#ASLI ENGINEERING

Things to remember while building Microservices

SWIPE

BY

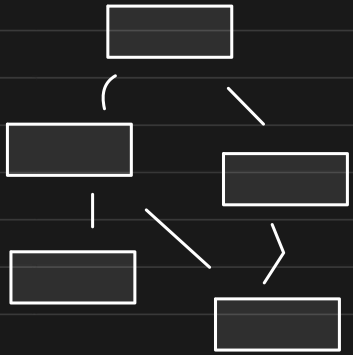
ARPIT BHAYANI

Things to remember while building Microservices

Huge scope for growth

A lot of core services and glue services are built when we adopt microservices.

Hence, there are many things that needs to be built, architect, scale and manage



Engineers can use this opportunity to step-up, own and earn some brownie points.

* not just some technical side only

we can showcase ownership, accountabity, attention to detail, leadership

Conflicts are inevitable

There is no one right way to build any system.

When teams work together, there would be arg about arch/decisions.

- data driven informed decisions
- watch and learn
- consult senior engineers
- okay to go along with design you disagree with ← sometimes/not always
- let seniors discuss and converge
- deciding in silo is fatal
- vision will evolve, adapt

ARPIT BHAYANI

Architechture evolves

Say, we have decided to use MySQL as our primary DB

but now our usecase / access pattern changed.

hence, we should be okay chaning the database.

The decision of deviating from what agreed upon is a big one.

This requires consensus and a solid reasoning.

Note: when we do ↑, document the reasons

Technical Debt

We cannot always build it the best way possible!

Mostly because we have to ship faster.

Engineering efforts are in a constant race against time

↳ Hence we cut corners

↳ make some quick inefficient decisions

Technical Debt

↓

These decisions prevent us from shipping / building faster in the future

It can also happen because the vision / requirements changed midway

Technical Debt is catastrophic and should be cleaned up periodically. Every sprint we should reserve ~10% of bandwith in cleaning it up.

Service Templates and Enforcing Standardization

We know how important is to standardize microservices but instead of relying on service owners,

build templates to spin up microservices

So, anyone using a template, would automotically follow the standard and best practices.

Ex: HTTP and gRPC for communication

Hysterix for circuit breaking

Prometheus for metrics

Redis as a cache

} clients integrated

Core idea: avoid duplicate efforts that are prone to deviate

Enforcing standardization may backfire

as engineers might feel strangled

→ This can be solved by providing proper reasoning

→ Updating the central template should be a collective effort where most teams are involved.

Keeping teams and engineers involved will reduce the negative sentiment.

Business >> Engineering

Has to be one of the most "offending" thing for an engineer but it is true...

Whatever we do, should be aligned with strategic goal of the business

Imagine,

Business wants to achieve profitability	ML Team provisions 50 GPU clusters

Some work should be conciously depriorititized

and this is not at all personal

* The projects we pick up, the tasks we do, should always be aligned with the business's strategic goals.

ARPIT BHAYANI