Microservices

Building Blocks

Content

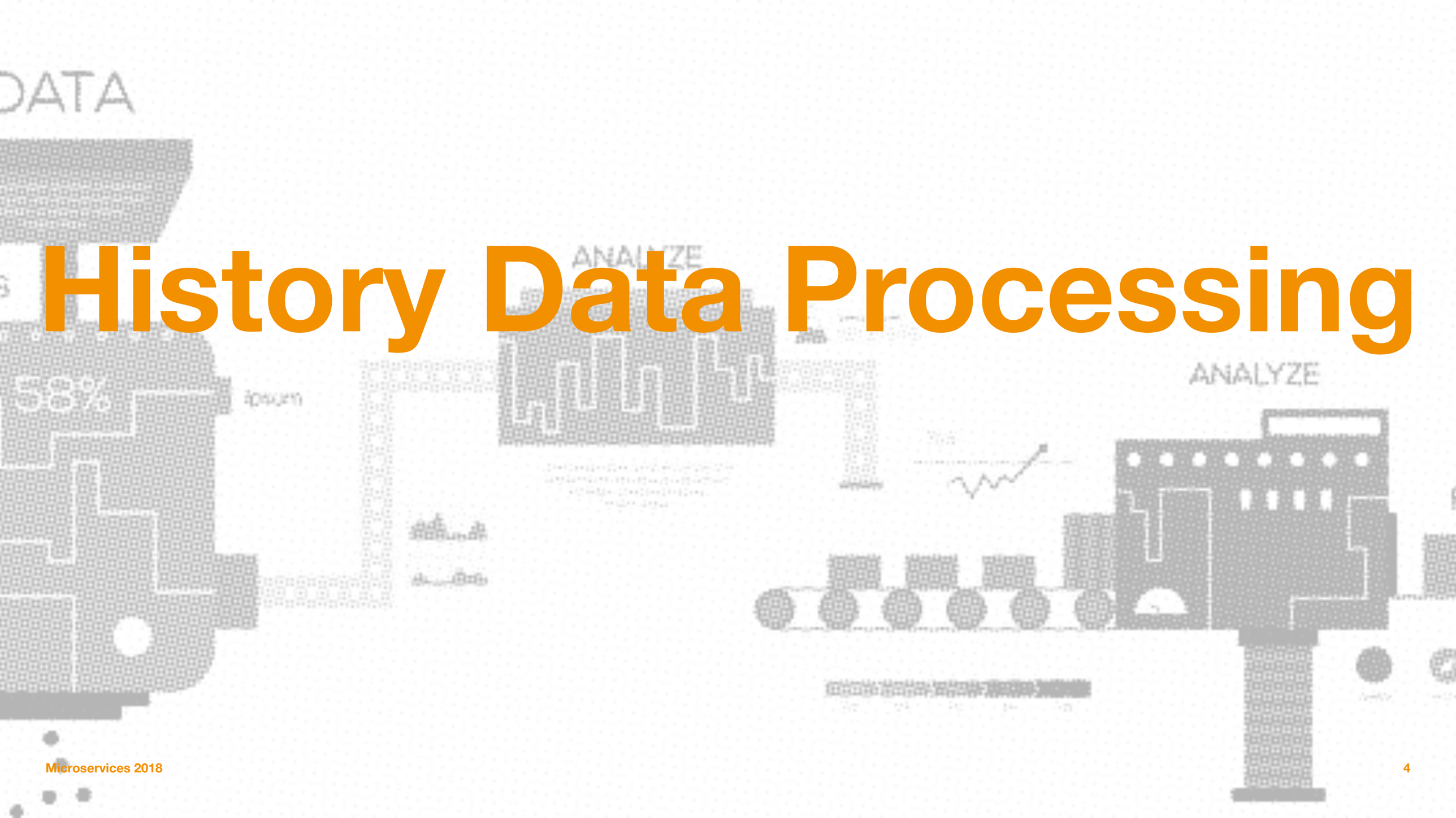
1. History of Data Processing & Services
2. Beyond Microservices
3. Monolithic vs. Microservice
4. Important Concepts

A Definition Data Processing / Services

Data processing is, generally, "the collection and manipulation of items of data to produce meaningful information." In this sense it can be considered as a subset of information processing, "the change (processing) of information in any manner detectable by an observer."

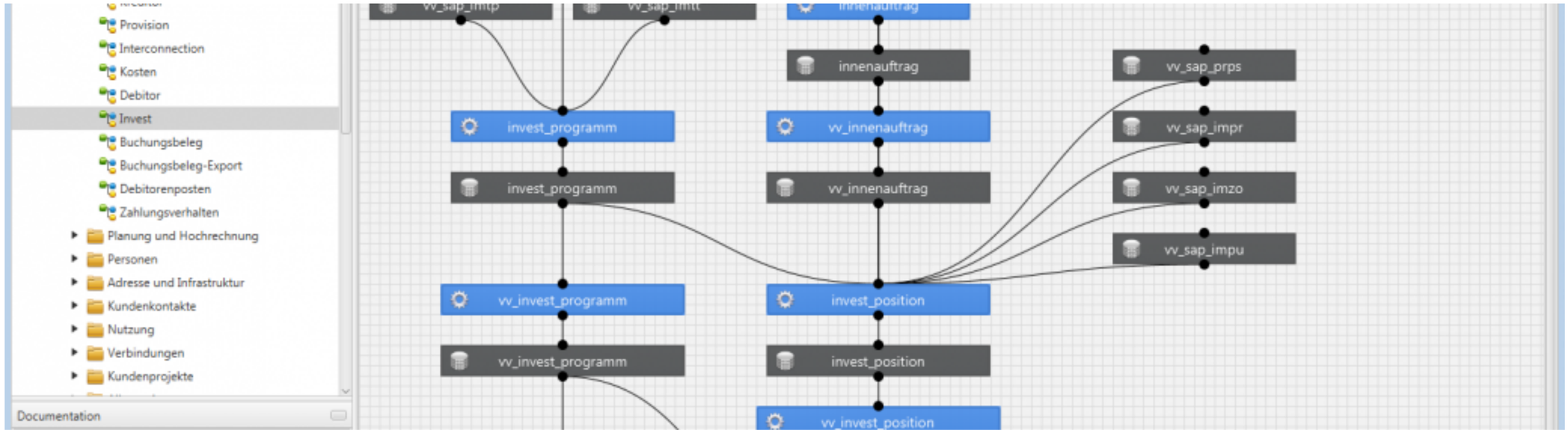
In the contexts of software architecture the term service refers to a software functionality or a set of software functionalities with a purpose that different clients can reuse for different purposes, together with the policies that should control its usage.

History Data Processing



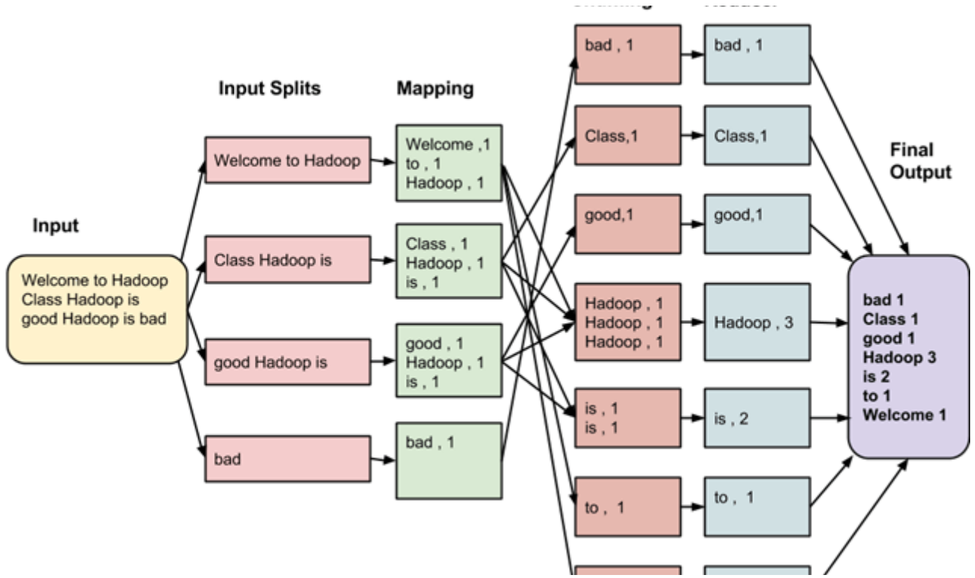
Offline Batch Processing

In computing, batch processing refers to a computer working through a queue or batch of separate jobs (programs) without manual intervention (non-interactive).



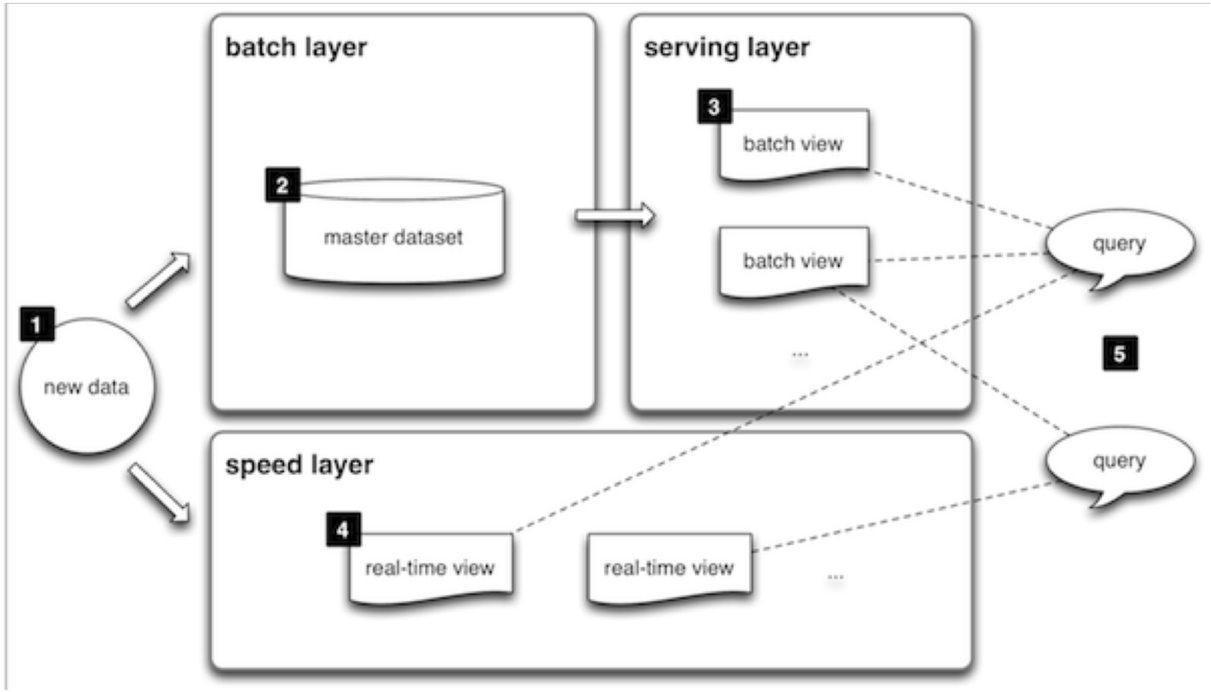
MapReduce

MapReduce is a programming model suitable for processing of huge data. Hadoop is capable of running MapReduce programs written in various languages.



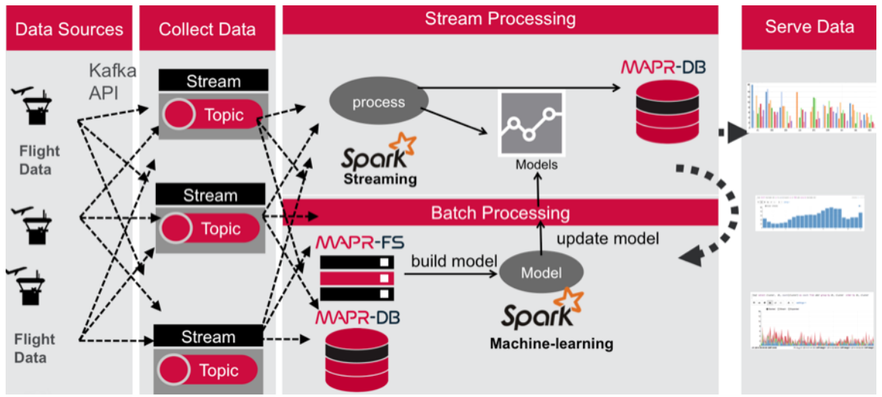
Lambda Architecture

The LA aims to satisfy the needs for a robust system that is fault-tolerant, both against hardware failures and human mistakes, being able to serve a wide range of workloads and use cases, and in which low-latency reads and updates are required. The resulting system should be linearly scalable, and it should scale out rather than up.

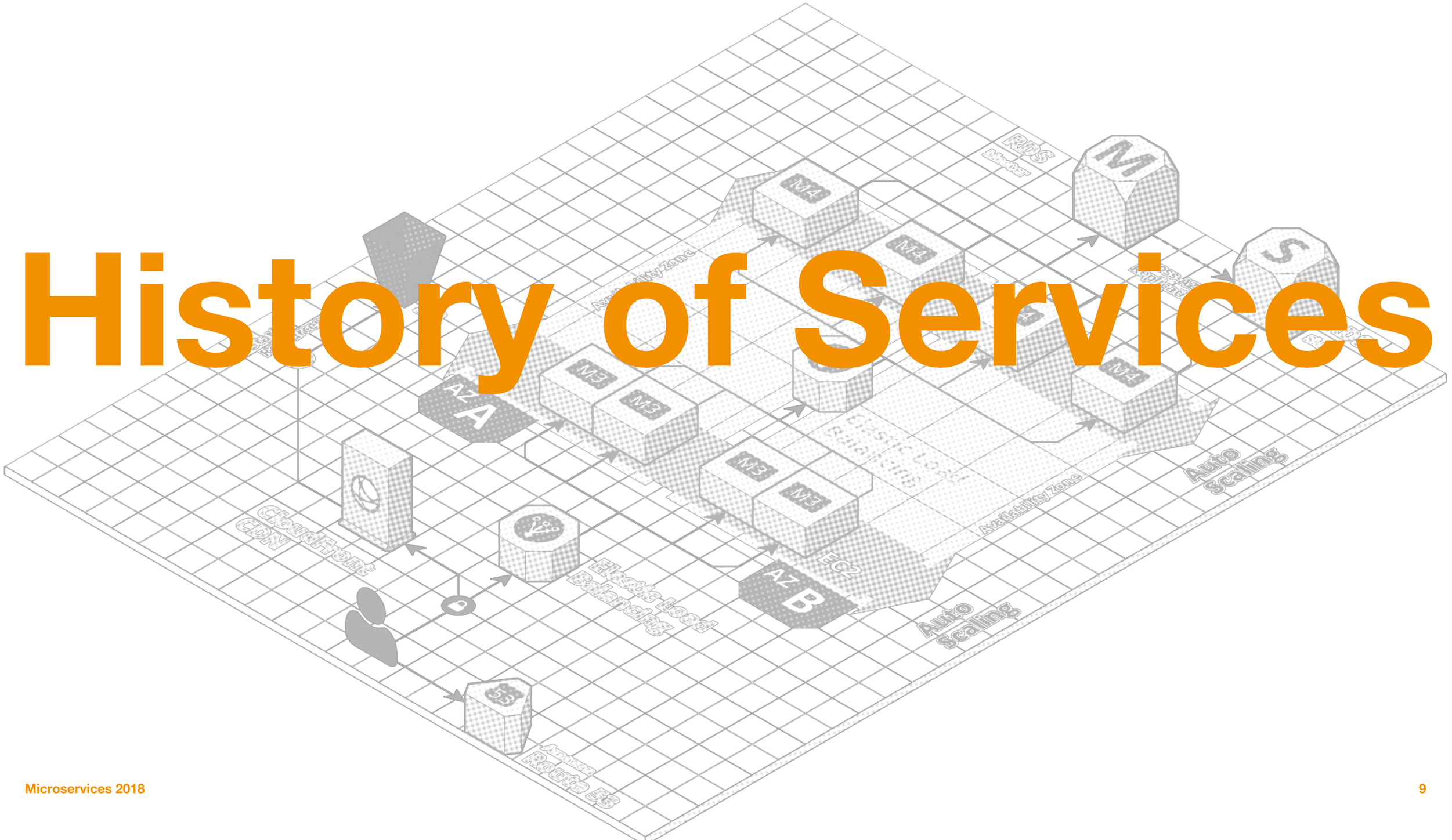


Live Stream Processing / Fast Data Architecture

Fast data is data in motion. It has different characteristics, different requirements and needs different technology to deal with it, technology that has the ability to analyze, decide, and act on – that is, offer recommendations, make decisions, or take other actions – as fast as data arrives, typically in milliseconds.

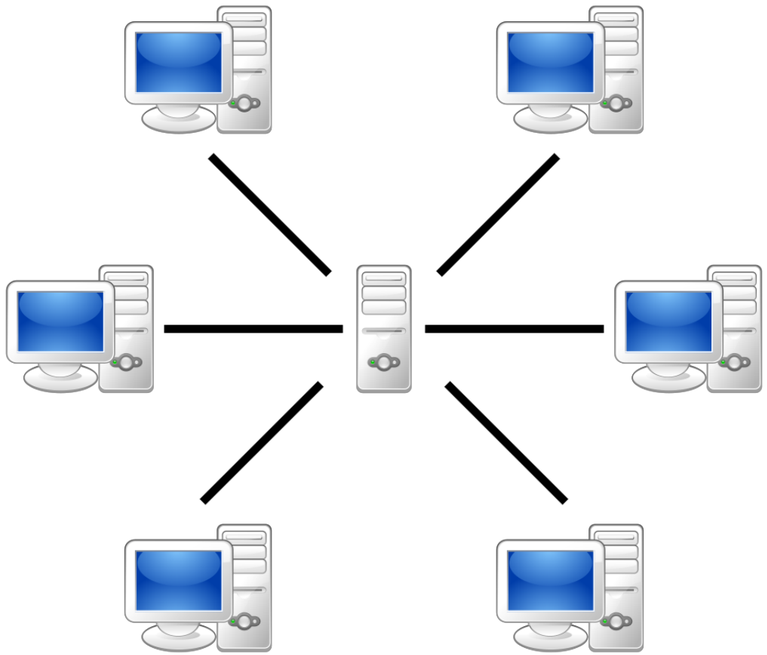


History of Services



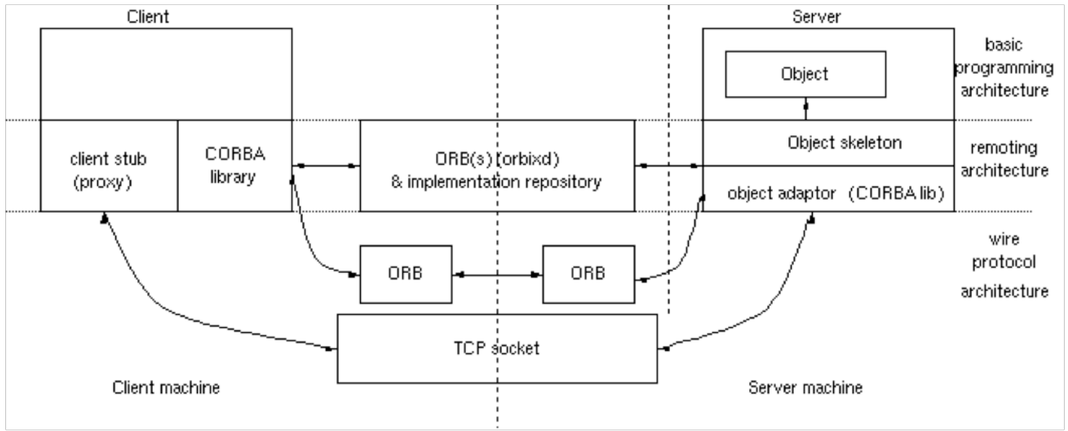
Client Server

e.g. FatClient / ThinClient 👀 VV Semester 4



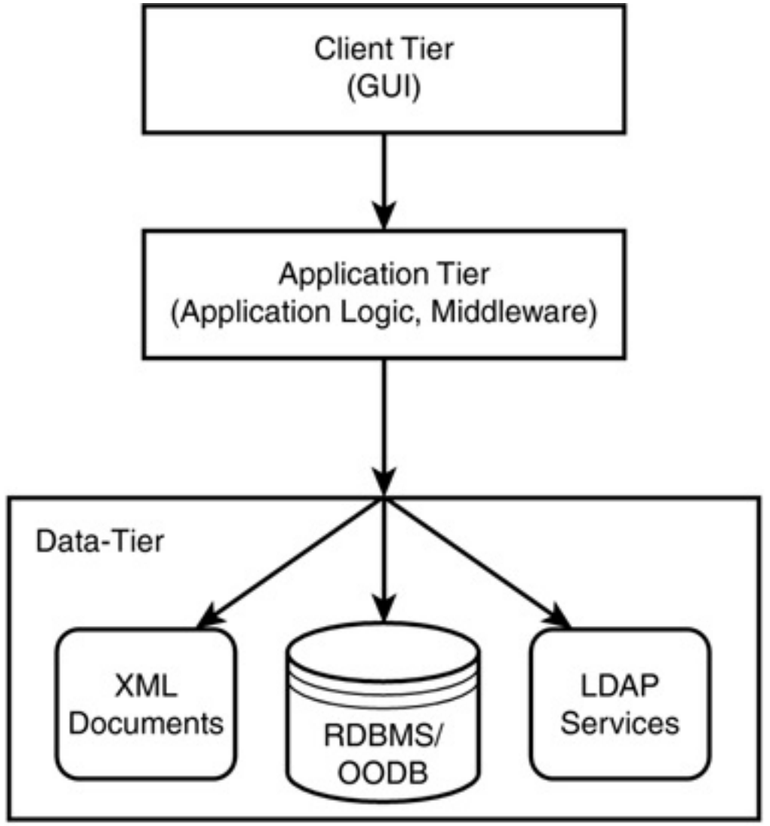
CORBA, DCOM & RMI

CORBA – Common Object Request Broker Architecture
DCOM – Distributed Component Object Model
Java/RMI – Java/Remote Method Invocation



EJB & N-Tier

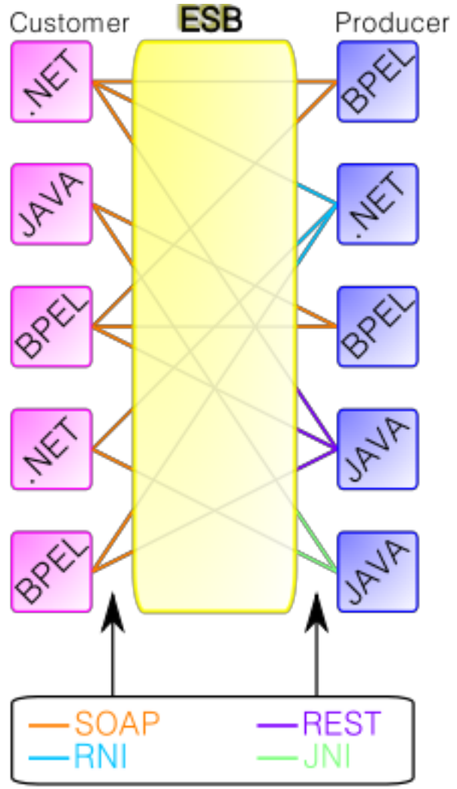
Enterprise Java Beans (EJB) is a development architecture for building ~~highly scalable~~ and robust enterprise level applications to be deployed on J2EE compliant Application Server such as JBOSS, Web Logic. EJB 3.0 is being a great shift from EJB 2.0 and makes development of EJB based applications quite easy.



3-Tier Architecture

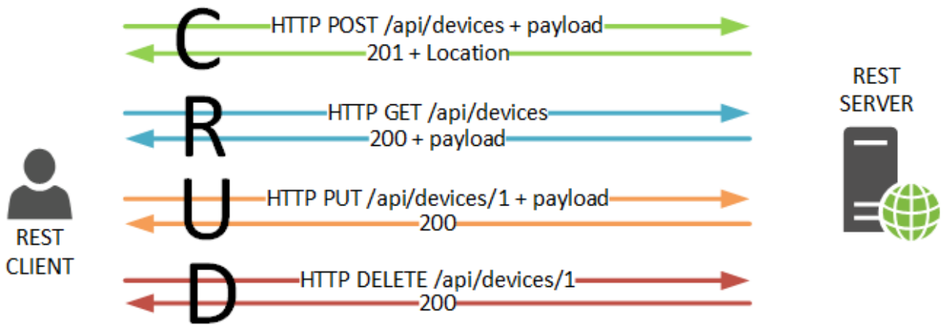
SOA & ESB

An enterprise service bus (ESB) implements a communication system between mutually interacting software applications in a service-oriented architecture (SOA). It implements a software architecture as depicted in the picture.



HTTP-API & REST

- Resources
  - Nouns, not Verbs
  - Coarse Grained, not Fine Grained
  - Architectural style for use-case scalability
- Keep it Simple
  - Collection Resource /users/
  - Instance Resource /users/1
- Behavior
  - GET, PUT, POST, DELETE ("CRUD"), Head
- Use HTTP Response Codes
- Use query params for offeet and limit

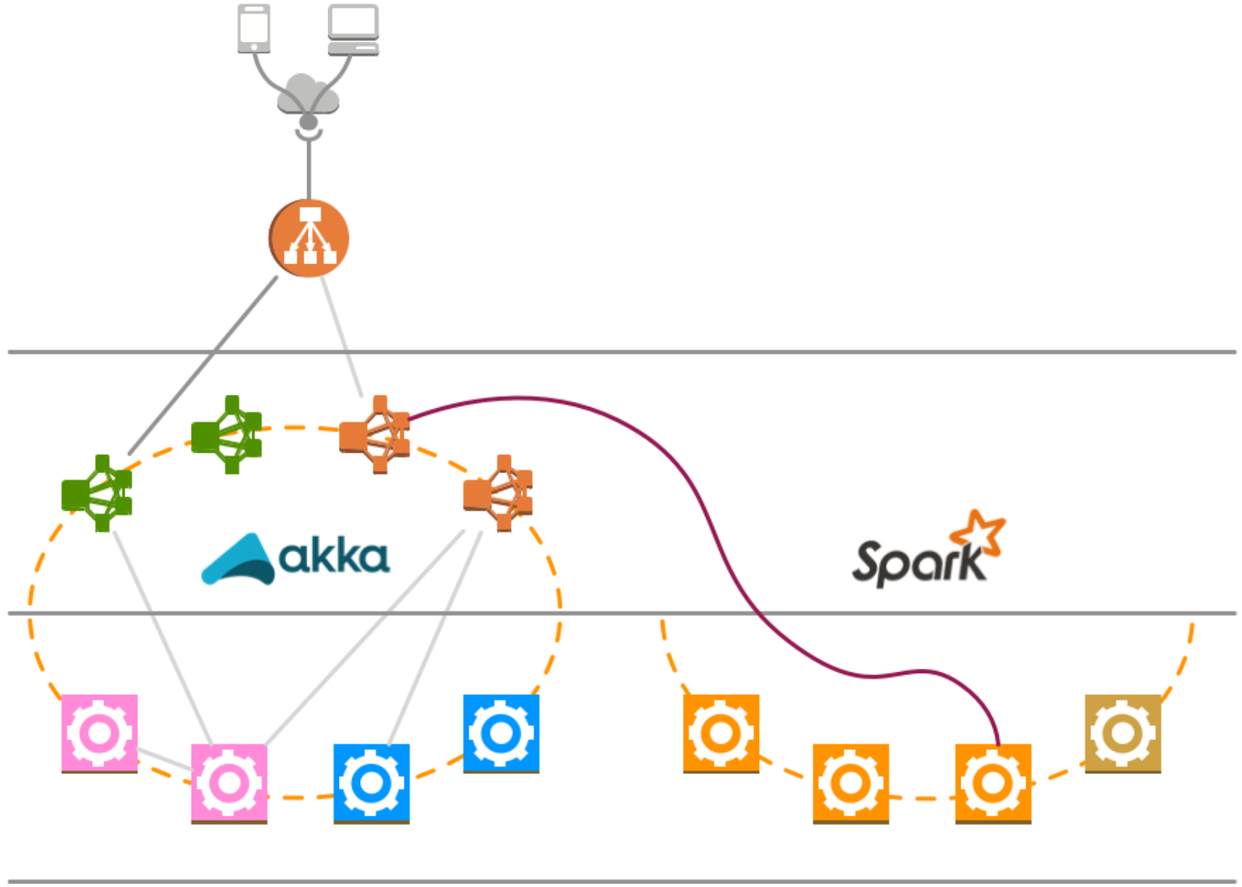


👀 VV Semester 4

Microservices

- A microservice is a software development technique that structures an application as a collection of loosely coupled services.
- It parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently

Microservices are NOT a Silver Bullet



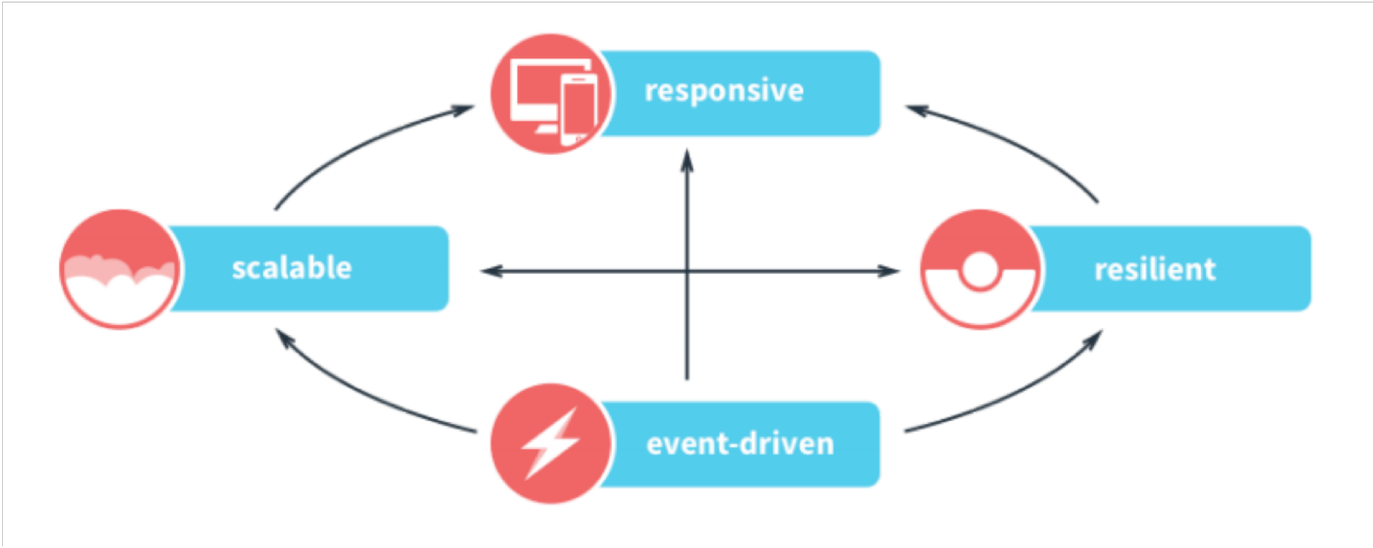
Beyond Microservices

Today Data Processing and Services have shared requirements:

- **Available**
- **Elastically scalable**
- **Resilient & self-healing**
- **Loosely coupley & upgradeable**

Reactive Systems[1]

Available (reponse), under failure (resilience) and load (elasticity), by being message-driven.



[1] The Reactive Manifesto.

Reactive Streams[2]

Provides a standard for asynchronous stream processing with non-blocking back pressure.

=> Fast Data Services are based on Reactive Streams.

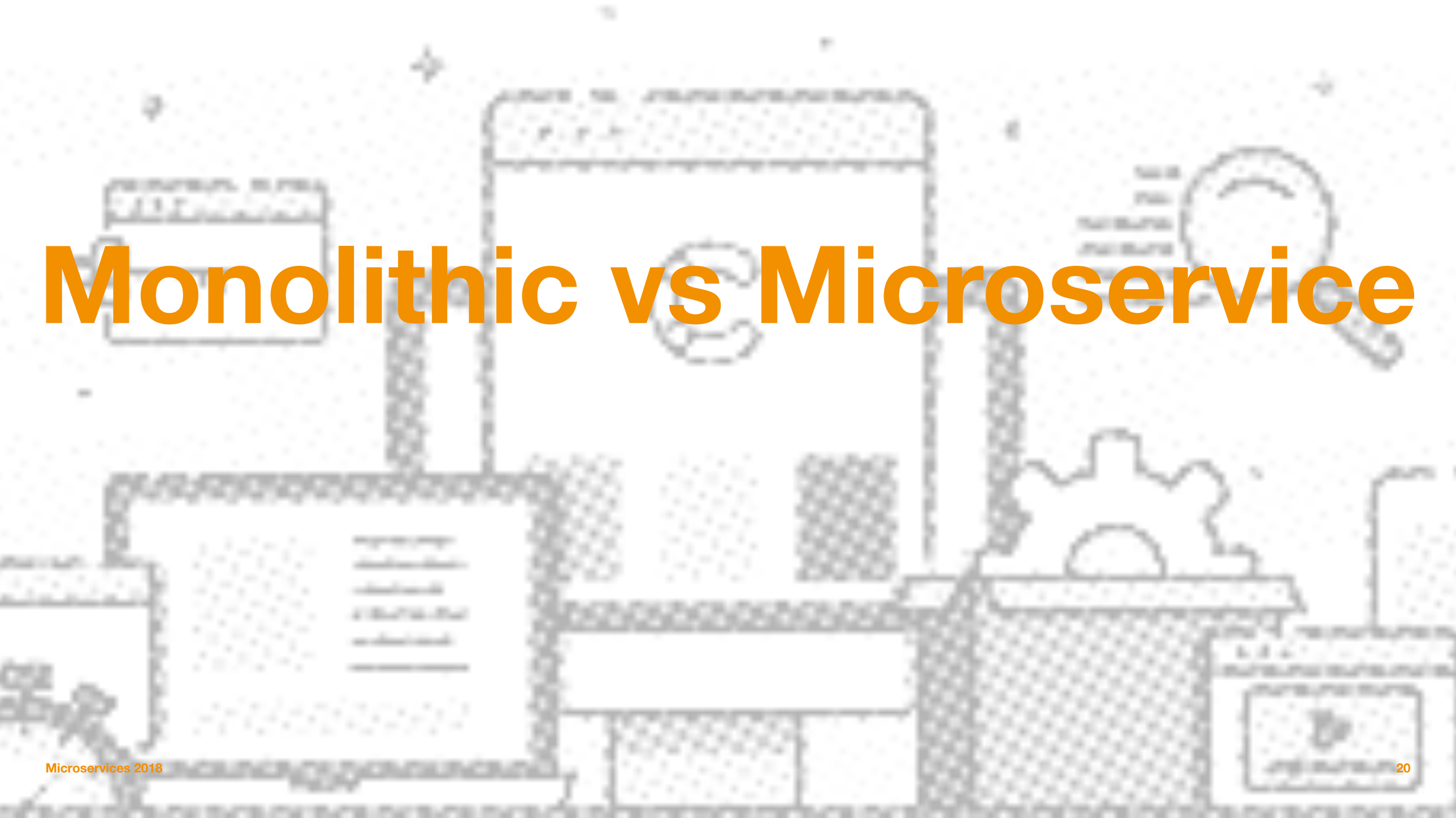
[2] Reactive Streams & java.util.concurrent.Flow JDK9+

Future: Convergence

- Evented Services are stream processing stages.
- Events-First DDD & EventStorming
- Data-driven Continuous Deployment
- User of the right tools for each component
- Microservices are a part of a streaming pipeline
- A pipeline can now be exposed as Microservices
- We can indepedently upgrade parts of the pipeline

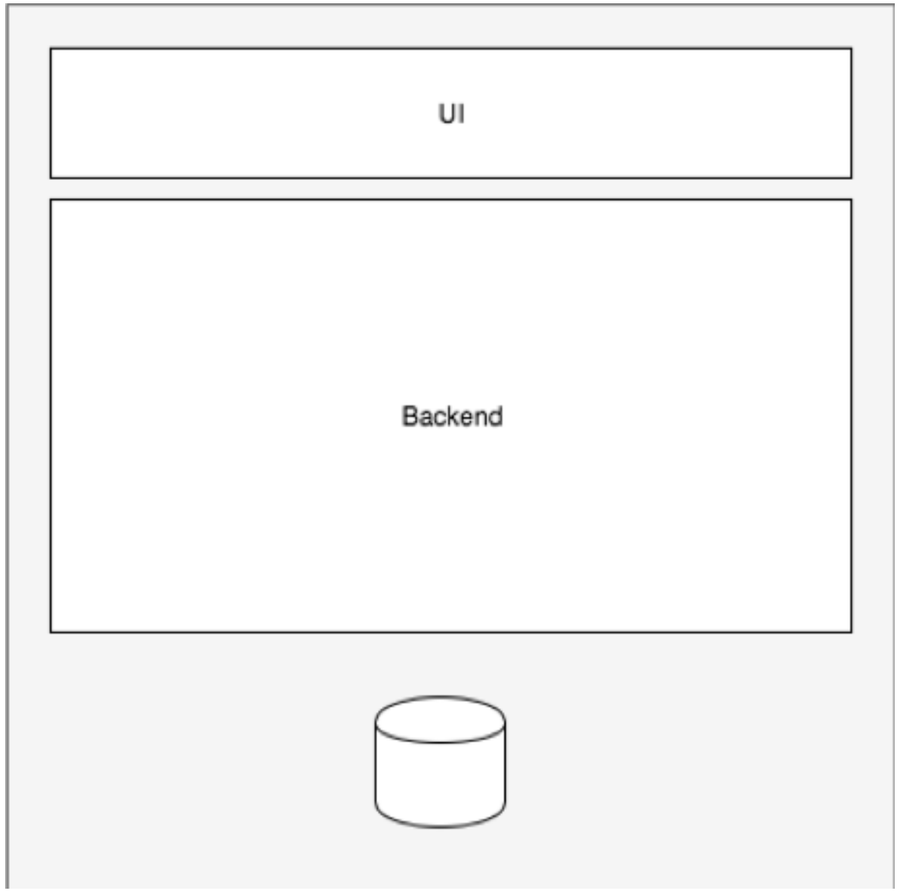
--> And then extend to the Edges! // Not this lecture!

Monolithic vs Microservice



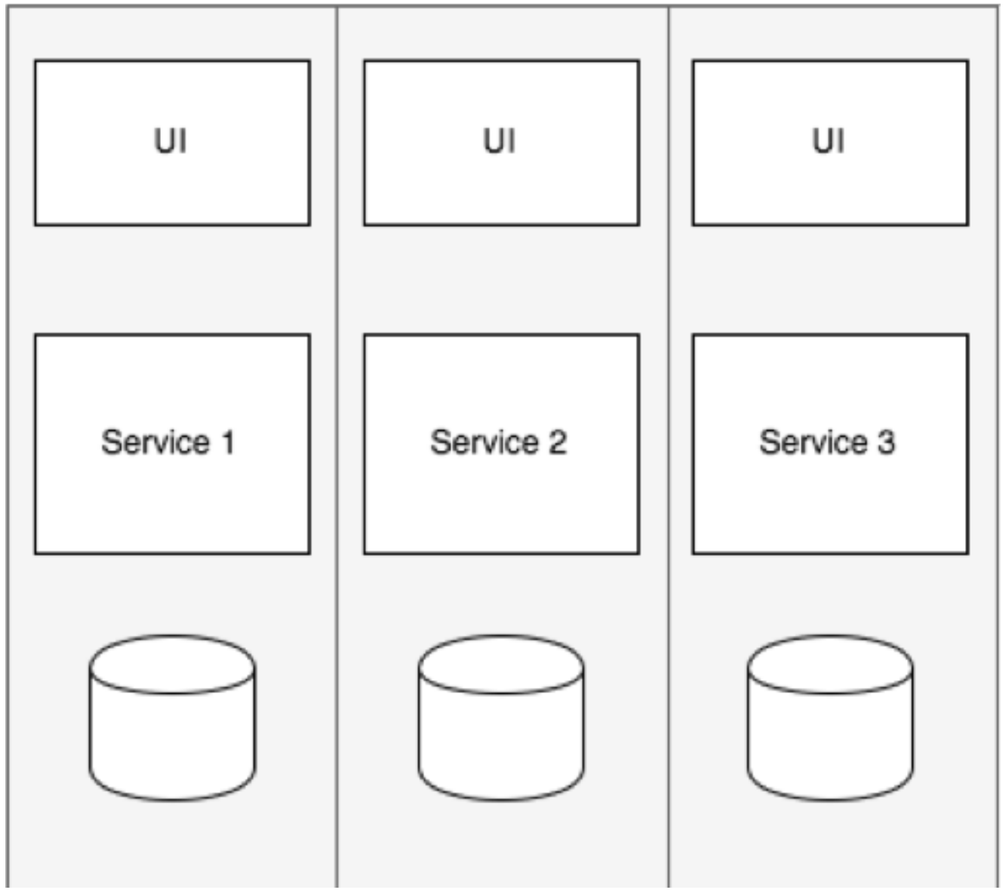
Monolithic vs. Microservice

- Most of the time just one huge project with one data source for all logic
- Scale out in monolithic systems is hard and often not possible



Monolithic vs. Microservice

- Services are sliced into small parts. Every service should use its own data source.
- Microservices are much more complex, but you can easily scale up and out.
- You need a high automation level

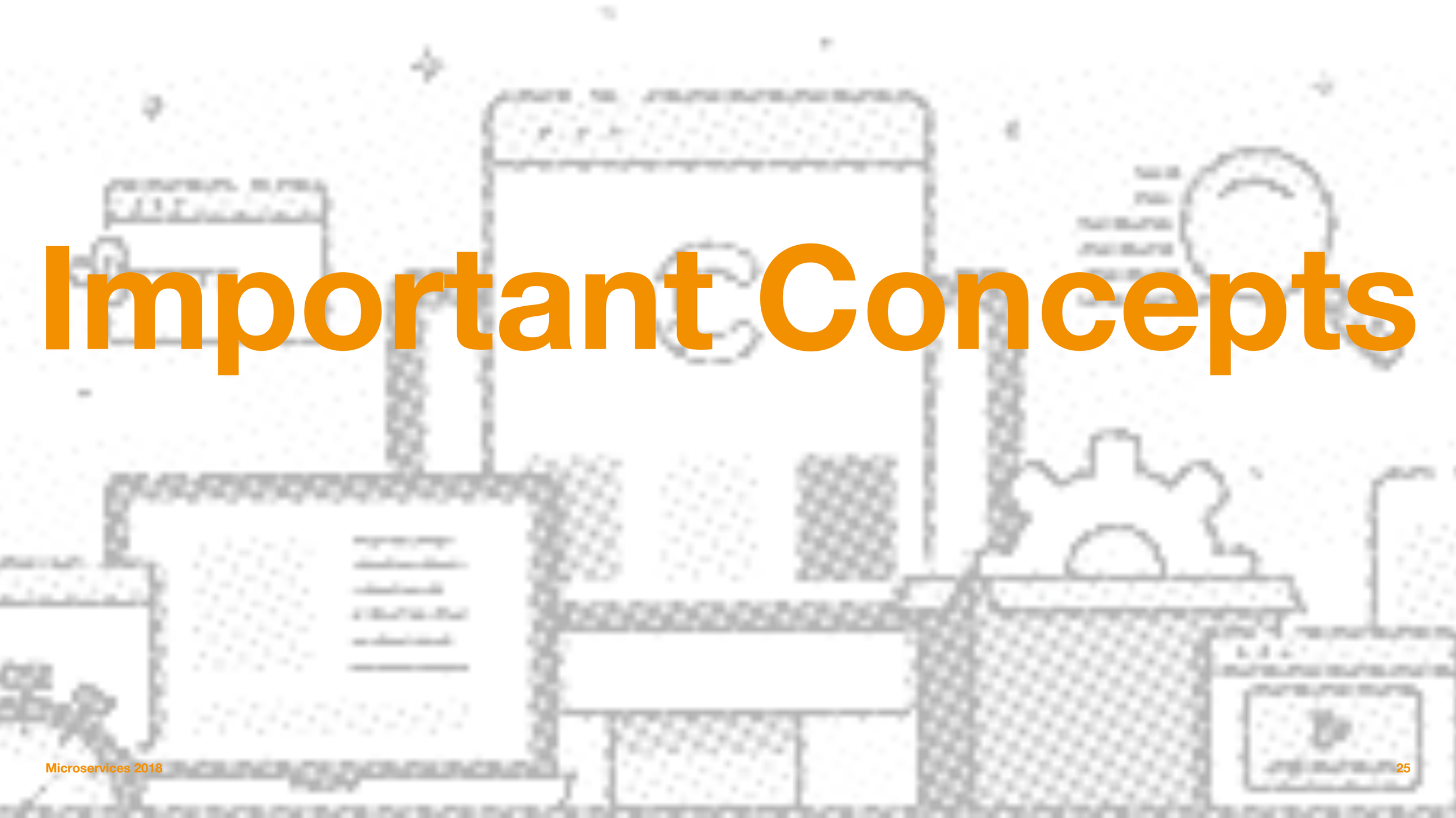


Pro Microservice

- scalable (up & out)
- replaceability
- inhomogeneous techstack
- bounded context
- loosely coupled
- Independent services & teams
- fast deployment and roll out
- caching
- highly automated

Con Microservice

- Good operation is hard
  - Monitoring
  - Logging
  - Deployment
  - Cluster-Setup
  - Storage
  - Backup
  - Recovery
- tech stack
- cascading failures
- latency



Important Concepts

Domain Driven Design (DDD)

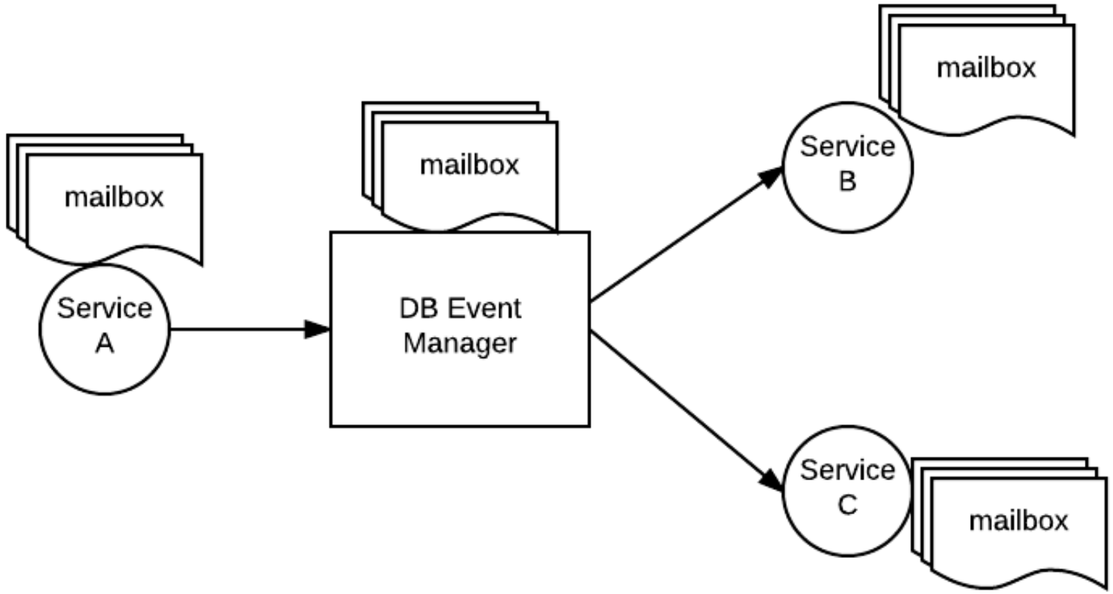
Define services corresponding to Domain-Driven Design (DDD) subdomains. DDD refers to the application's problem space as the domain.

A domain consists of multiple subdomains. Each subdomain corresponds to a different part of the business.

--> More about this Topic as external Talk

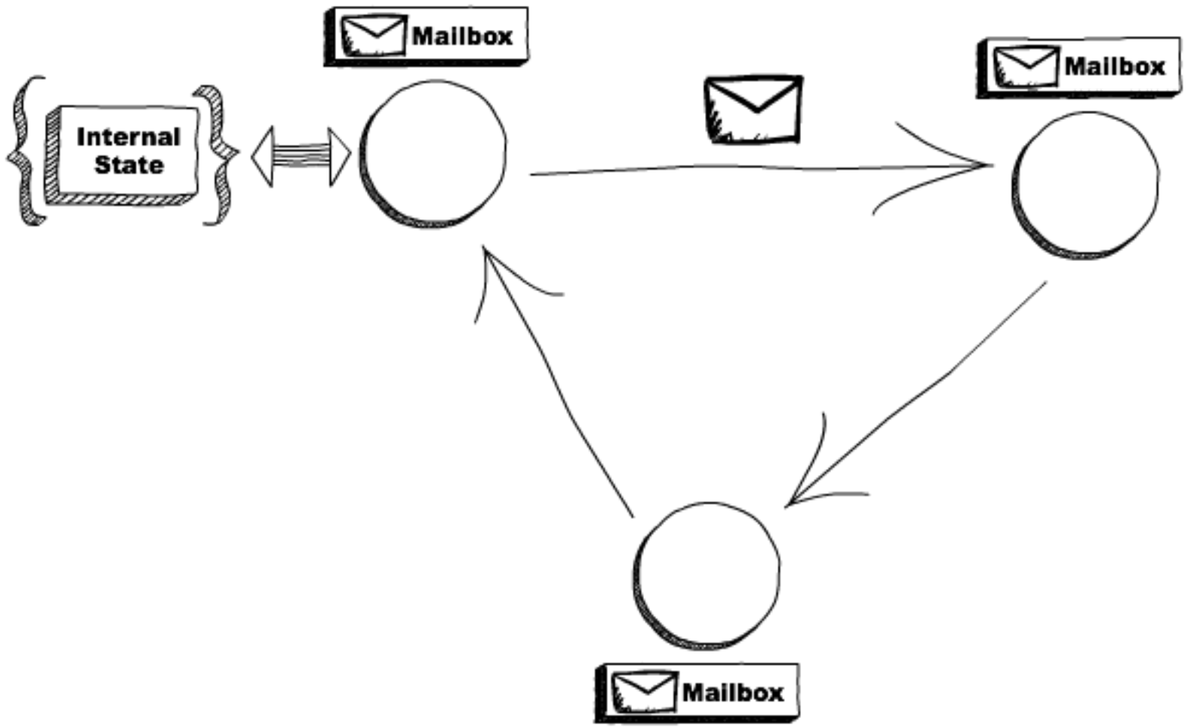
Event Driven

- flow of the program is determined by events
- Needed for data exchange between different services in a cluster
- A lot of programming languages use the actor pattern. akka is one implementation for scala and java
- Deep dive in lesson "Persistence" we will look at Event Sourcing and CQRS.



The Actor Model[3]

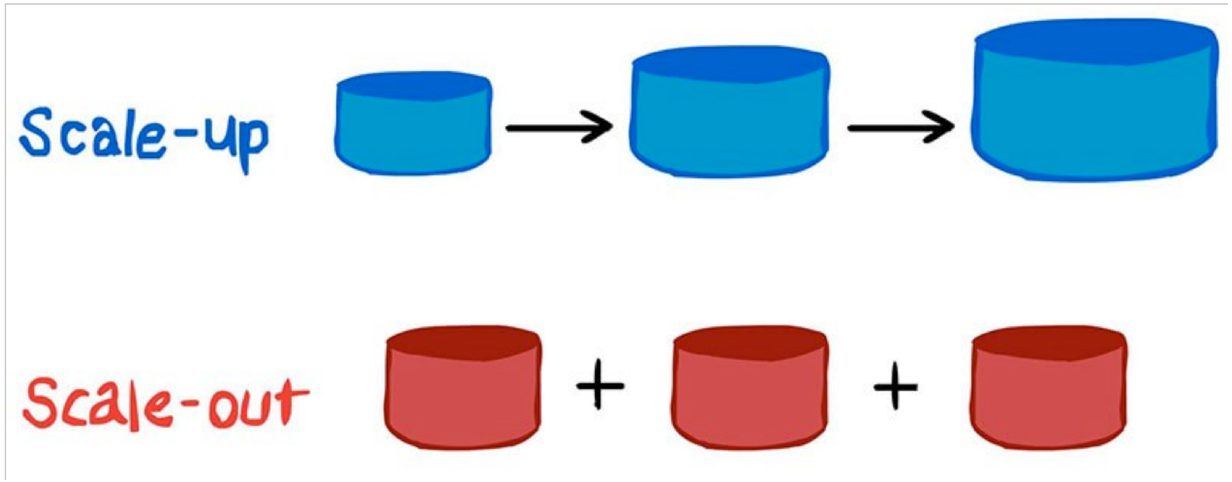
Actors communicate with each other by sending asynchronous messages. Those messages are stored in other actors' mailboxes until they're processed.



[3] The Actor Model - http://www.brianstorti.com/the-actor-model/

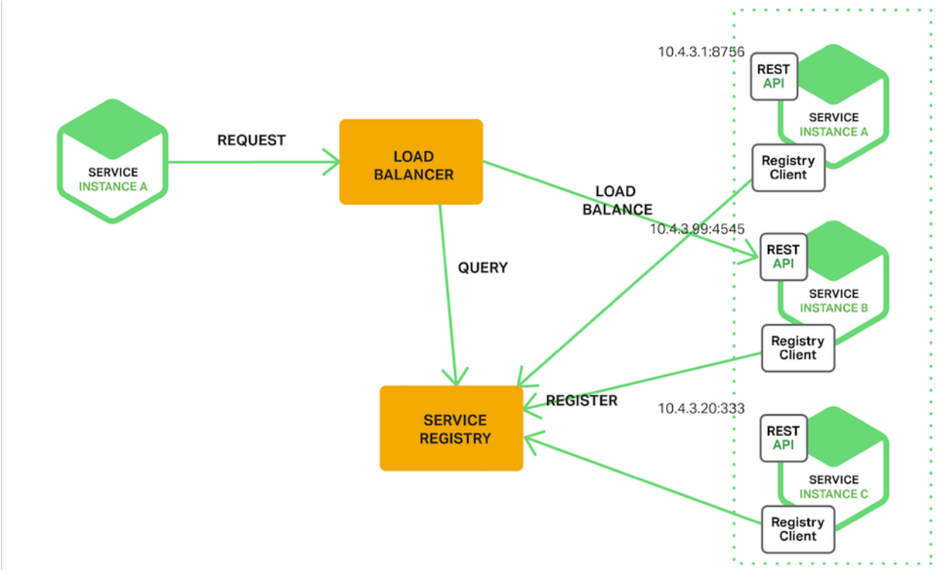
Scale Up & Scale Out

- Scale Up: Better Resources - e.g. CPU, RAM, Storage
- Scale Out: More Nodes



Load Balancer

How does the client of a service - the API gateway or another service - discover the location of a service instance? (Deep dive In lesson "Service Discovery & API Gateways")



Circuit Breaker

How to prevent a network or service failure from cascading to other services?

A service client should invoke a remote service via a proxy that functions in a similar fashion to an electrical circuit breaker.