第一章 课程介绍和学习路线

1、微服务架构SpringCloud课程介绍

简介：课程介绍和课程大纲讲解，讲课风格和重点内容理解技巧

2、技术选型和学后水平

简介：课程所需基础和技术选型讲解，学完课程可以到达怎样的程度

```
1、IDEA JDK8 Maven SpringBoot基础 Linux
2、理解掌握并开发SpringCloud里面主流架构和组件的基础使用，还有部分源码原理的理解
3、掌握学习的技巧和解决问题的思路
```

第二章 架构演进和分布式系统基础知识

1、传统架构演进到分布式架构

简介：讲解单机应用和分布式应用架构演进基础知识 (画图)

```
高可用 LVS+keepalive
1、单体应用：
    开发速度慢
    启动时间长
    依赖庞大
    等等
2、微服务
    易开发、理解和维护
    独立的部署和启动
    等
    不足：
        分布式系统-》分布式事务问题
        需要管理多个服务-》服务治理
```

2、微服务核心基础讲解

简介：讲解微服务核心知识： 网关、服务发现注册、配置中心、链路追踪、负载均衡器、熔断 1、网关：路由转发 + 过滤器 /api/v1/pruduct/ 商品服务 /api/v1/order/ 订单服务 /api/v1/user/ 用户服务 2、服务注册发现：调用和被调用方的信息维护 3、配置中心：管理配置，动态更新 application.properties 4、链路追踪：分析调用链路耗时 例子：下单-》查询商品服务获取商品价格-》查询用户信息-》保存数据库 5、负载均衡器：分发负载 6、熔断：保护自己和被调用方

3、常见的微服务框架

简介:讲解常用的微服务框架

```
consumer：调用方
provider：被调用方
一个接口一般都会充当两个角色（不是同时充当）

1、dubbo：zookeeper + dubbo + springmvc/springboot
    官方地址：http://dubbo.apache.org/#!/?lang=zh-cn
    配套
        通信方式：rpc
        注册中心：zookeper/redis
        配置中心：diamond
2、springcloud：全家桶+轻松嵌入第三方组件(Netflix 奈飞)
    官网：http://projects.spring.io/spring-cloud/
    配套
        通信方式：http restful
        注册中心：eruka/consul
        配置中心：config
        断路器：hystrix
        网关：zuul
        分布式追踪系统：sleuth+zipkin
学习资料：https://blog.csdn.net/zhangweiwei2020/article/details/78646252
```

4、微服务下电商项目基础模块设计

简介：微服务下电商项目基础模块设计 分离几个模块，课程围绕这个基础项目进行学习 小而精的方式学习微服务

1、用户服务
    1）用户信息接口
    2）登录接口
2、商品服务
    1）商品列表
    2）商品详情
3、订单服务
    1）我的订单
    2）下单接口

第三章 SpringCloud核心组件注册中心

1、什么是微服务的注册中心

简介：讲解什么是注册中心，常用的注册中心有哪些 (画图)

理解注册中心：服务管理,核心是有个服务注册表，心跳机制动态维护
服务提供者provider：启动的时候向注册中心上报自己的网络信息
服务消费者consumer：启动的时候向注册中心上报自己的网络信息，拉取provider的相关网络信息
为什么要用：
    微服务应用和机器越来越多，调用方需要知道接口的网络地址，如果靠配置文件的方式去控制网络地址，对于动态新增机器，维护带来很大问题
主流的注册中心：
    zookeeper、Eureka、consul、etcd 等

2、分布式应用知识CAP理论知识

简介：讲解分布式核心知识CAP理论

CAP定理：
    指的是在一个分布式系统中，Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性），三者不可同时获得。
一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值。（所有节点在同一时间的数据完全一致，越多节点，数据同步越耗时）
可用性（A）：负载过大后，集群整体是否还能响应客户端的读写请求。（服务一直可用，而且是正常响应时间）
分区容错性（P）：分区容忍性，就是高可用性，一个节点崩了，并不影响其它的节点（100个节点，挂了几个，不影响服务，越多机器越好）
CAP理论就是说在分布式存储系统中，最多只能实现上面的两点。而由于当前的网络硬件肯定会出现延迟丢包等问题，所以分区容忍性是我们必须需要实现的。所以我们只能在一致性和可用性之间进行权衡

3、分布式系统CAP原理常见面试题和注册中心选择

简介:讲解CAP原则在面试中回答和注册中心选择

```
    C A 满足的情况下，P不能满足的原因：
        数据同步(C)需要时间，也要正常的时间内响应(A)，那么机器数量就要少，所以P就不满足
    CP 满足的情况下，A不能满足的原因：
        数据同步(C)需要时间，机器数量也多(P)，但是同步数据需要时间，所以不能再正常时间内响应，所以A就不满足
    AP 满足的情况下，C不能满足的原因：
        机器数量也多(P)，正常的时间内响应(A)，那么数据就不能及时同步到其他节点，所以C不满足
    注册中心选择：
        Zookeeper：CP设计，保证了一致性，集群搭建的时候，某个节点失效，则会进行选举行的leader，或者半数以上节点不可用，则无法提供服务，因此可用性没法满足
        Eureka：AP原则，无主从节点，一个节点挂了，自动切换其他节点可以使用，去中心化
```

```
    结论：分布式系统中P,肯定要满足，所以只能在CA中二选一
        没有最好的选择，最好的选择是根据业务场景来进行架构设计
        如果要求一致性，则选择zookeeper，如金融行业

        如果要去可用性，则Eureka，如电商系统
```

4、SpringCloud微服务核心组件Eureka介绍和闭源后影响

简介：SpringCloud体系介绍

官方地址：http://projects.spring.io/spring-cloud/ Eureka的基础知识-->画图讲解交互流程，服务提供者<-->服务消费者 ; Eureka 2.x闭源后选择 参考：https://www.jianshu.com/p/d32ae141f680 https://blog.csdn.net/zjcjava/article/details/78608892

5、服务注册和发现Eureka Server搭建实战

简介：使用IDEA搭建Eureka服务中心Server端并启动，项目基本骨架介绍

```
官方文档：http://cloud.spring.io/spring-cloud-netflix/single/spring-cloud-
netflix.html#spring-cloud-eureka-server
第一步：创建项目
第二步：添加注解 @EnableEurekaServer
第三步：增加配置application.yml
        server:
          port: 8761
        eureka:
          instance:
            hostname: localhost
          client:
          #声明自己是个服务端
            registerWithEureka: false
            fetchRegistry: false
            serviceUrl:
              defaultZone:
http://${eureka.instance.hostname}:${server.port}/eureka/
第四步：访问注册中心页面
maven地址：https://www.cnblogs.com/sword-successful/p/6408281.html
```

6、服务注册和发现之Eureka Client搭建商品服务实战

简介：搭建用商品服务，并将服务注册到注册中心

```
1、创建一个SpirngBoot应用，增加服务注册和发现依赖
2、模拟商品信息，存储在内存中
3、开发商品列表接口，商品详情接口
4、配置文件加入注册中心地址
    使用eureka客户端 官方文档：http://cloud.spring.io/spring-cloud-netflix/single/spring-cloud-netflix.html#netflix-eureka-client-starter
```

7、Eureka服务注册中心配置控制台问题处理

简介：讲解服务注册中心管理后台，（后续还会细讲）

```
问题：eureka管理后台出现一串红色字体：是警告，说明有服务上线率低
EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.
关闭检查方法：eureka服务端配置文件加入
server:
    enable-self-preservation: false
注意：自我保护模式禁止关闭，默认是开启状态true
问题二：为什么只加一个注册中心地址，就可以注册
By having spring-cloud-starter-netflix-eureka-client on the classpath, your application automatically registers with the Eureka Server. Configuration is required to locate the Eureka server, as shown in the following example:
```

第四章 服务消费者ribbon和feign实战和注册中心高可用

1、常用的服务间调用方式讲解

简介：讲解常用的服务间的调用方式

```
RPC：
    远程过程调用，像调用本地服务(方法)一样调用服务器的服务
    支持同步、异步调用
    客户端和服务器之间建立TCP连接，可以一次建立一个，也可以多个调用复用一次链接
    PRC数据包小
        protobuf
        thrift
    rpc：编解码，序列化，链接，丢包，协议
Rest(Http)：
    http请求，支持多种协议和功能
    开发方便成本低
    http数据包大
    java开发：HttpClient，URLConnection
```

2、微服务调用方式之ribbon实战 订单调用商品服务

简介：实战电商项目 订单服务 调用商品服务获取商品信息

```
1、创建order_service项目
2、开发伪下单接口
3、使用ribbon. (类似httpClient,URLConnection)
    启动类增加注解
      @Bean
      @LoadBalanced
      public RestTemplate restTemplate() {
           return new RestTemplate();
      }
4、根据名称进行调用商品，获取商品详情
```

3、高级篇幅之Ribbon负载均衡源码分析实战

简介: 讲解ribbon服务间调用负载均衡源码分析

```
1、完善下单接口
2、分析@LoadBalanced
    1）首先从注册中心获取provider的列表
    2）通过一定的策略选择其中一个节点
    3）再返回给restTemplate调用
```

4、高级篇幅之服务间调用之负载均衡策略调整实战

简介：实战调整默认负载均衡策略实战

```
自定义负载均衡策略：http://cloud.spring.io/spring-cloud-
static/Finchley.RELEASE/single/spring-
cloud.html#_customizing_the_ribbon_client_by_setting_properties
在配置文件yml里面，自定义负载均衡策略
    #自定义负载均衡策略
    product-service:
      ribbon:
        NFLoadBalancerRuleClassName: com.netflix.loadbalancer.RandomRule
策略选择：
    1、如果每个机器配置一样，则建议不修改策略 (推荐)
    2、如果部分机器配置强，则可以改为 WeightedResponseTimeRule
```

5、微服务调用方式之feign 实战 订单调用商品服务

简介：改造电商项目 订单服务 调用商品服务获取商品信息

```
    Feign： 伪RPC客户端(本质还是用http)
    官方文档：https://cloud.spring.io/spring-cloud-openfeign/
    1、使用feign步骤讲解（新旧版本依赖名称不一样）
        加入依赖
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-starter-openfeign</artifactId>
            </dependency>
        启动类增加@EnableFeignClients
        增加一个接口 并@FeignClient(name="product-service")
```

```
2、编码实战
3、注意点：
    1、路径
    2、Http方法必须对应
    3、使用requestBody，应该使用@PostMapping
    4、多个参数的时候，通过@RequestParam（"id"） int id)方式调用
```

6、Feign核心源码解读和服务调用方式ribbon和Feign选择

简介: 讲解Feign核心源码解读和 服务间的调用方式ribbon、feign选择

```
1、ribbon和feign两个的区别和选择
    选择feign
        默认集成了ribbon
        写起来更加思路清晰和方便
        采用注解方式进行配置，配置熔断等方式方便
2、超时配置
    默认optons readtimeout是60，但是由于hystrix默认是1秒超时
    #修改调用超时时间
    feign:
      client:
        config:
          default:
            connectTimeout: 2000
            readTimeout: 2000
    模拟接口响应慢，线程睡眠新的方式
      try {
            TimeUnit.SECONDS.sleep(1);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

第五章 互联网架构服务降级熔断 Hystrix 实战

1、分布式核心知识之熔断、降级讲解

简介：系统负载过高，突发流量或者网络等各种异常情况介绍，常用的解决方案

```
1、熔断：
    保险丝，熔断服务，为了防止整个系统故障，包含子和下游服务
    下单服务 -》商品服务
             -》用户服务 （出现异常-》熔断）
2、降级：
    抛弃一些非核心的接口和数据
    旅行箱的例子：只带核心的物品，抛弃非核心的，等有条件的时候再去携带这些物品
3、熔断和降级互相交集
    相同点：
        1）从可用性和可靠性触发，为了防止系统崩溃
        2）最终让用户体验到的是某些功能暂时不能用
    不同点
        1）服务熔断一般是下游服务故障导致的，而服务降级一般是从整体系统负荷考虑，由调用方控制
```

2、Netflix开源组件断路器Hystrix介绍

简介：介绍Hystrix基础知识和使用场景

```
文档地址：
    https://github.com/Netflix/Hystrix
    https://github.com/Netflix/Hystrix/wiki
1、什么是Hystrix?
    1）hystrix对应的中文名字是“豪猪”

    2）hystrix   英[hɪst'rɪks] 美[hɪst'rɪks]
2、为什么要用?
    在一个分布式系统里，一个服务依赖多个服务，可能存在某个服务调用失败，
    比如超时、异常等，如何能够保证在一个依赖出问题的情况下，不会导致整体服务失败，
    通过Hystrix就可以解决
    http://cloud.spring.io/spring-cloud-netflix/single/spring-cloud-netflix.html#_circuit_breaker_hystrix_clients
3、提供了熔断、隔离、Fallback、cache、监控等功能
4、熔断后怎么处理?
    出现错误之后可以 fallback 错误的处理信息
    兜底数据
```

3、Feign结合Hystrix断路器开发实战《上》

简介：讲解SpringCloud整合断路器的使用，用户服务异常情况

```
1、加入依赖
注意：网上新旧版本问题，所以要以官网为主，不然部分注解会丢失
最新版本 2.0
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
    </dependency>
2、增加注解
    启动类里面增加注解
    @EnableCircuitBreaker
    注解越来越多-》 SpringCloudApplication注解
3、API接口编码实战
  熔断-》降级
    1）最外层api使用，好比异常处理（网络异常，参数或者内部调用问题）
        api方法上增加 @HystrixCommand(fallbackMethod = "saveOrderFail")

        编写fallback方法实现，方法签名一定要和api方法签名一致（注意点！！！）
补充： 修改maven仓库地址
pom.xml中修改
<repositories>
    <repository>
        <id>nexus-aliyun</id>
        <name>Nexus aliyun</name>
        <layout>default</layout>
        <url>http://maven.aliyun.com/nexus/content/groups/public</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
        <releases>
            <enabled>true</enabled>
        </releases>
    </repository>
</repositories>
```

4、Feign结合Hystrix断路器开发实战《下》

简介：讲解SpringCloud整合断路器的使用，用户服务异常情况

```
1、feign结合Hystrix
    1）开启feign支持hystrix  (注意，一定要开启，旧版本默认支持，新版本默认关闭)
        feign:
          hystrix:
            enabled: true
    2）FeignClient(name="xxx", fallback=xxx.class ), class需要继承当前
FeignClient的类
```

5、熔断降级服务异常报警通知实战

简介：完善服务熔断处理，报警机制完善

```
1、加入redis依赖
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-redis</artifactId>
    </dependency>
2、配置redis链接信息
  redis:
    database: 0
    host: 127.0.0.1
    port: 6379
    timeout: 2000
3、使用
    //监控报警
    String saveOrderKye = "save-order";
    String sendValue = redisTemplate.opsForValue().get(saveOrderKye);
    final String ip = request.getRemoteAddr();
    new Thread( ()->{
        if (StringUtils.isBlank(sendValue)) {
            System.out.println("紧急短信，用户下单失败，请离开查找原因,ip地址是
="+ip);
            //发送一个http请求，调用短信服务 TODO
            redisTemplate.opsForValue().set(saveOrderKye, "save-order-fail",
20, TimeUnit.SECONDS);
```

```
            }else{
                System.out.println("已经发送过短信，20秒内不重复发送");
            }
        }).start();
```

6、高级篇幅之深入源码剖析Hystrix降级策略和调整

简介：源码分析Hystrix降级策略和调整

```
1、查看默认讲解策略 HystrixCommandProperties
    1）execution.isolation.strategy    隔离策略
        THREAD 线程池隔离  （默认）
        SEMAPHORE 信号量
            信号量适用于接口并发量高的情况，如每秒数千次调用的情况，导致的线程开销过高，
通常只适用于非网络调用，执行速度快
    2）execution.isolation.thread.timeoutInMilliseconds  超时时间
        默认 1000毫秒
    3）execution.timeout.enabled 是否开启超时限制  （一定不要禁用）
    4）execution.isolation.semaphore.maxConcurrentRequests 隔离策略为 信号量的时
候，如果达到最大并发数时，后续请求会被拒绝，默认是10
官方文档：

https://github.com/Netflix/Hystrix/wiki/Configuration#execution.isolation.st
rategy
2、调整策略
    超时时间调整
hystrix:
  command:
    default:
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 4000
```

7、断路器Dashboard监控仪表盘实战

简介：讲解断路器Dashboard基础使用和查看

```
1、加入依赖
      <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-starter-netflix-hystrix-
dashboard</artifactId>
     </dependency>
     <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-actuator</artifactId>
     </dependency>
2、启动类增加注解
     @EnableHystrixDashboard

3、配置文件增加endpoint
management:
  endpoints:
    web:
      exposure:
        include: "*"
4、访问入口
     http://localhost:8781/hystrix
     Hystrix Dashboard输入： http://localhost:8781/actuator/hystrix.stream
参考资料
     默认开启监控配置
     https://docs.spring.io/spring-boot/docs/current-
SNAPSHOT/reference/htmlsingle/#boot-features-security-actuator
     配置文件类：
         spring-configuration-metadata.json
```

8、断路器监控仪表参数讲解和模拟熔断

简介：讲解 断路器监控仪表盘参数和模拟熔断

```
1、sse  server-send-event推送到前端
资料：https://github.com/Netflix/Hystrix/wiki/Dashboard
```

第六章 微服务网关zuul开发实战

1、微服务网关介绍和使用场景

简介：讲解网关的作用和使用场景 (画图)

```
1）什么是网关
        API Gateway，是系统的唯一对外的入口，介于客户端和服务器端之间的中间层，处理非业务功能 提供路由请求、鉴权、监控、缓存、限流等功能
                统一接入
                        智能路由
                        AB测试、灰度测试
                        负载均衡、容灾处理
                        日志埋点（类似Nignx日志）
                流量监控
                        限流处理
                        服务降级
                安全防护
                        鉴权处理
                        监控
                        机器网络隔离
2）主流的网关
        zuul：是Netflix开源的微服务网关，和Eureka,Ribbon,Hystrix等组件配合使用，Zuul 2.0比1.0的性能提高很多

        kong：由Mashape公司开源的，基于Nginx的API gateway

        nginx+lua：是一个高性能的HTTP和反向代理服务器,lua是脚本语言，让Nginx执行Lua脚本，并且高并发、非阻塞的处理各种请求
```

2、SpringCloud的网关组件zuul基本使用

简介：讲解zuul网关基本使用

```
1、加入依赖
2、启动类加入注解 @EnableZuulProxy
        默认集成断路器  @EnableCircuitBreaker
        默认访问规则
                http://gateway:port/service-id/**
                        例子：默认 /order-service/api/v1/order/save?user_id=2&product_id=1
                                自定义 /xdclass_order/api/v1/order/save?user_id=2&product_id=1
        自定义路由转发：
                zuul:
```

```
                routes:
                        order-service: /apigateway/**
        环境隔离配置:
                需求 : 不想让默认的服务对外暴露接口
                        /order-service/api/v1/order/save
                配置:
                zuul:
                        ignored-patterns:
                                - /*-service/api/v1/order/save
```

3、高级篇幅之Zuul常用问题分析和网关过滤器原理分析

简介：讲解Zuul网关原理和过滤器生命周期

```
1、路由名称定义问题
        路由映射重复覆盖问题
2、Http请求头过滤问题
3、过滤器执行顺序问题 ，过滤器的order值越小，越先执行
4、共享RequestContext，上下文对象
```

4、自定义Zuul过滤器实现登录鉴权实战

简介：自定义Zuul过滤器实现登录鉴权实战

```
1、新建一个filter包
2、新建一个类，实现ZuulFilter，重写里面的方法
3、在类顶部加注解，@Component,让Spring扫描
```

5、高级篇幅之高并发情况下接口限流特技

简介：谷歌guava框架介绍，网关限流使用

```
1、nginx层限流
2、网关层限流
```

6、Zuul微服务网关集群搭建

简介：微服务网关Zull集群搭建

```
1、nginx+lvs+keepalive
https://www.cnblogs.com/liuyisai/p/5990645.html
```

第七章 分布式链路追踪系统Sleuth和ZipKin实战

1、微服务下的链路追踪讲解和重要性

简介：讲解什么是分布式链路追踪系统，及使用好处

2、SpringCloud的链路追踪组件Sleuth实战

简介：讲解分布式链路追踪组件Sleuth实战

```
        1、官方文档
        http://cloud.spring.io/spring-cloud-static/Finchley.SR1/single/spring-cloud.html#sleuth-adding-project
        2、什么是Sleuth
                一个组件，专门用于记录链路数据的开源组件
                [order-service,96f95a0dd81fe3ab,852ef4cfcdecabf3,false]
                1、第一个值，spring.application.name的值
                2、第二个值，96f95a0dd81fe3ab ，sleuth生成的一个ID，叫Trace ID，用来标识一条请求链路，一条请求链路中包含一个Trace ID，多个Span ID
                3、第三个值，852ef4cfcdecabf3、spanid 基本的工作单元，获取元数据，如发送一个http
                4、第四个值：false，是否要将该信息输出到zipkin服务中来收集和展示。
        3、添加依赖
                <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-starter-sleuth</artifactId>
        </dependency>
```

3、SpringCloud的链路追踪组件Sleuth常见问题说明

简介：讲解分布式链路追踪组件Sleuth常见问题说明

4、可视化链路追踪系统Zipkin部署

简介：讲解Zipkin的介绍和部署

```
        1、什么是zipkin
                官网：https://zipkin.io/
                大规模分布式系统的APM工具（Application Performance Management）,基于Google Dapper的基础实现，和sleuth结合可以提供可视化web界面分析调用链路耗时情况
        2、同类产品
                鹰眼（EagleEye）
                CAT
                twitter开源zipkin，结合sleuth
                Pinpoint，运用JavaAgent字节码增强技术
                StackDriver Trace (Google)
        3、开始使用
                https://github.com/openzipkin/zipkin
                https://zipkin.io/pages/quickstart.html
                zipkin组成：Collector、Storage、Restful API、Web UI组成
        4、知识拓展：OpenTracing
        OpenTracing 已进入 CNCF，正在为全球的分布式追踪，提供统一的概念和数据标准。
        通过提供平台无关、厂商无关的 API，使得开发人员能够方便的添加（或更换）追踪系统的实现。
                http://blog.daocloud.io/cncf-3/
                https://www.zhihu.com/question/27994350
                https://yq.aliyun.com/articles/514488?utm_content=m_43347
```

5、高级篇幅之链路追踪组件Zipkin+Sleuth实战

简介：使用Zipkin+Sleuth业务分析调用链路分析实战

1、文档

http://cloud.spring.io/spring-cloud-static/Finchley.SR1/single/spring-cloud.html#_sleuth_with_zipkin_via_httpsleuth收集跟踪信息通过http请求发送给zipkin server，zipkinserver进行跟踪信息的存储以及提供Rest API即可，Zipkin UI调用其API接口进行数据展示默认存储是内存，可也用mysql、或者elasticsearch等存储

2、加入依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-zipkin</artifactId>
</dependency>
```

3、文档说明：http://cloud.spring.io/spring-cloud-static/Finchley.SR1/single/spring-cloud.html#_features_2

4、配置zipkin.base-url

5、配置采样百分闭spring.sleuth.sampler

推荐资料：

https://blog.csdn.net/jrn1012/article/details/77837710

第八章 微服务核心知识分布式配置中心Config实战

1、微服务下的分布式配置中心

简介：讲解什么是配置中心及使用前后的好处 (画图)

什么是配置中心：

一句话：统一管理配置，快速切换各个环境的配置

相关产品：

百度的disconf

地址:https://github.com/knightliao/disconf

阿里的diamand

地址：https://github.com/takeseem/diamond

springcloud的configs-server:

地址：http://cloud.spring.io/spring-cloud-config/

推荐干货文章：http://jm.taobao.org/2016/09/28/an-article-about-config-center/

2、SpringCloud的配置中心组件config-server实战

简介：讲解SpringCloud配置中心config-server实战

```
1、新建项目，创建config-server
2、启动类增加注解
        @EnableConfigServer
```

3、使用git服务器结合Config搭建分布式配置中心

简介：讲解使用git服务器结合Config搭建分布式配置中心

```
1、默认使用git存储配置中心
        使用git服务器，可以自己搭建gitlab服务器
        或者使用github、开源中国git、阿里云git
        794666918@qq.com
        xdclass.net123
        https://gitee.com/waitforxy/config_cloud.git
2、配置文件添加配置
        spring:
          application:
            name: config-server
        #git配置
          cloud:
            config:
              server:
                git:
                  uri: https://gitee.com/waitforxy/config_cloud
                  username: 794666918@qq.com
                  password: xdclass.net123
                  #超时时间
                  timeout: 5
                  #分支
                  default-label: master
3、访问方式（一定要注意语法，如果有问题，会出错）
        多种访问路径，可以通过启动日志去查看
        例子 http://localhost:9100/product-service.yml
        /{name}-{profiles}.properties
```

```
/{name}-{profiles}.yml
/{name}-{profiles}.json
/{label}/{name}-{profiles}.yml
name 服务器名称
profile 环境名称，开发、测试、生产
lable 仓库分支、默认master分支
```

4、分布式配置中心客户端使用实战

简介：微服务里面客户端接入配置中心实战

官方文档：http://cloud.spring.io/spring-cloud-config/single/spring-cloud-config.html#_spring_cloud_config_client

1、加入依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-client</artifactId>
</dependency>
```

2、修改对应服务的配置文件,把application.yml 改为 bootstrap.yml

```
#指定注册中心地址
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
#服务的名称
spring:
  application:
    name: product-service
  #指定从哪个配置中心读取
  cloud:
    config:
      discovery:
        service-id: CONFIG-SERVER
        enabled: true
      profile: test
      #建议用lable去区分环境，默认是lable是master分支
      #label: test
```

注意点：

```
1.配置文件要用bootstrap.yml
2.默认读取文件名是 服务名称
```

第九章 微服务消息总线Bus结合消息队列RabbitMQ实战

1、消息总线Bus介绍和使用场景

简介：讲解消息总线Bus介绍和使用场景

```
1、什么是消息
        一个事件，需要广播或者单独传递给某个接口
2、为什么使用这个
        配置更新了，但是其他系统不知道是否更新
```

2、消息队列和RabbitMQ基础介绍

简介：消息队列和RabbitMQ基础介绍

```
1、消息队列介绍
        参考：https://www.cnblogs.com/linjiqin/p/5720865.html
2、同类产品
        ActiveMQ
        RocketMQ
        Kafka
        等
3、SpringCloud默认推荐使用RabbitMQ
4、RabbitMQ介绍
        官方文档：http://www.rabbitmq.com/getstarted.html
        中文文档：http://rabbitmq.mr-ping.com/
```

3、实战系列使用Docker搭建RabbitMQ3.7

简介：使用Docker安装RabbitMQ

```
1、如果对Docker没基础，课程后续有讲解Docker，可以先跳转过去学习Docker
2、安装步骤
        1)拉取镜像：docker pull rabbitmq:management
        2)查看当前镜像列表：docker images
        3)删除指定镜像：docker rmi  IMAGE_ID  (如果需要强制删除加 -f)
```

```
                4)创建容器
                docker run -d --name="myrabbitmq" -p 5671:5671 -p 15672:15672
rabbitmq:management
                        -d: 后台运行容器，并返回容器ID
                        -p: 端口映射，格式为：主机(宿主)端口:容器端口
                        --name="rabbitmq": 为容器指定一个名称
        3、RabbitMQ默认创建了一个 guest 用户，密码也是 guest，如果访问不了记得查看防火
墙，端口或者云服务器的安全组
           管理后台：http://127.0.0.1:15672
其他安装方式：
Linux安装：https://blog.csdn.net/qq_34021712/article/details/72567786
windows安装：http://www.rabbitmq.com/install-windows.html
```

4、高级篇幅消息总线整合配置中心架构流程图

简介：讲解消息总线Bus结合config组件搭建配置中心项目架构图和操作流程

```
                启动
                rabbitmq：  docker run -d -p 5672:5672 -p 15672:15672
rabbitmq:management
                rabbitmq默认是5672,所以改为5672端口

                1、config-client加入依赖

                <!--配置中心结合消息队列-->

                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-actuator</artifactId>
                </dependency>
                <dependency>
                    <groupId>org.springframework.cloud</groupId>
                    <artifactId>spring-cloud-starter-bus-amqp</artifactId>
                </dependency>
                官方文档：http://cloud.spring.io/spring-cloud-bus/single/spring-
cloud-bus.html#_bus_refresh_endpoint
                文档里面 暴露端点 management.endpoints.web.exposure.include=bus-
refresh
                2、在配置文件中增加关于RabbitMQ的连接(如果是本机，则可以直接启动，采用默
认连接配置)
                        spring:
                          rabbitmq:
                            host: localhost
                            port: 5672
                            username: guest
                            password: guest
                        #暴露全部的监控信息
```

```
                management:
                  endpoints:
                    web:
                      exposure:
                        include: "*"
        3、需要刷新配置的地方，增加注解
                @RefreshScope
        4、访问验证 post方式：
                http://localhost:8773/actuator/bus-refresh
```

5、微服务相关项目改造配置中心

简介：把课程项目改造成配置中心讲解

```
1、git里面新增对应项目的配置文件，都要添加下面的配置
        #服务的名称
        spring:
          rabbitmq:
            host: localhost
            port: 5672
            username: guest
            password: guest
        #暴露全部的监控信息
        management:
          endpoints:
            web:
              exposure:
                include: "*"
2、项目里面添加maven依赖
    <!--配置中心客户端-->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-config-client</artifactId>
    </dependency>
    <!--config server-->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-bus-amqp</artifactId>
    </dependency>
3、修改application.properties为bootstrap.yml 并拷贝配置文件
```

```
                #指定注册中心地址
                eureka:
                  client:
                    serviceUrl:
                      defaultZone: http://localhost:8761/eureka/
                #服务的名称
                spring:
                  application:
                    name: order-service
                  #指定从哪个配置中心读取
                  cloud:
                    config:
                      discovery:
                        service-id: CONFIG-SERVER
                        enabled: true
                      profile: test
4、各个项目启动顺序
        1）注册中心
        2）配置中心
        3）对应的服务：商品服务、订单服务。。。
        4）启动网关
```

第十章、SpringCloud课程内容上半部分总结

1、微服务核心知识内容回顾

简介：回顾SpringCloud前面10章的基础内容

```
        1)介绍微服务的基础知识，核心组件，CAP原理
        2）SpringCloud注册中心 Eureka
        3）product-service / order-service
        4）伪RPC Ribbon / feign
        5）hystrix熔断
        6）服务网关介绍，zuul
        7）配置中心config-server
```

2、微服务下半部分知识 云服务器和Docker容器

简介：讲解云服务器和容器知识

第十一章 阿里云ECS服务器介绍和网络知识讲解

1、云服务器介绍和阿里云服务器ECS服务器选购

简介：什么是云服务器及目前主要的几个厂商介绍

```
        1、阿里云、腾讯云、亚马逊云
        阿里云：https://www.aliyun.com/
        腾讯云：https://cloud.tencent.com/
        亚马逊云：https://aws.amazon.com/
```

2、阿里云服务器远程登录和常用工具

简介：讲解阿里云服务器登录使用和常见终端工具

```
        1、windows工具    putty, xshell, security
                参考资料:

https://jingyan.baidu.com/article/e75057f210c6dcebc91a89dd.html
                        https://www.jb51.net/softjc/88235.html
        2、苹果系统MAC：  通过终端登录
                ssh root@ip  回车后输入密码
                ssh root@120.25.1.38
        3、可以尝试自己通过百度进行找文档，  安装mysql jdk  nginx maven git redis elk
```

3、互联网架构知识之网站部署上线基础准备

简介：讲解应用部署到可以公网访问需要步骤

1、一个http请求的故事
2、什么是cname和a记录
A记录和CNAME只可以同时生效一个，A记录优先

3、域名和ip的关系，DNS作用
参考资料：
https://blog.csdn.net/benbenzhuhwp/article/details/44704319
https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=0&rsv_idx=1&ch=1&tn=98050039_dg&wd=%E4%B8%80%E4%B8%AAhttp%E8%AF%B7%E6%B1%82%E7%9A%84%E8%AF%A6%E7%BB%86%E8%BF%87%E7%A8%8B&rsv_pq=80a65c5f00005961&rsv_t=a5fcWreuJzILdSwr4gI8pFqlO7HSu5BlhjwalyVzPiV9w2L%2BKEj78pPi1Qn6Vx4wXxI&rqlang=cn&rsv_enter=1&rsv_sug3=8&rsv_sug1=8&rsv_sug7=100&sug=%25E4%25B8%2580%25E4%25B8%25AAhttp%25E8%25AF%25B7%25E6%25B1%2582%25E7%259A%2584%25E8%25AF%25A6%25E7%25BB%2586%25E8%25BF%2587%25E7%25A8%258B&rsv_n=1

4、域名购买和配置解析实战

简介：域名购买和配置解析实战

1、购买域名，备案
阿里云 备案地址：https://beian.aliyun.com/
2、购买服务器，阿里云，腾讯云，亚马逊云aws
3、配置域名解析到服务器

第十二章 微服务必备技能Docker容器基础篇幅

1、微服务下的Docker介绍和使用场景

简介：Docker介绍和使用场景

1、什么是Dokcer
百科:一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口；
使用go语言编写，在LCX（linux容器）基础上进行的封装
简单来说：
1）就是可以快速部署启动应用
2）实现虚拟化，完整资源隔离
3）一次编写，四处运行（有一定的限制，比如Docker是基于Linux 64bit的，无法在32bit的linux/Windows/unix环境下使用）

2、为什么要用
1、提供一次性的环境，假如需要安装Mysql，则需要安装很多依赖库、版本等，如

果使用Docker则通过镜像就可以直接启动运行
2、快速动态扩容，使用docker部署了一个应用，可以制作成镜像，然后通过Dokcer快速启动
3、组建微服务架构，可以在一个机器上模拟出多个微服务，启动多个应用
4、更好的资源隔离和共享
一句话：开箱即用，快速部署，可移植性强，环境隔离

2、Linux云服务器Centos7安装Docker实战

简介：讲解阿里云ECS服务安装Docker实战

Linux Standard Base的缩写，lsb_release命令用来显示LSB和特定版本的相关信息
命令： lsb_release -a
阿里云安装手册：
https://help.aliyun.com/document_detail/51853.html?spm=a2c4g.11186623.6.820.RaToNY
常见问题：
https://blog.csdn.net/daluguishou/article/details/52080250

3、Docker仓库、镜像、容器核心知识讲解

简介：快速掌握Dokcer基础知识

1、概念：
Docker 镜像 - Docker images：
容器运行时的只读模板，操作系统+软件运行环境+用户程序

```
class User{
        private String userName;
        private int age;
}
```

Docker 容器 - Docker containers：
容器包含了某个应用运行所需要的全部环境

```
User user = new User()
```

Docker 仓库 - Docker registeries：
用来保存镜像，有公有和私有仓库，好比Maven的中央仓库和本地私服
镜像仓库：

```
                                        （参考）配置国内镜像仓库：
https://blog.csdn.net/zzy1078689276/article/details/77371782
                对比面向对象的方式
                Dokcer 里面的镜像 ： Java里面的类 Class
                Docker 里面的容器 ： Java里面的对象 Object
                通过类创建对象，通过镜像创建容器
```

4、Docker容器常见命令实战

简介：讲解Docker在云服务上的实际应用

```
        1、 常用命令（安装部署好Dokcer后，执行的命令是docker开头）,xxx是镜像名称
                搜索镜像：docker search xxx

                列出当前系统存在的镜像：docker images

                拉取镜像：docker pull xxx
                        xxx是具体某个镜像名称(格式 REPOSITORY:TAG)
                        REPOSITORY：表示镜像的仓库源,TAG：镜像的标签
                运行一个容器：docker run -d --name "xdclass_mq" -p 5672:5672 -p
15672:15672 rabbitmq:management
                        docker run - 运行一个容器
                        -d 后台运行
                        -p 端口映射
                        rabbitmq:management  (格式 REPOSITORY:TAG)，如果不指定
tag，默认使用最新的
                        --name "xxx"

                列举当前运行的容器：docker ps
                检查容器内部信息：docker inspect 容器名称
                删除镜像：docker rmi IMAGE_NAME
                         强制移除镜像不管是否有容器使用该镜像 增加 -f 参数，
                停止某个容器：docker stop 容器名称
                启动某个容器：docker start 容器名称
                移除某个容器：  docker rm 容器名称 （容器必须是停止状态）
        文档：
                https://blog.csdn.net/permike/article/details/51879578
```

5、实战应用之使用Docker部署Nginx服务器

简介：讲解使用Docker部署Nginx服务器实战 1、获取镜像 docker run (首先会从本地找镜像，如果有则直接启动，没有的话，从镜像仓库拉起，再启动) docker search nignx 2、列举 docker images 3、拉取 docker pull nignx 3、启动 docker run -d --name "xdclass_nginx" -p 8088:80 nginx docker run -d --name "xdclass_nginx2" -p 8089:80 nginx docker run -d --name "xdclass_nginx3" -p 8090:80 nginx 4、访问 如果是阿里云服务，记得配置安全组，腾讯云也需要配置，这个就是一个防火墙

6、公司中Docker镜像仓库使用讲解

简介：讲解一般公司中镜像仓库在的使用

```
        1、为啥要用镜像仓库
        2、官方公共镜像仓库和私有镜像仓库
                公共镜像仓库：
                                官方：https://hub.docker.com/，基于各个软件开发或者有软件提供商开发的
                                非官方：其他组织或者公司开发的镜像，供大家免费试用
                私有镜像仓库：
                                用于存放公司内部的镜像，不提供给外部试用；
                                SpringCloud 开发了一个支付系统 -》做成一个镜像 （操作系统+软件运行环境+用户程序）
```

7、高级篇幅之构建自己的镜像仓库

简介：使用阿里云搭建自己的镜像仓库

```
        1、阿里云镜像仓库：https://dev.aliyun.com/search.html
        点击管理控制台-》初次使用会提示开通，然后设置密码
        xdclass.net123
        2、使用阿里云私有镜像仓库
                1)登录： docker login --username=794666918@qq.com registry.cn-shenzhen.aliyuncs.com

                2) 推送本地镜像：
                docker tag [ImageId] registry.cn-shenzhen.aliyuncs.com/xdclass/xdclass_images:[镜像版本号]
                例子：
                docker tag 2f415b0e9a6e registry.cn-shenzhen.aliyuncs.com/xdclass/xdclass_images:xd_rabbitmq-v1.0.2
                docker push registry.cn-shenzhen.aliyuncs.com/xdclass/xdclass_images:xd_rabbitmq-v1.0.2
```

```
                3)拉取镜像
                        线上服务器拉取镜像:
                                docker login --username=794666918@qq.com
registry.cn-shenzhen.aliyuncs.com
                                docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:xd_rabbitmq-v1.0.2
                                启动容器:
                                docker run -d --name "xdclass_mq" -p 5672:5672
-p 15672:15672 2f415b0e9a6e
```

第十三章 微服务高级篇幅SpringCloud和Docker整合部署

1、高级篇幅之构建SpringBoot应用docker镜像上集

简介:使用Docker的maven插件，构建springboot应用

```
官方文档：https://spring.io/guides/gs/spring-boot-docker/

1、步骤：maven里面添加配置pom.xml

         <properties>
           <docker.image.prefix>xdclass</docker.image.prefix>
        </properties>

        <build>
                <finalName>docker-demo</finalName>
            <plugins>
                <plugin>
                    <groupId>com.spotify</groupId>
                    <artifactId>dockerfile-maven-plugin</artifactId>
                    <version>1.3.6</version>
                    <configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>
                        <buildArgs>

<JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
                        </buildArgs>
                    </configuration>
                </plugin>
            </plugins>
        </build>
        配置讲解
```

```
        Spotify 的 docker-maven-plugin 插件是用maven插件方式构建docker镜像的。
        ${project.build.finalName} 产出物名称，缺省为${project.artifactId}-
${project.version}
```

2、高级篇幅之构建SpringBoot应用docker镜像下集

简介:打包SpringCloud镜像并上传私有仓库并部署

```
                1、创建Dockerfile,默认是根目录，（可以修改为
src/main/docker/Dockerfile,如果修则需要制定路径）
                        什么是Dockerfile : 由一系列命令和参数构成的脚本，这些命令应用
于基础镜像，最终创建一个新的镜像
                        FROM openjdk:8-jdk-alpine
                        VOLUME /tmp
                        ARG JAR_FILE
                        COPY ${JAR_FILE} app.jar
                        ENTRYPOINT ["java","-jar","/app.jar"]
                        FROM <image>:<tag> 需要一个基础镜像，可以是公共的或者是私有
的， 后续构建会基于此镜像，如果同一个Dockerfile中建立多个镜像时，可以使用多个FROM指令

                        VOLUME  配置一个具有持久化功能的目录，主机 /var/lib/docker
目录下创建了一个临时文件，并链接到容器的/tmp。改步骤是可选的，如果涉及到文件系统的应用就很有
必要了。/tmp目录用来持久化到 Docker 数据文件夹，因为 Spring Boot 使用的内嵌 Tomcat 容器
默认使用/tmp作为工作目录
                        ARG  设置编译镜像时加入的参数， ENV 是设置容器的环境变量
                        COPY : 只支持将本地文件复制到容器 ,还有个ADD更强大但复杂点
                        ENTRYPOINT 容器启动时执行的命令
                        EXPOSE 8080 暴露镜像端口

                2、构建镜像
                mvn install dockerfile:build
                打标签
                docker tag a1b9fc71720d registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:docker-demo-v201808
                推送到镜像仓库
                docker push registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:docker-demo-v201808
                应用服务器拉取镜像
                docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:docker-demo-v201808
                docker run -d --name xdclass_docker_demo1 -p 8099:8080
a1b9fc71720d
```

```
                3、查看启动日志 docker logs -f  containerid
                文档:
https://yeasy.gitbooks.io/docker_practice/image/dockerfile/
```

3、实战系列之注册中心打包Docker镜像

简介：讲解使用Docker打包注册中心，上传私有镜像仓库并部署

```
        1、新增maven插件
                <properties>
                   <docker.image.prefix>xdclass</docker.image.prefix>
                </properties>
                <build>
                        <finalName>docker-demo</finalName>
                    <plugins>
                        <plugin>
                            <groupId>com.spotify</groupId>
                            <artifactId>dockerfile-maven-plugin</artifactId>
                            <version>1.3.6</version>
                            <configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>
                                <buildArgs>

<JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
                                </buildArgs>
                            </configuration>
                        </plugin>
                    </plugins>
                </build>

        2、新建Dockerfile
                        FROM openjdk:8-jdk-alpine
                        VOLUME /tmp
                        ARG JAR_FILE
                        COPY ${JAR_FILE} app.jar
                        ENTRYPOINT ["java","-jar","/app.jar"]
        3、打包：
                mvn install dockerfile:build
        4、推送阿里云镜像仓库
                阿里云镜像仓库：https://dev.aliyun.com/search.html
                docker tag 062d2ddf272a registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:eureka-v20180825
                docker push registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:eureka-v20180825
```

```
                docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:eureka-v20180825
        5、查看日志 docker logs -f  containerid
```

4、 实战系列之部署RabbitMQ和配置中心打包Docker镜像

简介：讲解使用Docker打包配置中心，和部署RabbitMQ

```
        1、服务地址 ssh root@47.106.120.173
        部署     rabbitmq：  docker run -d -p 5672:5672 -p 15672:15672
rabbitmq:management

        2、推送镜像
        docker tag 0f636543904e registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:config-server-v20180825
        docker push registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:config-server-v20180825
        docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:config-server-v20180825
```

5、常见问题处理之升级云服务器

简介：处理上节课出现的问题，升级服务器注意事项

```
        1、升级云服务器配置（购买配置后需要重启机器才生效）

        2、启动完成后，需要开启docker
                指令：  systemctl  start docker
        3、所有对外的都要经过网关才可以对外，应用间通信(除非跨机房)都用内网通信
```

6、实战系列之Docker部署Redis

简介：使用Docker安装redis

```
        1、搜索镜像 docker search redis
        2、拉取 docker pull docker.io/redis
        3、启动 docker run --name "xd_redis" -p 6379:6379 -d 4e8db158f18d
        参考:
        docker run --name "xd_redis" -p 6379:6379 -d 4e8db158f18d --
requirepass "123456" -v $PWD/data:/data
        4、访问redis容器里面，进行操作
        docker exec -it 295058d2b92e redis-cli
```

7、生产环境常见问题之配置中心访问

简介：讲解生产环境部署常见问题，配置中心访问路径变化

```
        1、配置中心访问出错，路径不对
        解决：修改所有的注册中心，增加下面配置
    instance:
    instance-id: ${spring.cloud.client.ip-address}:${server.port}
    prefer-ip-address: true
    docker tag 50a12cd66210 registry.cn-
shenzhen.aliyuncs.com/xdclass/xdclass_images:config-server-v20180826
```

8、实战系列之打包Docker镜像打包商品服务和订单服务、网关

简介:打包Docker镜像部署商品服务和订单服务、网关

```
    注意:
    1、maven打包构建，会触发单元测试，部分情况可以跳过,
    mvn install -Dmaven.test.skip=true dockerfile:build
    2、生产环境不能用localhost 或者 127.0.0.1, 一定要用内网通信ip (虚拟主机映射 hosts)
```

9、实战系列云服务器部署网关、订单、商品服务

简介：云服务部署商品服务、订单服务、网关服务

```
        1、拉取镜像、启动
         访问路径
         http://47.106.120.173:8781/api/v1/order/save?product_id=5&user_id=5
         http://47.106.120.173:9000/apigateway/order/api/v1/order/save?
user_id=5&product_id=3&token=232serer
```

第十四章 课程总结和常见问题处理

1、SpringCloud微服务常见问题和解决思路

简介：讲解SpringCloud开发实战常见问题，及解决思路

```
        1、技术选择：SpringCloud全家桶，每个组件又有多个替代，改怎么选择
                1）选择的时候根据公司里面团队人员熟悉程度，降低学习成本
                2）选择社区活跃的并且文档相对较多的，怎么判断活跃，可以看github上代码提交
和start数
        2、云服务选择：  腾讯云，阿里云 ，遇到问题可以提交工单，有专人跟进
        3、部署了应用，但是访问不了
                解决思路：
                        1)查看应用启动是否正常，如果有错误日志，复制错误日志去百度搜
索！！！！！！！特别重要
                        2）启动正常，则先在本机使用 CURL
"http://lcoalhost:8080/api/v1/user/find"   访问对应的接口，看是否有响应
```

3）启动正常，且curl有响应，则检查是否有关闭防火墙，或者开放对应的访问端口，开放端口才可以访问

4）腾讯云和阿里云都是有安全组，类似外层防火墙，一定要去web控制台检查是否有开启端口

4、内网和外网访问，鉴权问题安全问题

1）所有应用只能通过网关提供对外访问的入口

2）应用程序之间通讯，采用内网

5、程序出错或者异常：复制错误日志去百度搜索，网上有很多人会遇到类似的错误，多积累，特别强调

2、课程总结和后续技术规划

简介：总结SpringCloud课程和微服务后续课程规划

1、springCloud全家桶，技术选择和知识点特别多，一定要学会记笔记，微服务更多关注的是里面架构和数据流转，而不是具体的业务。

2、SpringCloud架构这些配置，一般使用一次后就不会多次修改了，进入公司主要还是开发业务，业务开发一般都用springboot，比较少让新人搭建SpringCloud的架构

3、开发业务，无非就是CRUD，增删改查，只不过是初级和高级的区别，封装成通用和不通用的区别

4、微服务和容器盛行的情况下，容器编排和自动缩扩容越来越重要

可以关注：k8s / service mesh /server less 等技术

总结：

1）后续会推出对应的课程，还有项目实战系列，大家记得关注 小D课堂，官网 ：https://xdclass.net,

2）也可以加我微信交流： jack794666918

3）购买对应的课程后，记得进我们小D课堂官方的交流群，我会在里面分享主流技术和答疑，面试经验等等，还会同步更新资料和还超级干货分享

重点来啦：免费赠送你干货文档大集合，包含前端，后端，测试，大数据，运维主流技术文档（持续更新）

https://mp.weixin.qq.com/s/qYnjcDYGFDQorWmSfE7lpQ