# Eureka

**Why Eureka server/service registry is needed in microservices?**

To understand why Eureka server needed in microservices architecture, lets start from simple

Understanding How one service calls another service REST endpoint for communication.

Say we need to call Employee payroll service to get payroll information for an Employee. The payroll service  which is deployed on localhost 8080 port and by passing employee primary key we got that information so we just call following

[**http://localhost:8080/payroll/1**](http://localhost:8080/payroll/1)**or**

[**http://127.0.0.1/payroll/1**](http://127.0.0.1/payroll/1)

Where localhost/127.0.0.1  is hostname/IP address and payroll is payroll service context, 1 is the Employee primary key.

But this is only possible when beforehand you know the hostname/IP addresses then you can configure your URL. So here Hostname or IP address is a constraint or a pain point.

If IP address of a  server/container is fixed then you can use that approach but

**what happens when your IP addresses and hostname are unpredictable?**

Nowadays on a Cloud platform, it is obvious that all the servers or containers use dynamic IPs for autoscaling. And the interesting thing is that in  Microservice architecture the key principle is your service can autoscaled as per load so Cloud platforms are ideal for Microservices.

What I try to say here is we can not predict the IP addresses of container/server beforehand so putting dependent services IP addresses in the config file is not a solution we need a more sophisticated technique to identify service, and Eureka server stepped in here.

<http://javaonfly.blogspot.com/2017/07/microservicespring-cloud-eureka-server.html>

<http://javaonfly.blogspot.com/2017/07/microservices-communication-feign-as.html> - feign client

# Zuul

Zuul acts as an API gateway or Edge service. It receives all the requests coming from the UI and then delegates the requests to internal microservices. So, we have to create a brand new microservice which is Zuul-enabled, and this service sits on top of all other microservices. It acts as an Edge service or client-facing service. Its Service API should be exposed to the client/UI. The client calls this service as a proxy for an internal microservice, then this service delegates the request to the appropriate service.

<https://dzone.com/articles/microservices-communication-zuul-api-gateway-1>