Q)What are Microservices?

Microservices is an architecture where the application is exposed as loosely coupled services that can be independently developed, deployed, and maintained.

Each service exposed is referred to as Microservice.

Each service performs a unique function. Speciality of this architecture is that polyglot architecture is supported. For example, if a team is working on one of the microservice using Java, Spring Boot, and MySQL, another team can work on another microservice using Python, Node JS, and NoSQL. Different microservices can use a different version of the same programming language.

Different microservices can use different programming languages.

Different microservices can use different architectures as well.

Q)Why Microservices?

In the case of monolith applications, there are several problems like Same code base for presentation, business layer, and data access layer.

Application is deployed as a single unit. Complex to maintain and scalability is an issue. Microservice solves the above problems.

Microservices are ideal when a monolith or a legacy application needs to be modernized. For new software development, if the key business drivers are to reduce time to market, scalable better software, lower costs, faster development, or cloud-native development, microservices are ideal.

Each service is independent and gives the flexibility to choose the programming language, database, and/or architecture. Distinct services can be developed, deployed, and maintained independently.

Q) What are the pros and cons of Microservice Architecture?

Pros of Microservice Architecture

1) Freedom to use different technologies

2) Each microservices focuses on single capability

3) Supports individual deployable units

4) Allow frequent software releases

5) Ensures security of each service

6) Multiple services are parallelly developed and deployed

Cons of Microservice Architecture

1) Management of a large number of services is difficult.

2) Communication between microservices is complex.

3) Increased efforts for configuration and other operations

4) Difficult to maintain transaction safety and data boundaries

5) Due to the decentralized nature of microservices, more microservices will mean more resources hence high Investment

6) Debugging of problems is harder unless the right instrumentation is followed during design and development.

7) Microservices will need a large team size with the right mix of experience in design, development, automation, deployments, tools, and testing.

Q) When to use microservices?

Reduce time to market,

Scalable better software,

Lower costs,

Faster development,

Cloud-native development It makes sense to adopt a microservices architecture, if the team size is big enough as each service will require its team to develop, deploy and manage.

Timeframe and skills of team members are a constraint.

If fast results are required, choose microservices architecture only if the team also has experience in microservices.

Do not use this architecture for simple application which can be managed by monolithic application . So you use ask yourself first do we really need microservice architecture

Q) What are the main features of Microservices? Microservices architecture breaks an application into smaller services, and it is possible to develop, deploy each service independently. This makes the introduction of new features in an application very easier. Decentralization: Microservices architecture leads to distributed systems. The data management is decentralized. There will be a monolithic database containing all data belonging to the application. Each service has the ownership of the data related to the business functionality of that service. Black box: Every microservice is defined as a black box. The details of the complexity are hidden from other services/components. Security: The Microservice platform itself should provide capabilities for certificate management, different types of credentials, authentication, and authentication based on RBAC (Role-based access model). Security is decoupled from the microservice development team as platform standardization help with it. Polyglot: This is one of the significant aspects of microservices architecture. For example, if a team is working on one of the microservice using Java, Spring Boot, and MySQL, another team can work on another microservice using Python, Node JS, and NoSQL. Different microservices can use a different version of the same programming language. Eg. Python 2.7 and Python 3.0 Different microservices can use different programming languages. Different microservices can use different architectures as well. Q) How do microservices communicate with each other? In the case of Microservice Architecture, there are 2 different types of inter-service communication between microservices. a. Synchronous communication b. Asynchronous communication Synchronous communication: In the case of Synchronous communication between microservices, the client service waits for the response within a time limit. The possible solution is using HTTP Protocol using via REST API for interservice communication. Asynchronous Communication: In the case of Asynchronous Communication, the client service doesn’t wait for the response from another service. When the client microservice calls another microservice, the thread is not blocked till a response comes from the server. The message producer service generates a message and sends the message to a message broker on a defined topic. The message producer waits for only the acknowledgment from the message broker to know that message is received by the broker. The consuming service subscribes to a topic in the messaging queue. All the messages belonging to that topic will be fed to the consuming system(s). The message producer service and consuming services don’t even know each other. The response is received in the same methodology through a message broker via defined message topics. Different messaging tools are based on the AMQP (Advanced Message Queuing Protocol). Some examples are given below. a. Apache Kafka b. RabbitMQ c. Apache ActiveMQ Q)What is the difference between Monolithic, SOA and Microservices Architecture? Main diff b/w SOA and MS architecture is Based on sharing of data and info. SOA shares and reuses as much as possible while MS focuses on sharing as little as possible