How to Configure Multiple Eureka Servers

There are multiple Eureka servers (to avoid Single Point Failure), each server needs to have synchronized registry details so that every server knows about the current state of every microservice registered in the Eureka server registry.

The Eureka server communicates its peer server as a client and clones the updated state of the registry, so the Eureka server itself acts as a client. We can perform this by just configuring the eureka.client.serviceUrl.defaultZone property.

The Eureka server works in two modes:

- Standalone: in local, we configure a stand-alone mode where we have only one Eureka server (localhost) and the same cloning property from itself.
- Clustered: we have multiple Eureka servers, each cloning its states from its peer.

```
spring.application.name=EmployeeEurekaServer
eureka.client.serviceUrl.defaultZone:http://localhost:9091/eureka/
server.port=9091
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Name	Description
spring.application.name	Unique name for a Eureka server service.
eureka.client.serviceUrl.defaultZone	It consults with other Eureka servers to sync the service registry. As it is in standalone mode, local server address is specified.
server.port	In which port the server will be bound.

eureka.client.register-with-eureka	This determines if this server registers itself as a client; as known, the Eureka server can also act as a client so that it can sync the registry. The value being false means it prevents itself from acting as a client.
eureka.client.fetch-registry	Does not register itself in the service registry.

Multiple Eureka Servers can be specified as below, in eureka client

Use a comma separated list of peers in `eureka.client.serviceUrl.defaultZone`.

```
eureka.client.serviceUrl.defaultZone=http://<peer1host>:<peer1port>/eureka,http
://<peer2host>:<peer2port>/eureka
```

But the register action only uses the first one to register. There remaining are used only if attempting to contact the first fails.

Configuration on Eureka Server: One Eureka server acts as client to another

```
eureka:
  client:
    registerWithEureka: true
    fetchRegistry: true
    eurekaServerPort: 8761
    eurekaServerURLContext: eureka
```

```
Good reference for Peer Awareness:

https://github.com/spring-cloud/spring-cloud-netflix/issues/1251

http://projects.spring.io/spring-cloud/spring-cloud.html#_peer_awareness
```