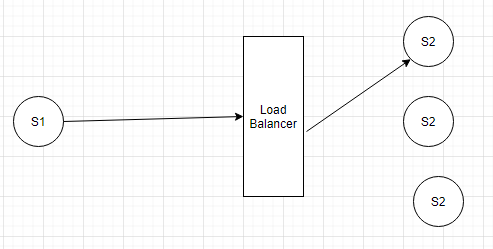
**Microservices**

Microservice can have many definitions.

* It is architectural style or an approach to developing a single application as a suit of small services, each running in its own process and communicating with light weight mechanisms, often HTTP resource API.
* Each service is independently deployable by fully automated deployment tools.
* Each can be written in different programming language. And different persistence can be used. Where as in monolith we have to stick on same language.
* If we compare microservice with monolith application it is easier to understand and developer can be more productive.
* Here we can accomplish parallel development. Where as in Monolith application it may create some issue.
* It can be easily scaled in terms of resources. Where as it is hard to scaled Monolith application.
* Each microservice can be deployable independently, it also improves fault isolation and conflicting resource can be resolved.

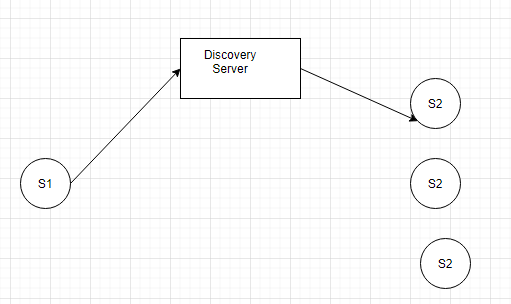
**Challenges and Solution of Microservices:**

1. In microservice generally one service calls another service take as example below.

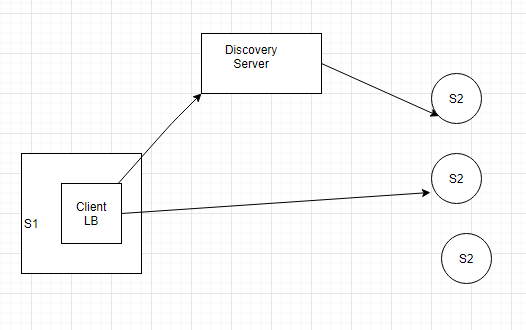


In above example we have two microservice i.e. S1 and S2, S1 is making a call to S2 and based on LB logic instance of S2 is assigned to that call. Here we can see we are having two remote calls. S1 to load balancer and then LB to S2. So, it will affect performance.

Now to over come this issue what microservice architect do that they have introduced another component call discovery server as shown in below diagram.



Discovery server will keep track of all the instance of S2. So, when call coming from S1 it will assign one instance of S2 to serve that request. But still our above problem is not resolve. So now what they do they have introduced another concept called client-side load balancing as shown in below diagram.



Here client side LB download all the registry of S2 instance from of discovery server and when S1 calls S2 based on some logic it will assign S2 instance to full fill the request. But here is another problem is that suppose if after sometime an instance of S2 is down then how client side LB know that S2 instance is down. Also, to download all the registry from discovery server it has to make a call to discovery server.

So, what Microservice architect has done during startup of S1 they have assigned one thread which actually get all the registry from discovery server and also after some seconds based on configuration it continuously synching the registry with discovery server and client-side LB. Similarly, Discovery server also call health check status of each S2 instance after some time interval and based on result it updates its registry and status.

For this purpose, Netflix have created Eureka (Discovery server) and Ribbon (Client side LB). what Netflix developer saw 30 secs is good time to synch and update registry as well as update the registry in Client-side LB.

Now suppose during that 30 sec one of instance of S2 went down. If we take 30 sec to update client-side LB and 30 sec to update registry and status of discovery server then during this time may be if an instance goes down then we will get 500 error. But in microservice architecture it was said that you should always right some fall back logic so that user never get 500 errors.