AGENDA

1:> what is Backend ✓

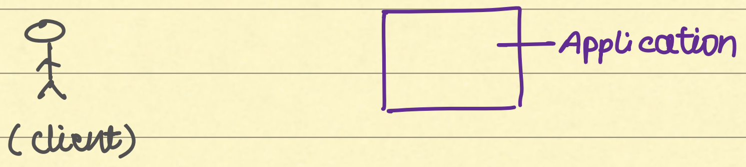
2:> Module Expectations •

3:> DOs & DONT's •

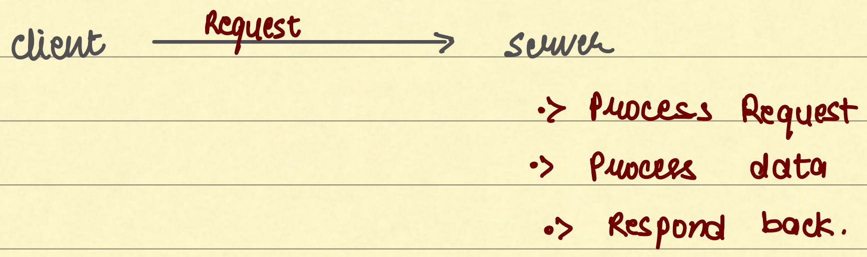
4:> curriculum •

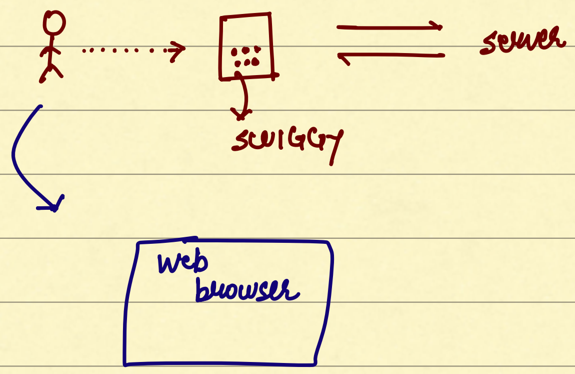
5:> Introduction to VCS •

start by 7:10 AM IST



① client
② server





CLIENT

website / App / smart device

SERVER

-> owns data
-> Machine which runs your Application
-> logic on servers can be vvv complex.

Amazon.com
↳ BE server

-> can be many machines

# BACKEND: Everything w.r.t servers

EXPECTATIONS FROM MODULE:

1·> Why BE ?

To have Good tech. Complex Projects
To have hands dirty with some latest tech
To be Ready from day-1

Learn About best Practices Accross Industry
learn About common terminologies
JWT / Auth / How to Connect to MICROSERVICES

DO'S AND DONT'S

DO's

1·> How Real world development looks like
- How to Google search
- How to use stack overflow / chatGPT
- How to work with any Framework.

2·> Discuss about common tools / concepts
Kafka / Authentication / postman

3·> - Get Project worthy to be added in resume
4·> working with Advanced infra layers
kafka / Redis / MongoDB ...

# DONT'S

1·> NO spoonfeeding
2·> NO memorising

*) CURRICULUM :

4 Parts

Part-1: Foundational concepts
git / command line / dev project

Part-2: Project

- set up the Project
- basic foundational things - Implement APIs/ connecting to DB
- common features: Auth/ Paging/ Filters/ sorting/ unit test etc

Part-3: Advanced concepts
- kafka
- Redis
- elasticsearch
- mongoDB
- distributed tracing
- Implementing Payments

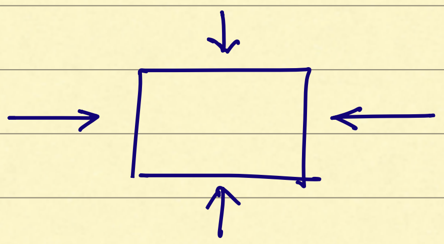
Part-4: Deployment
- AWS basics
- CI/CD
- docker setup / kube basics.

PROJECT: build E-comm platform.

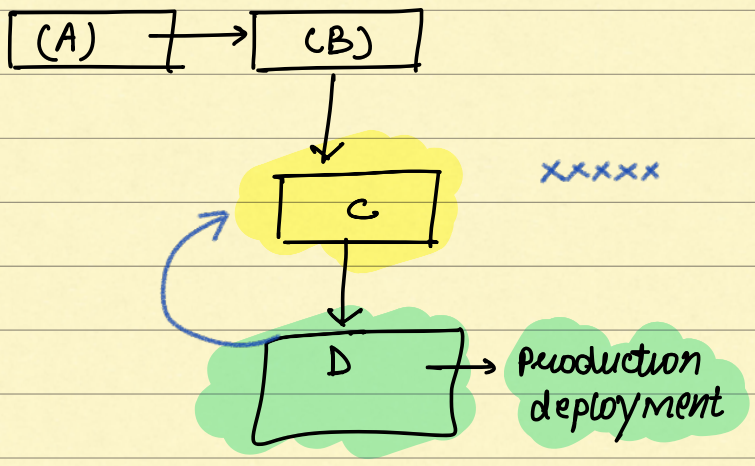
Git

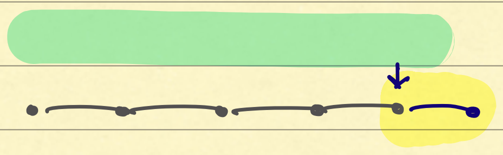
V.C.S

Version Control System



Eg: scaler





Reasons for keeping History of CodeBase:

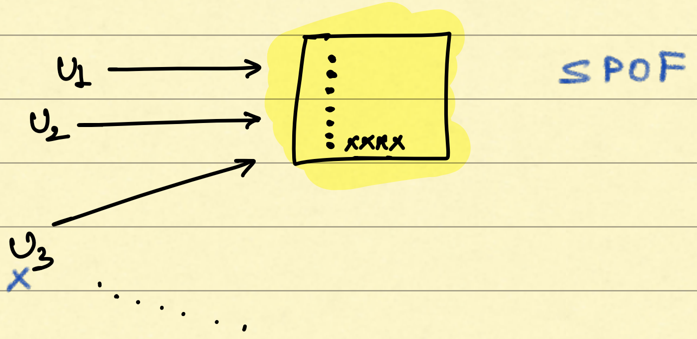
1.) Revert Easily
2.) In order to Review with latest changes
3.) To track changes

$$\begin{tikzcd}
& \text{VCS} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{CENTRALIZED} & & \text{DISTRIBUTED}
\end{tikzcd}$$

CENTRALIZED:

Everything is maintained on central server

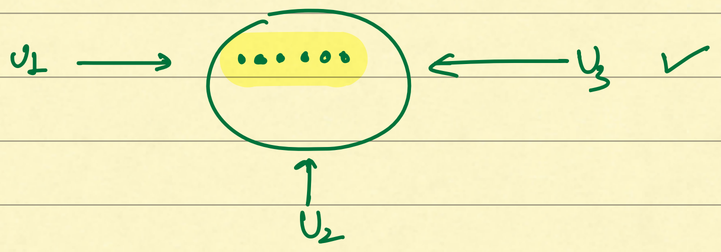
eg: Google Docs. / SVN



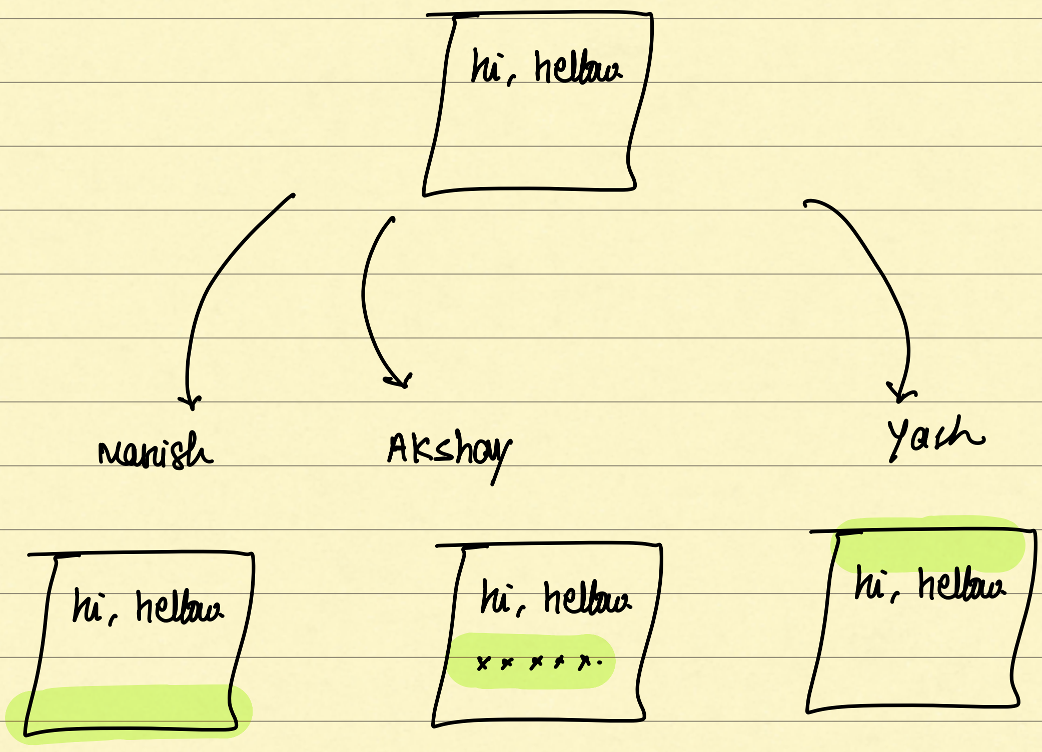
•> Complete history is stored on single server
•> You need to be ONLINE
•>

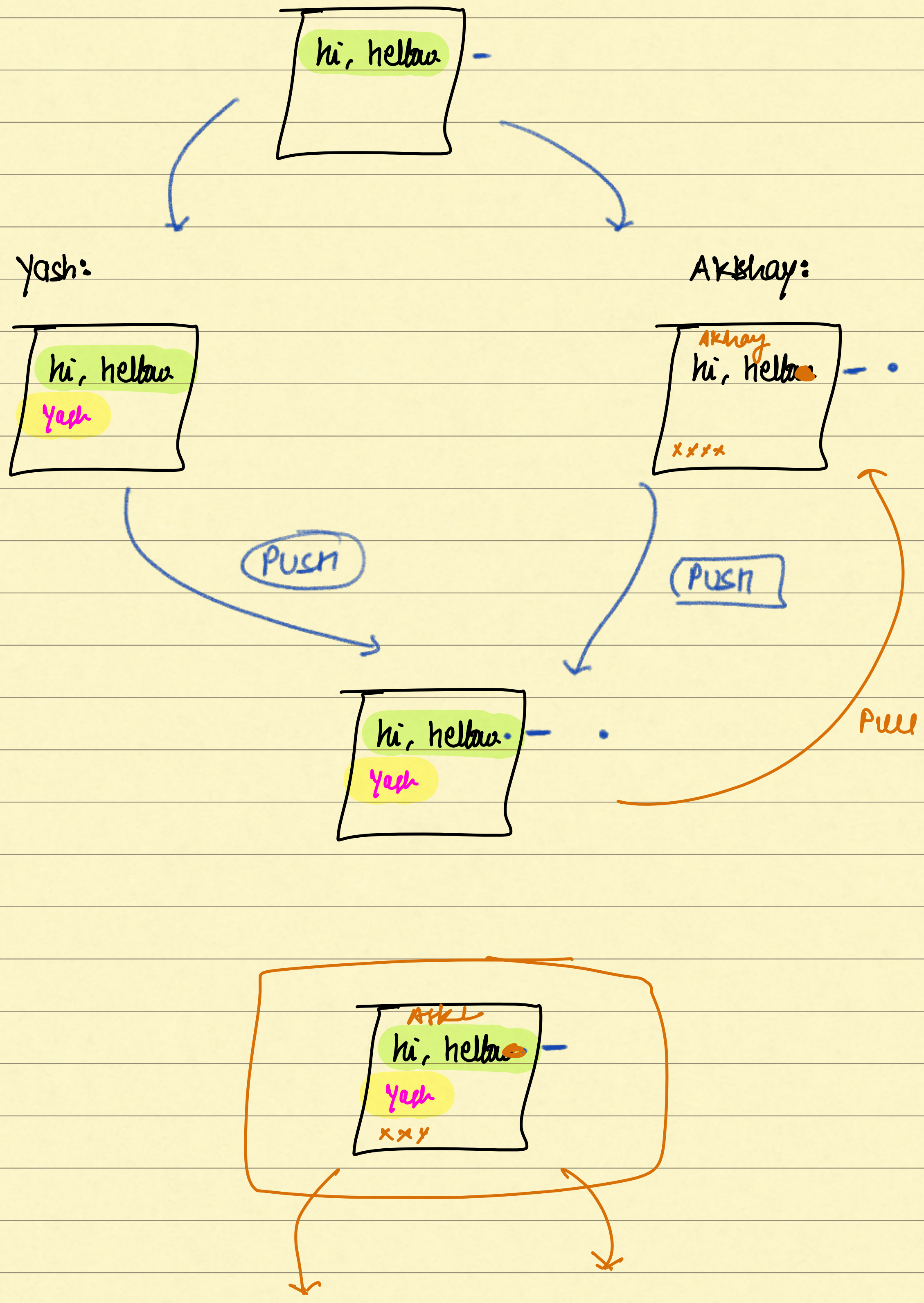
DISTRIBUTED:

- Allows people to work Independently even if offline
- NO SPOF
- complete history is stored on every machine



Google Docs




hi, hellow
Yash:
Akshay:
hi, hellow
yash
Akshay
hi, hellow
xxxx
Push
Push
hi, hellow
yash
Pull
Akki
hi, hellow
yash
xxy

Git [DISTRIBUTED]

Mostly used accross Industry

How Git Maintains History :

Commit

(very core term)

eg: Project

[empty folder]

$$\begin{tikzcd}
V1 \arrow[r, dash] \arrow[d] & V2 \arrow[r, dash] & V3 \arrow[r, dash] & V4 \cdots \\
\text{ROOT COMMIT} & & &
\end{tikzcd}$$

- commit message
- changes
- commit-id
- timestamp
- Author

→ **Commits are Immutable**

WHY:

??

C1 ······ $\boxed{C2}$ ······ C3 ······ C4

amend the commit

$\boxed{\text{git --amend}}$

Commit → 'id'

↓

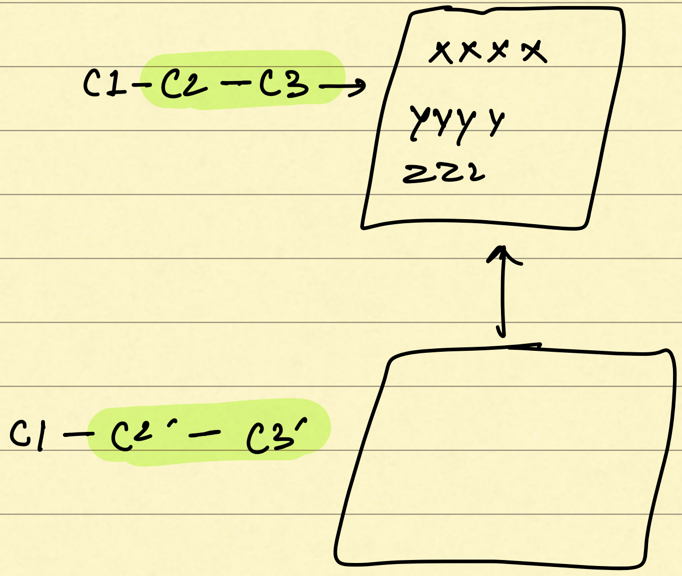
??

func (time + prev commit ids)

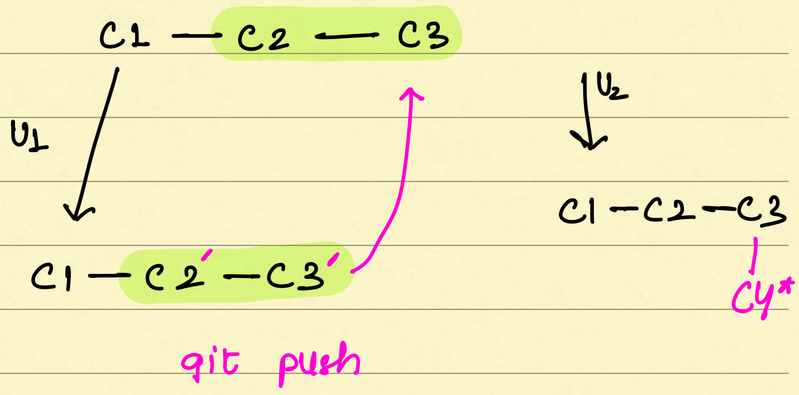
C1 —— C2 —— C3 —— $\boxed{C4^*}$

C1 —— C2* —— C3 —— C4 [expectation]

C1 —— C2' —— C3' —— C4' →



central server:


C1 – C2 – C3
U1
U2
C1 – C2' – C3'
git push
C1 – C2 – C3
C4*