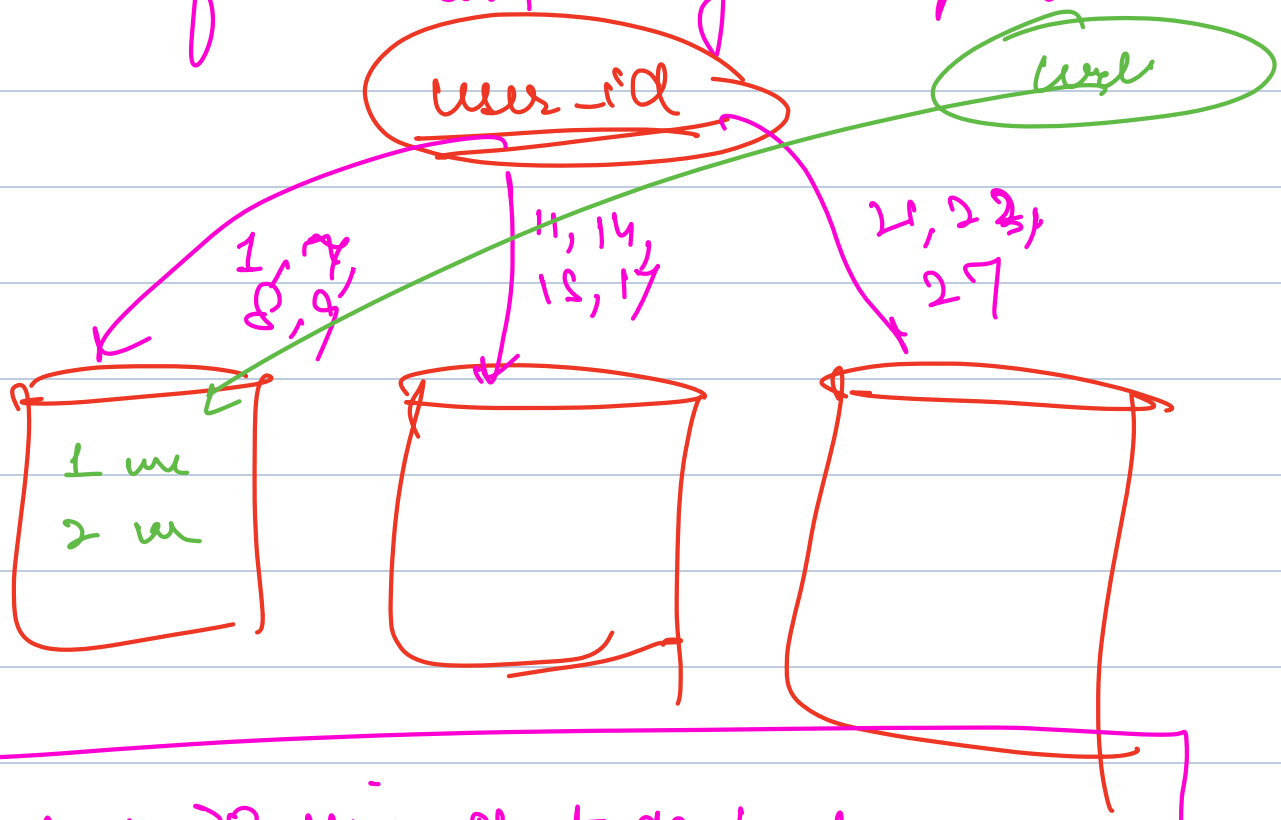
Store Bookmark()

fetch Bookmark(u_id)

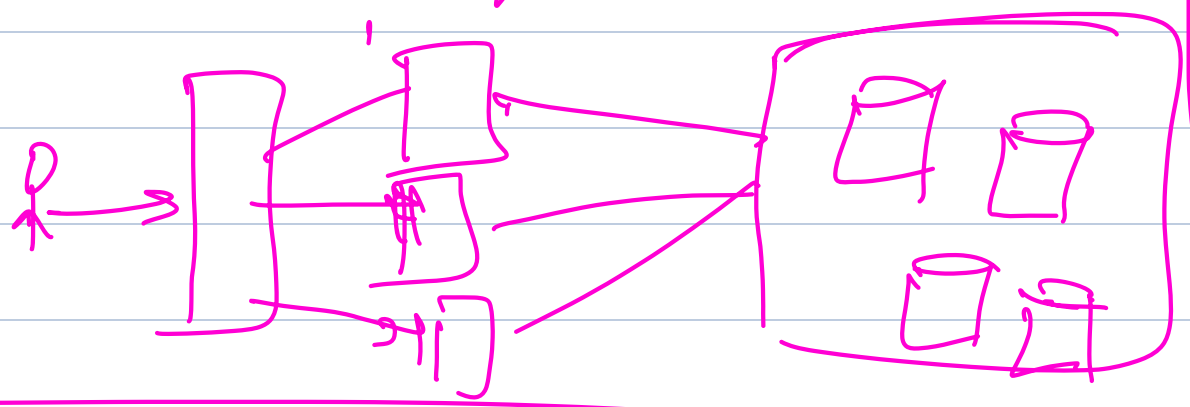
count of Book(url)

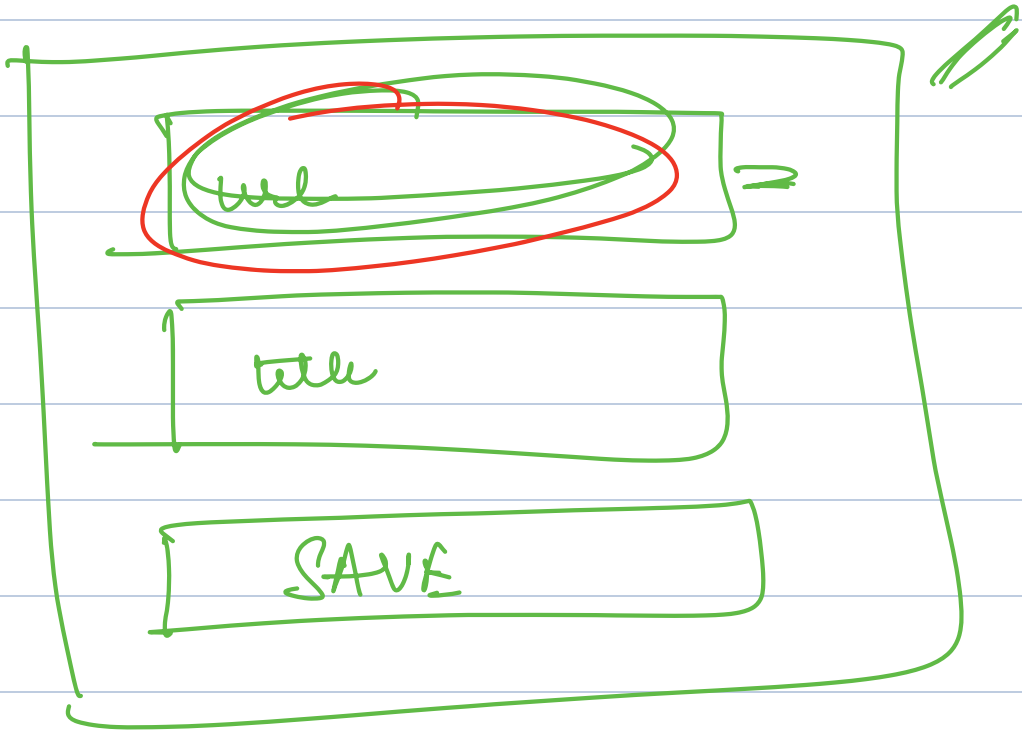
Randomly	B_id	url	title	U_id
1	1	1	1	1
ALL	ALL	ALL	ALL	1
ALL	ALL	1	ALL	ALL

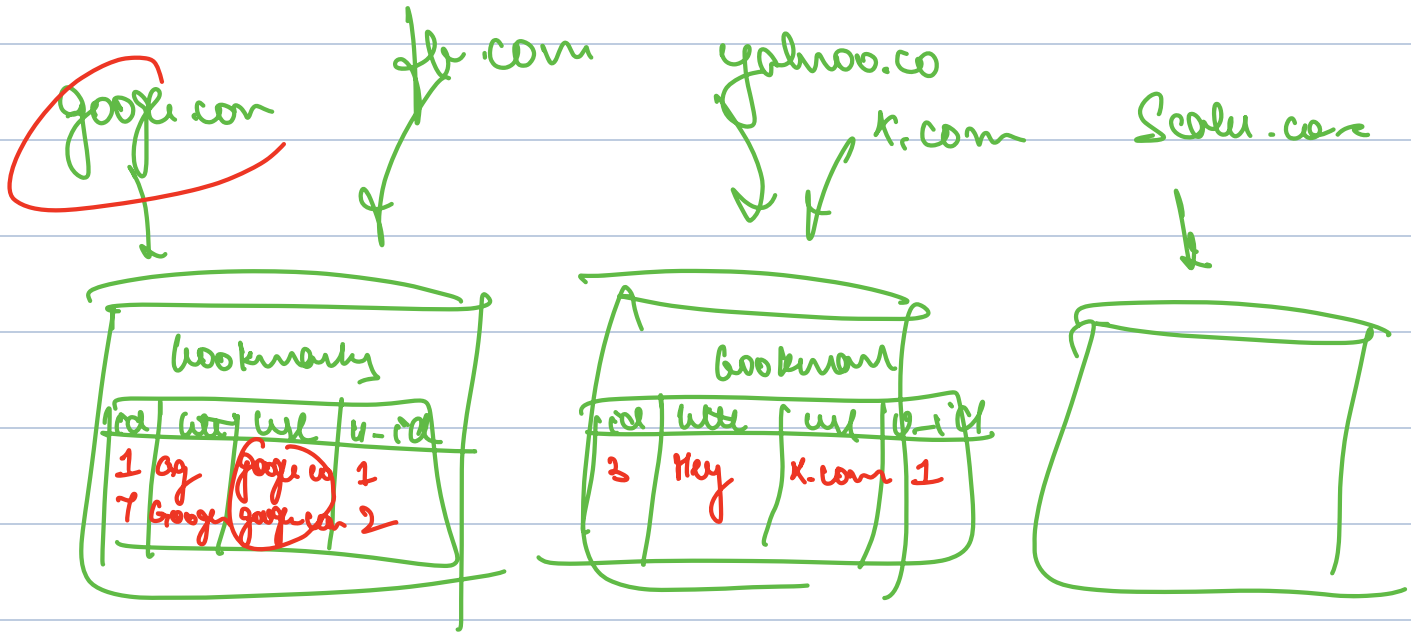
facebook.com → My Home Page xo xo



Last 30 Min of today's class







Get the count of bookmarks created of a URL

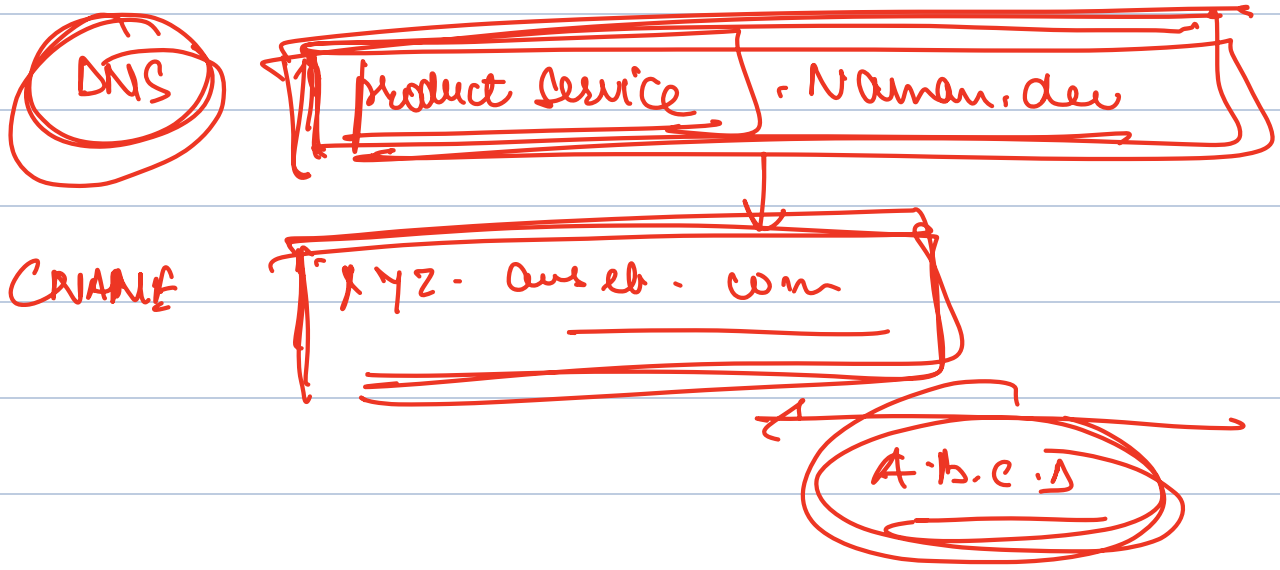
Order by rank, name

getShard (sharding-key) {

Also

}

AWS Elastic Bean Stalk Curls



Vercel