

Kotlin电商

架构文档

2019-9-23

北京

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 版本 | 修订者 | 备注 |
| 2019.9.1 | v1.0 | 于新海 |  |
| 2019.9.23 | v1.1 | 于新海 |  |

目录

[系统分析 5](#_Toc20140041)

[数据量分析 5](#_Toc20140042)

[项目难点分析 5](#_Toc20140043)

[整体架构 6](#_Toc20140044)

[逻辑架构 6](#_Toc20140045)

[物理架构 7](#_Toc20140046)

[系统划分 7](#_Toc20140047)

[客户端划分： 7](#_Toc20140048)

[服务器端组件划分： 7](#_Toc20140049)

[微服务划分： 8](#_Toc20140050)

[系统设计 9](#_Toc20140051)

[关键组件 9](#_Toc20140052)

[缓存 10](#_Toc20140053)

[存储 10](#_Toc20140054)

[网络 10](#_Toc20140055)

[计划任务 10](#_Toc20140056)

[规则引擎 10](#_Toc20140057)

[工作流 11](#_Toc20140058)

[平台设置 11](#_Toc20140059)

[支付流程 11](#_Toc20140060)

[电子签章 11](#_Toc20140061)

[推荐引擎 11](#_Toc20140062)

[日志收集分析 11](#_Toc20140063)

[链路追踪Zipkin 11](#_Toc20140064)

[软件环境 12](#_Toc20140065)

[H5客户端 12](#_Toc20140066)

[Android客户端 12](#_Toc20140067)

[IOS客户端 12](#_Toc20140068)

[服务器端 12](#_Toc20140069)

[Java开发环境 13](#_Toc20140070)

[服务器参数 14](#_Toc20140071)

[域名分配 14](#_Toc20140072)

[二级域名 14](#_Toc20140073)

[端口号 14](#_Toc20140074)

[Tomcat8参数设置 15](#_Toc20140075)

[Nginx参数 16](#_Toc20140076)

[安全性 16](#_Toc20140077)

[普通Web接口的有效性验证 16](#_Toc20140078)

[提供给第三方供应商的接口 16](#_Toc20140079)

[防Sql注入 17](#_Toc20140080)

[防盗刷 17](#_Toc20140081)

[群发邮件 防止封号 17](#_Toc20140082)

[防盗链 17](#_Toc20140083)

[设计规范 17](#_Toc20140084)

[Java开发规范 17](#_Toc20140085)

[接口设计规范 17](#_Toc20140086)

[跨域 18](#_Toc20140087)

[幂等性 18](#_Toc20140088)

[返回结构 18](#_Toc20140089)

[数据库设计 20](#_Toc20140090)

[关系型数据库 20](#_Toc20140091)

[Redis设计规范 21](#_Toc20140092)

[项目管理工具 21](#_Toc20140093)

[文档管理： 21](#_Toc20140094)

[代码管理 21](#_Toc20140095)

[Bug管理 22](#_Toc20140096)

[部署集成（CI，CD）Devops流水线 22](#_Toc20140097)

[Git分支： 22](#_Toc20140098)

[打包部署： 22](#_Toc20140099)

[沟通工具 23](#_Toc20140100)

# 系统分析

根据客户需求分析出大致的数据量：

1. 使用用户量。
2. 每个用户生产的数据量，总数据量，存储容器，计算数据库及磁盘容量。
3. 评估系统的流量及高峰流量，计算带宽及每个服务的负载数量。

## 数据指标

Mysql: 每秒200次插入

Mongo: 每秒10000次插入

TimescaleDB：每秒15000次插入

## 数据量分析

用户量：C端用户： 2000万，B端用户：30万。平台10个业务微服务

Tps：

总 2000万\*0.3活跃\* 80%集中到4个小时 \* 100次连接 = 3.4万

接口占 20% = 6800

单节点接口处理能力每秒按400算，负载到 17 个节点。每个微服务用2个节点做负载。

Sku：

30万\*10个产品\*8个Sku = 2400万

图片量：

30万 \* 10个产品 \* 10个详情\* 100K = 3T

30万 \* 10个宣传图 \* 100K = 1T

视频量：

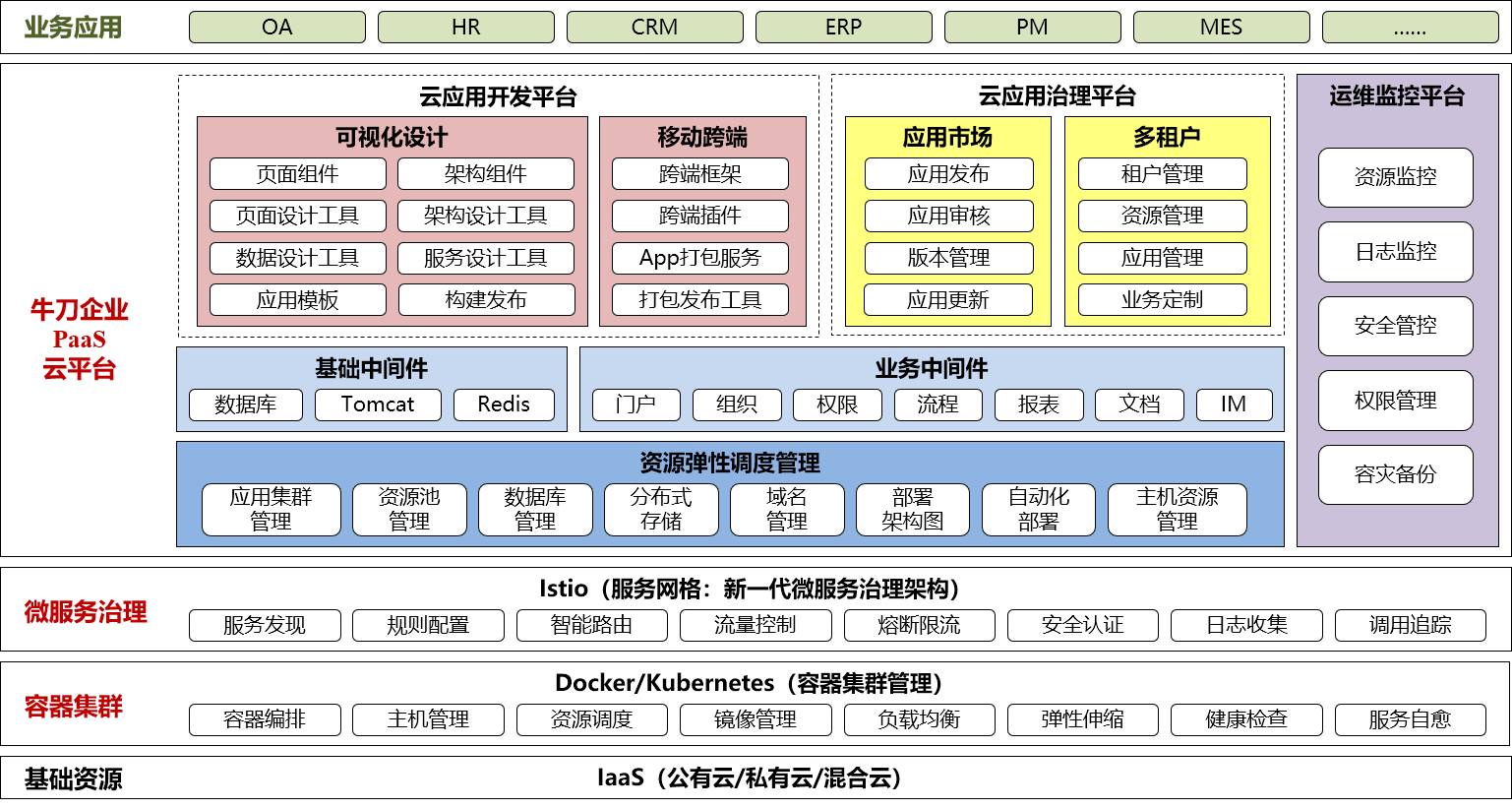
30万\*10个产品\*1个视频\* 10MB = 30T

## 项目难点分析

* 数据安全性。防拖库，防劫持。
* 优惠券、营销活动等计算规则。
* 支付，分润，数据的完整性、一致性。
* 活动集中时段，保障系统的高可用。如秒杀
* 推荐算法。
* K8s部署，Istio流量治理。

# 整体架构

## 技术架构



## 逻辑架构



## 物理架构

Mysql 采用 Mariadb galera cloud 集群 3个主节点 + 读写分离3个从节点

MongoDb采用分片副本集 6节点

Redis 原生集群 6点节

Es集群

存储：FastDfs 3台，或MinIO分布式文件系统。

Ceph 用于保存k8s 的数据库及配置文件

## 系统划分

### 客户端划分：

* 官网
* C端Android，买家端
* C端Ios，买家端
* C端小程序，买家端
* B端 商户端 Web端
* B端 商户端 Android
* B端 商户端 Ios
* 骑手端，服务端 Android端
* 骑手端，服务端 Ios端
* Boss平台，运营平台Web端
* 代理商平台Web端

### 服务器端组件划分：

* KeepAlived + Nginx
* K8s环境
* 分布式文件集群
* MariaDb Galera Cloud
* Redis Cluster
* Mongo Sharding Replica Set
* Mq 集群
* Es 集群
* Zookeeper
* EFK组件
* Skywalking Server

### 微服务划分：

#### 系统微服务

* 注册中心
* 配置中心
* 网关

#### 中台微服务

* 用户：登录，注册，token续期，C端用户，B端商户，服务者(骑手), 微信登录，QQ登录
* 第三方服务：发短信，消息推送，运营统计，邮件，分享，物流、配送，语音识别，OCR识别。
* IM：
* 客服：工作台，话术库，问题库，派单，智能客服。
* 商铺：CMS，装修。
* 商品：Sku设置，详情，分类。供应链。
* 订单：生成订单，评价，电子发票。
* 支付：C2B，
* 财务：B2B，分润
* 活动，优惠券，积分：规则引擎
* 文件存储服务：视频，图片
* 电子签章
* 集成第三方地图
* 搜索引擎：
* 推荐：流式计算Flink
* 舆情分析：运营用。
* 爬虫：抓取
* 广告
* 任务调度
* EFK

#### 业务微服务

* 官网服务接口
* 用户端微服务：C端
* 商户端微服务:
* 服务端微服务：骑手
* Boss平台、城市合伙人平台 微服务：
* 客服即时通讯 微服务
* 开放平台：对外是提供Api

# 系统设计

## 关键组件

### 缓存

使用Redis做热点数据缓存，对缓存进行依赖设置，对破坏缓存的接口，做统一管理。

### 存储

* 业务图片，视频：分布式存储方案采用MinIO 或 FastDFS部署
* K8s数据存储：Ceph

### 数据

#### 上报数据

时序数据库可以使用 TimeScaleDB 或 MongoDb

时序数据库前面用Mq缓冲。

#### 消息推送

要推送到App端的消息，先推送到Mq，消费者拿到消息顺序推送给使用者，同时记录到Mongo数据库表中。

### 网络

静态文件的CDN

动态文件的DSA

### 计划任务

循环任务使用 @Scheduled(fixedDelay = 10000)

定时任务使用 @Scheduled(cron = "0 0 0 \* \* ?")

每天晚上11点到5点执行。

* 文件备份
* 数据库备份
* 商家服务过期
* 活动过期

### 规则引擎

使用Drools在后台动态维护规则，包括：营销活动，分销规则，组合商品，可用优惠券。

用户在选择商品，添加到购物车后，系统在结算时，自动使用后台添加的规则引擎，选择可用优惠

### 工作流

JBpm5

所有审批，使用工任务流，如财务审核

### 平台设置

多商户Saas系统，可以定义角色，权限，RBAC

### 支付流程

### 电子签章

### 推荐引擎

### 日志收集分析

采用 EFK （Elasticsearch , FileBeat, Kibana）进行日志收集显示。包括以下日志：

* 访问日志：
* 安全日志：登录，登出，所在地，IMEI码。
* 操作日志：CRUD Sql日志及重要的业务日志。
* 异常日志：

### 链路追踪Skywalking

所有URL调用，请求参数，返回参数，执行时长。

### 监控 Grafana

### 数字运营指标

通过Cookie标识客户端Id，用于Uv。

通过行为分析得到用户最后关闭的页面，分析跳出率。

页面访问时长，其它页面可以通过行为分析得到：

最后一页关闭不能确定。

每个页面都需要有一个请求。

## 软件环境

采用最新稳定版本，原生开发，方便后期维护升级。大版本一年升级一次。

### H5客户端

采用 Vue + ElementUI ，H5 + Css3 ，手脚架：vu cli ，兼容主流浏览器：Chrome64+，IE10+，对手机，PC不同分辨率进行适配。

官网需要静态化。

### Android客户端

兼容Android6及以上版本，采用Kotlin + Java。可以采用主流的： Retrofit + mvp + Rxjava模式 ， 也可采用最新的：flutter 和jetpack框架

### IOS客户端

兼容IOS9以上版本，采用 Swift+ObjectC。主流框架：AFNetWorking，FMDB，CoreData，Masonry，ImageSD，Sqlite

### 服务器端

硬件：单节点 4Core16G内存，千兆网卡，Ceph，MinIO（或FastDfs，或MongoDB File Server）。

软件运行环境：

* CentOs7+稳定版本
* JDK8+，建议最新的 JDK12
* Tomcat8+，建议最新的Tomcat9
* Nginx:1.10，对外使用HTTPS（Https CDN）
* KeepAlived:2.0
* K8s:1.15.3
* Docker:18.09+

数据库：

* MariaDb
* MongoDb
* Redis
* ElasticSearch
* RabbitMq 或 Kafaka

### Java开发环境

开发环境：

* Idea + Alibaba规范插件
* Git
* XShell，XFtp
* Office，Visio，Project
* Robo 3T
* HeidiSql、Navicate
* WebStorm
* PostMan接口测试工具
* Fiddler，抓包工具
* Maven

开发组件：

* SpringBoot2+ , SpringCloud / Dubbo
* Swagger
* Spring
* Spring-Jdbc
* Spring-Mybatis，Spring-Mybatis-generator
* DBCP 或 Durid
* Cache
* MyCat 或 。。。
* Spring Oauth
* Spring Security

## 服务器参数

### 域名分配

* Lwljuyang.com
* Lwljuyang.net
* Lwljuyang.cn

### 二级域名

形式： xx.域名.com ， 二级域名尽量短。

1. 直接访问域名，表示 HTML 访问
2. i 表示图片服务
3. v表示视频服务
4. f 表示其它文件服务。
5. s 表示接口服务
6. r 表示资源服务，css,js,等。

### 端口号

数据库服务器的实际端口号 = 默认端口号 + 200 , 如果实际端口非法或被占用，则 -200。如：

1. mariadb：3506
2. mongo : 27217
3. redis： 6579
4. rabbitMq： 5872
5. Es：9500

### Tomcat8参数设置

针对 4Core+8G内存，每个节点启动一个服务设置。如果是4Core16G内存，每个节点启动两个服务。

JVM参数设置，可通过参数指定，或在Seten.sh 中通过 export CATALINA\_OPTS 指定：

-Xmx6400m : 堆内存的最小大小，= 8G \* 80% = 6.4G

-Xms6400m 同, 避免内存变化.

-XX:NewRatio=3：年轻代(包括Eden和两个Survivor区)与年老代的比值(除去持久代)

-XX:+UseParallelGC

-XX:ParallelGCThreads=8 ： 8核并行收集

-XX:+UseParallelOldGC

-XX:MaxGCPauseMillis=100

-XX:-UseAdaptiveSizePolicy : 禁用: <https://blog.csdn.net/rickiyeat/article/details/78906235>

-XX:MaxTenuringThreshold=10 ： 垃圾最大年龄

server.xml

<Executor

name="tomcatThreadPool"

namePrefix="catalina-exec-"

maxThreads="500"

minSpareThreads="100"

prestartminSpareThreads = "true"

maxQueueSize = "100"

/>

<Connector

executor="tomcatThreadPool"

port="8080"

protocol="org.apache.coyote.http11.Http11Nio2Protocol"

connectionTimeout="20000"

maxConnections="10000"

redirectPort="8443"

enableLookups="false"

acceptCount="100"

maxPostSize="10485760"

compression="on"

disableUploadTimeout="true"

compressionMinSize="2048"

acceptorThreadCount="2"

compressableMimeType="text/html,text/xml,text/plain,text/css,text/javascript,application/javascript"

URIEncoding="utf-8"

/>

# 注释掉 ajp

<!-- <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" /> -->

### Nginx参数

* 启用 gzip
* 启用 http2
* 启用 https

## 客户端

视频，图片的上传，要客户端压缩到720P，即最大：1080\*720。压缩后再上传到服务器。

## 安全性

### 普通Web接口的有效性验证

采用Jwt 的 token 机制，token 有效时间30分钟，验证token 有效性。

### 提供给第三方供应商的接口

第三方供应商都会有一个唯一的商户Id: MerchantId，以及密钥 ？

采用AppId + token + 传输数据签名 方式。

### 防Sql注入

对Sql参数使用参数化。

### 防盗刷

对短信下发接口做规则校验：

1. 同一个外网IP，每分钟限发100个。
2. 同一手机号，每小时限发 10个。每天限发100个。

### 群发邮件 防止封号

### 防盗链

针对视频和图片，做以下规则：

# 设计规范

## Java开发规范

* 参见附件：《Java开发手册》v1.5.0 华山版.pdf
* Idea 安装 《Alibaba Java Code Guideline》，各开发人员自行进行检查，所有代码注释必须表述清晰。
* 明确业务中的枚举。并映射到实体字段。

## 接口设计规范

接口的状态码具有通用含义，请求返回数据使用 ContentType: application/json;utf8; 内容使用 JSON 数据。

### 跨域

由于网页与接口使用不同的域名。所以，服务器网关对接口做仅支持指定域名的跨域支持、处理。

### 幂等性

查询，删除，天生幂等性

新增数据，更新数据，生成订单，支付

1. 打开页面，服务器Redis生成 token
2. 新增或更新时，带入 token，请求接口
3. 服务器删除token，删除不成功，返回错误，防止重复数据和重复更新。
4. 执行新增或更新逻辑

### 返回结构

#### 普通返回结构

* msg：String
* code：String
* data: Any
* cause: String

msg: 表示接口错误时的错误消息, 如果接口成功，返回空字符串。必须返回。

code: 可选，除非必要时返回，否则为空。

cause: 可选 ，当参数校验失败时，指出是哪个参数。如果成功，返回空字符串。

data: 当需要返回具体的数据时, 使用 data 。 非必须返回。

#### 列表请求结构

* skip：Int
* take：Int
* 自定义查询参数

skip: 跳过行数。

take: 每页条数。

注：

\_bid\_：通过 header 传递，表示上次该请求返回数据被客户端缓存，该数据缓存的

#### 列表返回结构

列表返回结构 继承 普通返回结构，额外添加两个字段：

* total: Int
* list: Array

list: 表示返回的列表数据。

total: 总条数，如果为 -1 表示服务器没有查询总条数，客户端使用之前的缓存总条数。

注：

\_bid\_：通过 header 传递，表示该数据的 md5 + CheckSum

#### 列表返回状态码

状态码使用标准的HTTP状态码。

* 200： 成功
* 500： 内部服务器错误
* 401： （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。

## 数据库设计

标准规范：

* 大驼峰命名
* 有注释
* 表名，列名，大小写不敏感

统一名称：

* Id
* Name名称
* Remark备注
* Code 编码
* CreateAt 创建时间
* CreateBy 创建者
* Version 版本号
* UpdateAt 更新时间
* UpdateBy 更新者

### 关系型数据库

使用 MariaDb 替代MySql，使用 MariaDb Galera Cloud 多主集群，通过MaxScale 对外提供读写分离。

* 必须有主键
* InnoDB引擎
* 查询必须分页，避免大数据量返回。
* 由于存在同步的性能损耗，尽量把日志数据放到 MongoDb中。
* 对增量数据进行归档，如：订单仅保留今年及上年的，之前的订单放到 order\_${Year} 中。

## Redis设计规范

参见：《Redis设计规范》.pdf

## 项目管理工具

### 文档管理：

公司内部多人共享：采用Git，能保留版本，管理多种格式，如：Word，Excel，PPT，Viso，效果图片，原型等。

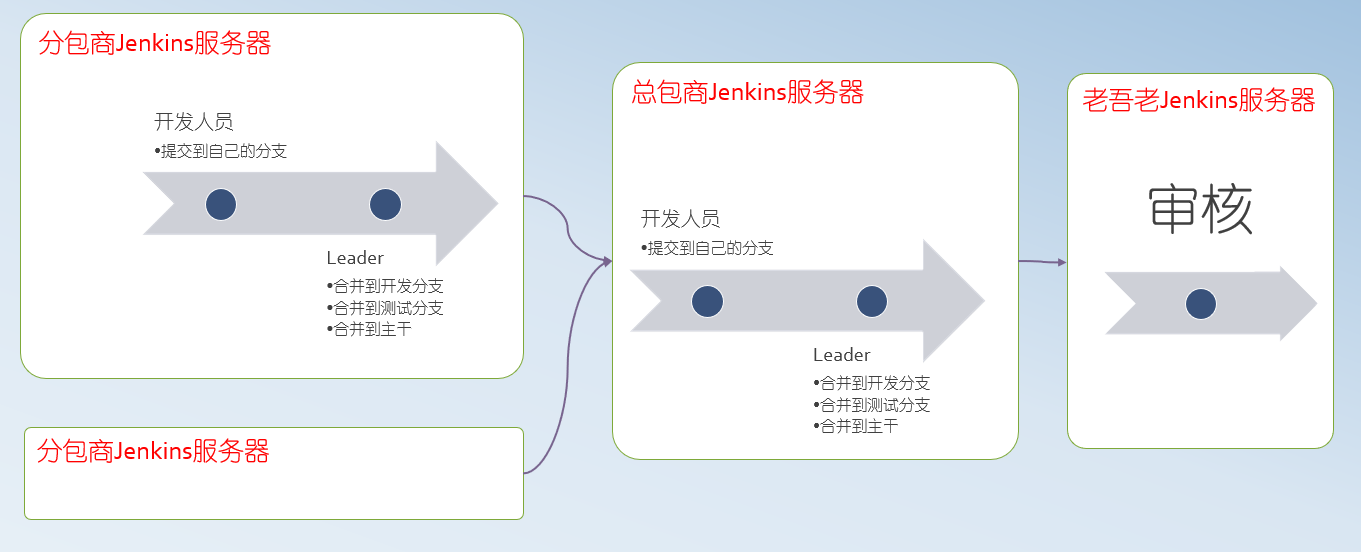
注：

对PSD等大文件，只有少部分人使用的文件，保存到另外的Git库。

如果是对外提供给第三方，或，使用 Wiki 方式。不必保留历史版本。

### 代码管理

分包，总包，分别搭建 Git 服务器，并添加下游的镜像库。 采用开源Gitea。



分支：

1. 每个开发人员创建一个自己的分支，用于日常开发。
2. dev，每个人的功能开发完成后，提交到 dev 分支。
3. test, 项目经理把 dev 提交到 test , 交付测试。
4. master, 测试经理测试完成后， 把 test 提交到 master。

### Bug管理

统一部署，采用 禅道进行管理。由产品项目决定。

## 部署集成（CI，CD）Devops流水线

### Git分支：

使用 docker Gitea 部署

### 打包部署：

采用Jenkins 部署，Git + K8s。

约定：

1. 每个K8s中的Node，可以部署多个Pod应用，但每个Pod不能是同一个应用。
2. 每个Pod挂载服务器文件目录（或分布式文件目录）。如，应用名称：app1-0.0.1.jar
   1. 服务器文件目录： /data/nfs/pv1/ , 建立软链接:/app/ , (和 docker中的文件夹一致，便于理解。)
   2. 建立应用包名的文件夹：app1-0.0.1.jar/
   3. 挂载到docker中的目录为：/app/

其中 /app/app1-0.0.1.jar是目录，下面的文件为：

1. app1-0.0.1.jar 程序包
2. lib 运行时依赖的Jar包文件夹。
3. ~~app1-0.0.1.jar.sh 运行脚本。~~
4. app1-0.0.1.yaml 是k8s创建微服务的配置文件
5. log/${pod\_name}/ 每个Pod运行时生成的日志。

约定 @/ = /app/app1-0.0.1.jar/

环境部署步骤：

1. 微服务首次打包时，分离出系统组件的Jar包，并把相应的Jar包上传到 @/lib/
2. 微服务打Jar包，上传到 @/app1-0.0.1.jar
3. 通过工具生成yaml文件，确认文件内容正确，并上传。
4. K8s主服务器上执行 kubctl apply -f @/app1-0.0.1.jar.yaml 创建应用

### 沟通工具

腾迅通RTX，如何部署？

* 微信
* QQ群
* 网盘