Microservices

* Small autonomous service that work together

* Challenges:
  1] Bounded Context. : Right boundary based on domain knowledge
  2] Config mngmnt : diff server, instance, db ...
  3] Dynamic Scale up & down : load balancing
  4] Visibility : Centralised log, monitoring
  5] Pack of Goods : have fault tolerance if one micro down not all micro should be affected

Spring cloud Netflix mngs →

* Spring cloud config server : Centralised Config for all servers

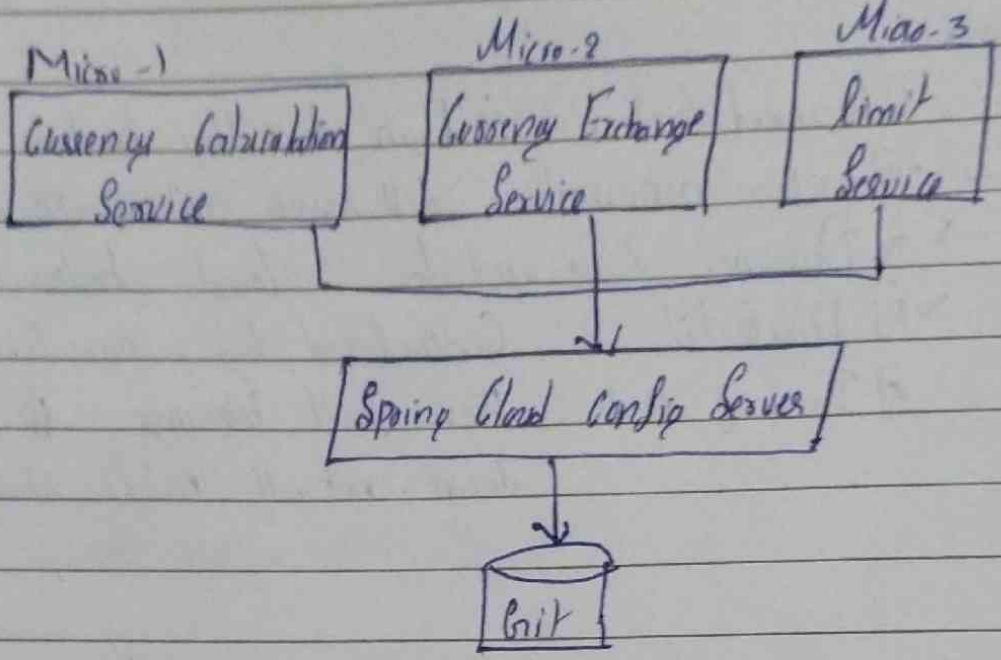
* Naming Server (Eureka) : Registering Micro service
  → Ribbon - client side load bal. Provide instance of micro's
  → Feign -

* Zipkin Distributed Tracing
  Netflix API gateway Zuul - common feature login ..

* Hystrix : fault tolerance

* Advtg of Micro-service :
  1] Diff Pgm lang.
  2] Diff DB / Centralised DB
  3] Dynamic Scaling

✡ Spring Cloud Config Server:



Proposed Structure:

1) Currency Calculation Service
2) Currency Exchange Service
3) Limits Service

4) git local Config
5) Spring cloud config server

6) netflix-eureka naming Server

7) netflix-zuul-apigateway-Server

* Steps to Connect limit Service connect to spring-cloud

1] Rename application.properties to bootstrap.properties

Following are the possible values in bootstrap.properties

* spring.application.name = limit-service
* spring.cloud.config.uri = http://localhost:8888

2] Create a class for reading properties from ~~limit-pro~~ spring-cloud

```
@Component                                  Same as application.name
@ConfigurationProperties("limit-service")
  Public class   Myclass
   {
      // define all the properties pojo to be fetched from
      // Spring cloud

      Put   int minlimit;
        :

   }
```

3] Dependency Required

1] Devtools
2] Web
3] Actuator
3] Spring-cloud-Starter-config

* Steps to Connect Spring-cloud to Git Repo

1] Main class
// http://localhost:8888/{application-name}/{Profile}
@EnableConfigServer
@SpringBootApplication
:
:

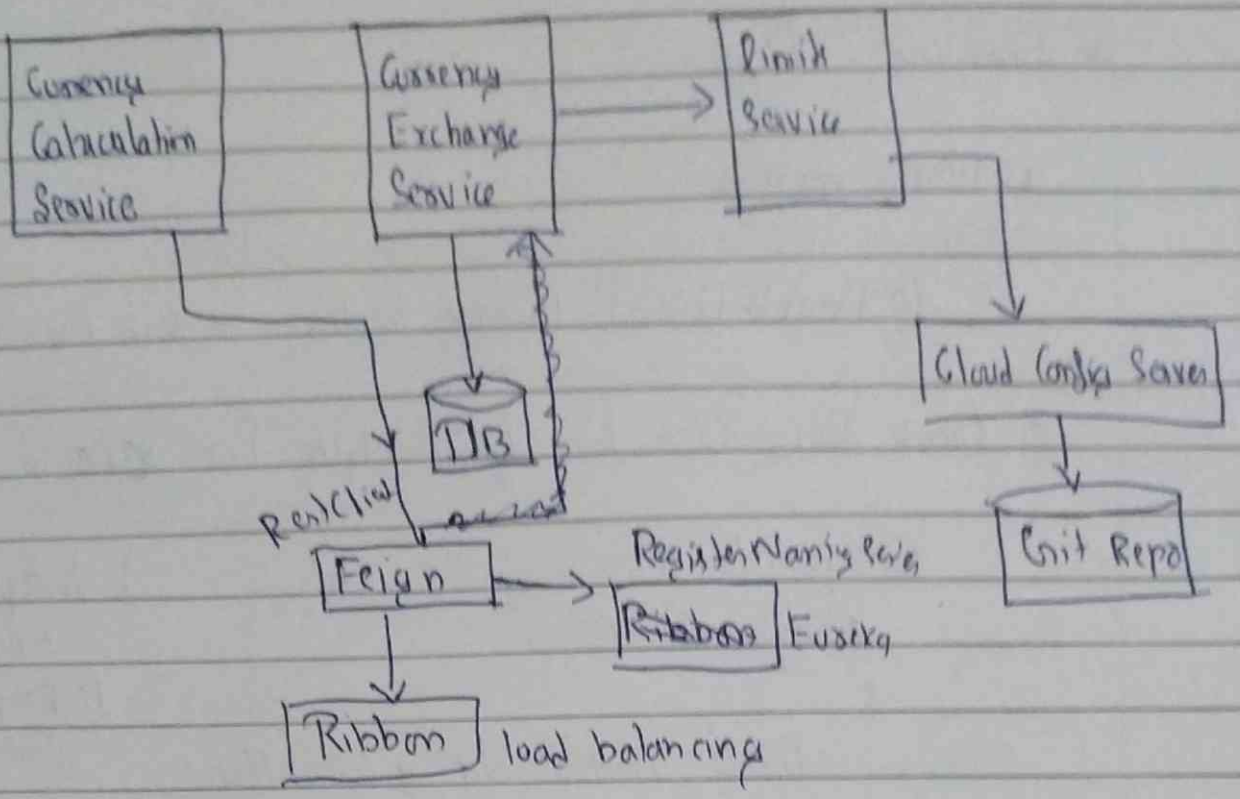
2] Create a local Git Repo
Right on project and add it as link source
Create limit-service-dev.properties in linked Repo & Comm
application-name Qa Prod

3] Application.properties:
spring.application.name = spring-config-cloud-server
server.port = 8888
spring.config.cloud.server.git.uri = file:///D:/localGitRepo

4] Dependency:
* Devtools
* spring-cloud-config-server

Note:
* if a property is not defined in dev properties & profiling is of type dev then in such case default properties are refered.
* spring-bus: Bcoz we need to restart ~~server~~ microS everytime on any changes of local-git repo this can be made dynamic/deployable auto by using Spring-bus

DOMS



1] Create Currency Exchange Service micro and expose a service "/currency/exchange" for DB Connectivity.

2] Create and Run multiple instance of CES [Currency Exchange Service] [Done in eclipse by adding Port in Vmargs]

3] Create Currency Calculation Service. Create an API "/currency/calculate". Ways to call another Micro i.e CES in this case

a] Using Restemplate
- * Resttemplate
- * HTTPEntity
- * HTTPHeaders.

b] Using Feign:

* Enabling Feign

1] Main method

@EnableFeign("Package location to scan feign Proxy class")

2] Create Proxy class for Called Micro [i.e MES in our case]

@FeignClient(name="" , url="") // Spring application name here
// http://localhost:5050 Path here as Url
interface CESProxy
{
// add method def of Called class
}

3] Dependency:

<artifactId> spring-cloud-feign-starter </artifactId>

Note:

* @PathVariable

if called API micro has @PathVariable Kindly used below syntax

m1(@PathVariable("UserId") int UserId)

instead of

m1(@PathVariable int UserID)