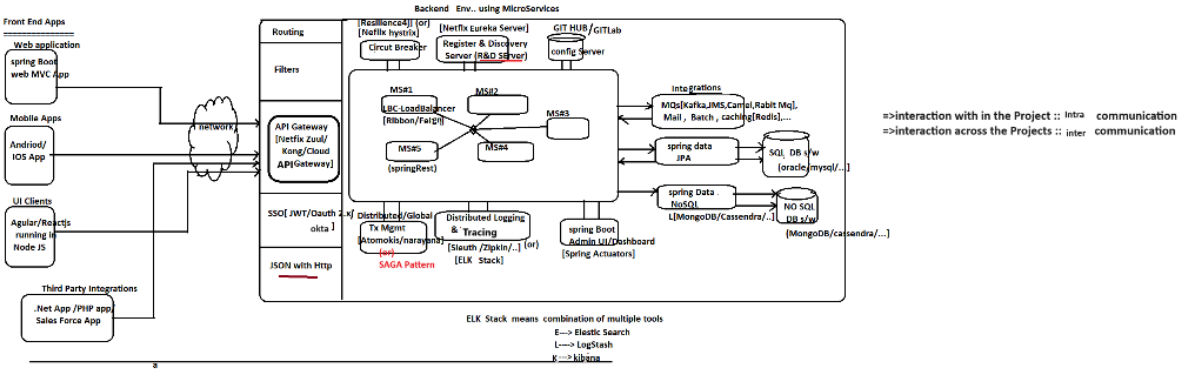
MicroServices App = Developing the services as Restful Apps + Tools + Design patterns + etc...

MicroServices App in spring boot = Developing the services as Spring boot Rest Apps + Spring boot cloud module + Design Patterns + third party tools services/tools

Overview of MicroService Architecture



=>interaction with in the Project :: Intra communication
=>interaction across the Projects :: inter communication

ELK Stack means combination of multiple tools
E---> Elestic Search
L----> LogStash
K ---> kibana

Micro service Architecture is Desinging or specification which provides set of rules and guidelines to develop Project as set of loosely coupled /De coupled Services .. and this can be implemented using spring boot +spring cloud + Netflix and lots of other tools.

=> Every module in the project will be developed as seperate microservice in the form of spring RestController with the support of spring boot env..

=> Once the Microservic architecture project is ready .. It will be deployed in cloud env..like AWS/AZure/Google Cloud and etc.. with DevOPS tools Jenkins with CI/CD + Docker + Ansible + Kubernate+.... (dev ops tools)

=>After developing each Micro SErvice as seperate Spring Rest App/Project it must be published in a common place called R & D SErver (Register And Discovery Server) like Netflix Eureka server

=> One MicroSErvice can find another MicroService in R & D SErver and can be used for Communication Through the same old R & D Server.. then only other micro services can find them for communication

=> One MicroService can find and communicate with another microservice only when it is published in R & D server ..

=> The common properties with same values of Multiple Micro services can be placed outside the Micro services in place called Config server like GIT Hub

=> If any exception is raised in the execution of one microService then it has to be informed to Admin UI /Dashboard with the support of Circuit Braker like Netflix hystrix. (or) rqliance4j

=> if any MicroService is having more demand then we allow to create multiple instances dynamically.. in that situation to pick up right istance with less Load factor from other MicroServices we take the support of Load Balancer Clients (LBC) like Ribbon ,Feign ,Http LoadBalancer and etc..

=> since we are devlooping every MicrService as Spring Rest App in spring boot env.. So we can make these Micro services Apps integrating with lots of other facilities like MQs (Message Queries), Mail, Caching, Batch PRocessing and etc.. [Integrations]

=> MicroServices can interact with SQL Db s/ws using spring data jpa

=> MicroServices can interact with No SQL Db s/ws using spring data Nosql modules like spring data mongoDB ,spring data cassendra and etc..

=> To Monitor and Manage all the MircorSErvices of the Project .. we try to spring boot Admin UI/Dashboard that is created using support of spring boot Actuators which are also useful providign non-funcationel features like Health metrics on the projects., thread pool info, memory info and etc..

=> Since Project contains multiple microSErvices interacting with each other..So we need to perform logging and tracing activity across the multiple micro services as needed with the support Distributed Logging and tracking tools like sluath and zipkin or ELK stack

=> The MicroServices of the Project can have different types of Clients (Front and Apps) like mobile Apps , web mvc Apps, Third party Apps , UI Technologies App and etc..

=> To use all these microServices and tools from different types of Clients /Front end Apps we need one common entry and exit point concept nothing but API Gateway like spring cloud gateway/ netflix zuul/Kong and etc.. providing facilities to apply Filters , Routers , SEcurity like SSO (Single Sign On) and etc..

=> The process of executing related logics by applying do everything or nothing principle is called Transaction Management.

Two types of Tx Mgmt

=> Local Tx Mgmt (all the activities of Tx Mgmt takes place on the single resource/DB s/w)
eg: Transfer money operation b/w the two accounts of same bank

=> Distributed Tx Mtmt (The activities of Tx Mgmt takes place on multiple resources / DB s/ws)
eg: Transfer money operation b/w the two accounts of two differnet banks

=> if the micro services in communication are completing job/task .. it is recommanded to enable Distributed TxMgmt on them becoz the task will be completed across the multiple micro services using multiple DB s/ws

=> Funtional Features are main operations in project
eg:: withdraw, deposite , transfer money and etc. are main features of the Project

=>Non-Funtional Features are the supporting operations in the project
eg: memory info , health matrix , threads info and etc..

MicroServices App

=

Developing the services as Restful Apps

+ Tools + Design patterns + etc...

MicroServices App in spring boot = Developing the services as Spring boot Rest Apps

+

boot

Spring cloud module + Design Patterns + third party tools services/tools

Overview of MicroService Archicteture

Front End Apps

Web application spring Boot

web MVC App

Mobile Apps

Andriod/

IOS App

Routing

Backend Env.. using MicroServices Resilience4j! (or) [Netflx Eureka Server] [Neflix hystrix]

GIT HUB/GITLab

Circut Breaker

Register & Discovery Server (R&D Server)

config Server

Filters

MS#1

MS#2

network

API Gateway [Netfix Zuul/ Kong/Cloud API Gateway]

LBC-LoadBalancer [Ribbon/Feign]

MS#3

MS#5

MS#4

(springRest)

UI Clients

Agular/Reactjs

running in

Node JS

SSO[JWT/OAuth 2.x Distributed/Global

Distributed Logging

okta]

Tx Mgmt Atomokis/narayana-

(or)

JSON with Http

SAGA Pattern

& Tracing [Sleuth /Zipkin/..] (or) [ELK Stack]

Third Party Integrations

.Net App /PHP app

Sales Force App

spring Boot

Admin UI/Dashboard

[Spring Actuators]

ELK Stack means combination of multiple tools

E---> Elestic Search L----> LogStash

K---> kibana

Integrations

MQs[Kafka,JMS,Camel, Rabit Mq], Mail, Batch, caching[Redis],...

spring data

JPA

SQL DB s/w (oracle/mysql/..]

communication =>interaction with in the Project :: Intra =>interaction across the Projects :: inter communication

spring Data NoSQL

NO SQL

L[MongoDB/Cassendra/..] DB s/w

(MongoDB/cassendra/...]

a

Micro service Architecture is Desinging or specification which provides set of rules and guidelines

to develop Project as set of loosely coupled /De coupled Services .. and this can implemented using spring boot +spring cloud + Netflix and lots of other tools.

=> Every module in the project will be develop as seperate microservice: in the form of spring RestController with the support of spring boot env..

=> Once the Microservic architecture project is ready.. It will be deployed in cloud env..like AWS/Azure/Google Cloud and etc.. witho tools Jenkins with CI/CD + Docker + Ansible + Kubernates+.... (dev ops tools)

=>After developing each Micro Service as seperate Spring Rest App/Project it must be published in a common place called R & D SErver (Register And Discovery Server) like Netflix Eureka server

=> One MicroService can find another MicroService in R & D Server and can be used for Communication

Through the same old R & D Server.. then only other micro services can find them for communication

=> One MicroService can find and communicate with another microservice only when it is published in R & D server..

=> The common properties with same values of Multiple Micro services can be placed outside the Micro services in place called Config server like GIT Hub

=> if any exception is raised in the execution of one microService then it has to be informed

to Admin UI /Dashboard with the support of Circuit Breker like Netflix hystrix. (or) reliance4j

=> if any MicroService is having more demand then we allow to create multiple instances dynamically..

In that situation to pick up right istance with less Load factor from other MicroServices we take the support of Load Balancer Clients (LBC) like Ribbon, Feign,Http Load Balancer and etc..

=> since we are devleoping every MicrService as Spring Rest App in spring boot env.. So we can make these Micro services Apps integrating with lots of other facilities like MQs (Message Queries), Mail, Caching, Batch Processing and etc.. [Integrations]

=> MicroServices can interact with SQL DB s/ws using spring data jpa

=> MicroServices can interact with No SQL DB s/ws using spring data Nosql modules like spring data mongoDB,spring data cassendra and etc..

use

=> To Monitor and Manage all the MircorServices of the Project .. we try to spring boot Admin UI/Dashboard that is created using support of spring boot Actuators which are also useful providign non-funcational features like Health metrics on the projects., thread pool info, memory info and etc..

=> Since Project contains multiple microServices interacting with each other..So we need to perform logging and tracing activity across the multiple micro services as needed with the support Distributed Logging and tracking tools like slueth and zipkin or ELK stack

=> The MicroServices of the Project can have different types of Clients (Front end Apps) like mobile Apps, web mvc Apps, Third party Apps, UI Technologies App and etc..

=> To use all these microservices and tools from different types of Clients /Front end Apps

we need one common entry and exit point concept nothing but API Gateway like spring cloud gateway/ netflix zuul/Kong and etc.. providing facilities to apply Filters, Routers, SEcurity like SSO (Single Sign On) and etc..

=> The process of executing related logics by applying do everything or nothing principle is called Transaction Management.

Two types of Tx Mgmt

=> Local Tx Mgmt (all the activities of Tx Mgmt takes place on the single resource/DB s/w)

eg: Transfer money operation b/w the two accounts of same bank

=> Distributed Tx Mtmt (The activities of Tx Mgmt takes place on multiple resources / DB s/ws)

eg: Transfer money operation b/w the two accounts of two differnet banks

0

=> Funtional Features are main operations in project

eg:: withdraw, deposite, transfer money and etc. are main features of the Project

=>Non-Funtional Features are the supporting operations in the project

eg: memory info, health matrix, threads info and etc..

=> if the micro services in communication are completing job/task .. it is recommanded to enable Distributed TxMgmt on them

becoz the task will be completed across the multiple micro services using multiple DB s/ws