Back To Module Home

Architecture of Scalable Applications

0% completed

Different Tiers in Software Architecture

Monolith and Microservices

Monolith and Microservices Quiz

Conclusion

Mark Module as Completed

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#

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

Leverage the heterogeneous technologies#

Every microservice interacts with each other via a REST API gateway interface. A system with microservices can leverage the polyglot persistence architecture and other heterogeneous technologies like *Java, Python, Ruby, NodeJS,* etc.

Polyglot persistence uses multiple database types, like *SQL* and *NoSQL*, together in the 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 they can be scaled independently without impacting other services.

Cons of microservices architecture#

Management complexity#

Microservices is a distributed environment with several services powered by clusters of servers. This makes system management and monitoring complex.

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

We need skilled resources and even have to set up a dedicated team just to manage these services.

Strong consistency#

Sometimes, *strong consistency* is hard to guarantee in a distributed environment, especially when trying to achieve a single consistent state across several microservices.

Things are eventually consistent across the nodes, and this limitation is due to the distributed design.

In the database chapter, we will discuss both *strong* and *eventual consistency* in detail.

When should you pick a microservices architecture?#

The microservice architecture fits best for complex use cases; for apps that need to expand quick from adding new features standpoint. A social network application is a good example of a complex use case.

A typical social networking application has various components such as *messaging, real-time chat, LIVE video streaming, photo uploads, post like and share features,* etc.

This use case fits best for a microservices architecture. Also, the microservices architecture enables a business move fast. A business can separately develop, test, deploy an application feature without affecting the current features. There is not much need for *regression testing* and so on.

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

So, by now, we have seen three ways to proceed with the design of our application:

- Picking a monolithic architecture
- Picking a microservice architecture
- 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

suggest keeping things simple and thoroughly understanding the requirements before deciding on an architecture.

In the next lesson, let's understand the trade-offs involved when choosing *monolith* and *microservices* architecture.

What is Microservice Architecture?

Next

Monolith and Microservices– Underst…

Mark as Completed

Report an Issue