# ForestPath

Microservice oriented software architecture

# Overview

- What is a microservice?
- What problems led us to microservices?
- How microservices solves problems?

# What is a microservice?

- Piece of software which is responsible for specific context and only that.
- It can be developed independently of other microservices in the system.
- It can be deployed without stopping the system.
- It can be rewritten within short period of time.
- It can be excluded from system if deemed unnecessary.
- It can use any stack (OS, programming language, database, framework).

# What problems led us to microservices?

- Legacy systems
- Scaling
- Technology
- Staff

# Legacy systems

- They become too large and maintaining them becomes too hard.
- Adding new features is hard and expensive.
- Software, which system relies upon, may become unsupported.
- You have to move your software to updated version even though you are ok with old one and the only reason for that is lack of support

# Scaling

- Software is not modularized and you have only a unit of software, you can only do vertical scaling or if it impossible you can only put your software on a more powerful machine.

# Technology

- Technologies (programing languages, frameworks, databases) come and go and it's great if you have ability to switch to a new one.
- When a system is developed it may become apparent that initially chosen stack (OS, programming language, database, frontend framework) was not optimal and a different stack is required.

# Staff

- As software gets more mature, the personnel supporting it may become irreplaceable part of business and it leads to problems for both parties:
  - personnel would want to switch to new exciting technologies, but maintenance requires their full attention
  - business which relies on a specific personnel will have a hard time changing
  - software's stack is outdated and finding talent to support it becomes harder and more expensive

# How microservices solves problems?

- Scaling
- Technology
- Personnel

# Scaling

- Microservices communicate using messages, thus it allows for both: horizontal (adding more services, distributing load between them, adding more machines) and vertical (adding more power to machines) scaling.
- Microservices do not interrupt system when they need to restart or be redeployed, which allows us to add more microservices on demand.

# Technology

- Because communication is done over text messages, as long as software can do that, you are free to use any stack (OS, programming language, database, frontend framework).

# Staff

- It encourages talent rotation because microservice is not complex in and of itself and a team will be able to understand code in a few weeks; or rewriting it from scratch may be more suitable option.
- Existing personnel inevitably will explore new technologies and may be in business' interest to switch to it or develop a new microservice that new technology.

# Review

- A microservice is a piece of software which communicates with other microservice using messages.
- Legacy systems become:
  - too hard to maintain
  - too important to kill
  - too expensive to rewrite
- Microservices are small and simple by definition which allows for:
  - easier maintenance
  - removing it from system
  - rewrite it using new technology or new team