Micro-service Architecture  
-----------------------------------

Microservice architecture is a method of developing as a set of independently deployable,

small, modular services in which each service runs a unique process and communicates

It should be design based on the bounded Context

Gateway

Registry server

Config server

API services

Eureka :

It is server to register other services in Micro-service platform

ZULL proxy:

It is server-side load balancer by Netflix   
 Allows a browser to consume services from multiple hosts without managing cross-origin-resource

Ribbon:

Client-side load balancer setup in gateway

Ribbon + Feign client

Offer client side load balancer

Hystrics circuit breaker:

Fall back method for error handling

A typical Spring – micro service :

The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.

Infrastructure as a service (IaaS)  
Platform as a service (PaaS)Software as a service (SaaS)

Micro-service development management (Actuator)

Generate UI for our end points , has 2 parts UI and server side.

SwaggerConfig : Enable swagger is based on io.springfox which creates Docket

Swagger UI: Io.springfox-swagger-ui

Logs : We can manage log files at run time from info, debug or trace   
 It works with LoggerContext of Logback sl4j comes with 2 REST APIs :

1. List all loggerList();
2. PUT Rest controller to change thetype of logs

**Spring boot Actuator**

Spring boot has built-in API for health check

https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html

**Spring cloud config server:**

Provides an HTTP resource-based API for external configuration

@EnableConfigServer

By default is runs on 8080

**Bootstarp.yml** 🡪 contain name of application and profile environment

**Environment variables in ConfigServer :**

Environmnet repository is where we save our property values

Structure of repository :

The default implementation of **Environment Repository** is GIT back-end. This is good for Audit changes or any other physical update

To change the location of repository we can use

spring.cloud.conifg.server.git.uri = file:// 🡨 local

spring.cloud.conifg.server.git.uri = GIT url

Difference between Local and GIT :

Server directly goes to local repository without cloning from GIT

Even in that case, it is better to use the ssh: protocol for a shared filesystem repository, so that the server can clone it and use a local working copy as a cache.

Place holder:

Spring Config su[pports place holder for Apppliction,label and profile

spring:

cloud:

config:

server:

git:

uri: https://github.com/myorg/{application**}**

### Pattern Matching and Multiple Repositories

Comes with pattern matching on the application and profile name

spring:

cloud:

config:

server:

git:

uri: https://github.com/spring-cloud-samples/config-repo

repos:

simple: https://github.com/simple/config-repo

special:

pattern: special\*/dev\*,\*special\*/dev\*

uri: https://github.com/special/config-repo

local:

pattern: local\*

uri: file:/home/configsvc/config-repo

The pattern property in the repo is actually an array, so you can use a YAML array (or [0], [1], etc. suffixes in properties files

Type of ConfigServer URI :

It could be SSH or HTTPS

Force-pull :

If the local copy of repository gets dirty then forc-pull = true will save the problem

Delete untrack branches

In order to keep local repository branches clean and up to remote - deleteUntrackedBranches property could be set. It will make Spring Cloud Config Server **force**delete untracked branches from local repository. Example:

Vault backend:

It is secure for certain properties such as password we use Vault profile

By default Vault server runs on 8200

**Health indicator**

## Health Indicator

Config Server comes with a Health Indicator that checks whether the configured EnvironmentRepository is working

spring:

cloud:

config:

server:

health:

repositories:

myservice:

label: mylabel

myservice-dev:

name: myservice

profiles: development

Security in Config server:

For repository the same as GIT we have

SSH

Basic auth

If it is basic Auth we can have user name and password in conifgServer yml file

git:

uri: https://github.com/spring-cloud-samples/config-repo

username: trolley

password: strongpassword

for SSH we can have private key copied in there.