#ASLI ENGINEERING

Best Practices to make service integration easy

SWIPE

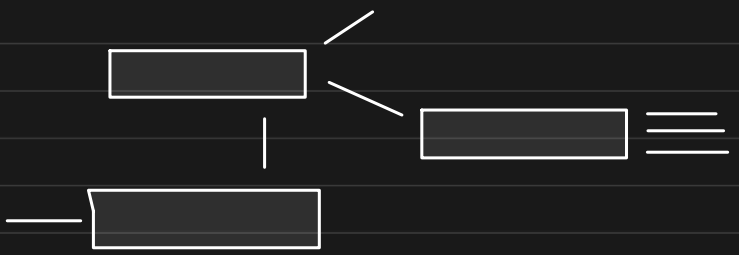
BY

ARPIT BHAYANI

Preparing for integration

Running microservices in isolation does not make a lot of sense

They work together with other services to get things done



So, how to get microservices integration right ??

Forward and Backward Compatability

Make changes to your services, such that the others don't need to change at all.

This is compatability

Key places where you need to keep a close eye

- database migrations
- API responses
- Messages for async communication

Some do's and don'ts

- never change the type of the column / attribute
  - ↳ instead prefer creating a new column with new type
    this would help us avoid forced changes
- never delete a column / attribute
  - ↳ instead deprecate it and give your dependent services enough time to remove dependency on it

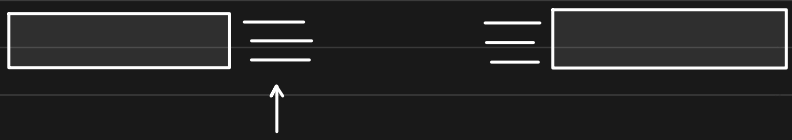
Make API interface tech agnostic

Things evolve very quickly in the world of s/w development

any interfacing technique you pick should be coupled with underlying tech

This would allow other services to pick a stack best suited for their use-case.

For example,



an RPC marshalling format

that would only work when

other service is using Java along with a particular lib

Dead Simple Consumption

Microservices are built to interact with other microservices.

Hence, it should by VERY simple for other services to talk to your service

- simple API
- simple data format
- common protocols

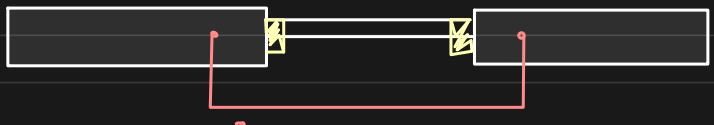
It would not matter at all if you have the most optimized and beautiful code, but a very poor interface practices

BE CONSUMER CENTRIC

Do not use weird formats, API routes, interfacing protocols. Keep it Simple!

Hide Internal Implementation Details

Never let the interfacing services learn internal implementation



We do not want
Microservice be bound to internal implementation

eg: how DB is structured
how messages are passed
direct linking nested jars etc.

All communication should happen over public API interface
This would allow us to upgrade internal & implementation
logic without breaking or being dependent on other services

* This also affects Technical Debt
↓
which would eventually
reduce shipping velocity