PWL # 8:
"Microservices: The Journey So Far and Challenges Ahead"

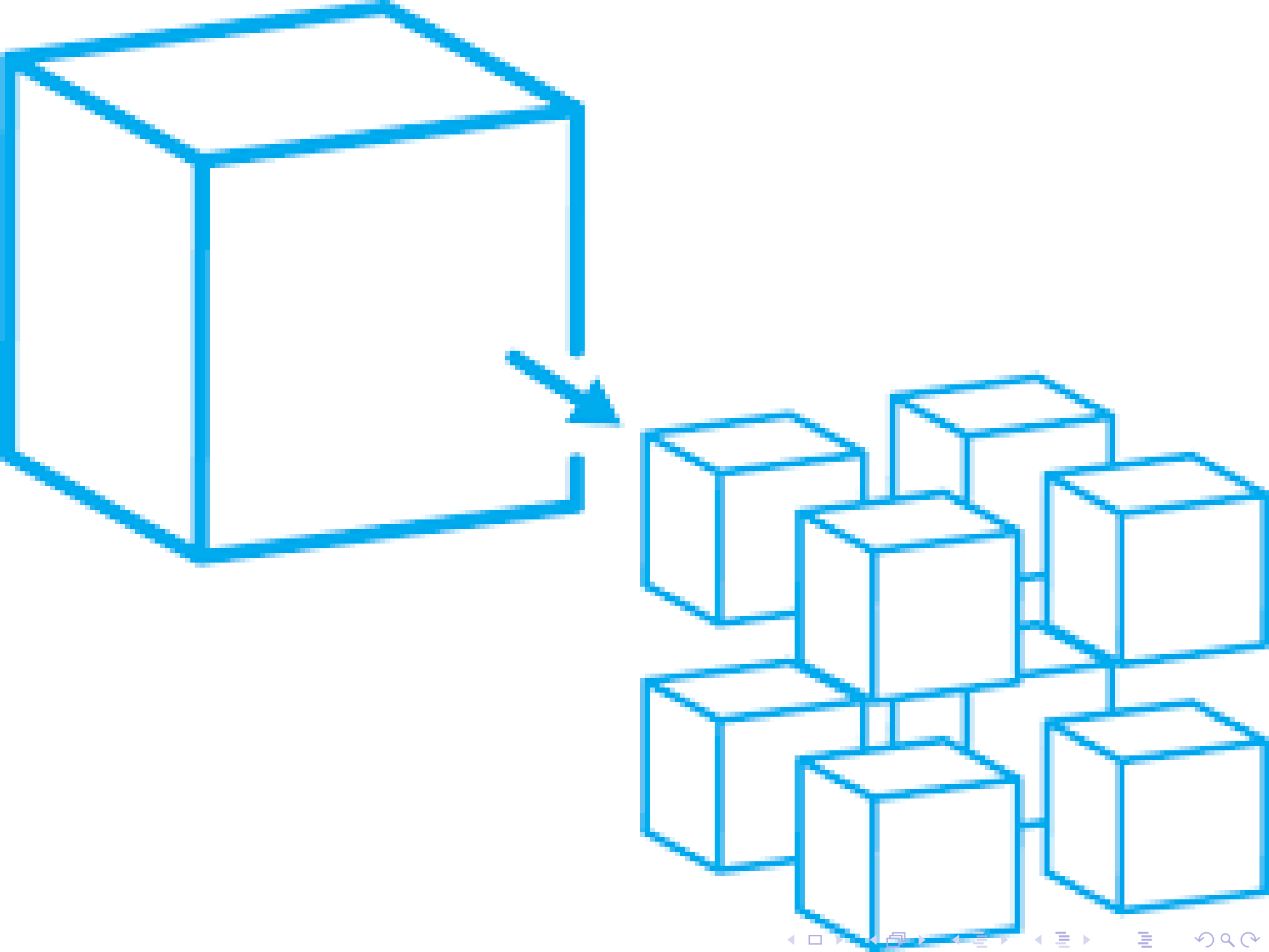
Pooyan Jamshidi, Claus Pahl, Nabor C. Mendonça, James Lewis, and Stefan Tilkov

Papers We Love ❤ Brasília

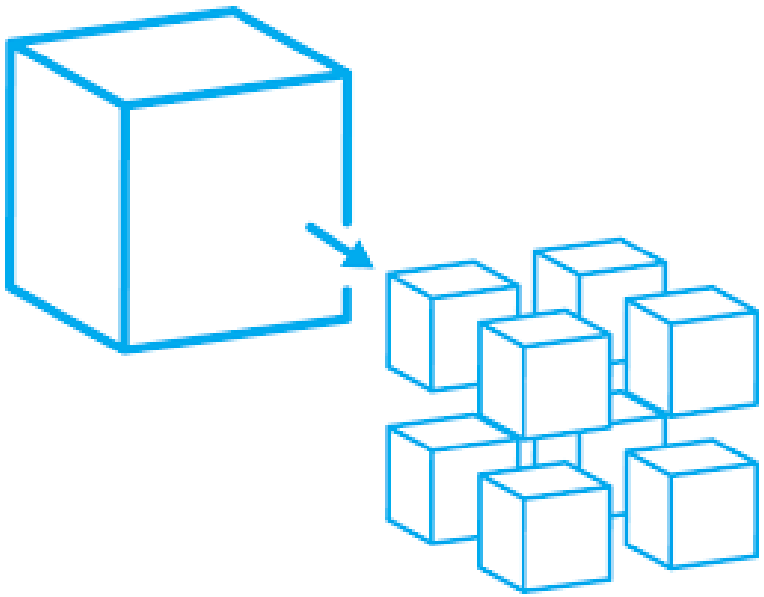
Presenter: Alessandro Leite

June 07, 2018

Outline

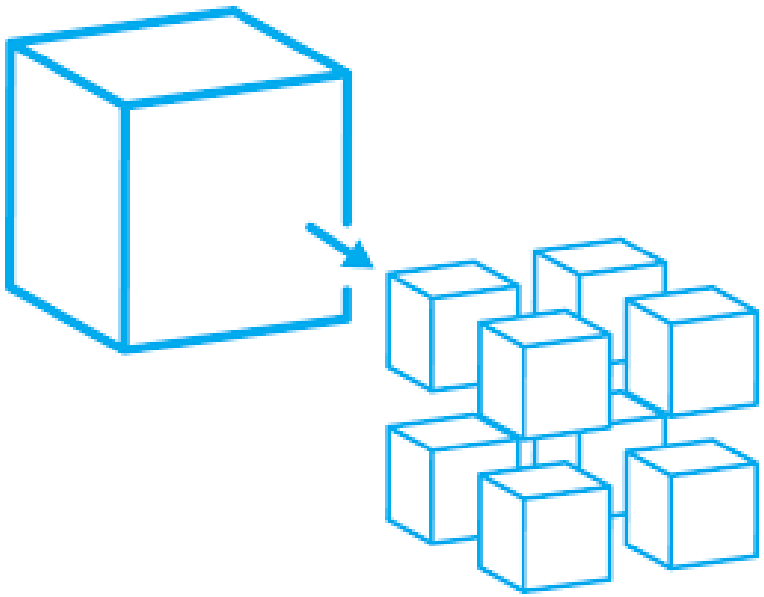


Microservices, what are you talking about?



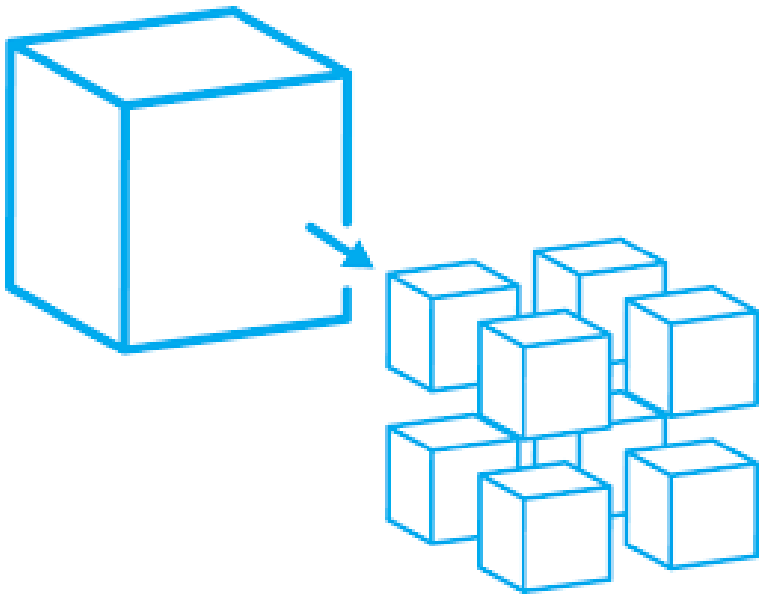
- Microservices comprise a strategy to **software** and **system architectures** that relies on well-established concepts of **modularization**

Microservices, what are you talking about?



- Microservices comprise a strategy to **software** and **system architectures** that relies on well-established concepts of **modularization**
- An emphasis on **technical boundaries**, where each **module** – microservice is **implemented** and **operated** as a **small** and **independent system**, offering access to its internal logic and data through a **well-defined network interface**.

Microservices, what are you talking about?



- Microservices comprise a strategy to **software** and **system architectures** that relies on well-established concepts of **modularization**
- An emphasis on **technical boundaries**, where each **module** – microservice is **implemented** and **operated** as a **small** and **independent system**, offering access to its internal logic and data through a **well-defined network interface**.
- Each **microservice** becomes an **independent unit** of **development**, **deployment**, **versioning**, and **scale**.

Microservices vs Service-Oriented Architecture (SOA)

- **SOA** usually relies on enterprise service bus (**ESB**)
- **SOA** is often associated with web services protocols, tools, and formats such as **SOAP**, **WSDL**, and the **WS-*** family of standards
- **SOA** is commonly viewed as an **integration solution**
- **Microservices** rely only on **lightweight technologies**
- **Microservices** commonly rely on **REST** and **HTTP** or other formats perceived as being **native** for **web development**
- **Microservices** are typically applied to **build individual** software **applications**

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices
  - fully automated software integration and delivery machinery

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices
  - fully automated software integration and delivery machinery
- **Improved scalability**

Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices
  - fully automated software integration and delivery machinery
- **Improved scalability**
  - **services** can be **scaled independently**, according to their specific requirements (i.e., system's runtime scalability viewpoint)

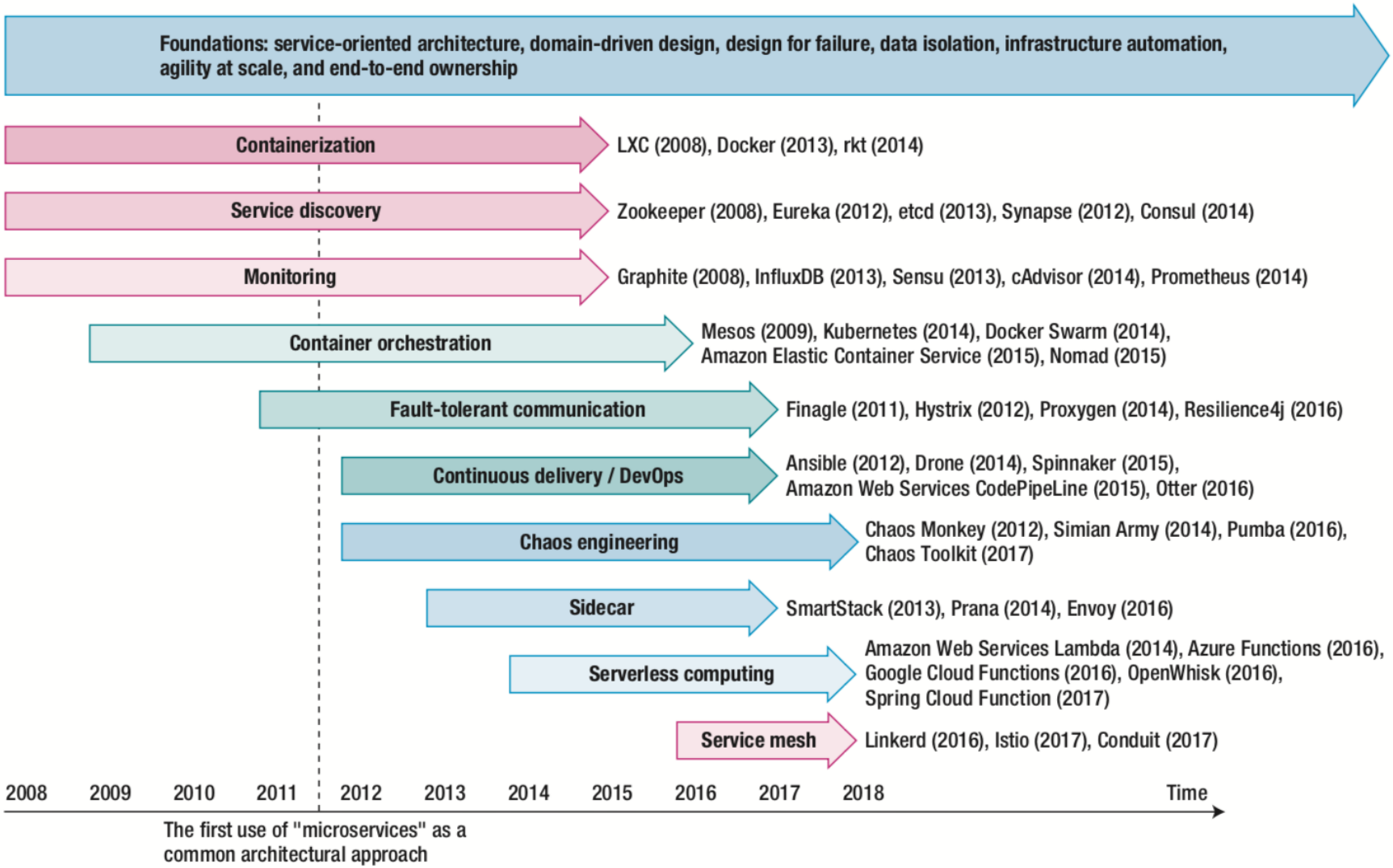
Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices
  - fully automated software integration and delivery machinery
- **Improved scalability**
  - **services** can be **scaled independently**, according to their specific requirements (i.e., system's runtime scalability viewpoint)
  - **services** can be **developed**, **deployed**, and **operated** by **different teams**, enabling the option for parallel introduction of new features (i.e., development viewpoint)

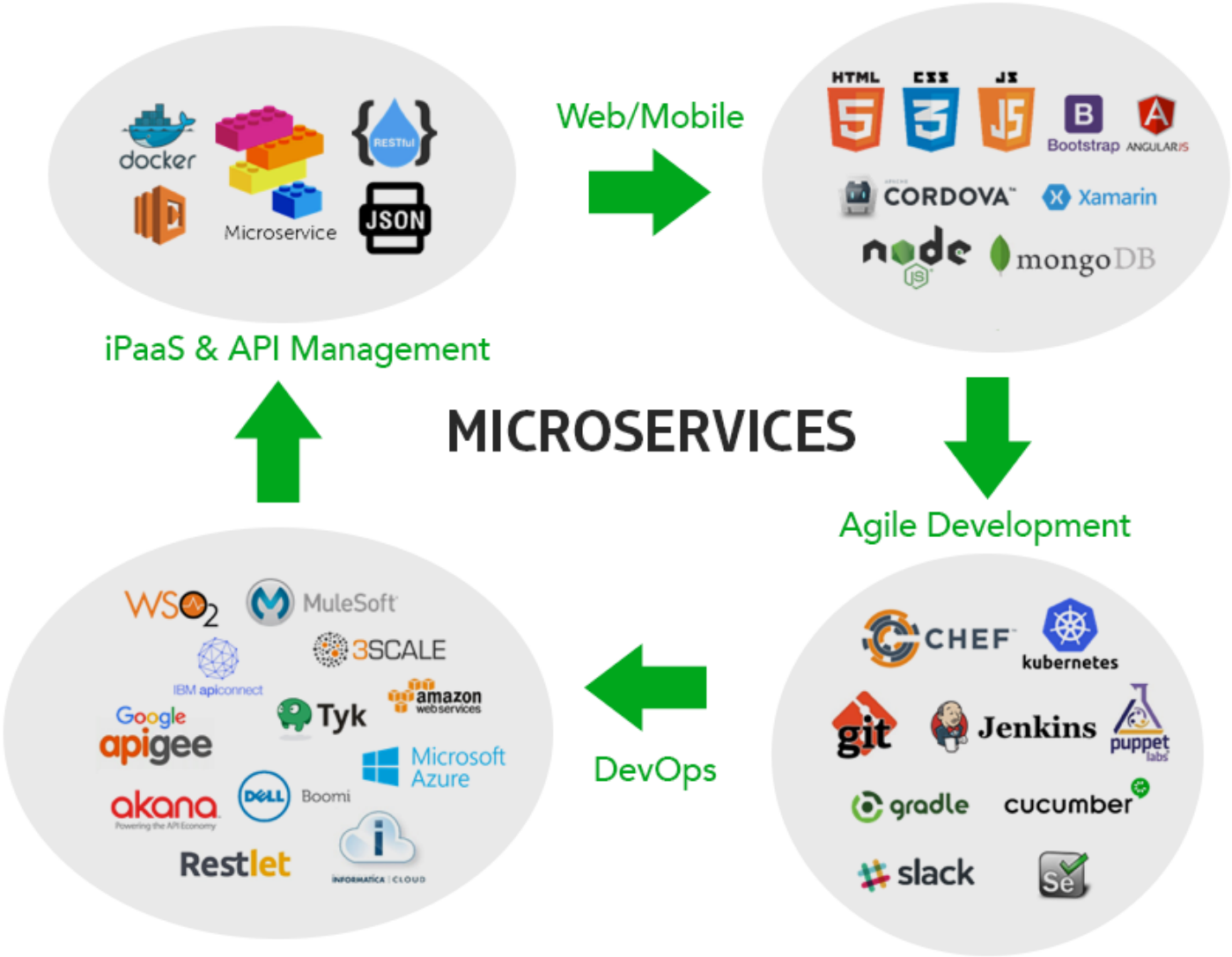
Microservices' benefits

- **Faster delivery**: **delivery** software ready **to production as soon as possible**
  - lightweight container technologies
  - DevOps practices
  - fully automated software integration and delivery machinery
- **Improved scalability**
  - **services** can be **scaled independently**, according to their specific requirements (i.e., system's runtime scalability viewpoint)
  - **services** can be **developed**, **deployed**, and **operated** by **different teams**, enabling the option for parallel introduction of new features (i.e., development viewpoint)
- **Greater autonomy**: each microservice offers an **autonomous** and **bounded unit** of both **development** and **runtime decisions**

Microservice evolution

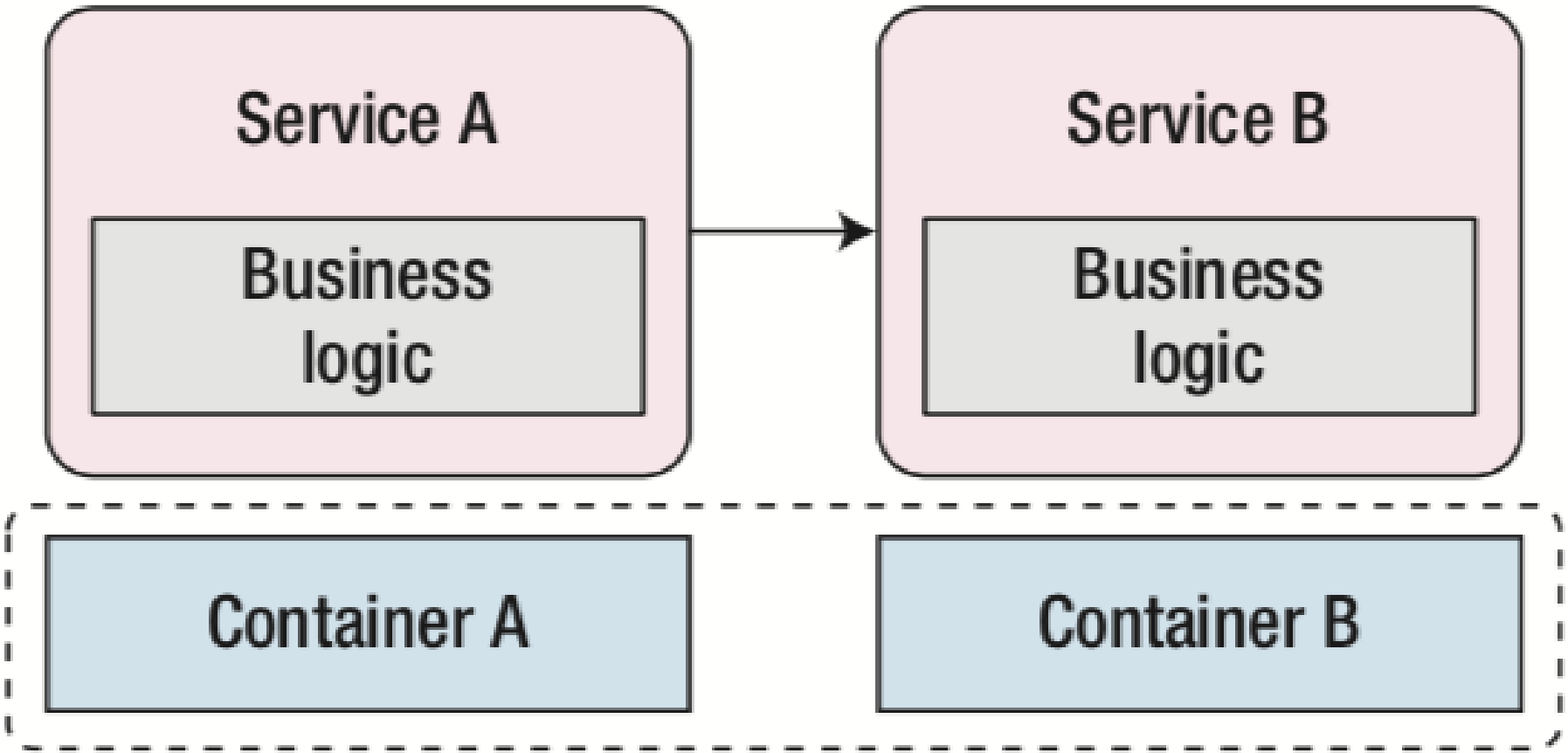


Technological perspective



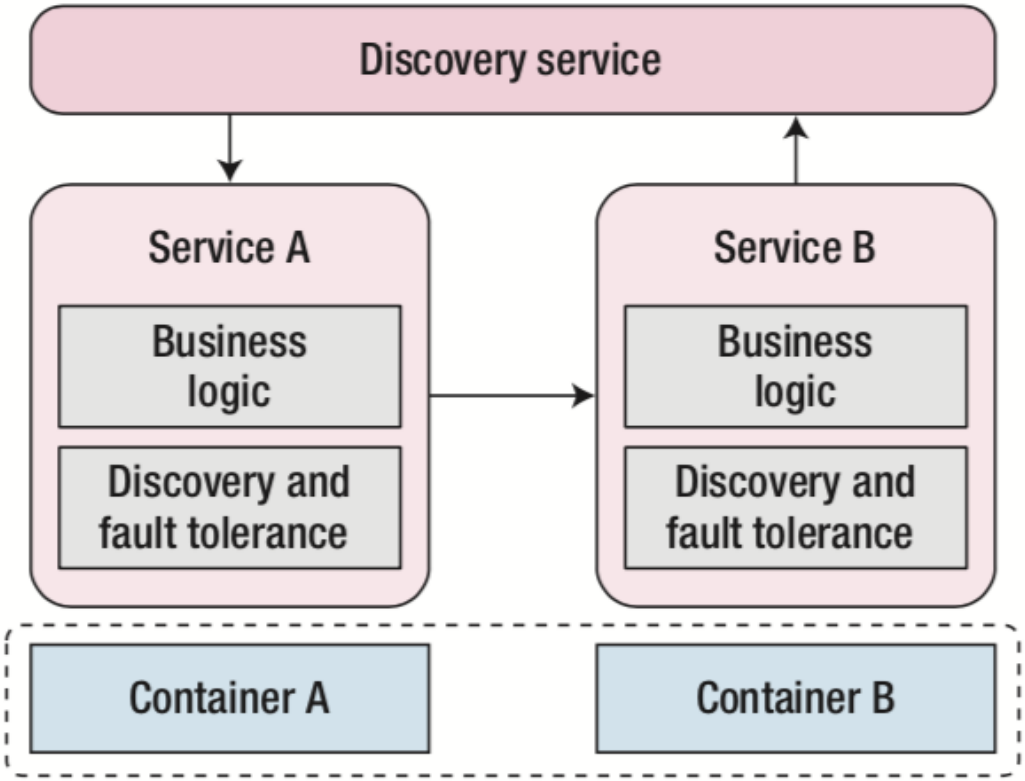
Microservices have evolved along four generations

- **First generation**: lightweight container technologies



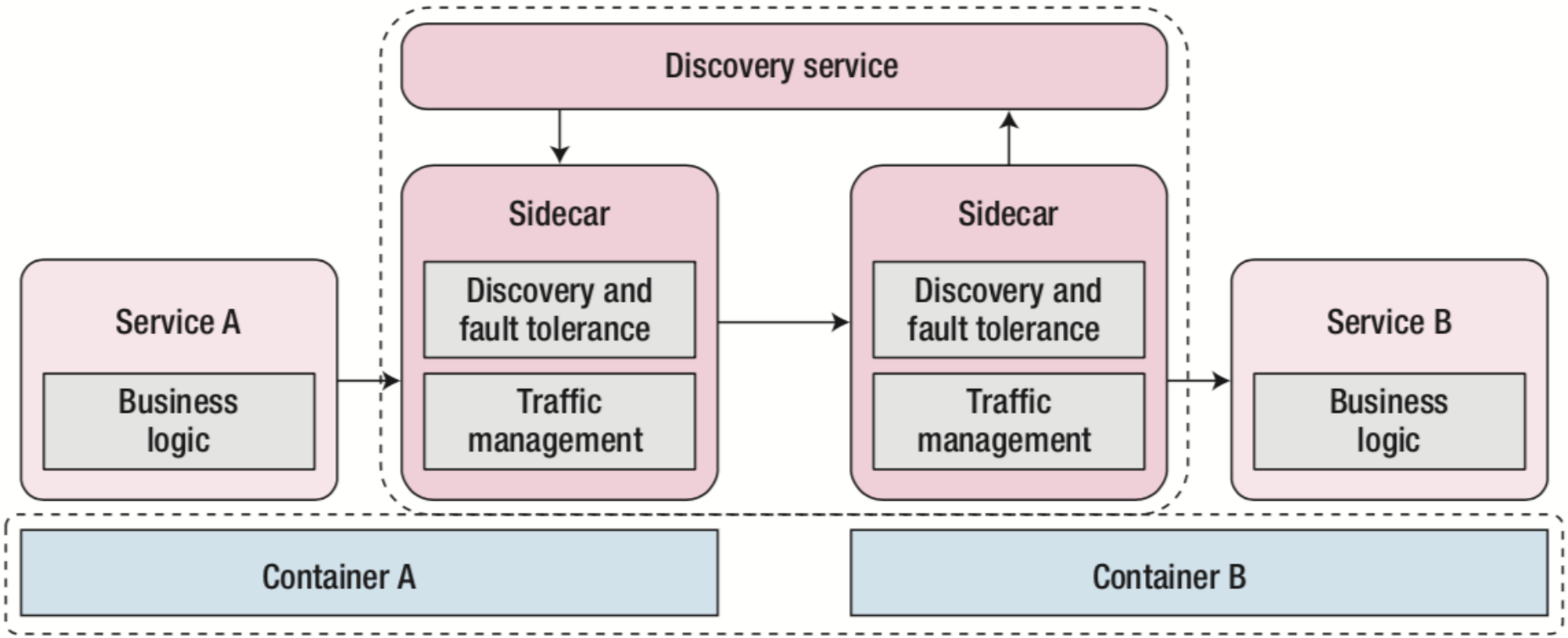
Microservices have evolved along four generations

- **Second generation**: service discovery and reusable fault-tolerant communication libraries



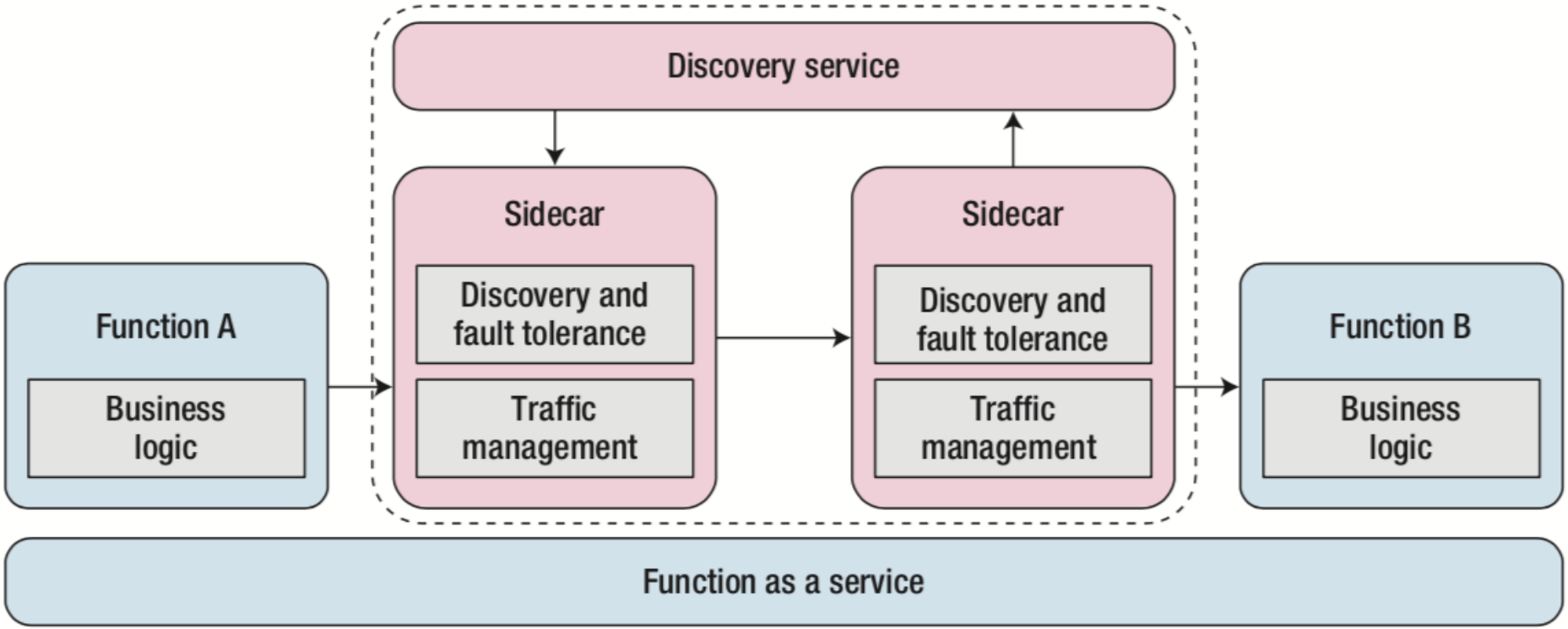
Microservices have evolved along four generations

- **Third generation**: introduction of standard service proxies, as transparent service intermediates



Microservices have evolved along four generations

- **Fourth generation**: function-as-a-service (FaaS) and serverless computing



Microservices aren't and will never be a silver bullet

- There are still many open questions, regarding:
  - Service modularisation and refactoring
  - Service granularity
  - Front-end integration
  - Resource monitoring and management
  - Failure, recovery, and self-repair
  - Organisational culture and coordination

That's all Folks!