

Microservices Architecture

Audience Course Microservices Architecture

The course Microservices Architecture course is intended for developers, architects and others who want to understand the why and how of a Microservices Architecture.

Prerequisites training Microservices Architecture

In order to participate in the course Microservices Architecture, general knowledge of software development and software design is desirable. Previous knowledge of modern programming languages is beneficial to understanding.

Realization course Microservices Architecture

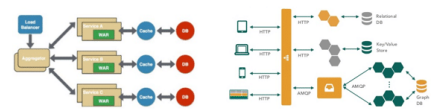
The theory is discussed on the basis of presentations. Illustrative demos are used to clarify the concepts. There is ample opportunity to practice and theory and practice are interchanged. Course times are from 9:30 to 16:30.

Certification Microservices Architecture

After successful completion of the course participants receive an official certificate Microservices Architecture.

Duration: 3 days
Price: € 1850
[Open Schedule](#)


Micro Service Architecture



Content Course Microservices Architecture

In the course Microservices Architecture participants learn the advantages of developing applications in a Microservices Architecture compared to a Monolithic Architecture. In the Microservices Architecture, small teams are responsible for development and deployment of the microservices, each of which can have its own database and user interface.

Intro Microservices

The course Microservices Architecture starts with a discussion of the problems of monolithic applications. Subsequently the characteristics of a Microservices Architecture such as independent deployability, independent scaling and separate databases are discussed.

Design Considerations

Next it is explained how according to Conway's law the architecture of applications is reflected in the organizational structure. The switch to a Microservices Architecture therefore requires adjustments in the organization. The principles of Domain Driven Design are also treated.

Interprocess Communication

Microservices are separate processes and communicate via interprocess communication. Attention is paid to synchronous communication via REST, asynchronous communication via messaging and communication via a binary protocol such as Thrift.

Micro Frontends

Then it is time for a discussion of the user interfaces of Microservices. Various approaches for integrating a user interface are discussed such as custom elements, server side templates and built-time integration of JavaScript libraries.

Data Management

Data management in a Microservices Architecture is also covered. The different patterns for data storage such as database per service, shared database, the saga pattern and event sourcing are discussed in this respect.

Discovery and Deployment

Finally options for service discovery are covered, such as client and server side discovery and the use of service registrars. Deployment options of microservices such as in virtual machines and in containers are also treated.

Modules Course Microservices Architecture

Module 1 : Intro Microservices	Module 2 : Design Considerations	Module 3 : Interprocess Communication
Microservices Architecture Monolithic Applications Software Monolith Problems of Monoliths Layered Architecture Growing Beyond Limits Microservices Characteristics Underlying Principles Independent Deployability Independent Scaling Separated Databases Size of Microservices Frontend Monolith Micro Frontend Architecture Blurry Service Boundaries	Conways's Law Law as Enabler Domain Driven Design Building Blocks Bounded Context Reactive Manifesto Reactive Microservices Microservices with UI Microservices Benefits Effective Modularization Replaceability Microservices Continuous Delivery Pipeline Free Technology Choice Team Independence Microservices and Languages	Communicating Processes Interaction Styles Service Communications Styles Defining API's Netflix Hystrix Asynchronous Messaging Messaging Models Publish and Subscribe Point to Point Advantages of Messaging Synchronous IPC REST Services Resource URI Access Thrift Content Negotiation
Module 4 : Micro Frontends	Module 5 : Reactive Microservices	Module 6 : Data Management
What are Micro Frontends? Monolith versus Micro Frontends Benefits Micro Frontends Delivery Pipeline per Service Avoid Shared Artifacts FrontEnd Integration ESI Composition Links and JavaScript Custom Elements Integration Approaches Server Side Templates Built Time Integration Shared Component Libraries Cross Application Communication Backend Communication	What is Reactive? Reactive Programming Reactive Extensions Observables Reactive Systems Elasticity and Resilience Reactive Microservices Asynchronous Development Event Loop Reactor Pattern Multireactor Pattern Verticles Callbacks vs Observables RxJava API Monitoring	Distributed Data Complex Data Access Polyglot Architecture Private Access Design Patterns Database per Service Shared Database Pattern Saga Pattern Event Publishing Consuming Events Responding to Events Base Model Transactions Local Transactions Database Transaction Log Event Sourcing
Module 7 : Service Discovery	Module 8 : Deployment Strategies	Module 9 : Security
Why Service Discovery Finding Services Client Side Discovery Service Registry Server Side Discovery Load Balancers Service Registries Self Registration Pattern Third Party Registration Pattern Service Registrars Netflix Eureka HashiCorp Consul	Deployment Patterns Virtual Machines Creating Virtual Machines Drawbacks of VM's Containers Containers versus VM's Container Orchestration Kubernetes Multiple Instances Per Host Service Instance per Host Service Instance per Container Serverless Deployment	Microservices Security Security Challenges Key Security Fundamentals Confidentiality Edge Security Securing with OAuth2 Authorization Server Interactions Actors OAuth2.0 Flow OAuth Roles Application Registration Securing with API Gateway Zuul Proxy and OAuth2