***What is service discovery?***

*Service discovery helps you discover, track, and monitor the health of services within a network. Service discovery registers and maintains a record of all your services in a service catalog. This service catalog acts as a single source of truth that allows your services to query and communicate with each other. The Need for Service Discovery.*

*How will servers communicate with each other? We will simply make a REST call and all these servers will communicate with each other using the REST API. But the real challenging part is when a server wants to connect to another server then before this server connects to this server it needs to know the IP address, and the port number where this particular application is running in the server. Where you have thousands and thousands of applications, how you will manage the server IP? How you’ll maintain their port number?*

*A microservice needs to know the location (IP address and port) of every service it communicates with. If we don’t employ a Service Discovery mechanism, service locations become coupled, leading to a system that’s difficult to maintain. Dynamically determining the location of an application service isn’t a trivial matter.*

***How does service discovery work?***

*Service discovery uses a service's identity instead of traditional access information (IP address and port). This allows you to dynamically map services and track any changes within a service catalog. Service consumers (users or other services) then use DNS to dynamically retrieve other service's access information from the service catalog. The lifecycle of a service may look like the following:*

*A service consumer communicates with the "Web" service via a unique Consul DNS entry provided by the service catalog.*

*A new instance of the "Web" service registers itself to the service catalog with its IP address and port. As new instances of your services are registered to the service catalog, they will participate in the load balancing pool for handling service consumer requests.*

*The service catalog is dynamically updated as new instances of the service are added and legacy or unhealthy service instances are removed. Removed services will no longer participate in the load balancing pool for handling service consumer requests.*

***What is Service Discovery in Microservices?***

*In a microservices application, the set of active service instances changes frequently across a large, dynamic environment. These service instances rely on a service catalog to retrieve the most up-to-date access information from the respective services. A reliable service catalog is especially important for service discovery in microservices to ensure healthy, scalable, and highly responsive application operation.*

*The Client‑Side Discovery Pattern - When using client‑side discovery, the client is responsible for determining the network locations of available service instances and load balancing requests across them. The client queries a service registry, which is a database of available service instances. The client then uses a load‑balancing algorithm to select one of the available service instances and makes a request.*

*When making a request to a service, the client obtains the location of a service instance by querying a Service Registry, which knows the locations of all service instances. It’s an advantage because it saves an extra hop that we would’ve had with a dedicated load balancer. It’s a disadvantage because the Service Consumer must implement the load balancing logic.*

*Server-Side Service Discovery - The alternate approach to Service Discovery is the Server-Side Discovery model, which uses an intermediary that acts as a Load Balancer. The client makes a request to a service via a load balancer that acts as an orchestrator. The load balancer queries the Service Registry and routes each request to an available service instance. In this approach, a dedicated actor, the Load Balancer, does the job of load balancing. This is the main advantage of this approach. Indeed, creating this level of abstraction makes the Service Consumer lighter, as it doesn’t have to deal with the lookup procedure. As a matter of fact, there’s no need to implement the discovery logic separately for each language and framework that the Service Consumer uses.*

*On the other hand, we must set up and manage the Load Balancer, unless it’s already provided in the deployment environment.*

*What are the Advantages of Service Discovery (Server-side & Client-side)?*

*The advantage of Server-side service discovery is that it makes the client application lighter as it does not have to deal with the lookup procedure and makes a request for services to the router.*

*The advantage of Client-side service discovery is that the client application does not have to traffic through a router or a load balancer and therefore can avoid that extra hop.*