When should you pick Microservices Architecture?

In this lesson, you will learn about the pros and cons of microservice architecture and when to pick it for your project.

We'll cover the following

- Pros of microservice architecture
  - No Single Points of failure
  - Leverage the heterogeneous technologies
  - Independent and continuous deployments
- Cons of microservices architecture
  - Complexities in management
  - No strong consistency
- When should you pick a microservices architecture?

Pros of microservice architecture#

No Single Points of failure#

Since microservices is a loosely coupled architecture, there is no single point of failure. Therefore, even if a few of the services go down, the application as a whole is still up.

Leverage the heterogeneous technologies#

Every component interacts with each other via a *REST API Gateway interface*. The components can leverage the polyglot persistence architecture and other heterogeneous technologies together like *Java*, *Python*, *Ruby*, *NodeJS*, etc.

Polyglot persistence uses multiple database types, like *SQL* and *NoSQL* together in an architecture. We will discuss this in detail in the database lesson.

Independent and continuous deployments#

The deployments can be independent and continuous. We can have dedicated teams for every microservice, and it can be scaled independently without impacting other services.

Cons of microservices architecture#

Complexities in management#

Microservices is a distributed environment where there are so many nodes running together. As a result, managing and monitoring them gets complex.

We need to set up additional components to manage microservices such as a node manager like *Apache Zookeeper*, which is a *distributed tracing* service for monitoring the nodes etc.

We need more skilled resources and maybe a dedicated team to manage these services.

No strong consistency#

Strong consistency is hard to guarantee in a distributed environment. Things are *eventually consistent* across the nodes, and this limitation is due to the distributed design.

We will discuss both strong and eventual consistencies in the database chapter.

When should you pick a microservices architecture?#

The microservice architecture fits best for complex use cases and for apps that expect traffic to increase exponentially in the future, like a fancy social network application.

A typical social networking application has various components such as messaging, real-time chat, LIVE video streaming, image uploads, Like and Share features, etc.

In this scenario, I would suggest developing each component separately, keeping the *Single Responsibility* and the *Separation of Concerns* principles in mind.

Writing every feature in a single codebase would take no time to become

Writing every feature in a single codebase would take no time to become a mess.

So, by now, we have gone through three approaches in the context of monolithic and microservices:

1. Picking a monolithic architecture
2. Picking a microservice architecture
3. Starting with a monolithic architecture and later scaling out into a microservice architecture.

Picking a monolithic or a microservice architecture largely depends on our use case.

I'll suggest keeping things simple and having a thorough understanding of the requirements. Get the lay of the land, build something only when you need it, and keep evolving the code iteratively. This is the right way to go.

← Back

What is Microservice Architecture?

Next →

Monolith and Microservices– Understa...

Mark as Completed

Report an Issue