**MICROSERVICES**

**INTRODUCTION:**

Software architecture is a high level structure for the development of software systems. As software becomes more common and widespread, different architectural styles are evolved. There are so many software architecture in which microservices architecture has picked up a lot of grip and importance.

**BACKGROUND:**

Before the invention of microservices there were monolithic architectures.Monolith means composed all in one piece. Any application consists of a client side interface, a database for storing and handling data, and server side logic. A monolithic application is one single unit. In a monolithic application, server side logic is pushed together in the form of a monolith, with all the logic for handling requests running in a single process. Monolithic software only supports one programming language. In monolithic architecture if application is too large and complex to understand, it is challenging to make changes fast and correctly. The size of the application can slow down the start-up time. Redeploy the entire application on each update must be needed.Monolithic applications can also be challenging to scale when different modules have conflicting resource requirements.

**MICROSERVICES:**

Microservices or microservice architecture is an architectural style that structures an application as a collection of services that are highly maintainable,loosely coupled and independently deployable. It also enables the continuous delivery and deployment of large and complex applications. Microservices are small in size so adding new features are very easy and fast. Developers can work on different microservices autonomously. Microservices solve these challenges of monolithic systems by being as modular as possible.They help build an application as a suite of small services, each running in its own process and are independently deployable. These services may be written in different programming languages and may use different data storage techniques. While this results in the development of systems that are scalable and flexible. Microservices are often connect via APIs. Microservices solve these challenges of monolithic systems by being as modular as possible. Microservices have many benefits for Agile and DevOps team as Netflix, eBay, Amazon, Twitter, PayPal, and other tech stars that have evolved from monolithic to microservices architecture. A monolith application is built as a single, autonomous unit. In microservices if error is fixed, it can be deployed only for the respective service instead of redeploying an entire application.

**COMPARISON BETWEEN MONOLITHIC AND MICROSERVICES ARCHITECTURE:**

A picture containing screenshot

Description automatically generated

**BENEFITS :**

* **Improved fault isolation**: Larger applications can remain mostly unaffected by the failure of a single module.
* **Flexibility**: Microservices provide the flexibility to try out a new technology stack on an individual service as needed. There won’t be as many dependency concerns and rolling back changes becomes much easier. With less code in play, there is more flexibility.
* **Ease of understanding:**With added simplicity, developers can better understand the functionality of a service.
* **Smaller and faster deployments**: Smaller codebases and scope means quicker deployments, which also allow to start observing the benefits of Continuous Deployment.
* **Scalability**: Since your services are separate, you can more easily scale the most needed ones at the appropriate times, as opposed to the whole application. When done correctly, this can impact cost savings.

**DRAWBACKS:**

* **Communication between services is complex**: Since everything is now an independent service, you have to carefully handle requests traveling between your modules.
* **More services**: Multiple databases and transaction management can be painful.
* **Debugging problems**: Each service has its own set of logs to go through. Log, logs, and more logs.
* **Security challenges:** Securing one software application is hard. Securing hundreds of microservices which are distributed systems is quite challenging.
* **Large vs small product companies**: Microservices are great for large companies, but can be slower to implement and too complicated for small companies who need to create and iterate quickly.

**CONCLUSION:**

Designing microservices is difficult as compared to monolithic architectures. As application development trends continue to evolve, the debate between using microservices monolithic architectures will only become more pronounced. Developers should know which architecture work for their specific task.

For smaller companies, starting with a monolithic application can be simpler, faster, and cheaper and if the product hasn’t gotten too mature, it can still be migrated to microservices at an appropriate time. The huge companies with millions of users are obvious examples of the best use case for microservices, as they need to ensure the uptime, scalability that the added modularity can provide.