BEFORE RUNNING WITH MICROSERVICES, LEARN TO WALK!

André Dutra
Developer Experience
Manager @ Zé Delivery

WHOAMI - ANDRÉ DUTRA

Married and father of two beautiful kids

Weekend chef and bbq master

Moved by music

Basketball S2

Tv Shows, Movies, Board games

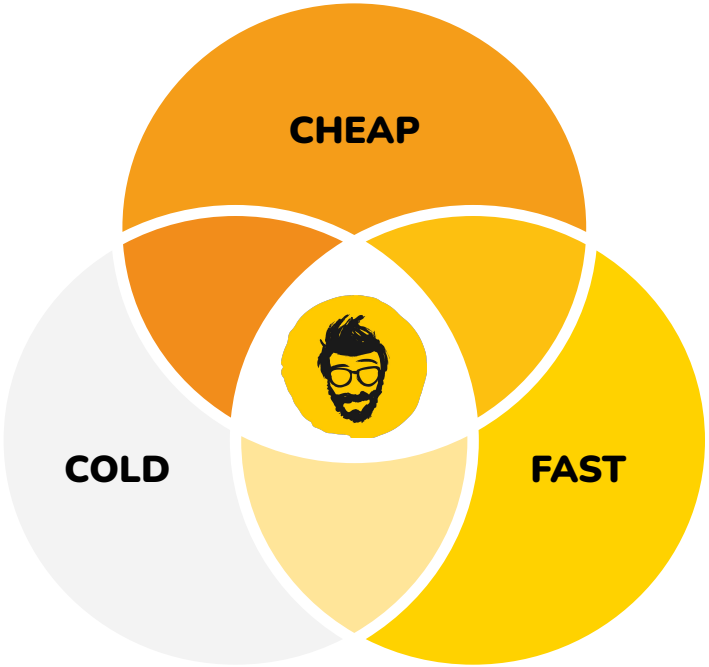
Technology Enthusiast

During a cleanup in my bag I found 6 different headphones

Oh, and, to pay the bills I work currently at Zé Delivery :D

What about you, have you ordered a Zé Delivery?

All people deserve a drink where, how and whenever they want!



YES, WE HAVE POSITIONS!

Talk to me or take a look at our open positions:
https://jobs.kenoby.com/zedelivery

PODE CONTAR COM O

MICROSERVICES WHAT?

01

But what exactly are microservices?

DEFINITION

The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

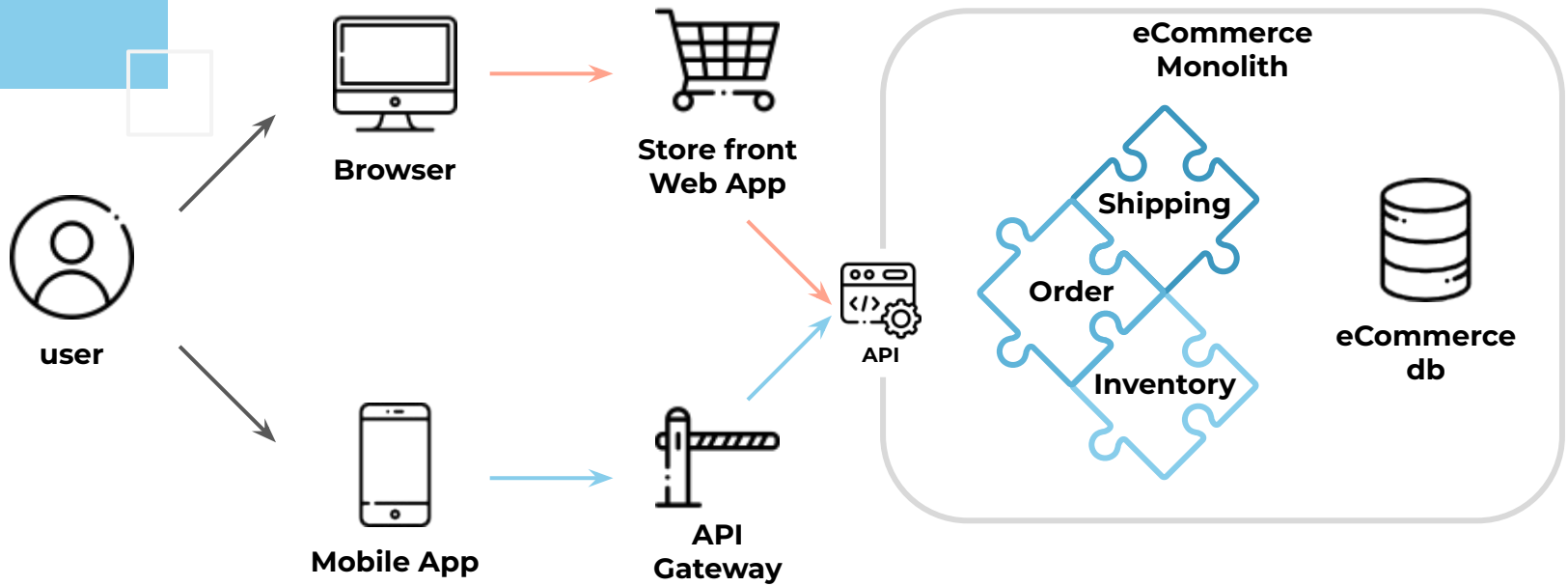
Source: https://martinfowler.com/articles/microservices.html

CHARACTERISTICS

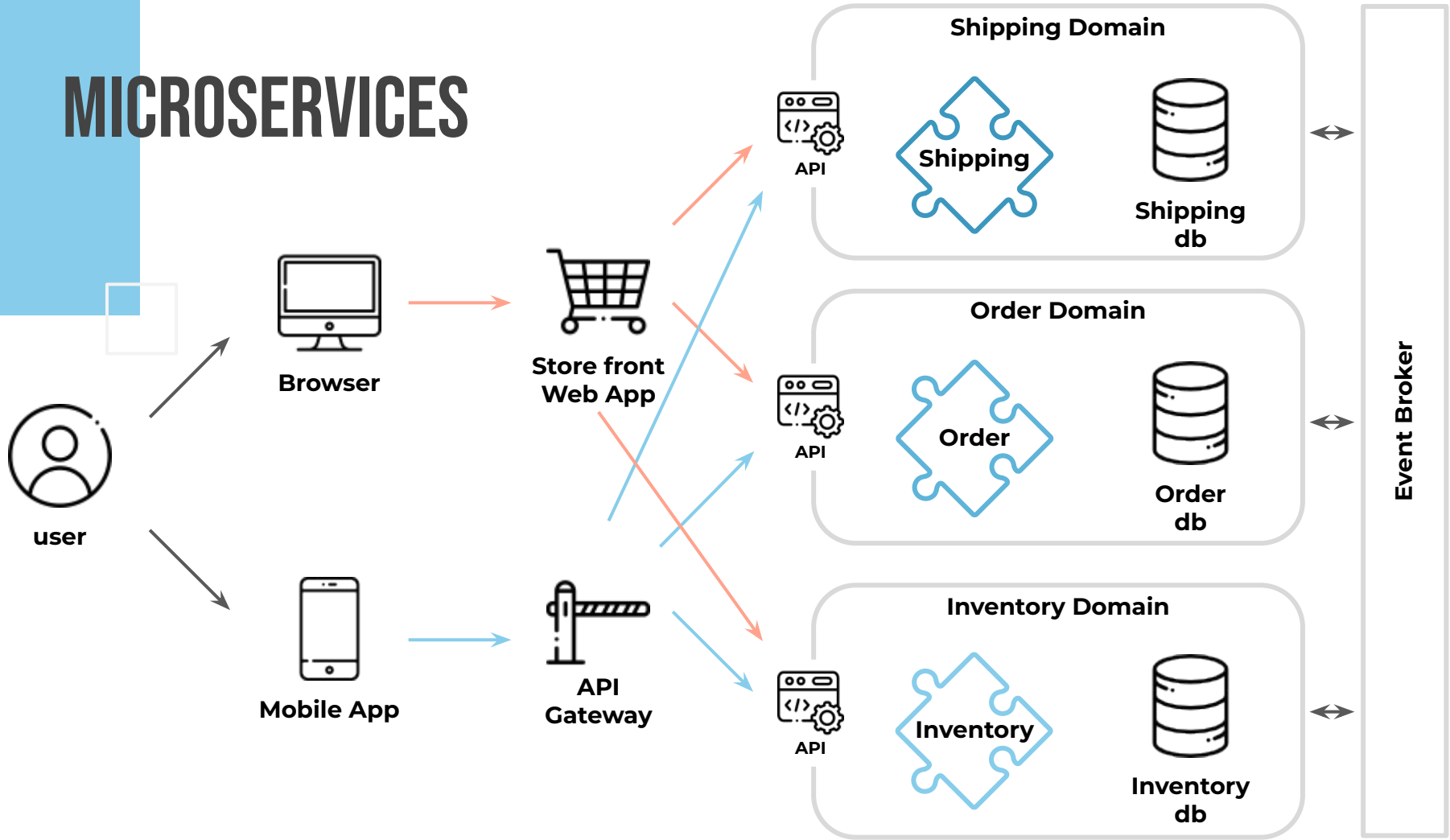
- SERVICES ARE COMPLETELY INDEPENDENT
- DECENTRALIZED DATA
- MULTIDISCIPLINARY TEAMS ORGANIZED BY BUSINESS CAPABILITIES
- DESIGN FOR FAILURE
- AUTOMATION, AUTOMATION, AUTOMATION
- EVENTS BECOME RELEVANT

EXAMPLE: MONOLITH AND MICROSERVICES

MONOLITH


Browser
Store front
Web App
eCommerce
Monolith
user
API
Shipping
Order
Inventory
eCommerce
db
Mobile App
API
Gateway

MICROSERVICES



PROS AND CONS 02

What are the pros and cons of choosing this type of architecture?

PROS

MODULAR COMPONENTS

Team specialization, product extensibility and reusability

INDEPENDENT DEPLOYS

Smaller codebase, faster deployments, lower risk, independent scale

MULTIPLE TECHNOLOGIES

Use the most appropriate technology for the proposed problem (eg SQL and NoSQL)

CONS

DISTRIBUTED SYSTEM

Test, debug, coordinate deployment and manage communication more difficult

ADDED COMPLEXITY

Many more components and technological diversity to deal with

EVENTUAL CONSISTENCY

Learn to handle distributed transactions and adjust user experience accordingly

WHEN TO USE? 03

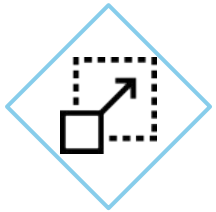
And what does that really help me with?

When to use?

USE WHEN YOU NEED:

AGILITY

Faster deployments, teams with less cognitive load



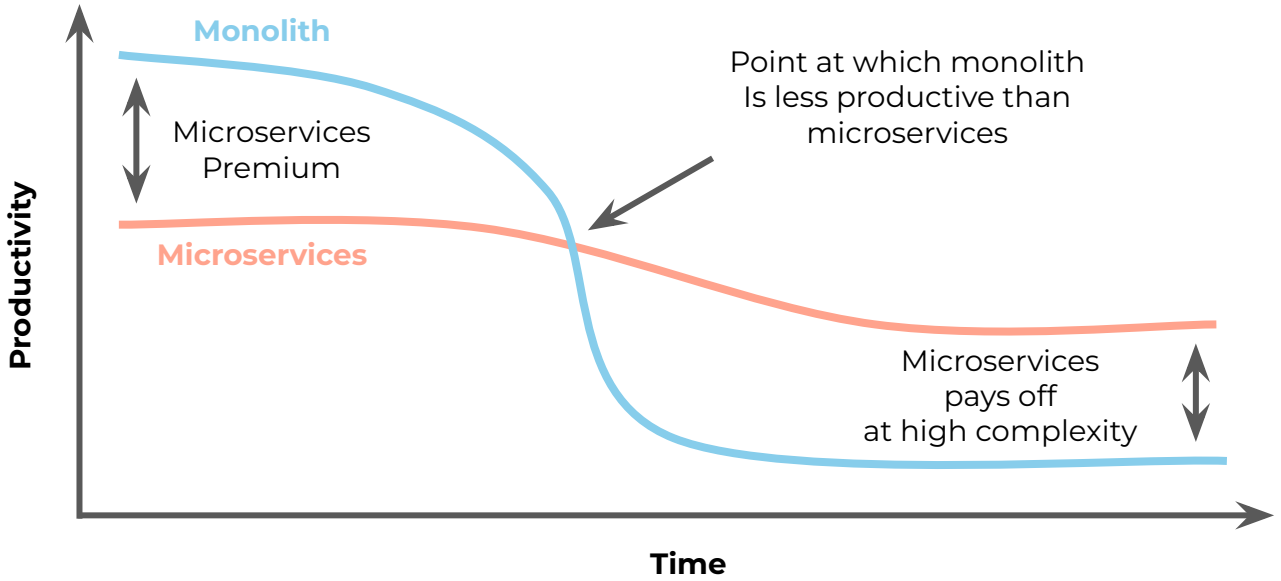
SCALABILITY

Scaling the application (or part of it) more fluid

RESILIENCY

Improve application fault tolerance

WELL... NOT THAT SIMPLE



Source: https://semaphoreci.com/blog/bad-microservices

So my primary guideline would be **don't even consider microservices unless you have a system that's too complex to manage as a monolith**

- MARTIN FOWLER

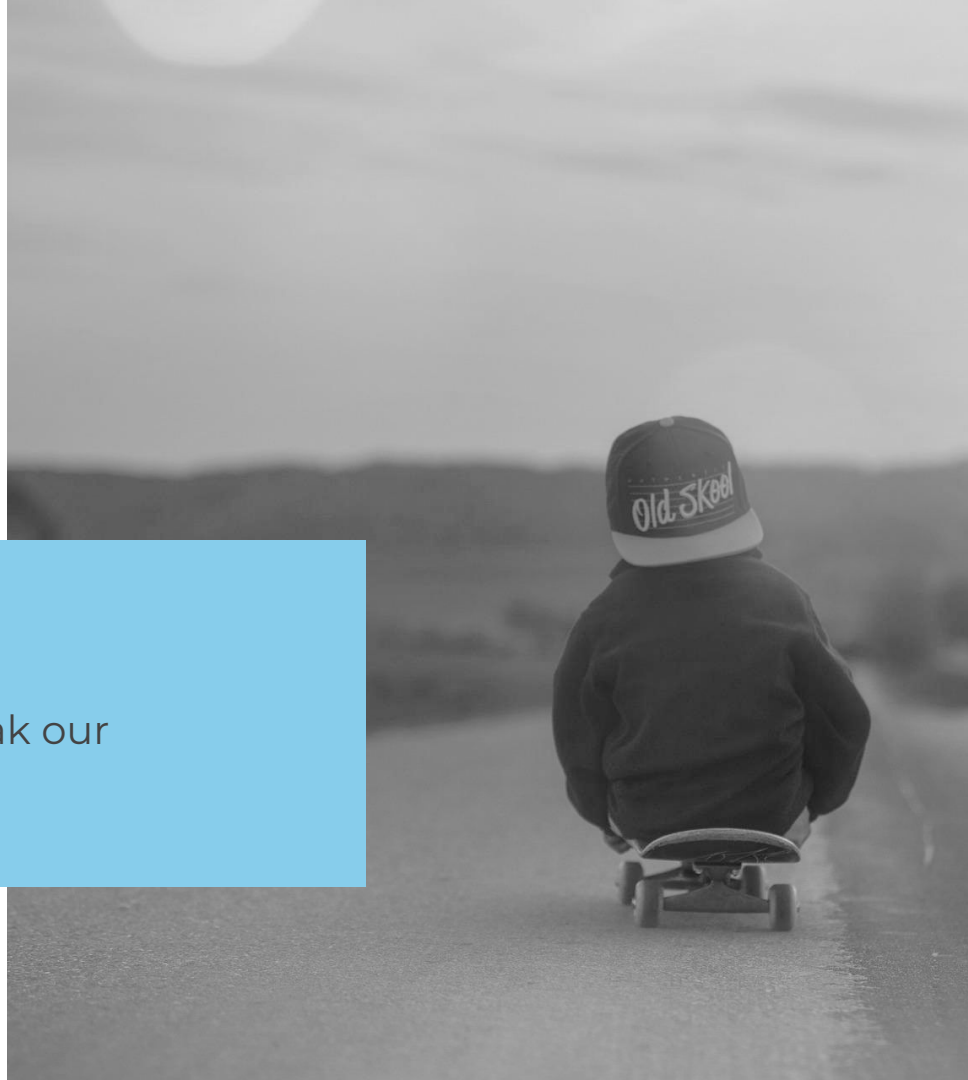
Author of Patterns of Enterprise Application Architecture

ARE WE READY?

04

Is my team ready?

Are we at the point to break our monolith?

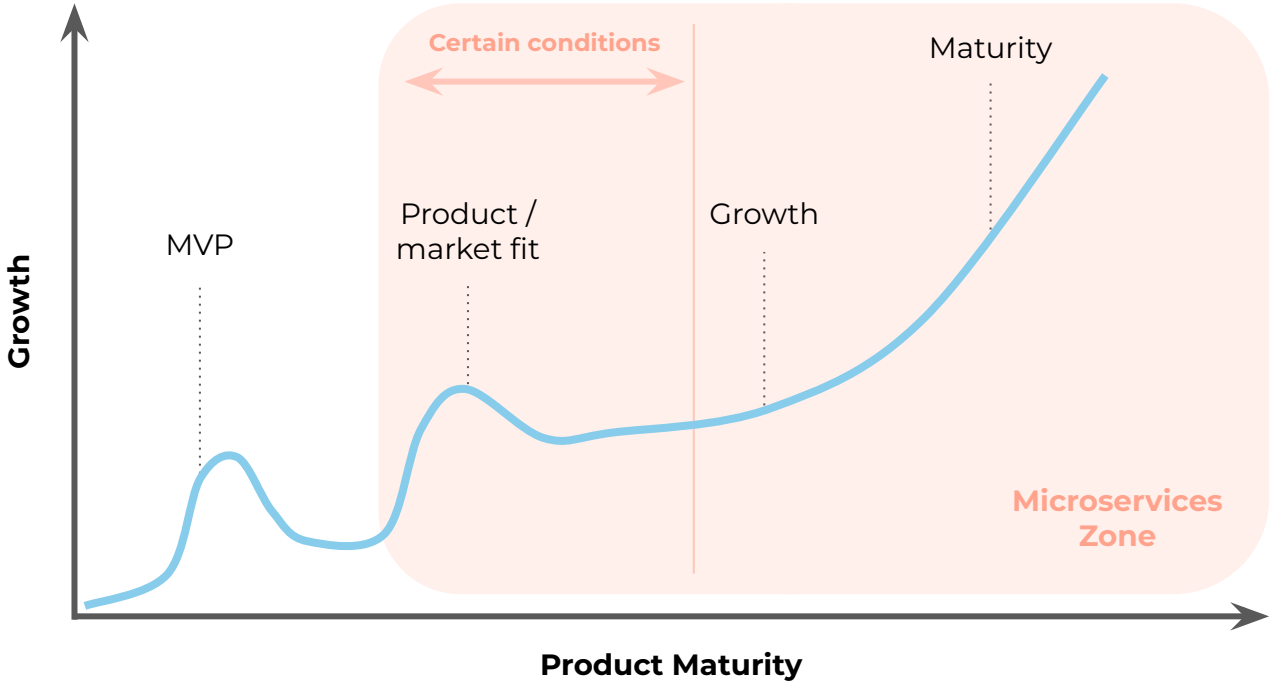


Any decent answer to an interesting question begins, "it depends..."

- KENT BECK

Author of Extreme Programming

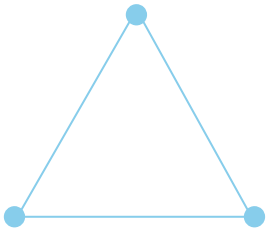
WHICH IS OUR PRODUCT STAGE?



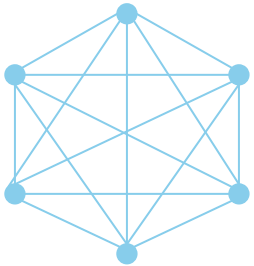
HOW BIG IS OUR TEAM?

There is science behind the *'two pizza teams'* rule:

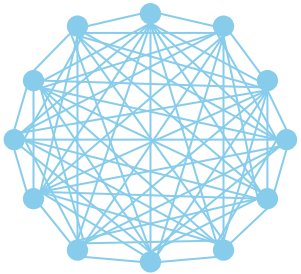
$$\text{\# of links} = \frac{n\,(n-1)}{2}$$



3 people
3 links



6 people
15 links



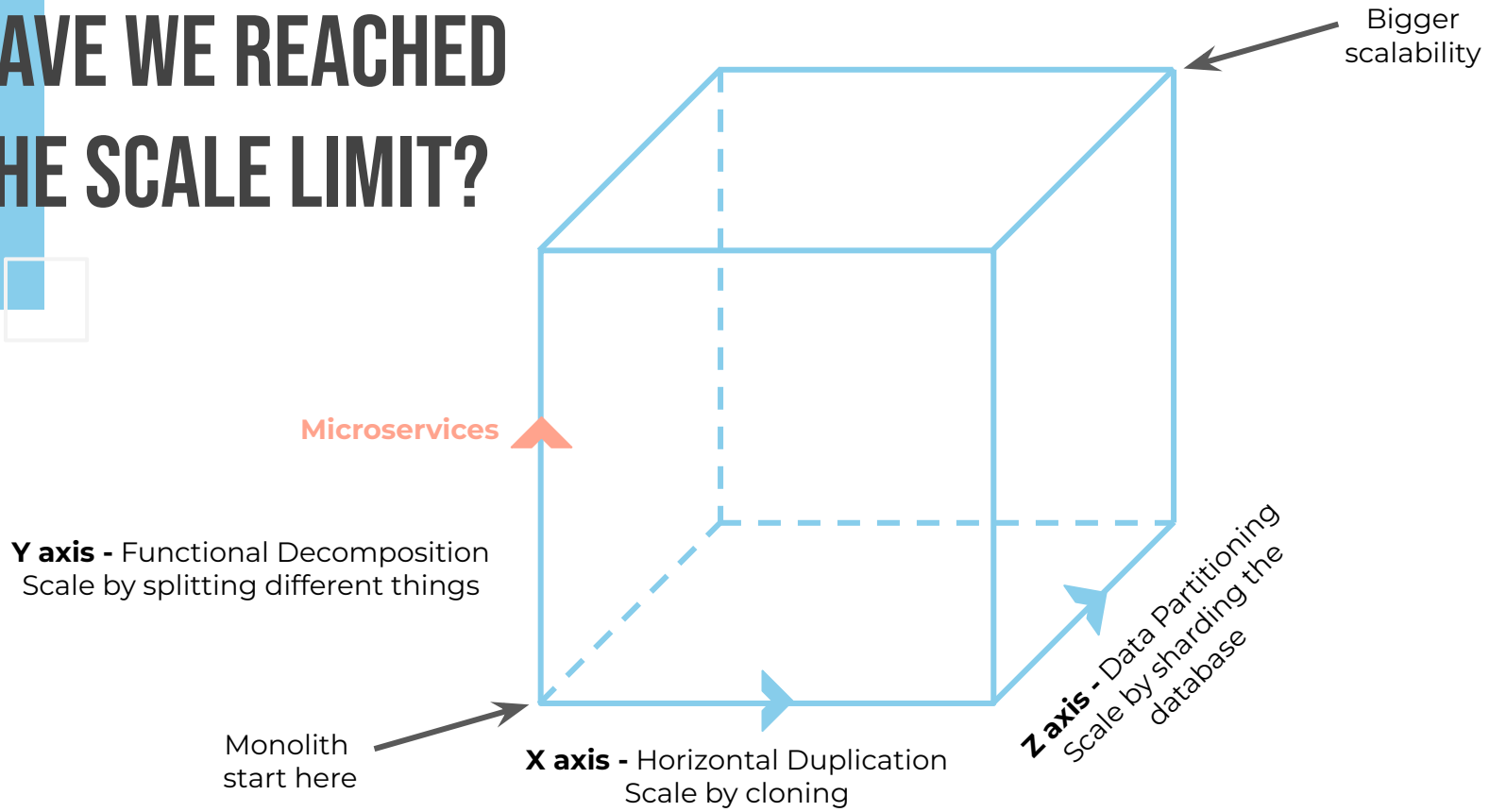
12 people
66 links

50 people
1225 links

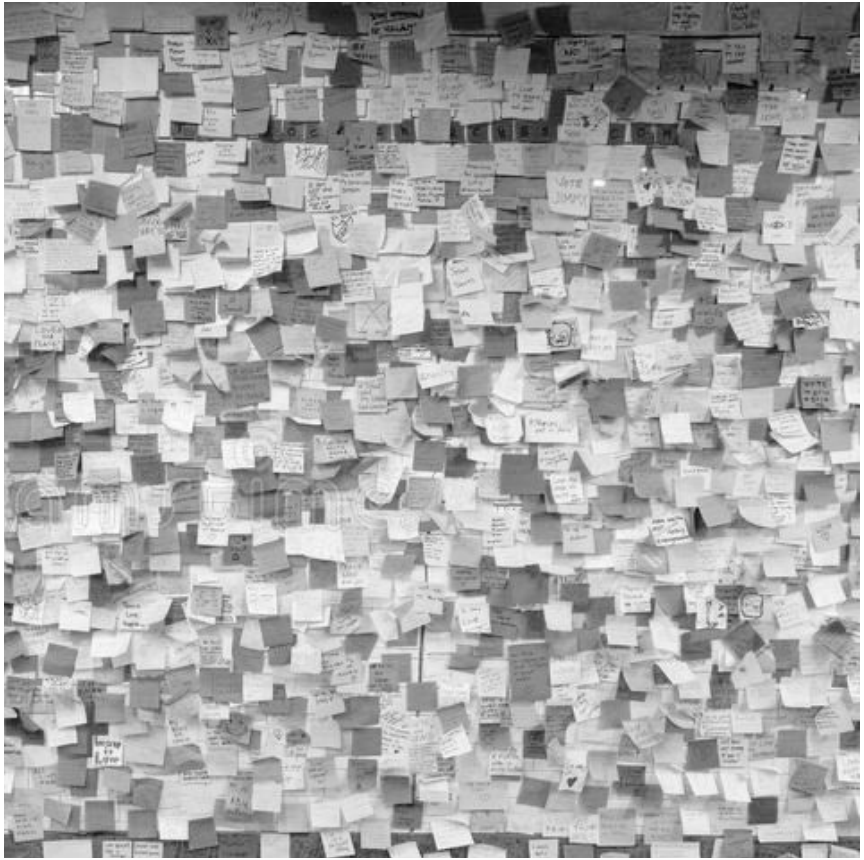
2x people
4.4x links

Source: https://blog.nuclino.com/two-pizza-teams-the-science-behind-jeff-bezos-rule

HAVE WE REACHED THE SCALE LIMIT?

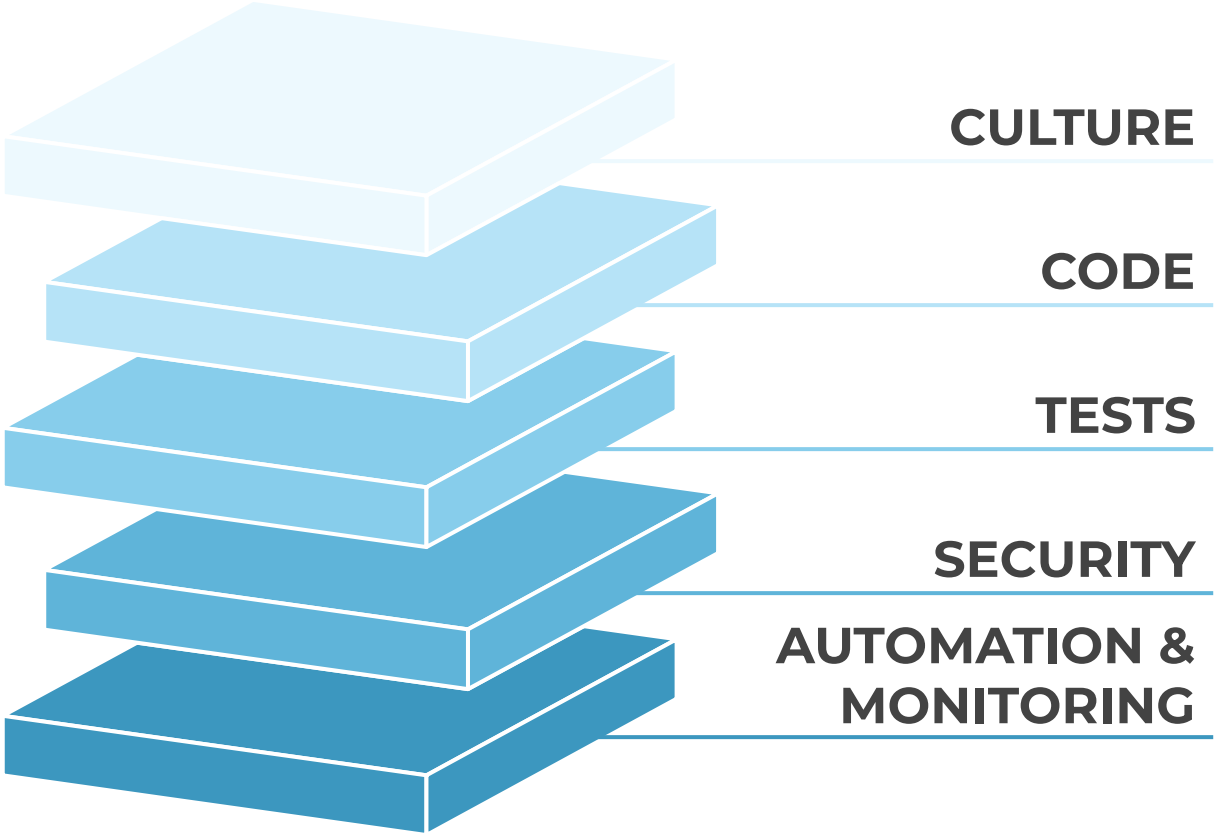


Source: http://theartofscalability.com/



HOW MATURE ARE OUR SOFTWARE ENGINEERING PRACTICES?

SOME DIMENSIONS



CULTURE

How the teams are organized in my company?

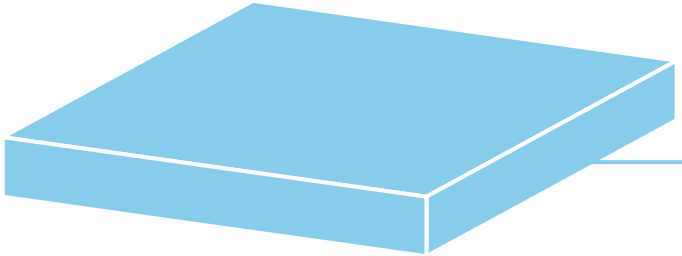
How are projects/initiatives prioritized by the company? How do we estimate work?

How much do we understand agility and really practice its principles? How often do we get new features to production?

How are we in terms of teamwork, communication and collaboration?

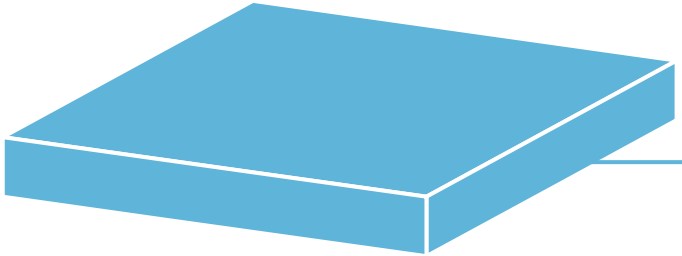
CODE

- How readable is our code? How is the quality of our code? How often do we deploy to production?
- How do we deal with refactoring and technical debt?
- Do we apply principles such as Clean architecture, SOLID, DRY, YAGNI and KISS?
- How we manage version control? Do we do code review and do we have an approval process?



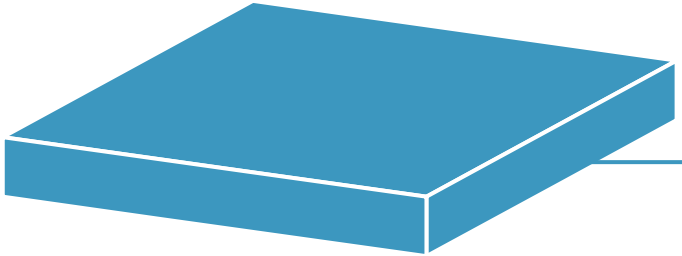
TESTS

- How do we handle unit tests? Integration tests? Performance tests?
- Is TDD understood and applied?
- What is the level of automation of our tests?
- Who is responsible for writing and maintaining tests on our team?



SECURITY

- Is security a relevant topic for the team when dealing with new features?
- What do we understand about OWASP TOP 10, SAST and DAST?
- Do we have a security team? What they do? Blue Team? Red Team? Penetration Testing?
- How do we handle and track vulnerabilities?



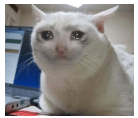
AUTOMATION & MONITORING

- Are the dev, QA, and production environments consistent?
- Where do we have automation? Build? Deploy? CI/CD pipeline? IaC? Configuration Management?
- How do we monitor our products? How do we query the logs?
- When something goes wrong, how are we notified? Is it possible to roll back cleanly and reliably?

IN SUMMARY: HOW MATURE IS OUR TEAM?

NETFLIX

Microservices Zone



Unstable environment for software development

INITIAL

Fair quality software product by following certain rules, quality standard and guidelines while developing

REPEATABLE

Good quality software product, documentation of standards and procedures, peer reviews, standards of development processes

DEFINED

Quantitative quality goals are set for the organization (for software products as well as software processes) Predictive quality within some limits. Training programs to enhance knowledge and team skills. Intergroup coordination

MANAGED

Continuous process improvement using quantitative feedback to improve productivity, quality and cycle time for the software product. Methods and tools are implemented to enhance defect prevention

OPTIMIZING

Capability Maturity Model

FINAL STORY...

Ahhhhh.... 😢

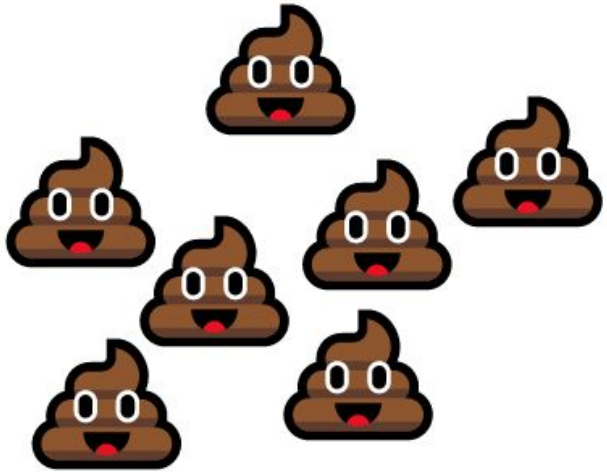
Monoliths are the future because the problem people are trying to solve with microservices doesn't really line up with reality

- KELSEY HIGHTOWER

Author of Kubernetes: Up and Running

AND WHAT HAPPENS IS...

Monolith



Microservices

THANK YOU!

André Dutra

byteswithcoffee.com

/andrelmdutra

andre.dutra@ze.delivery