Challenges of Monolithic Architecture:

Scalability,

Monolith Architecture:

• The original architecture

• All software components are executed in a single process

• No distribution of any kind

• Strong coupling between all classes

• Usually implemented as Silo

Monolith Architecture Pros:

• Easier to design

• Performance

Service Oriented Architecture:

• First coined in 1998

• Apps are services exposing functionality to the outside world

• Services expose metadata to declare their functionality

• Usually implemented using SOAP & WSDL

• Usually implemented with ESB

SOA Pros:

• Sharing Data & Functionality

• Polyglot Between Services

With monolith, all the components must be developed using the

same development platform

• Not always the best for the task

• Can’t use specific platform for specific features

Characteristics of Microservices:

1.Componentization via Services:

• Modular design is always a good idea

• Components are the parts that together compose the software

• Modularity can be achieved using:

• Libraries – called directly within the process

• Services – called by out-of-process mechanism (Web API, RPC)

• In Microservices we prefer using Services for the componentization

• Libraries can be used inside the service

2.Organized Around Business Capabilities:

• Traditional projects have teams with horizontal responsibilities –UI, API, Logic, DB etc

• With Microservices, every service is handled by a single team, responsible for all aspects.

• With Microservices, every service handles a well-defined business capability.

3.Products not Projects:

• With traditional projects, the goal is to deliver a working code

• No lasting relationship with the customer

• Often no acquaintance with the customer

• After delivering – the team moves on to the next project

4.Smart Endpoints and Dumb Pipes:

• Traditional SOA projects used two complicated mechanisms:

• Made inter-service communication complicated and difficult to maintain

• Microservices systems use “dumb pipes” – simple protocols

• Strive to use what the web already offers

• Usually – REST API, the simplest API in existence

Decentralized Governance: With Microservices each team makes its own decisions , Which dev platform to use, Which database to use, How logs are created

Decentralized Data Management

Infrastructure Automation

Design for Failure

Evolutionary Design

Microservices

What are Microservices?

Definition: According to Sam Newman, "Microservices are the small services that work together."

According to James Lewis and Martin Fowler, "The microservice architectural style is an approach to develop a single application as a suite of small services. Each microservice runs its process and communicates with lightweight mechanisms. These services are built around business capabilities and independently developed by fully automated deployment machinery."

There is a bare minimum of centralized management of these services, which may be written in different programming language and use different data storage technologies.

Points to remember

• These are the services which are exposed by REST.

• These are small well-chosen deployable units.

• The services must be cloud-enabled.

The microservice defines an approach to the architecture that divides an application into a pool of loosely coupled services that implements business requirements. It is next to Service-Oriented Architecture (SOA). The most important feature of the microservice-based architecture is that it can perform continuous delivery of a large and complex application.

Microservice helps in breaking the application and build a logically independent smaller applications. For example, we can build a cloud application with the help of Amazon AWS with minimum efforts.

principles of Microservices

There are the following principles of Microservices:

• Single Responsibility principle

• Modelled around business domain

• Isolate Failure

• Infrastructure automation

• Deploy independently

Single Responsibility Principle

The single responsibility principle states that a class or a module in a program should have only one responsibility. Any microservice cannot serve more than one responsibility, at a time.

Modeled around business domain

Microservice never restrict itself from accepting appropriate technology stack or database. The stack or database is most suitable for solving the business purpose.

Advantages of Microservices

• Microservices are self-contained, independent deployment module.

• The cost of scaling is comparatively less than the monolithic architecture.

• Microservices are independently manageable services. It can enable more and more services as the need arises. It minimizes the impact on existing service.

• It is possible to change or upgrade each service individually rather than upgrading in the entire application.

• Microservices allows us to develop an application which is organic (an application which latterly upgrades by adding more functions or modules) in nature.

• It enables event streaming technology to enable easy integration in comparison to heavyweight interposes communication.

• Microservices follows the single responsibility principle.

• The demanding service can be deployed on multiple servers to enhance performance.

• Less dependency and easy to test.

• Dynamic scaling.

• Faster release cycle.

Disadvantages of Microservices

• Microservices has all the associated complexities of the distributed system.

• There is a higher chance of failure during communication between different services.

• Difficult to manage a large number of services.

• The developer needs to solve the problem, such as network latency and load balancing.

• Complex testing over a distributed environment.

Microservices Monitoring

Monitoring is the control system of the microservices. As the microservices are more complex and harder to understand its performance and troubleshoot the problems. Given the vivid changes to software delivery, it is required to monitor the service. There are five principles of monitoring microservices, as follows:

• Monitor container and what's inside them.

• Alert on service performance.

• Monitor services that are elastic and multi-location.

• Monitor APIs.

• Monitor the organizational structure.

These principles allow us to address technological changes associated with the microservices and organizational changes related to them.

Microservices Monitoring Tool

There are three monitoring tools are as follows:

• Hystrix dashboard

• Eureka admin dashboard

• Spring boot admin dashboard

Microservice Virtualization

Microservices virtualization is the method to simulate the behavior of specific components in various component-based application like cloud-based application, SOA, and API driven architecture. Service virtualization also reduces cost and save time. By combining service virtualization, an organization can develop the application which can be delivered from various locations and dissimilar environments