



#ASLI ENGINEERING

What are Microservices after all?



BY

ARPIT BHAYANI

What are Microservices?

Microservices are processes / workflows

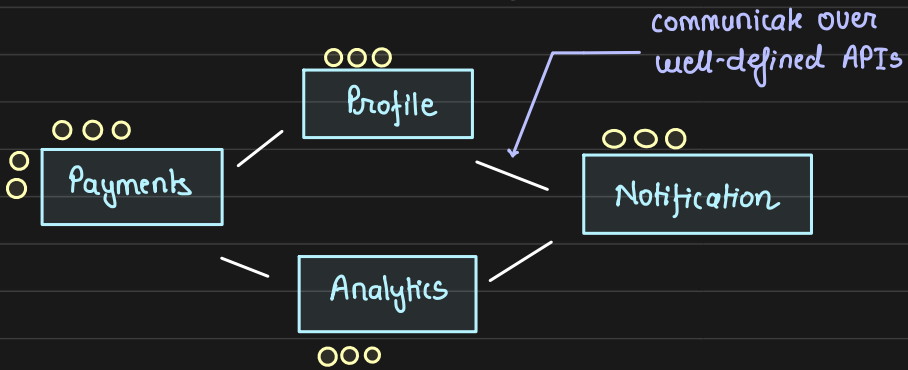
that are

- Highly maintainable
- loosely coupled

Microservices are like reusable functions, but bigger and over network

Microservices are typically structured around

Business needs and are owned by a *small team*



With microservices, we are trying to optimize

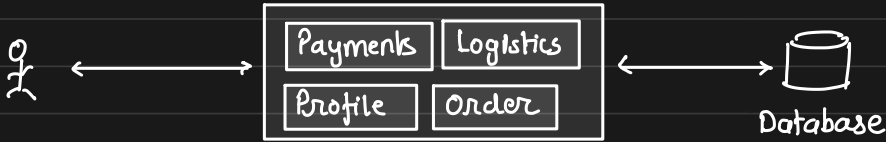
- rapid feature delivery
- easy to upgrade the stack

Microservices are no silver bullet

and have several drawbacks

Precursor to microservices is **Monolith Architecture**

Almost all products start monolith and slowly move to microservices



Business logic of payments, profile, orders, logistics are all part of the same JAR/Codebase/Binary that are deployed across all the servers.

Advantages simple to develop, build, test, and scale

↓ ↓

everything part all modules part
of same codebase of same artifact

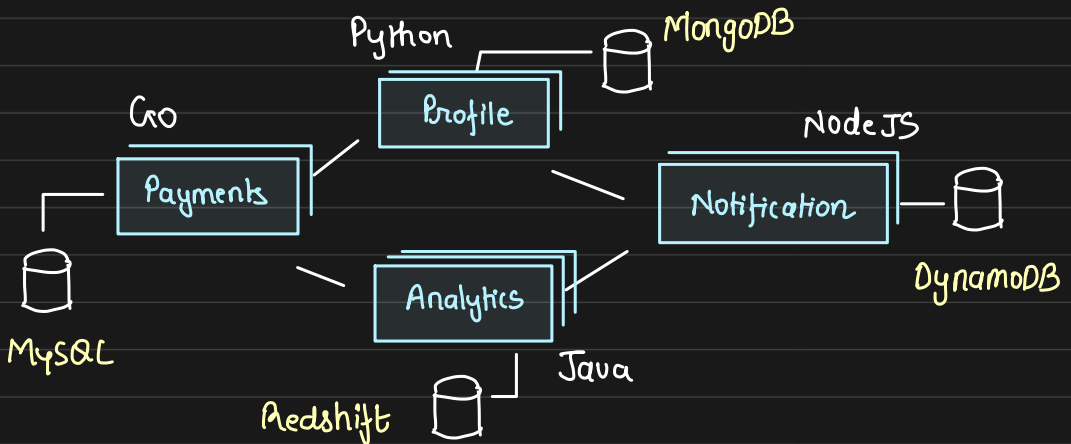
↗ same JAR
↘ deployed
across all STU

Disadvantages

- Tight coupling b/w modules
- Bulky container / binary / deployment artifact
- Homogeneous single Tech Stack
- Bug in one module affects others
- Scaling one module requires scaling everything
- Large monolithic codebase is intimidating and eventually slows down delivery

Monolith to microservices

Club functions that do related work and create a microservice
Start small, with few modules and continue to separate them
Eventually leading to architecture like this ↓



Characteristics of Microservices

- autonomous → decisions and operations are independent
 - specialized → focus on solving one problem really well
 - built for business → organized around business needs and teams
- ↗ evolutionary

Advantages of adopting microservices

- Agility → small independent teams moving faster
- Scaling → scaling is precise, independent and owned by team
- Tech Freedom → Picking right stack for the service
- Simple to understand → limited responsibilities
- Reusability of resources → build once, use across spectrum
- Faster defect isolation → isolate the failed service quickly
"circuit breaker"

Anti-patterns

- do not start with microservices
- do not make services too small
- use as many tools as possible, do not build from scratch