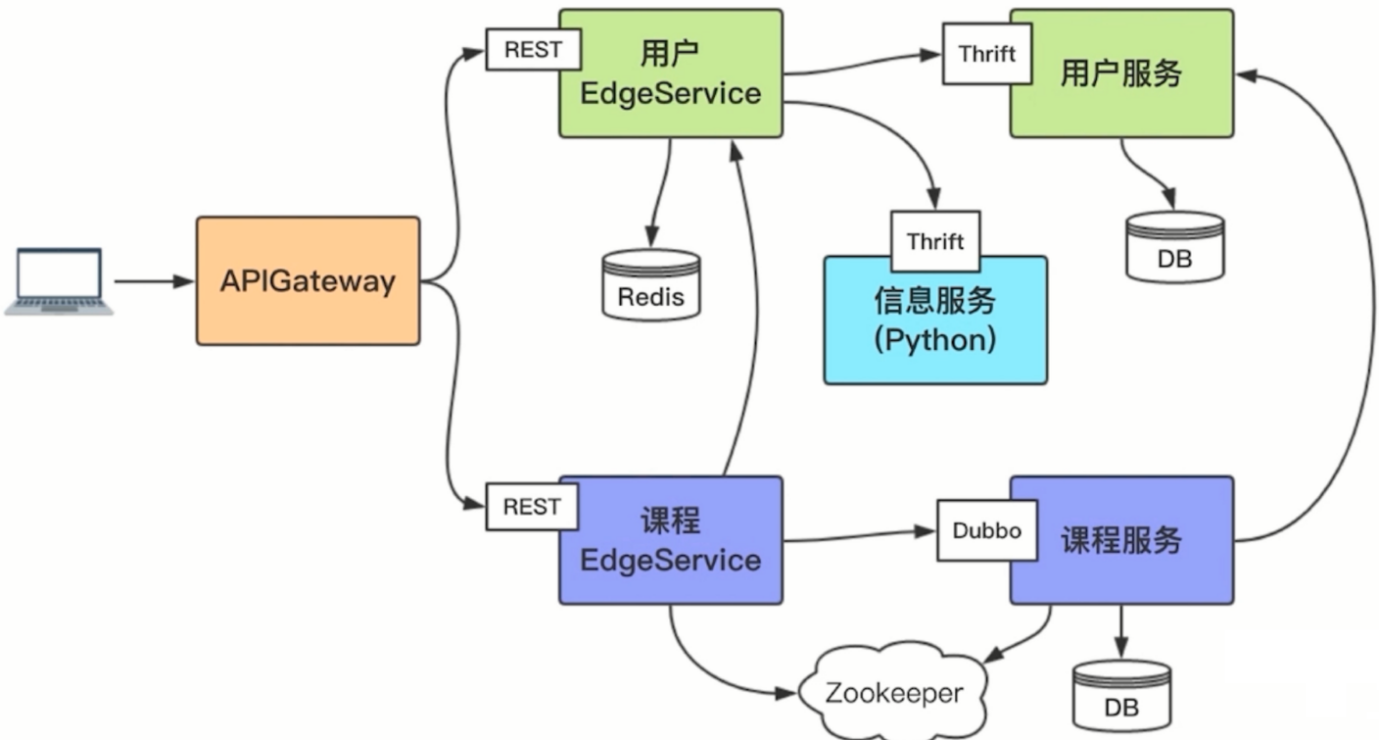
# Docker+Kubernetes(k8s)微服务容器化实践

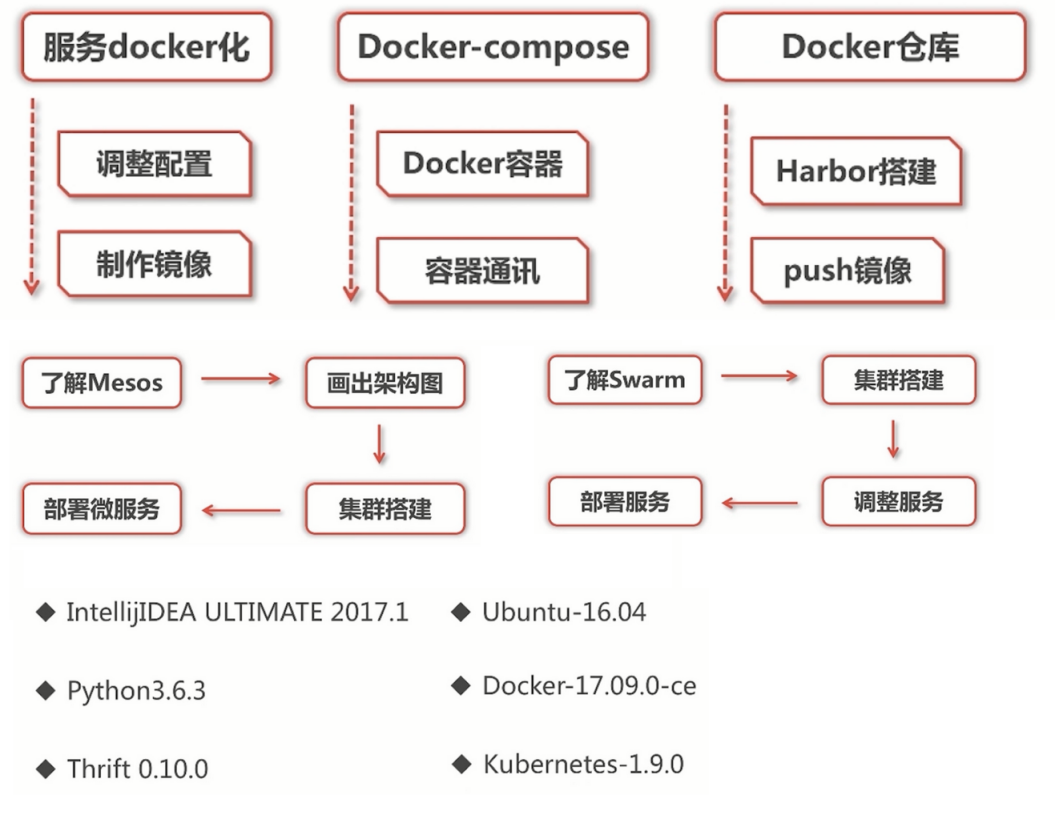
## 第一章 初识微服务

### 1.1概述

（1）Web应用整体框架

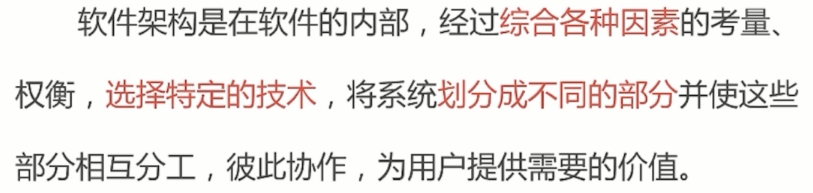


（2）容器化与DevOps工具

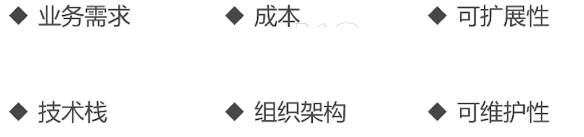


### 1.2软件架构的进化

·软件架构定义



·影响因素



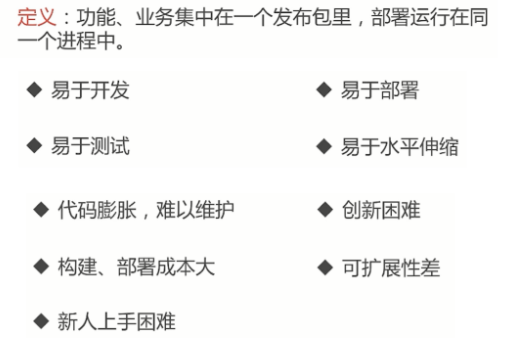
·软件架构进化

·一层架构

·MVC

·dubbo

·单体架构



### 1.3微服务

使用一套小服务来开发单个应用的方法，每个服务运行在独立的进程里，一般采用轻量级的通讯机制互联，并且他们可以通过自动化的方式部署。

·多微才算微

·代码量？

·开发时间

·不可度量

·微服务特征

·单一职责

·轻量级通讯

·隔离性

·有自己的数据

·技术多样性

（1）Grpc 使用HTTP2,序列化效率最高

（2）Dubbo使用的是python序列化

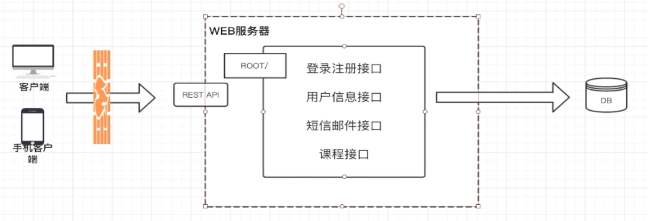
·微服务诞生背景

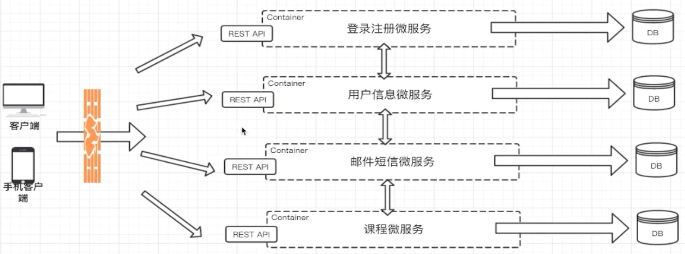
·互联网行业的快速发展

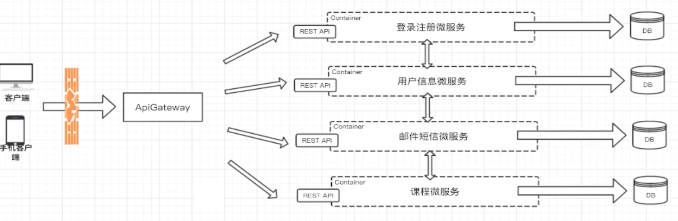
·敏捷开发，精益方法深入人心

·容器技术的成熟

### 1.4画出微服务架构图







### 1.5微服务架构的优势

·独立性

·敏捷性

·技术栈灵活

·高效团队

不足：

·额外的工作

·数据一致性

·沟通成本（接口变更）

## 微服务带来的问题及解决方案分析

### 2.1微服务结构带来的问题

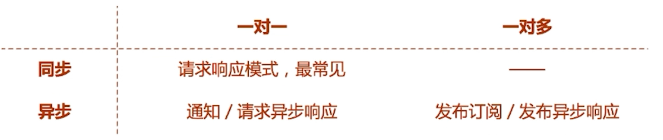
·微服务间如何通讯

·微服务如何发现彼此

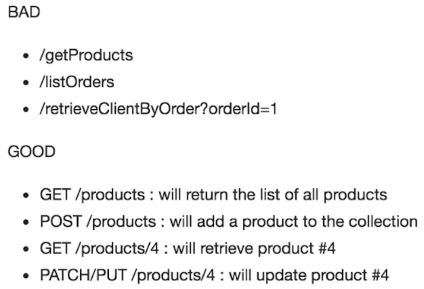
·微服务怎样部署？更新？扩容？

### 2.2微服务间如何通讯

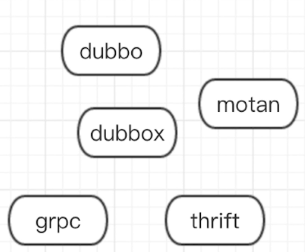
·从通讯模式角度考虑



·REST API



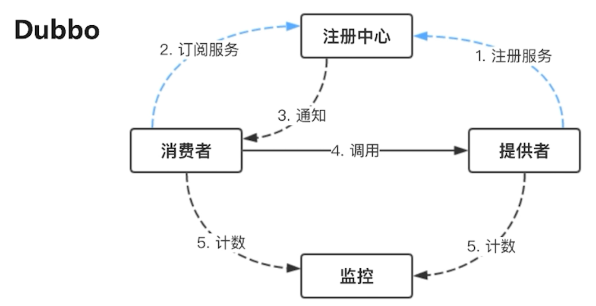
·RPC框架

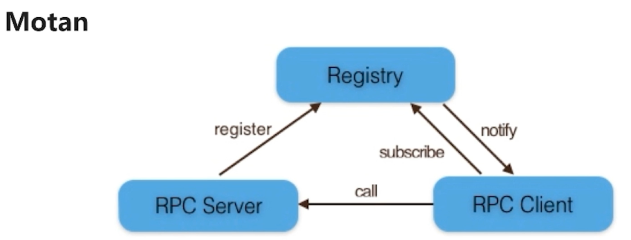


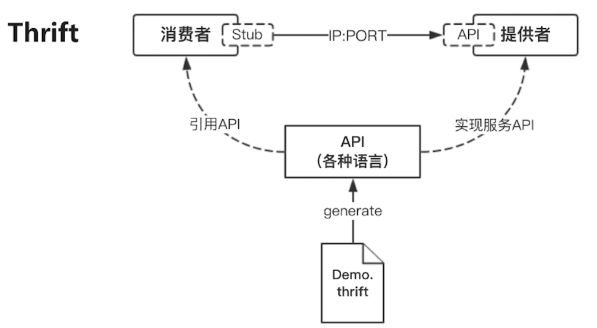
·MQ

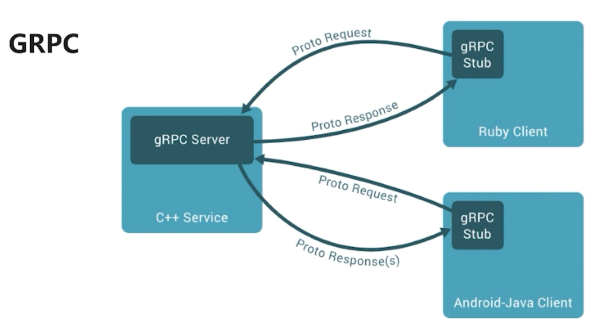
·如何选择RPC框架

1. I/O、线程调度模型
2. 序列化方式，二进制、JSON：影响RPC通讯效率
3. 多语言支持
4. 服务治理





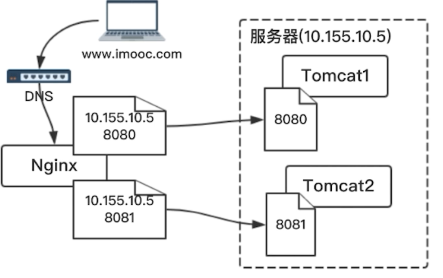




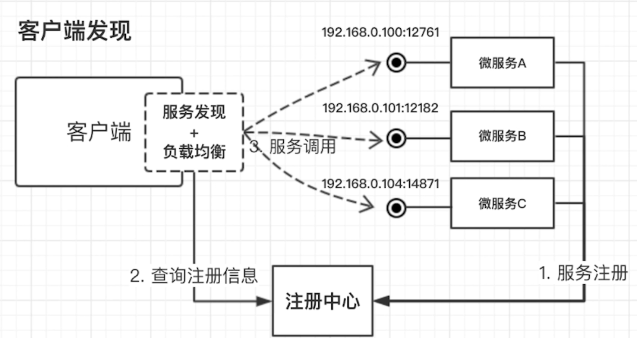


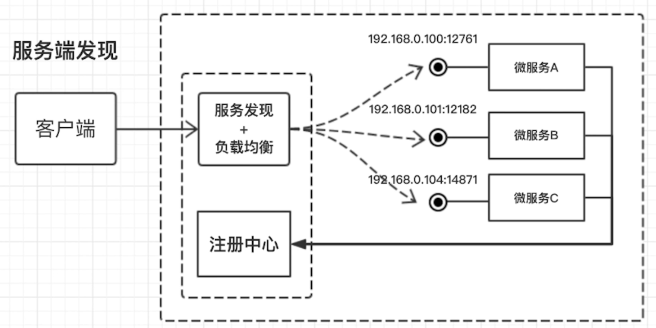
### 2.3服务发现

·传统服务VS微服务



·服务发现





·服务部署、更新和扩容

·服务编排

·流行的服务编排工具



### 2.4SpringBoot & SpringCloud

·SpringBoot与微服务

·SpringBoot核心功能

·独立运行 java -jar xxx.jar

·内嵌web服务器

·简化配置

·准生产的应用监控

·微服务SpringCloud

·统一配置管理

·服务注册与发现

·服务间调用

·负载均衡

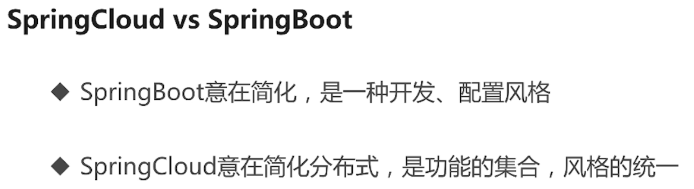
·分布式锁

·分布式Session

·深入理解

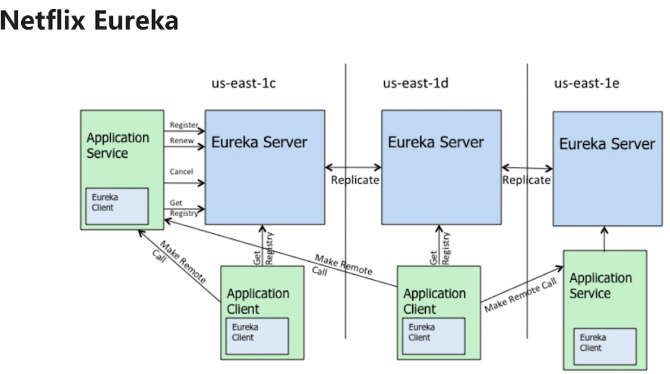
·一系列框架

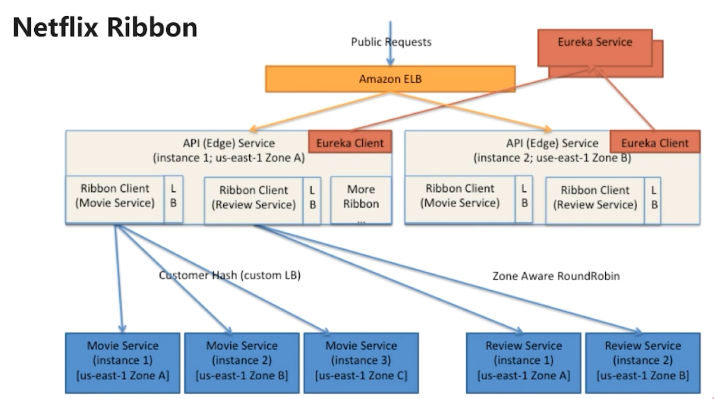
·简化java的分布式开发



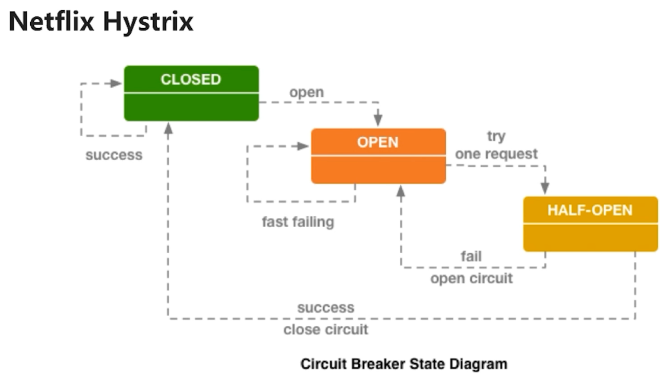
·SpringCloud核心组件

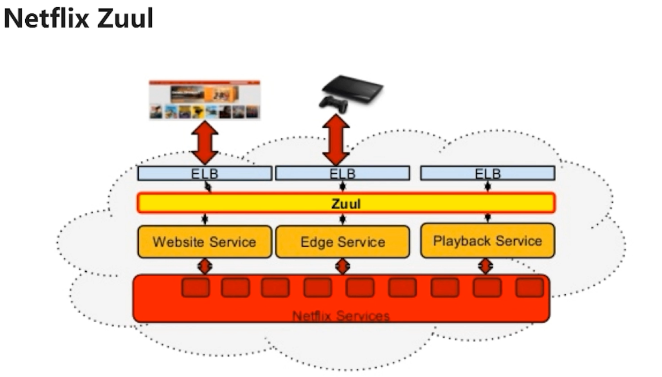


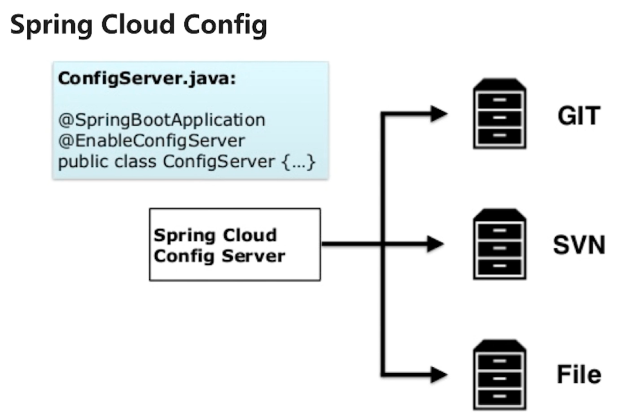




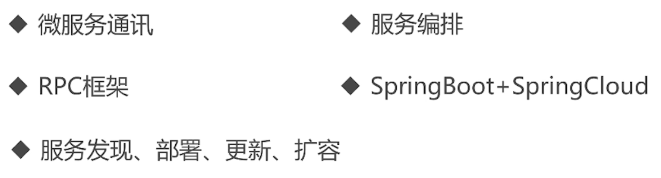
·Edge Service：Web RESTAPI







### 2.5小节



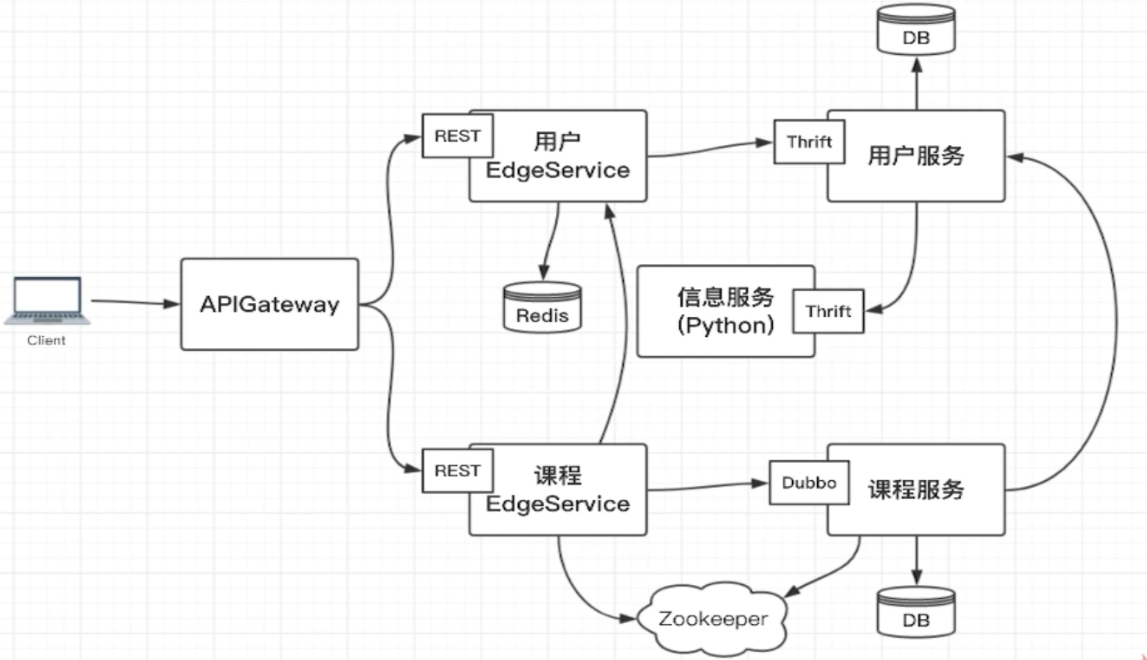
## 微服务开发

### 3.1微服务业务分析

·业务场景

·用户可以注册和登录

·登录用户可以对课程进行CRUD



### 3.2Thirft安装和验证

|  |
| --- |
| namespace java com.byf.thrift.demo  namespace py thrift.demo  service DemoService {  void sayHello(1:string name);  } |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen java demo.thrift  C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen py demo.thrift |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test 的目录  2019/09/22 14:57 <DIR> .  2019/09/22 14:57 <DIR> ..  2019/09/22 14:56 120 demo.thrift  2019/09/22 14:56 <DIR> gen-java  2019/09/22 14:57 <DIR> gen-py  2019/09/22 14:52 3,887,104 thrift-0.12.0.exe |

### 3.3Python开发信息服务

|  |
| --- |
| namespace java com.byf.thrift.message namespace py message.api  service MessageService {  bool sendMobileMessage(1:string mobile, 2:string message);  bool sendEmailMessage(1:string email, 2:string message); } |

|  |
| --- |
| G:\MicroService\microservice\message-thrift-python-service\thrift>thrift-0.12.0.exe --gen py -out ../ message.t  hrift |

idea installed package failed目录缺少packaging\_tool.py

Intellij IDEA安装SDKs失败：

将目录F:\Program Files\JetBrains\IntelliJ IDEA 2019.1.1\plugins\python\helpers

拷贝至Intellij IDEA配置目录：C:\Users\BYF\.IntelliJIdea2019.1\config

|  |
| --- |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**None**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |
| org.apache.thrift.transport.TTransportException: java.net.ConnectException: Connection refused: connect  客户端连接时报错，修改None为具体IP地址 |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**"127.0.0.1"**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |

### 3.4dubbo上手示例

<http://dubbo.apache.org/en-us/docs/user/quick-start.html>

|  |
| --- |
| <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- provider's application name, used for tracing dependency relationship -->* <**dubbo:application name="demo-provider"**>  <**dubbo:parameter key="qos.enable" value="true"**/>  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"**/>  <**dubbo:parameter key="qos.port" value="33333"**/>  </**dubbo:application**>  *<!-- use multicast registry center to export service -->* <**dubbo:registry address="multicast://224.5.6.7:1234?unicast=false"**/>  *<!-- use dubbo protocol to export service on port 20880 -->* <**dubbo:protocol name="dubbo" port="20880"**/>  *<!-- service implementation, as same as regular local bean -->* <**bean id="demoService" class="org.apache.dubbo.demo.provider.DemoServiceImpl"**/>  *<!-- declare the service interface to be exported -->* <**dubbo:service interface="org.apache.dubbo.demo.DemoService" ref="demoService"**/> </**beans**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- consumer's application name, used for tracing dependency relationship (not a matching criterion),  don't set it same as provider -->* <**dubbo:application name="demo-consumer"**>  <**dubbo:parameter key="qos.enable" value="true"** />  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"** />  <**dubbo:parameter key="qos.port" value="33333"** />  </**dubbo:application**>  *<!-- use multicast registry center to discover service -->  <!--<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" check="false" />-->  <!-- <dubbo:registry valid="false" check="false" zookeeperProtocol="false" id="org.apache.dubbo.config.RegistryConfig" prefix="dubbo.registries." />-->  <!-- generate proxy for the remote service, then demoService can be used in the same way as the  local regular interface -->* <**dubbo:reference id="demoService" check="false" interface="org.apache.dubbo.demo.DemoService" url="127.0.0.1:20880"**/> </**beans**> |

-Ddubbo.application.qos.enable=true -Ddubbo.application.qos.port=33333 -Ddubbo.application.qos.accept.foreign.ip=false

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-consumer</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-provider</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *###set log levels###* **log4j.rootLogger**=**info, stdout** *###output to the console###* **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.out log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**[%d{dd/MM/yy hh:mm:ss:sss z}] %t %5p %c{2}: %m%n** |

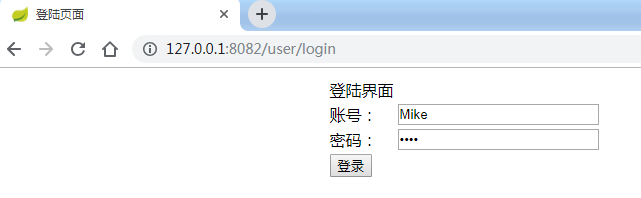
### 3.5课程服务

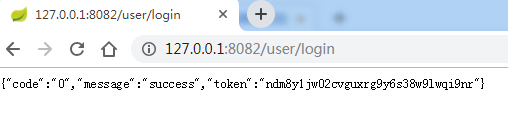
|  |
| --- |
| namespace java com.byf.thrift.user  struct UserInfo {  1:i32 id,  2:string username,  3:string password,  4:string realName,  5:string mobile,  6:string email,  7:string intro,  8:i32 stars }  service UserService {  UserInfo getUserById(1:i32 id);  UserInfo getTeacherById(1:i32 id);   UserInfo getUserByName(1:string username);  void registerUser(1:UserInfo userInfo); } |

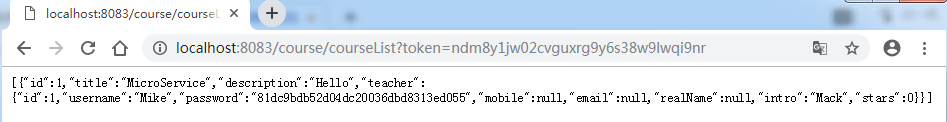
G:\MicroService\microservice\user-thrift-service-api\thrift>thrift-0.12.0.exe --gen java -out ../src/main/java user

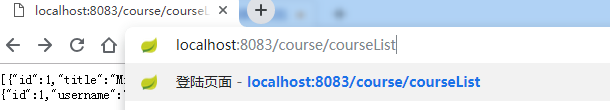
\_service.thrift

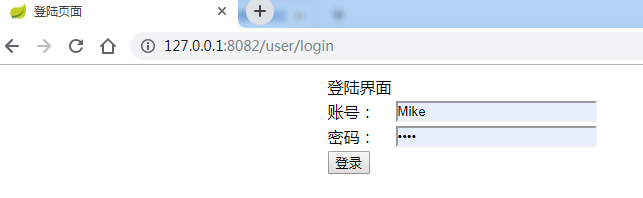
|  |
| --- |
| java.lang.IllegalStateException: Failed to check the status of the service com.byf.course.service.ICourseService. No provider available for the service com.byf.course.service.ICourseService from the url zookeeper://127.0.0.1:2181/com.alibaba.dubbo.registry.RegistryService?application=course-service&dubbo=2.5.3&interface=com.byf.course.service.ICourseService&methods=courseList&pid=7732&side=consumer&timestamp=1569339917977 to the consumer 192.168.0.107 use dubbo version 2.5.3 |
| 原因已找到。错误在CourseServiceImpl类中@Service注解用的是dubbo类中的，不是Spring中的！！！一定要注意 com.alibaba.dubbo.config.annotation.Service |











## 第4章微服务编排前奏

### 4.1服务Docker化

idea maven package无法将依赖的jar包打入一个jar包

|  |
| --- |
| <**plugin**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-maven-plugin</**artifactId**>  <**executions**>  <**execution**>  <**goals**>  <**goal**>repackage</**goal**>  </**goals**>  </**execution**>  </**executions**> </**plugin**> |

Centos7上安装docker

|  |
| --- |
| **一、安装docker** 1、Docker 要求 CentOS 系统的内核版本高于 3.10 ，查看本页面的前提条件来验证你的CentOS 版本是否支持 Docker 。  通过 **uname -r**命令查看你当前的内核版本  $ uname -r  2、使用 root 权限登录 Centos。确保 yum 包更新到最新。  $ sudo yum update  3、卸载旧版本(如果安装过旧版本的话)  $ sudo yum remove docker docker-common docker-selinux docker-engine  4、安装需要的软件包， yum-util 提供yum-config-manager功能，另外两个是devicemapper驱动依赖的  $ sudo yum install -y yum-utils device-mapper-persistent-data lvm2  5、设置yum源  $ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo  IMG_256  6、可以查看所有仓库中所有docker版本，并选择特定版本安装  $ yum list docker-ce --showduplicates | sort -r  IMG_257  7、安装docker  $ sudo yum install docker-ce #由于repo中默认只开启stable仓库，故这里安装的是最新稳定版17.12.0  $ sudo yum install <FQPN> # 例如：sudo yum install docker-ce-17.12.0.ce  IMG_258  8、启动并加入开机启动  $ sudo systemctl start docker  $ sudo systemctl enable docker  9、验证安装是否成功(有client和service两部分表示docker安装启动都成功了)  $ docker version  IMG_259 |

拉取镜像

|  |
| --- |
| Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: TLS handshake timeout  [root@localhost /etc/docker]#vi daemon.json  {  "registry-mirrors":["https://6kx4zyno.mirror.aliyuncs.com"]  }  [root@localhost /etc/docker]#systemctl restart docker.service  [root@localhost /etc/docker]#systemctl status docker.service |

|  |
| --- |
| Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]    -d, --detach=false 指定容器运行于前台还是后台，默认为false  -i, --interactive=false 打开STDIN，用于控制台交互  -t, --tty=false 分配tty设备，该可以支持终端登录，默认为false  -u, --user="" 指定容器的用户  -a, --attach=[] 登录容器（必须是以docker run -d启动的容器）  -w, --workdir="" 指定容器的工作目录  -c, --cpu-shares=0 设置容器CPU权重，在CPU共享场景使用  -e, --env=[] 指定环境变量，容器中可以使用该环境变量  -m, --memory="" 指定容器的内存上限  -P, --publish-all=false 指定容器暴露的端口  -p, --publish=[] 指定容器暴露的端口  -h, --hostname="" 指定容器的主机名  -v, --volume=[] 给容器挂载存储卷，挂载到容器的某个目录  --volumes-from=[] 给容器挂载其他容器上的卷，挂载到容器的某个目录  --cap-add=[] 添加权限，权限清单详见：http://linux.die.net/man/7/capabilities  --cap-drop=[] 删除权限，权限清单详见：http://linux.die.net/man/7/capabilities  --cidfile="" 运行容器后，在指定文件中写入容器PID值，一种典型的监控系统用法  --cpuset="" 设置容器可以使用哪些CPU，此参数可以用来容器独占CPU  --device=[] 添加主机设备给容器，相当于设备直通  --dns=[] 指定容器的dns服务器  --dns-search=[] 指定容器的dns搜索域名，写入到容器的/etc/resolv.conf文件  --entrypoint="" 覆盖image的入口点  --env-file=[] 指定环境变量文件，文件格式为每行一个环境变量  --expose=[] 指定容器暴露的端口，即修改镜像的暴露端口  --link=[] 指定容器间的关联，使用其他容器的IP、env等信息  --lxc-conf=[] 指定容器的配置文件，只有在指定--exec-driver=lxc时使用  --name="" 指定容器名字，后续可以通过名字进行容器管理，links特性需要使用名字  --net="bridge" 容器网络设置:  bridge 使用docker daemon指定的网桥  host //容器使用主机的网络  container:NAME\_or\_ID >//使用其他容器的网路，共享IP和PORT等网络资源  none 容器使用自己的网络（类似--net=bridge），但是不进行配置  --privileged=false 指定容器是否为特权容器，特权容器拥有所有的capabilities  --restart="no" 指定容器停止后的重启策略:  no：容器退出时不重启  on-failure：容器故障退出（返回值非零）时重启  always：容器退出时总是重启  --rm=false 指定容器停止后自动删除容器(不支持以docker run -d启动的容器)  --sig-proxy=true 设置由代理接受并处理信号，但是SIGCHLD、SIGSTOP和SIGKILL不能被代理 |

|  |
| --- |
| [root@localhost /etc/docker]#docker pull openjdk:8-jre  8-jre: Pulling from library/openjdk  092586df9206: Pull complete  ef599477fae0: Pull complete  4530c6472b5d: Pull complete  f68761487d74: Pull complete  44e59d0ee089: Pull complete  81c7d41ff281: Pull complete  Digest: sha256:b2096eb6ac13c88e431e100db54c470f48c4228b1f2503a8419dbb9a7d3df8f6  Status: Downloaded newer image for openjdk:8-jre  docker.io/library/openjdk:8-jre  [root@localhost /etc/docker]#docker images  REPOSITORY TAG IMAGE ID CREATED SIZE  openjdk 8-jre fd62519d2906 2 weeks ago 246MB |

|  |
| --- |
| Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: TLS handshake timeout  [root@localhost /etc/docker]#vi daemon.json  {  "registry-mirrors":["https://6kx4zyno.mirror.aliyuncs.com"]  }  [root@localhost /etc/docker]#systemctl restart docker.service  [root@localhost /etc/docker]#systemctl status docker.service  docker run -it --name=jdk8 --hostname=centos --entrypoint bash openjdk:8-jre  192.168.1.108  [root@localhost /soft/microservice/user-thrift-service]#ll  total 4  -rw-r--r--. 1 root root 175 Sep 28 18:08 Dockerfile  drwxr-xr-x. 2 root root 50 Sep 28 18:08 target  [root@localhost /soft/microservice/user-thrift-service]#docker build -t user-service:latest .  Sending build context to Docker daemon 12.07MB  Step 1/4 : FROM openjdk:8-jre  ---> fd62519d2906  Step 2/4 : MAINTAINER BYF baiyifan@163.com  ---> Running in bcc414d0ec59  Removing intermediate container bcc414d0ec59  ---> 2c6cc8a8208b  Step 3/4 : COPY target/user-thrift-service-1.0-SNAPSHOT.jar /user-service.jar  ---> 33ddd61c0254  Step 4/4 : ENTRYPOINT ["java", "-jar", "/user-service.jar"]  ---> Running in f467607aac7c  Removing intermediate container f467607aac7c  ---> 8edde3390dc2  Successfully built 8edde3390dc2  Successfully tagged user-service:latest  [root@localhost /soft/microservice/user-thrift-service]#docker images  REPOSITORY TAG IMAGE ID CREATED SIZE  user-service latest 8edde3390dc2 5 seconds ago 258MB  openjdk 8-jre fd62519d2906 2 weeks ago 246MB  docker rm user-service  docker run -d --name=user-service --hostname=centos user-service:latest --mysql.address=192.168.1.108  docker rm user-edge-service  docker run -d --name=user-edge-service --hostname=centos user-edge-service:latest --redis.address=192.168.1.108  docker rm course-service  docker run -d --name=course-service --hostname=centos course-service:latest --zookeeper.address=192.168.1.108 --mysql.address=192.168.1.108  docker stop course-edge-service  docker rm course-edge-service  docker run -d --name=course-edge-service --hostname=centos course-edge-service:latest --zookeeper.address=192.168.1.108  docker run -it --name=course-edge-service --hostname=centos course-edge-service:latest --zookeeper.address=192.168.1.108  docker stop api-gateway-zuul  docker rm api-gateway-zuul  docker run -d --name=api-gateway-zuul --hostname=centos api-gateway-zuul:latest  docker run -it --name=api-gateway-zuul --hostname=centos api-gateway-zuul:latest |