**Topic - Industrial Analysis with Microservices**

- Architecture / AR Patterns / Design Patterns
- Enterprise Solutions
- Top Software trends for 2024
- Monolith to Micro Migrations
- Service Mesh
- Service Fabric
- Kubernetes
- SECOND WAVE OF MICROSERVICES ADOPTION
- EDGE COMPUTING DRIVES MICROSERVICES
- WEB OF MICROSERVICES
- MICROSERVICES FAILURES

2005 – micro web services
2010 – microservices
Initially adopted by Netflix and Amazon

From monolith to microservices – 10 examples of global transitions

Amazon, Bestbuy.com, Coca Cola, eBay, Etsy, Gilt.com, Netflix, Spotify, Uber, Zalando

https://www.divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others

https://www.alliedmarketresearch.com/microservices-architecture-market

- How to start with microservices as a career option

Step -1
---
Basic Explanation - Theory - Software Archi Style - L1

Ref: Master+Microservices+with+SpringBoot,Docker,Kubernetes

https://spring.io/tools
https://www.jetbrains.com/idea/download/?section=windows

Enterprise approach for Software development - Enterprise Approach Software Development.ppt

Step -2
---

API – API - L1

REST API in action - with Spring Boot
Code1 - Basic – API.txt

Welcome_ Learning REST APIs -L1.mp4

Step -3
---

What Are Microservices? - What are Microservices - L1
(Definition Only)

Step -4
---

Platforms Developing Microservices - Technology for Microservices - L1

Step -5
---

Spring boot as the technology to develop microservices – Spring – L1

Step -5
---

Microservices examples - IntelliJ
Payments and Order microservices
Recording 3 – 27th minute onwards

Step -6
---
Enterprise Solutions - Software Archi Style - L1
(Bit Detail)
Recoding -4 – first 10 minutes
Enterprise Approach Software Development.ppt

Step -7
---

Spring – MVC - Spring MVC – L1

Video – 4 – from minute 11

Step -8
---

Migrating Spring Boot's Build to Gradle - https://spring.io/blog/2020/06/08/migrating-spring-boot-s-build-to-gradle

https://github.com/spring-projects/spring-boot/tree/main/spring-boot-project/spring-boot-starters/spring-boot-starter-web

Step -9
---
Video 4 – minute 16 onwards

Why Migrate from Monolith to microservices?

- The pain of monolith Architecture, what is it? understand the disease of monolith before start treatment

1. Testing the monolith

One change, just adding a column to a table - just a simple change in a single API will cause to deploy the entire regression testing suite and deploy the testing team to test each and every API
Even though no changes in any other controller
this is same with implementing new features
with Microservices time to market will be reduced.

2. Scalability

As the architecture of microservices is based on small components, scaling is so much easier than in a monolith architecture. This means that when big changes happen in a certain component, the others remain untouched and run correctly. This is a great approach for companies that work on a large number of devices and applications. This can be achieved by designing CI / CD pipelines with Kubernetes.

3. Flexibility

The microservices approach doesn't limit developers in terms of tools and technologies. Each service can be built with a different framework and language. That simplifies the selection of the most suitable technology for your service's specific needs. Because of the independence in design and implementation the fact that components are different doesn't affect the work of other microservices or the communication between them by no means.

4. Increased productivity

Microservices are easy to develop and this process doesn't take up too much time. They allow several teams to work on different components from the same app without affecting the work of other services. By introducing an DevOps Culture Team Silos can be further break down, there's no need for waiting for one team to finish so that the other can start working. Moreover, microservices can be easily moved and modified. This allows testers to test separate services individually and get more done in a shorter period of time, without having to wait for the entire application to be ready.

5. Easier communication in distributed teams

Since it is lousily coupled architecture remote teams can function much easier if they are working with microservices architecture. Not like the interdependency between team when developing monolith application, Microservices give developers the freedom to work independently without having to wait for the entire team to make every decision.

6. Maintenance

Debugging an application has finally become pain-free. Due to the smaller size of microservices, detecting an error and repairing it gets done in no-time. Teams have more control over these small instances of code and the probability to deliver error-free applications is larger.

Let's say one of the API's having a Bug, Like Memory Leak
Just think about a pricing change in an Ecommerce application
Big Business impact, since u need to spend time to find which API causing the issue

with Microservices API's works independently with their own resources

Companies that are growing where it requires highly scabble and load balancing in applications should seriously consider microservices as an architecture. The most common reasons teams migrate to microservices are scalability and productivity challenges. Projects that are growing require more developers, so allowing them to focus on a single service of the application makes things much easier.

Monolith – Micro Migrations – Example

https://virtusa.udemy.com/course/master-microservices-with-spring-docker-kubernetes/learn/lecture/26024406#overview

Step -10
---

Containerization - Enterprise Solutioning with Containerization of Microservices - L1