1. **What is Micro Service Architecture?**

Micro service Architecture is an architectural development style which builds an application as a collection of small autonomous services developed for a business domain.

1. **What are the advantages of micro services?**

Here, are some significant advantages of using Micro services:

* Technology diversity, e.g., Micro services can mix easily with other frameworks, libraries,  and databases
* Fault isolation, e., a process failure should not bring the whole system down.
* Greater support for smaller and parallel team
* Independent deployment
* Deployment time reduce

**3. What are main differences between Micro services and Monolithic Architecture?**

|  |  |
| --- | --- |
| **Microservices** | **Monolithic Architecture** |
| Microservices are loosely coupled architecture. | Monolithic architecture is mostly tightly coupled. |
| Changes done in a single data model does not affect other Microservices. | Any changes in the data model affect the entire database |
| Continuous deployment is possible. | Continuous deployment becomes difficult. |

1. **What is monolithic architecture?**

Monolithic architecture is like a big container in which all the software components of an application are clubbed inside a single package.

1. **What are it's disadvantages?** 
   1. **Inflexible –** Monolithic applications cannot be built using different technologies
   2. **Unreliable –** Even if one feature of the system does not work, then the entire system does not work
   3. **Unscalable –** Applications cannot be scaled easily since each time the application needs to be updated, the complete system has to be rebuilt
   4. **Blocks Continous Development –** Many features of the applications cannot be built and deployed at the same time
   5. **Slow Development –** Development in monolithic applications take lot of time to be built since each and every feature has to be built one after the other
   6. **Not Fit For Complex Applications –**Features of complex applications have tightly coupled dependencies
2. **Challenges of microservice architecture.**

Developing a number of smaller microservices sounds easy, but the challenges often faced while developing them are as follows.

* **Automate the Components**: Difficult to automate because there are a number of smaller components. So for each component, we have to follow the stages of  Build, Deploy and, Monitor.
* **Perceptibility**: Maintaining a large number of components together becomes difficult to deploy, maintain, monitor and identify problems. It requires great perceptibility around all the components.
* **Configuration Management**: Maintaining the configurations for the components across the various environments becomes tough sometimes.
* **Debugging**: Difficult to find out each and every service for an error. It is essential to maintain centralized logging and dashboards to debug problems.

1. **What is cohesion and coupling? Why do we need high cohesion and low coupling?**

Cohesion refers to the degree to which the elements of a module/class belong together, it is suggested that the related code should be close to each other, so we should strive for high cohesion and bind all related code together as close as possible. It has to do with the elements within the module/class.

Coupling refers to the degree to which the different modules/classes depend on each other, it is suggested that all modules should be independent as far as possible, that's why low coupling. It has to do with the elements among different modules/classes.

1. **What is rest? What are it's uses?**

**Representational State Transfer (REST)/RESTful** web services are an architectural style to help computer systems communicate over the internet. This makes microservices easier to understand and implement.

Microservices can be implemented with or without RESTful APIs, but it’s always easier to build loosely coupled microservices using RESTful APIs.

1. **How independent micro-services communicate with each other?**

It depends upon your project needs. However, in most cases, developers use HTTP/REST with JSON or Binary protocol. However, they can use any communication protocol.

1. **What is Spring Cloud?**

Spring cloud is an Integration software that integrates with external systems. It allows micro services framework to build applications which perform restricted amounts of data processing.

### ****What problems are solved by Spring Cloud?****

While developing distributed microservices with Spring Boot we face few issues which are solved by Spring Cloud.

* **The complexity associated with distributed systems –**This includes network issues, Latency overhead, Bandwidth issues, security issues.
* **Ability to handle Service Discovery –**Service discovery allows processes and services in a cluster to find each other and communicate.
* **Solved redundancy issues –**Redundancy issues often occur in distributed systems.
* **Load balancing –**Improves the distribution of workloads across multiple computing resources, such as a computer cluster, network links, central processing units.
* **Reduces performance issues –**Reduces performance issues due to various operational overheads.

### What is DRY in Microservices architecture?

**DRY** stands for **Don’t Repeat Yourself**. It basically promotes the concept of reusing the code. This results in developing and sharing the libraries which in turn result in tight coupling.

#### **What is difference between REST & Microservice?**

Microservice is an architectural style that structures an application as a collection of self contained, loosely coupled services.  
RESTful services is a way to implement microservices.

1. **What is load-balancer in microservices architecture? How is it implemented in Spring Cloud?**

Load balancing improves the distribution of workloads across multiple computing resources. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy.

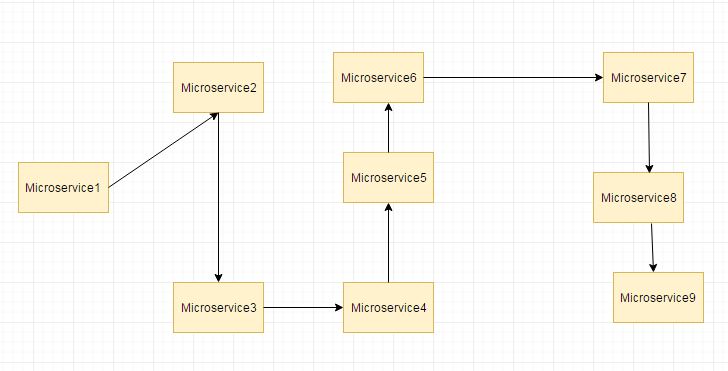
In Spring Cloud this can be implemented using Netflix Ribbon.

1. **What is Netflix Hystrix ?**

Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.

1. **In which business scenario to use Netflix Hystrix ?**

Usually for systems developed using Microservices architecture, there are many microservices involved. These microservices collaborate with each other.



Suppose if the microservice 9 in the above diagram failed, then using the traditional approach we will propagate an exception. But this will still cause the whole system to crash anyways. This is where hystrix comes into picture.

1. **What are the features of Hystrix?**

**FallBack method, Circuit Breaker**

1. **What is fallback method?**

**FallBack method-** Due to some reason if exposed service throws an exception, fallback method is defined using Hystrix. This fallback method should have the same return type as the exposed service. In case of exception in the exposed service the fallback method will return some value.

1. **What is circuit breaker?**

**Circuit Breaker-**If the exceptions keep on occuring then the Hystrix circuit will break and the method in exposed service will be skipped all together and directly call the fallback method.The purpose of circuit breaker is to give time to recover from exception.

#### **What is a Gateway in microservice ?**

Clients do not directly invoke microservices, they go through the Gateway. The Gateway in turn calls the microservices and returns the response to the client.

The Gateway decouples the microservices from the clients.

1. **Why do we need zuul proxy?**

**Zuul** is an edge service that **proxies** requests to multiple backing services. It provides a unified “front door” to your system, which allows a browser, mobile app, or other user interface to consume services from multiple hosts without managing cross-origin resource sharing (CORS) and authentication for each one.

1. **What is service discovery?**

**Service discovery** is how applications and (micro)services locate each other on a network. Service discovery implementations include both:

* a central server (or servers) that maintain a global view of addresses and
* clients that connect to the central server to update and retrieve addresses.