#ASLI ENGINEERING

Should we standardize Microservices?

SWIPE

BY

ARPIT BHAYANI

Standardizing Microservices

We love creating Microservices!

But there should be a standard way to creak one

A "Good" service

- define what a "good" service is (for your org)
- what are the capabilities that would make it
  - manageable
  - observable
  - debuggable

There has to be a common set of characteristics that every microservice of an org should have

But then what about autonomy?

The biggest advantage of microservices is autonomy.

Every service is allowed to take its own decision

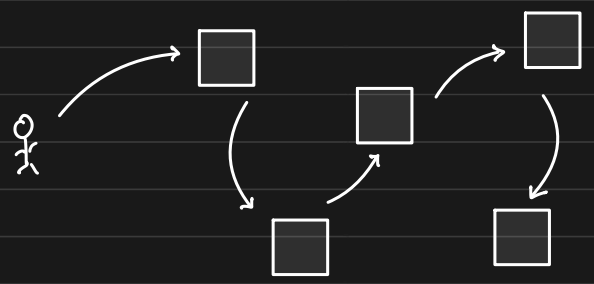
But bringing in standardization challenges independence

Hence, we need to Strike a balance

Standardization is important, as it allows us to keep entire system coherent & uniform

Monitoring

It is essential to know how we trace cross-service view of a request



↓
System-wide

request-wise view → Distributed tracing
using Zipkin, Aws x-ray

We also need to know

- how every server is doing → CPU, Memory, Disk consumption
- how every service is doing → Health check

Everything at one place

Possible Techs

Prometheus, Graphite, New Relic, DataDog, etc

[set alerting on top of these metrics]

Collect metrics: CPU, RAM, Disk, Network

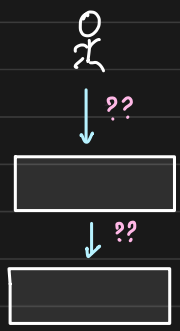
Collect logs: Application, Process, User log, SSH, OS

Collect app metrics: 2xx, 3xx, 4xx, 4xx, response time
request count, #servers

Interfaces

How would two services talk to each other ?

How would an end user talk to a service ?



Have a few ways to achieve this, but not many.

eg: have HTTP/REST and gRPC

Also, within a particular protocol, standardize

- how to define routes ? number of segments and resources
- how to name endpoints ? singular or plural
- how to paginate documents ? limit/offset or token

A critical thing to standardize : Versioning

/v1/users or /users/v1 or v1.a.com/users

Connection timeout, retry strategy, payload type

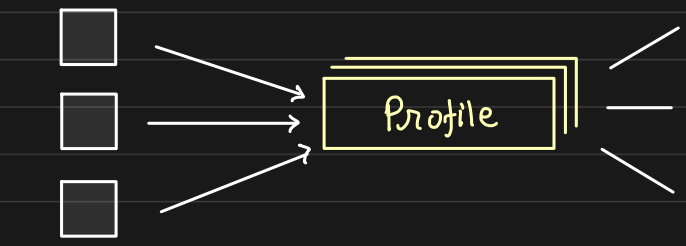
↓ not too small, nor large

↓ exponential backoff

↓ JSON, XML, TXT

Tolerance

What if one service bombards other



One bad service can take down the entire infrastructure.

Hence, each service should shield itself

few strategies to help us achieve this

- ration the number of calls from each service

  300 calls per second from service 1 to profile

- limit the number of outgoing calls from a service
- have an ability to cut-off incoming call from a service
- have an ability to cut-off outgoing call to a service