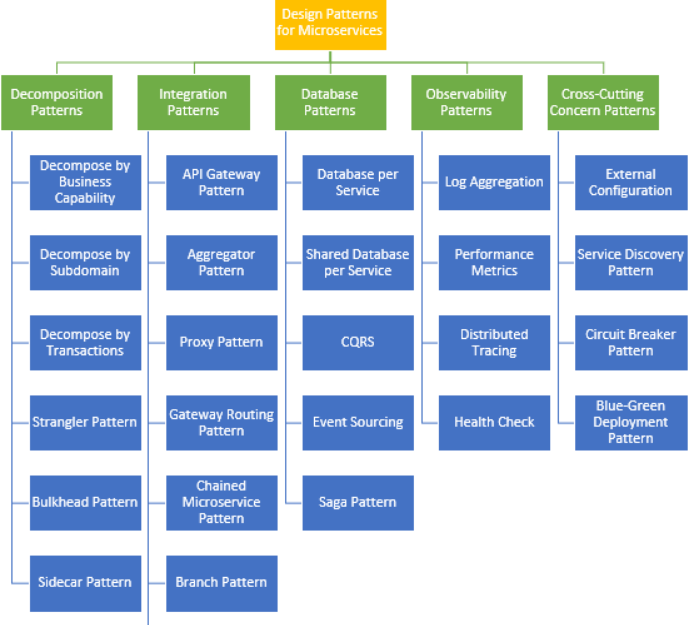
**Microservices**



**Typical problems in the monolith are**

* The complexity of adding new functionalities on top of the existing application.
* Performance issues due to a recent spike in transactions.
* Managing compliance and security in the modern world.
* Upgrading technology issues and its depended frameworks.

**What to decople and when from monolithic app**

* Based on vertical capabilities that are important to the business and subject to frequent change.
* Each step of migration should represent an atomic improvement to the overall architecture.

**Advantages of Micro services**

* Highly maintainable and testable
* Loosely coupled
* Independently deployable
* Organized around business capabilities
* Owned by a small team
* improve the speed, the frequency and reliability of software delivery

**Microservice Issues, Challenges, and Hurdles**

**Complexity of the Architecture**

The architecture for microservice applications is much more complex than for legacy applications, as the teams need to manage many different components.

**Closer Monitoring**

In order to avoid chaos, close monitoring is absolutely crucial.

**More Complicated Security System**

A higher number of components means a higher risk of attacks by hackers and other cybercriminals.

**Logging**

Since the log messages generated by microservices are distributed across multiple hosts, without a good strategy for logging you will be unable to understand the issues that might occur in the application.

**Risk of Slow Performance**

As each microservice consumes resources, the burden on servers is greater than with monolithic applications. This increase in resource usage may cause the application to run slower. In order to fix it, you need to deploy additional servers.

**Higher Maintenance Costs**

The cost of maintaining separate environments may turn out to be higher than the cost of managing monolith applications.

**Issues With Consistency**

As microservices need multiple resources to update, developers need to make sure the components are synchronized to avoid unexpected bugs. There is also a need for greater communication between developers and other team members.