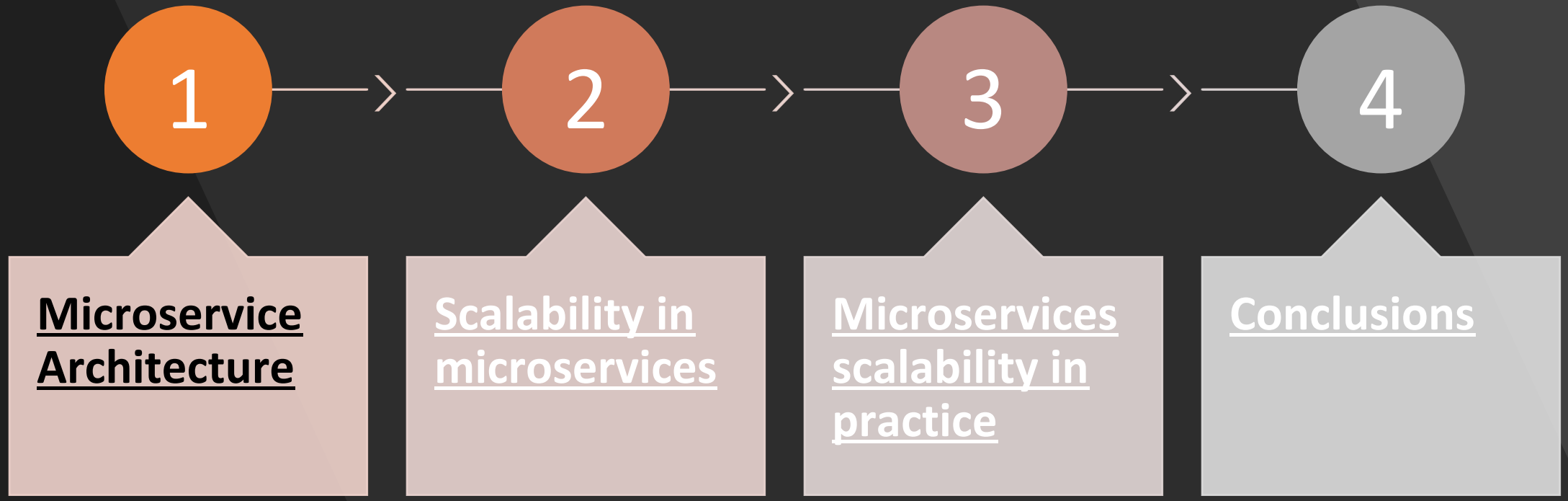
Microservices: How to make your application scale

Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, Larisa Safina

A.P. Ershov Informatics Conference
(the PSI Conference Series, 11th edition)
June 27–29, 2017, Moscow, Russia

Agenda


1
2
3
4
Microservice Architecture
Scalability in microservices
Microservices scalability in practice
Conclusions

1: Microservice Architecture

Microservices

Inspired by **SOA**

Developed around **business capabilities**

Each microservice implements a limited amount of functionality and runs its own process

Uses lightweight **communication** mechanisms

Supports pervasive distribution and scalability

What does this all mean?

- A Microservice is not just “a very small service”
- There is not a predefined “size limit” that define whether a service is a microservice or not
- Indeed “microservice” is a somehow misleading definition
  - Or better there is not definition at all, or not a unique one

Distinctive Characteristics

- **Size** : The size is comparatively small wrt. a typical service
  - Focus on providing only a **single business capability**
  - Benefits in terms of service maintainability and extendibility

- **Bounded context :** related functionalities are combined into a single business capability, which is then implemented as a service

- **Independency** : Each service is operationally independent from other, and the only form of communication between services is through their published interfaces

SOA vs. Microservices

- In SOA Services are not required to be **self-contained** with data and UI
  - No focus on **independent deployment** units and its consequences

- Focused on enabling **business-level programming** through **business processing engines** and languages such as BPEL and BPMN

2: Microservices scalability

Microservices scalability (1)

Portability

- Containers technologies
- Horizontally scaling

https://www.docker.com/

Elasticity

- Natural for the clouds
- Dynamically scaling

Microservices scalability (2)

Availability

- Ease of replication
- Updates and evolution on the fly

Robustness

- Fault tolerance over independent processes and containerization
- Logical vs physical environment

3: Microservices scalability in practice

Language-based

- The fine granularity of microservices moves the complexity of applications from the implementation of services to their **coordination**

- **Communication, interfaces, and dependencies are central to the development of microservice applications**

- Such concepts should be available as ***first class entities*** in a language that targets microservices

Programming language for microservices

- Four concepts are identified to be ***first class entities*** in a programming language for microservices
  - **Interfaces**
  - **Ports**
  - **Workflows**
  - **Processes**

- **Jolie** (Java Orchestration Language Interpreter Engine) includes all of them

Jolie Programming Language

- A language for microservice
  - Imperative with standard constructs such as **assignments**, **conditionals** and **loops**
  - Constructs dealing with distribution, communication and services
  - Variables are trees to for easy marshal/unmarshal (XML)
  - Separation of concerns between **behaviour and deployment** information

- Jolie takes inspiration from **WS-BPEL** and **CCS**
  - transfers these ideas into a full-fledged programming language

Applications

Netflix (Chaos Monkey)

Pioneers who moved from monolith to microservices to ensure scalability

IoT and Smart Buildings

Easy to separate the logic into small components

4: Conclusions

Microservices and scalability, summary

1

Microservices architecture is **more complex** than one based on monoliths

- The cost of growing and scaling easily

2

The future is certainly not challenge-free

- Security issues
- Quality of packages and correctness
- Computational capabilities

3

No silver bullet

- Multiple independent entities introduce extra overhead for deployment, administration, monitoring, and security
- Sometimes microservices are not the solution

Additional References

1. *N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina -* **Microservices: yesterday, today, and tomorrow**. In *Present and Ulterior Software Engineering*, Springer, 2017
2. *N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina.* ***Microservices: How to make your application scale.*** In *PSI 11th edition. Springer, 2017*
3. *N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara -* **Microservices: Migration of a mission critical system.** Technical report April 2017 https://arxiv.org/abs/1704.04173
4. *Manuel Mazzara Fabrizio Montesi Claudio Guidi, Ivan Lanese.* **Microservices: a language-based approach.** In *Present and Ulterior Software Engineering.* Springer, 2017
5. *Alexey Bandura, Nikita Kurilenko, Manuel Mazzara, Victor Rivera, Larisa Safina, Alexander Tchitchigin -* ***Jolie Community on the Rise****.* 9th IEEE International Conference on Service-Oriented Computing, 2016
6. *Larisa Safina, Manuel Mazzara, Fabrizio Montesi, Victor Rivera -* ***Data-Driven Workflows for Microservices: Genericity in Jolie.*** 30th IEEE International Conference on Advanced Information Networking and Applications, 2016.
7. *Alexander Tchitchigin, Larisa Safina, Manuel Mazzara, Mohamed Elwakil, Fabrizio Montesi, Victor Rivera.* **Refinement types in jolie.** In Spring/Summer Young Researchers Colloquium on Software Engineering, SYRCoSE, 2016.