#ASLI ENGINEERING

10 Challenges in Implementing Microservices

SWIPE

BY

ARPIT BHAYANI

Challenges in implementing Microservices

We always talk about benefits of adopting Microservices
but grass is not always green

Benefits are all well documented, but what about challenges?

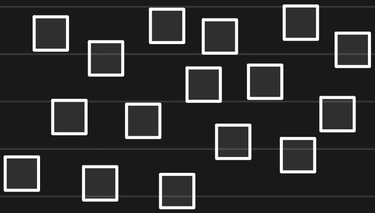
Here are some challenges that we need to tackle

Managing Microservices

As number of microservices increase,
managing them becomes a challenge

a new microservice should be well planned

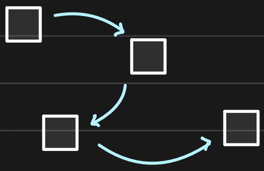
Things can go out of hand very quickly



Battle tested ↷

Instead of building tooling from scratch, use existing tools.

Monitoring and logging



Having blindspot in a microservices architecture
could be catastrophic.

1. Every component / service / server should be monitored
2. Debugging a root cause, spanning services, should be easy

↓

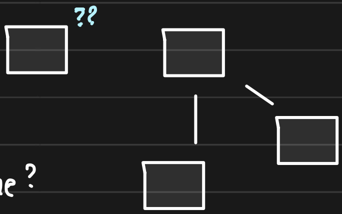
Distributed tracing becomes super critical

eg. Zipkin

ARPIT BHAYANI

Service Discovery

With there being 100s of microservices spread across 1000s of servers, how would we find whom to talk to get it done?



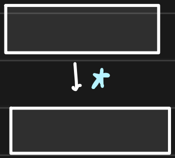
Service discovery could be done through

1. central service registry
2. LB based discovery
3. service mesh

All of these approaches in turn have their own advantages / disadvantages

Authentication and Authorization

We cannot keep services open, even internally. So, along with auth for external services Auth needs to be setup for inter-service communication



- a central auth (internal) service issuing JWT tokens

Configuration Management

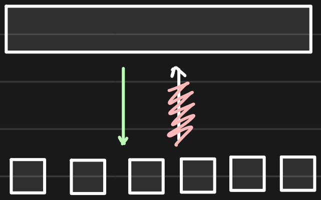
Each service has its own set of "secrets" & configurations. There needs to be a way to store and access configurations centrally.

everyone doing it on their own is waste of time

ARPIT BHAYANI

There's no going back

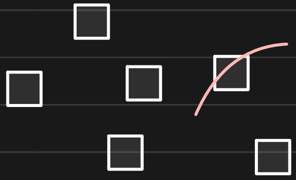
It is very difficult to switch back to Monolith from microservices



1. Tech Diversity
2. Teams have tested autonomity
3. People have adopted new tools and processes

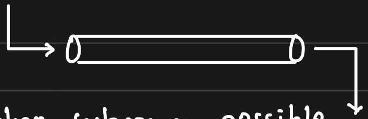
Fault Tolerance - There are more ways to fail

Outages are inevitable, but it is important that outage in one service is not bringing down all of them.



Hence try to model services around

- loose coupling
- Asynchronous Dependency



Use broker wherever possible

Internal and External Testing

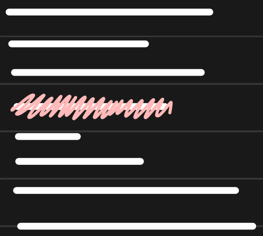
Testing becomes complex with microservices
It is hard to get 100% Isolated environments

+ Hard to simulated Distributed Failures

ARPIT BHAYANI

Design with Failure in mind

Counter-intuitive approach to development
But very critical in building robust services



Assume the code breaks after every single line
↳ solve and re-iterate

Dependency management is a nightmare

Managing dependency across services is tough.
There are 3 kinds of dependencies to think about

1. Service dependency

Sync dependency may trigger cascading failures

2. library / Module dependency

Without proper versioning or
backward compatability, rolling out changes becomes painfully slow

3. Data dependency

Services relying on data coming from other service
will hamper the end-user experience

ARPIT BHAYANI