Microservices

Monolithic Approach

Use case: Hospital Management System

1. Patient Registration

2. Patient Clinical

3. Bed Management

4. Claim Management

Problem:

1. Huge code base

2. Difficult to fix bugs

3. Difficult to add new feature

Microservice Approach

Use case: Hospital Management System

4 different microservice application

1. Patient Registration

2. Patient Clinical

3. Bed Management

4. Claim Management

Features:

1. Small and Focused

2. Autonomous (package and deploy on own machine)

3. Communication through network calls via http/rest

4. API is exposed for each microservice

Golden Rule:

Can we change and deploy our application without impacting other services?

Why Microservices?

Characteristics of Microservices:

1. Heterogeous (written in different programming language/run on different platform/communicate using rest)
2. Robustness (if one component fails all application do not fail there is gracefully degrade the service)
3. Scalable (should be able to scale on multiple servers)
4. Easy to deploy (quickly push products to production)
5. Reusable (one service used by multiple service)
6. Replaceable (if we have better service we can replace it with third party vendor)

Cloud Computing

1. Storage (RDS)

2. Networking

3. DB Security

4. Auto Scaling

5. Load Balancing

6. Fault Tolerance

1. On demand (whenever we ask we will be charged for specific service)

2. Ubiquitous (found everywhere)

3. Network

4. Shared (Google Drive)

Cloud Providers (existing cloud providers with low maintainence)

1. AWS

2. Azure

3. GCP

4. Oracle

5. IBM

6. Alibaba

Types of Clouds

1. Public (AWS, AZURE, GCP)

2. Private

3. Hybrid (public + public or public + private)

(Data Center or DB + apps hosted on Azure(webserver, load balancing)

(public + public)=(AWS RDS + Azure(.net))

Types of Service Models

1. Saas (networking, storage, services, virtualization,os,middleware,runtime,data, application services)

eg: all infra will be on the fly

RDS (Relational Database Service)

S3 (Simple Storage Service)

Lambda

Serverless architecture (just give app we will run for you)

2. Paas (networking, storage, services, virtualization,os,middleware,runtime)

eg: EBS(elastic bean stalk(EBS), simple notification service(SNS))

3. Iaas (networking, storage, services, virtualization) eg: CloudFormation, Terraform

AWS

1. Environment where they are deployed (EBS or elastic bean stalk)
2. Scaling (Auto Scale)
3. Load Balancing (ELB or elastic load balancer)
4. Security (Identity Access Management (IAM)/ Security Group)
5. Health Check and Monitoring (Cloud Watch)

Creating Microservices

1. Create DB tables (coupon and product)
2. Create a Spring Boot Starter Project with the required dependencies (web, jpa, mysql)
3. Create model class and annotate with JPA annotations
4. Create a Spring Repository using Interface and extends JPARepository
5. Create a RestController with 2 method to createCoupon and getCoupon

@RequestMapping(value = "/coupns", method = RequestMethod.***POST***)

**public** Coupon create(@RequestBody Coupon coupon) {

**return** coupon;

}

@RequestMapping(value = "/coupons/{code}", method = RequestMethod.***GET***)

**public** Coupon getCoupon(@PathVariable("code") String couponCode) {

**return** couponRepository.findByCode(couponCode);

}

1. Update application.properties file
2. Test the application in postman