

# 10W QPS真刀实操\_以及基于ZK+Netty手写分布式测试工具

---

## 参考链接

系统架构知识图谱（一张价值10w的系统架构知识图谱）

<https://www.processon.com/view/link/60fb9421637689719d246739>

秒杀系统的架构

<https://www.processon.com/view/link/61148c2b1e08536191d8f92f>

## 10W QPS真刀实操的要点

---

### 这个是一个复杂的工程

---

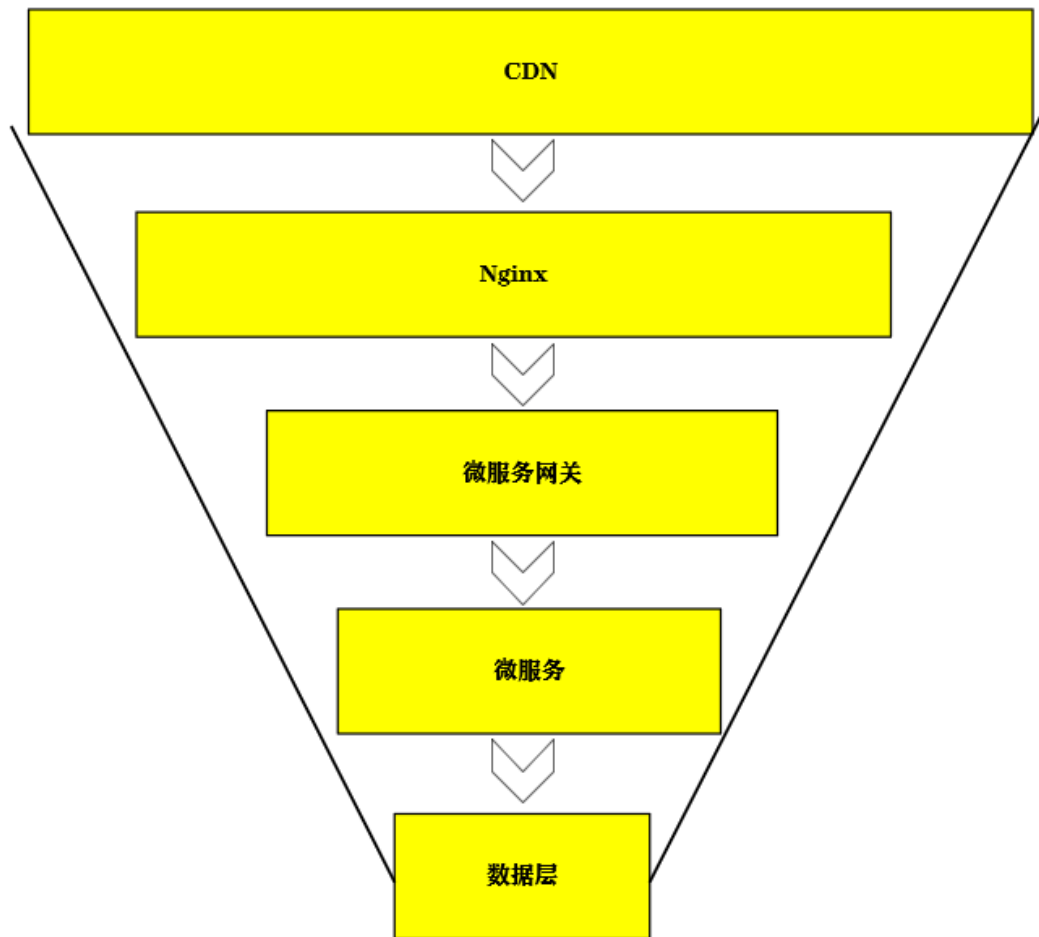
但是很有意义

### 漏斗模型中的请求分层过滤

---

漏斗型请求处理模型的核心策略，对请求进行 **分层过滤**。

而针对漏斗型请求处理模型（如秒杀场景）一种核心策略，就是对请求进行分层过滤，从而过滤掉一些无效的请求。



## 案例：秒杀系统的分层过滤

比如,在秒杀系统中, 请求分别经过 CDN、Nginx (商品详情)、微服务 (如交seckill) 和数据库 这几层, 那么:

- 1 大部分数据和流量在用户浏览器或者 CDN 上获取, 这一层可以拦截大部分静态资源的读取;
- 2、经过第二层 Nginx (商品详情) 时, 尽量得走 Nginx Cache, 过滤一些可以直接访问Nginx缓存的请求;

经过第二层 Nginx (商品详情) 时, 也可以进行流控, 还可以进行黑名单过滤, 拦截掉一些无效的流量;

- 3、再到服务层, 进入微服务网关时, 可以做用户的授权检验, 对系统做好保护和限流, 这样数据量和请求就 进一步减少;

- 4、业务层, 还可以进行数据的有效性、一致性过滤, 这里又减少了一些流量。

这样就像漏斗一样, 尽量把数据量和请求量一层一层地过滤和减少了。

分层过滤的核心思想是: 在不同的层次尽可能地过滤掉无效请求, 让“漏斗”最末端的才是有效 请求。而要达到这种效果, 我们就必须对数据做分层的校验。

## 分层过滤的基本原则是:

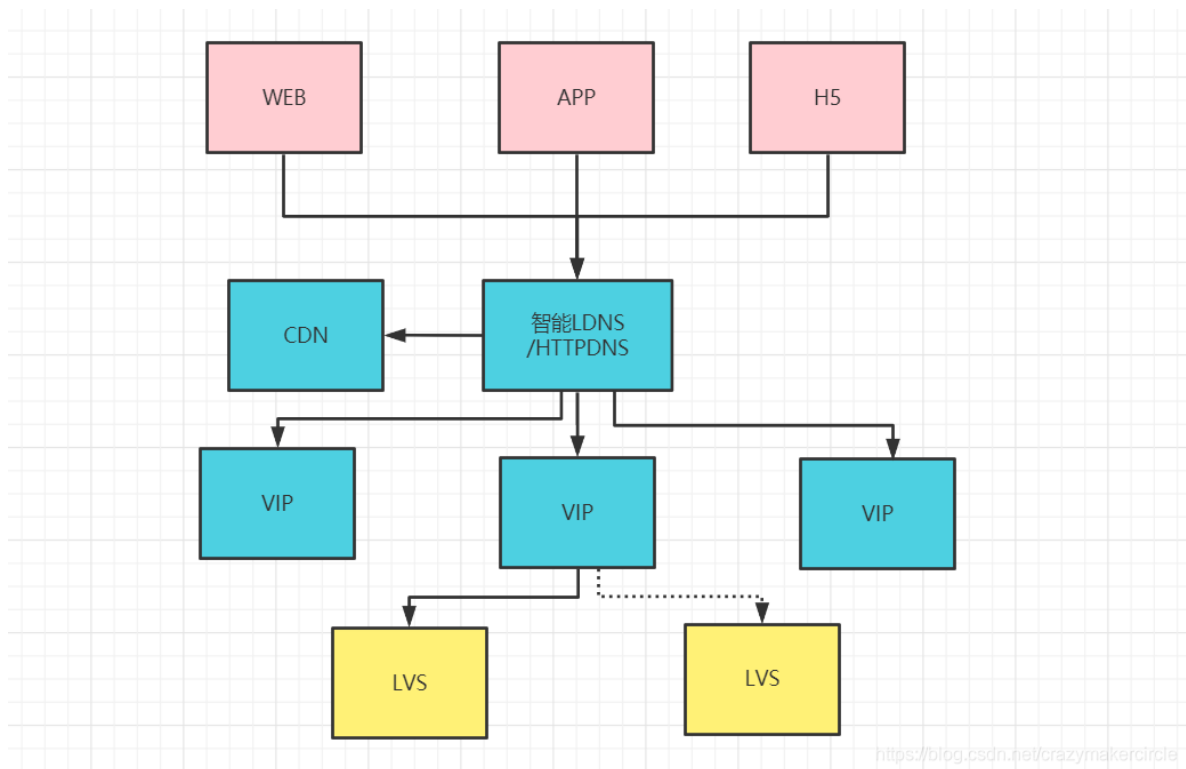
通过在不同的层次尽可能地过滤掉无效请求，尽早处理掉请求。

- 通过CDN过滤掉大量的图片，静态资源的请求。
- 读请求尽量命中缓存，不要穿透到数据库；
- 尽量将动态读数据请求，命中在三级缓存,或者二级缓存，过滤掉无效的数据读；
- 对写入操作进行削峰，争取批量写入，提高写入的吞吐量；
- 分层限流：防止系统雪崩

## 接入层

流量分发、负载均衡

按照用户规模，流量规模（吞吐量规模），接入层的架构方案不一样



slb 负载均衡， gslb

## LVS简介

LVS是Linux Virtual Server的简称，也就是Linux虚拟服务器，由章文嵩博士发起的自由软件项目，它的官方网站是[www.linuxvirtualserver.org](http://www.linuxvirtualserver.org)。通过LVS提供的负载均衡技术和Linux操作系统实现一个高性能、高可用的服务器群集，它具有良好可靠性、可扩展性和可操作性。从而以低廉的成本实现最优的服务性能。LVS 是一个实现负载均衡集群的开源软件项目，LVS 架构从逻辑上可分为调度层、Server 集群层和共享存储。

负载均衡(LB) 集群的架构和原理很简单，就是当用户的请求过来时，会直接分发到 Director Server 上，然后它把用户的请求根据设置好的调度算法，智能均衡地分发到后端真正服务器 (real server) 上。为了避免不同机器上用户请求得到的数据不一样，需要用到了共享存储，这样保证所有用户请求的数据是一样的。

## Lvs 的优点？

抗负载能力强，因为 lvs 工作方式的逻辑是非常之简单，而且工作在网络 4 层仅做请求分发之用，没有流量，所以在效率上基本不需要太过考虑。

有完整的双机热备方案，当节点出现故障时，lvs 会自动判别，所以系统整体是非常稳定的。

基本上能支持所有应用，因为 lvs 工作在 4 层，所以它可以对几乎所有应用做负载均衡，包括 http、数据库、聊天室等等。

## lvs 负载均衡机制

lvs 是四层负载均衡，也就是说建立在 OSI 模型的第四层——传输层之上

传输层上有 TCP/UDP，lvs 支持 TCP/UDP 的负载均衡

因为 LVS 是四层负载均衡，因此它相对于其它高层负载均衡的解决办法，比如 DNS 域名轮流解析、应用层负载的调度、客户端的调度等，它的效率是非常高的

lvs 的转发可以通过修改 IP 地址实现（NAT 模式）

lvs 的转发还可以通过修改直接路由实现（DR 模式）

## lvs 与 nginx 对比？

负载度 lvs 优于 nginx

稳定度 lvs 优于 nginx

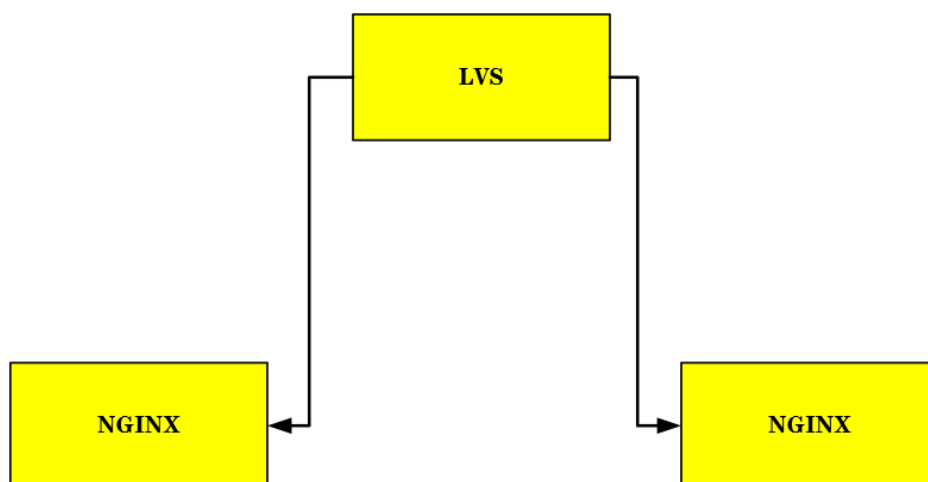
服务器性能要求 lvs 优于 nginx

网络层数的效率 lvs 优于 nginx □ 网络七层：应用层、会话层、表示层、传输层、网络层、链路层、物理层

功能多少 nginx 优于 lvs

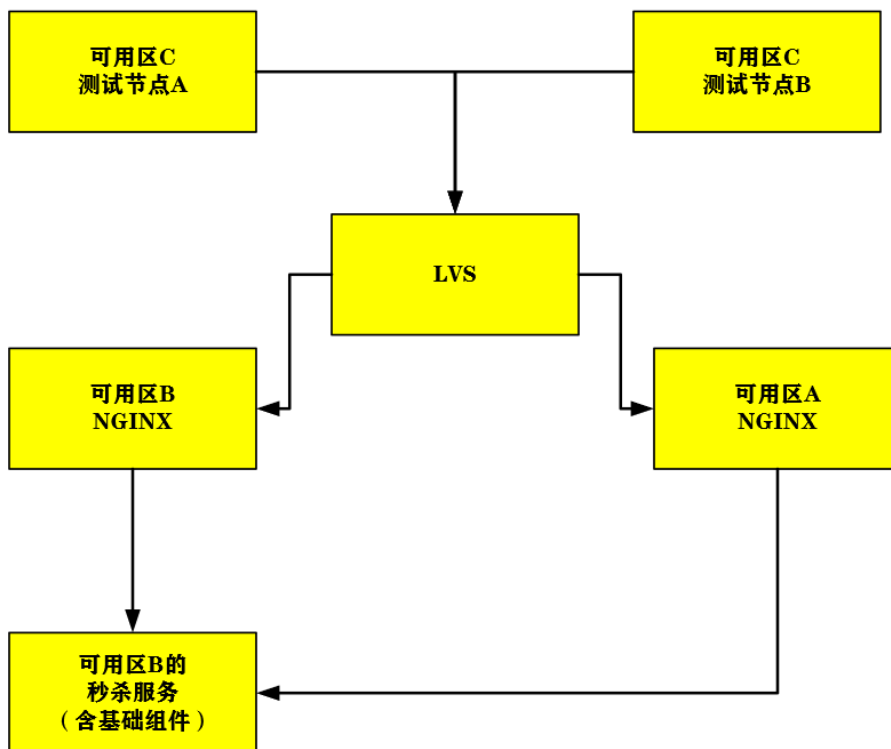
# 10W QPS实操的系统架构

## 10W QPS实操的接入层逻辑架构



# 10W QPS实操的物理架构

阿里云购买5个8核、32g的ecs服务器，来进行实操，具体的架构如下。



CSDN @架构师-尼恩

## 阿里云机器购买

### 购买虚拟机

三个可用区，四个虚拟机

a、b区各一个，作为接入主机

c区两个，做为 服务层的主机，和测试主机

### 登录&注册

登录阿里云

百度搜索阿里云，进入阿里云官网（[访问](#)）。如下图所示。



全球领先、安全、稳定的云计算产品

计算、存储、网络、安全、大数据、人工智能，普惠科技助您飞跃发展

在上图的页面的右上角，找到“登录”按钮，点击以后，会弹出如下登录界面：

密码登录 [扫码登录](#)

淘宝及1688会员可直接使用会员名登录

请在这里输入你的用户名

登录密码

**登录**

[忘记密码](#) [忘记会员名](#) [免费注册](#)

其他方式登录：

在这个登录界面中，输入用户名和密码。

查看全部产品 >

热门产品

- 弹性计算
- 存储
- 数据库
- 安全
- 大数据
- 人工智能
- 网络与CDN
- 视频服务
- 容器与中间件
- 开发与运维
- 物联网IoT
- 混合云
- 企业应用与云通信

搜索云产品

热门推荐

- 云服务器 ECS
- 域名注册
- 对象存储 OSS
- 短信服务
- 商标服务
- 云数据库 RDS MySQL 版
- 负载均衡 SLB
- CDN
- 日志服务 SLS
- 弹性公网 IP

新晋畅销

- 无影云桌面
- 智能语音交互
- 文字识别
- DataV 数据可视化
- 云速成美站
- 工商财税
- 云数据库 MongoDB 版
- 数据库备份 DBS
- 邮件推送
- 号码隐私保护

新品发布 >

- 交通云控平台商业化发布
- 智能用户增长商业化发布
- 微服务引擎 MSE 专业版发布
- 云安全访问服务
- 智能外呼机器人
- 事件总线
- Databricks 数据洞察 **NEW**
- 云数据库专属集群 MyBase
- 相册与网盘服务
- 服务网格 ASM
- Prometheus 监控服务 (公测中)
- 移动安全加固

产品动态 >

- 空中架构师云速搭 CADT 全新发布
- DataV6.0 新品发布
- 阿里云容器网络文件系统重磅发布
- 走近云原生的开源大数据统一平台
- 云防火墙新增开通美国西部硅谷地域
- 云数据库 MySQL 发布备份上云功能
- 短信服务新用户0元试用
- 日志服务发布新版智能告警功能
- 弹性公网 IP 香港精品线路发布

## 选择付费类型

阿里云 购物车 订单 备案 简体中文 crazymaker

云服务器 ECS 一键购买 自定义购买 节省计划 购买历史 产品价格 购买云盘 产品控制

1 基础配置 2 网络和安全组 3 系统配置 (选项) 4 分组设置 (选项) 5 确认订单

付费模式

包年包月 按量付费 抢占式实例

按量付费 ECS 支持停机后部分资源不收费功能，可以有效降低成本，了解相关限制和触发条件 查看详情

搭配节省计划，按量账单最高可享受 **2.4折折扣**，且计划内产品不受地域/升降配/变更规格族限制，前往介绍页了解更多

使用资源保障服务进行**容量预留**，确保您的应用拥有扩容所需容量，点击了解更多

地域及可用区

华北6 (乌兰察布) 随机分配 可用区B 可用区A 可用区C

如何选择地域

不同地域的实例之间内网互不相通；选择靠近您客户的地域，可降低网络时延，提高您客户的访问速度。

实例规格

分类选型 场景化选型

当前代 所有代 **8 vCPU 16 GiB**

筛选 选择 vCPU 选择内存 搜索规格名称，如：ecs.g5.large I/O 优化实例 是否支持IPv6

规格族	实例规格	vCPU	内存	处理器主频/睿频	内网带宽	内网收发包	存储IOPS 基准/峰值	IPv6	参考价格	处理器型号
计算型 c7	ecs.c7.2xlarge	8 vCPU	16 GiB	~3.5 GHz	最高 10 Gbps	160 万 PPS	5 万/11 万	是	¥ 1.613 /时	Intel Xeon(Cascade Lake) Platinum 8369B

CSDN @架构师-尼恩

## 选择系统类型

规格族	实例规格	vCPU	内存	处理器主频/睿频	内网带宽	内网收发包	存储IOPS 基准/峰值	参考价格	处理器型号
通用型 g6	ecs.g6.large	2 vCPU	8 GiB	2.5 GHz/3.2 GHz	最高 3 Gbps	30 万 PPS	1.05 万/-	¥ 0.5 /时	Intel Xeon(Cascade Lake) Platinum 8269CY
通用型 g6	ecs.g6.xlarge	4 vCPU	16 GiB	2.5 GHz/3.2 GHz	最高 5 Gbps	50 万 PPS	2.1 万/-	¥ 1.0 /时	Intel Xeon(Cascade Lake) Platinum 8269CY

当前选择实例: ecs.g6.xlarge (4 vCPU 16 GiB, 通用型 g6)

购买实例数量:  台 当前所选实例规格在当前可用区已开通 0 台，最多还可开通 64 台。如需更多配额，您可前往控制台提升

建议使用 弹性伸缩 Auto Scaling 产品 来创建/伸缩 ECS 实例。创建伸缩组时扩容策略可选择“成本优化策略”，更加经济高效，去创建

最小购买数量:  台 当 ECS 的 库存数量 < 最小购买数量，会创建失败；当 购买数量 >= 库存数量 >= 最小购买数量，会按照 库存数量 来创建。

镜像

公共镜像 自定义镜像 共享镜像 镜像市场

CentOS 7.2 64位 安全加固

存储

系统盘

云盘参数和性能: ESSD云盘 40 GiB 2280 IOPS 性能级别: PLO (单盘IOPS性能上限1万) 随实例释放

不同云盘性能指标不同，查看 云盘性能指标

# 选择网络类型

云服务器 ECS 一键购买 自定义购买

节省计划 购买历史 产品价格

基础配置 网络和安全组 系统配置 (选项) 分组设置 (选项)

网络 如何选择网络

专有网络  默认交换机

默认专有网络 默认交换机

如需创建新的专有网络, 您可 [前往控制台创建](#)。 交换机所在可用区: 华北 1 可用区 C 交换机网段: - 您可 [前往控制台创建](#)。

公网 IP 分配公网 IPv4 地址

系统会分配公网 IP, 也可采用更加灵活的弹性公网 IP 方案, 了解 [如何配置并绑定弹性公网 IP 地址](#)。

公网带宽计费 按使用流量 按固定带宽

后付费模式, 按使用流量 (单位为GB) 计费, 每小时扣费, 请保证余额充足。

带宽峰值 1M 25M 50M 75M 100M 5 Mbps

阿里云免费提供最高 5Gbps 的恶意流量攻击防护。 [了解更多 | 提升防护能力](#)

# 选择输入登录密码和节点名称

登录凭证  密钥对  自定义密码  创建后设置

密钥对安全强度远高于常规自定义密码, 可以避免暴力破解威胁, 建议您使用密钥对创建实例。

登录名 root

登录密码 .....  
8~30 个字符, 必须同时包含三项 (大写字母、小写字母、数字、[!@#%&\*^\_+=|~`<>.,/?] 中的特殊符号), 其中 Windows 实例不能以斜线号 (/) 开头

确认密码 .....  
请牢记您所设置的密码, 如遗忘可登录ECS控制台重置密码。

实例名称 seckill-ginx-20210904 如何自定义有序实例名称

2~128 个字符, 以大小写字母或中文开头, 可包含数字、点号 (.)、下划线 (\_)、半角冒号 (:) 或连字符 (-)

描述 输入描述  
长度为 2~256 个字符, 不能以 http:// 或 https:// 开头

主机名 seckill-ginx-20210904 如何自定义有序主机名

Linux 等其他操作系统: 长度为 2~64 个字符, 允许使用点号 (.) 分隔字符成多段, 每段允许使用大小写字母、数字或连字符 (-), 但不能连续使用点号 (.) 或连字符 (-), 不能以点号 (.) 或连字符 (-) 开头或结尾。

有序后缀  为实例名称和主机名添加有序后缀

实例释放保护  防止通过控制台或 API 误删除释放

公网带宽: 1Mbps 按使用流量 配置费用: ¥ 2.154 /时 公网流量费用: ¥ 0.800 /GB

上一步: 网络和安全组 下一步: 分组设置 确认

# ecs详情

云服务器 ECS 一键购买 自定义购买

节省计划 购买历史 产品价格 购买云盘 产品控制台

基础配置 网络和安全性 系统配置 (选项) 分组设置 (选项) 5 确认订单

所选配置

基础配置 [编辑](#) 付费模式: 按量付费 地域及可用区: 乌兰察布 可用区C 实例规格: 通用型 g7 / ecs.g7.xlarge (8vCPU 32GiB)  
 购买数量: 1 台 镜像: CentOS 7.4 64位 (安全加固) 系统盘: ESSD云盘 40GiB, 随实例释放, PL1 (单盘OPS性能上限5万)

网络和安全性 [编辑](#) 网络: 专有网络 VPC: 默认专有网络 交换机: 默认交换机  
 公网带宽: 按使用流量 1Mbps 安全组: 1) 默认安全组 (自定义端口)

系统配置 [编辑](#) 登录凭证: 自定义密码 实例名称: seckill-ginx-20210904 主机名: seckill-ginx-20210904  
 实例元数据访问模式: 普通模式 (兼容加固模式)

[保存为启动模板](#) [生成Open API最佳实践脚本](#) [保存当前购买配置为ROS模板](#)

使用期限  设置自动释放服务时间 ECS实例将在您预约的时间点进行释放, 实例释放后数据及IP地址不会被保留且无法找回, 请谨慎操作。

服务协议  《云服务器 ECS 服务条款》

公网带宽: 1Mbps 按使用流量

配置费用: ¥ 2.154 /时 公网流量费用: ¥ 0.800 /GB

[上一步: 分组设置](#) [立即购买](#)

## 控制台详情

阿里云 工作台 账号全部资源 华北6 (乌兰察布)

云服务器 ECS 实例

实例使用须知

创建实例 选择实例属性项搜索, 或者输入关键字检索

高级搜索 创建诊断 刷新 删除

搜索项: 实例ID: i-0jdyt0g07q1fzrq1aj 清除

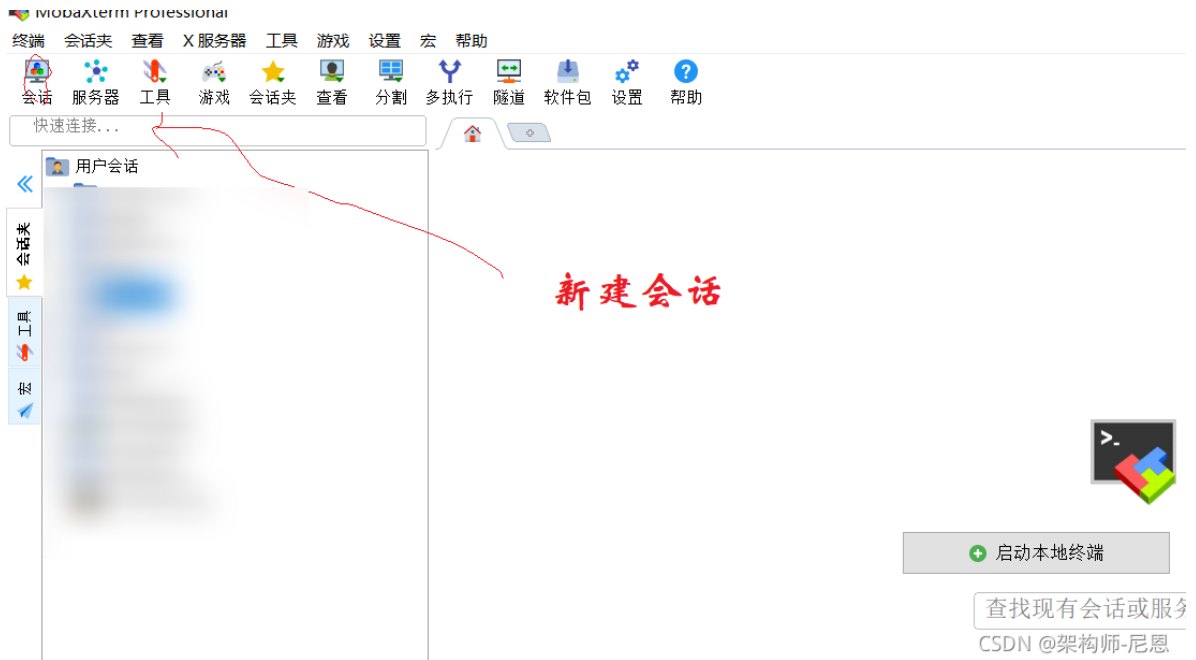
实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-0jdyt0g07q1fzrq1aj seckill-ginx-20210904			乌兰察布 可用区C		待启动	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.xlarge 1Mbps (峰值)	按量 2021年9月4日 10:32 创建	管理 更改实例规格   更多

共有1条, 每页显示: 20 条 < 1 >

# 云主机的登录与配置

## 云主机的登录

### 新建ssh会话



# MobaXterm

## 输入ecs的IP



从阿里云控制台去复制一下ip, 注意, 是公网ip

## 实例

创建实例 选择实例属性项搜索, 或者输入关键字识别搜索 高级搜索 创建诊断

检索项: 实例ID: i-0jldyt0g07gj1fzrq1aj 清除

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-0jldyt0g07gj1fzrq1aj seckill-ginx-20210904			乌兰察布 可用区C	8.130.31.52 (公) 172.26.9.105 (私有)	运行中	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 10:32 创建	管理 更改

启动 停止 重启 重置实例密码 续费 按量付费转包年包月 释放设置 更多

共有1条, 每页显示: 20 条 << >>

CSDN @架构师-尼恩

## 输入密码后, 登录成功

输入购买的时候设置的密码

```
4. 8.130.31.52 (root) x
```

```
? MobaXterm 20.0 ?  
(SSH client, X-server and networking tools)  
  
> SSH session to root@8.130.31.52  
? SSH compression : x  
? SSH-browser      : ✓  
? X11-forwarding  : x (disabled or not supported by server)  
? DISPLAY         : 192.168.0.9:0.0  
  
> For more info, ctrl+click on help or visit our website  
  
Last login: Sat Sep  4 10:42:04 2021 from 103.202.198.105  
Welcome to Alibaba Cloud Elastic Compute Service !  
[root@seckill-ginx-20210904 ~]#
```

CSDN @架构师-尼恩

## 查看核心数量和内存大小

查看CPU

```
grep 'core id' /proc/cpuinfo | sort -u | wc -l
```

查看内存

```
[root@seckill-ginx-20210904 ~]# grep 'core id' /proc/cpuinfo | sort -u | wc -l
4
[root@seckill-ginx-20210904 ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:           30G          375M          30G          312K          352M          30G
Swap:           0B           0B           0B
```

## 文件句柄数的配置

---

### 查看最大句柄数限制

```
#查看单个进程最大句柄数

ulimit -n

cat /etc/security/limits.conf

#查看系统打开句柄最大数量

cat /proc/sys/fs/file-max
```

结果:

```
[root@seckill-ginx-20210904 ~]# cat /etc/security/limits.conf
# /etc/security/limits.conf
#
#This file sets the resource limits for the users logged in via PAM.
#It does not affect resource limits of the system services.
#
#Also note that configuration files in /etc/security/limits.d directory,
#which are read in alphabetical order, override the settings in this
#file in case the domain is the same or more specific.
#That means for example that setting a limit for wildcard domain here
```

```

#can be overridden with a wildcard setting in a config file in the
#subdirectory, but a user specific setting here can be overridden only
#with a user specific setting in the subdirectory.
#
#Each line describes a limit for a user in the form:
#
#<domain>          <type> <item> <value>
#
#Where:
#<domain> can be:
#     - a user name
#     - a group name, with @group syntax
#     - the wildcard *, for default entry
#     - the wildcard %, can be also used with %group syntax,
#         for maxlogin limit
#
#<type> can have the two values:
#     - "soft" for enforcing the soft limits
#     - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#     - core - limits the core file size (KB)
#     - data - max data size (KB)
#     - fsize - maximum filesize (KB)
#     - memlock - max locked-in-memory address space (KB)
#     - nofile - max number of open file descriptors
#     - rss - max resident set size (KB)
#     - stack - max stack size (KB)
#     - cpu - max CPU time (MIN)
#     - nproc - max number of processes
#     - as - address space limit (KB)
#     - maxlogins - max number of logins for this user
#     - maxsyslogins - max number of logins on the system
#     - priority - the priority to run user process with
#     - locks - max number of file locks the user can hold
#     - sigpending - max number of pending signals
#     - msgqueue - max memory used by POSIX message queues (bytes)
#     - nice - max nice priority allowed to raise to values: [-20, 19]
#     - rtprio - max realtime priority
#
#<domain>          <type> <item>          <value>
#
#*                soft   core           0
#*                hard   rss           10000
#@student         hard   nproc        20
#@faculty        soft   nproc        20
#@faculty        hard   nproc        50
#ftp             hard   nproc        0
#@student        -       maxlogins    4

# End of file
root soft nofile 65535
root hard nofile 65535
* soft nofile 65535
* hard nofile 65535

```

```
[root@seckill-ginx-20210904 ~]# cat /proc/sys/fs/file-max
```

```
3195223
```

## 修改最大句柄数限制

```
cat /proc/sys/fs/file-max
```

```
#追加
cat >> /etc/security/limits.conf <<EOF
* soft nproc 65535
* hard nproc 65535
* hard nofile 1200000
* soft nofile 1200000
EOF

#编辑
vi /etc/security/limits.conf

#追加
cat >> /etc/sysctl.conf <<EOF
fs.nr_open=2000000
fs.file-max=10000000
EOF

#编辑
vi /etc/sysctl.conf

#生效
sysctl -p

cat /etc/security/limits.conf

cat /proc/sys/fs/file-max
```

## 为啥要调整最大文件句柄

我们常常会遇到类似“Socket/File: Can't open so many files”，“无法打开更多进程”，或是coredump过大等问题，这些都可以设置资源限制来解决。

## 局部文件句柄数和全局文件句柄数的详解

### 增加局部文件句柄数

- 临时修改当前局部文件句柄数：

```
ulimit -n 10000
```

这是临时设置，系统重启后设置会丢失

- 永久修改当前局部文件句柄数：修改文件/etc/security/limits.conf，

```
#编辑
vim /etc/security/limits.conf
#查看
cat /etc/security/limits.conf
```

- 文件末尾加上：

```
* hard nfile 1200000
* soft nfile 1200000
```

- 修改文件后需要重启机器，才能生效

## 修改后的效果

```
##*                soft    core    0
##*                hard    rss     10000
#@student          hard    nproc   20
#@faculty          soft    nproc   20
#@faculty          hard    nproc   50
#ftp               hard    nproc   0
#@student          -      maxlogins 4
* soft nproc 65535
* hard nproc 65535
* soft nfile 6553500
* hard nfile 6553500
# End of file
```

soft nproc：单个用户可用的最大进程数量(超过会警告);  
hard nproc：单个用户可用的最大进程数量(超过会报错);  
soft nfile：可打开的文件描述符的最大数(超过会警告);  
hard nfile：可打开的文件描述符的最大数(超过会报错);

## 局部文件句柄数上限配置

可通过修改/etc/sysctl.conf 修改 fs.nr\_open 值，sysctl -p生效

```
//查看  
cat /etc/sysctl.conf  
  
//修改  
vim /etc/sysctl.conf  
  
fs.nr_open=150000
```

## 生效的shell指令

```
sysctl -p
```

新的问题：局部的不能大过全局的限制

还有一个Linux kernel能分配的全局文件句柄数限制

---

## 全局句柄数限制的配置

全局句柄数限制，字面上看file-max，file-max 确实像是对应最大文件数，而在Linux内核文档中它们两的解释是：

file-max：

```
The value in file-max denotes the maximum number of file-  
handles that the Linux kernel will allocate. When you get lots  
of error messages about running out of file handles, you might  
want to increase this limit
```

---

## 查看全局文件句柄的限制

局部的不能大过全局的限制，查看我所有进程能够打开的最大文件数是多少

```
[root@cdh1 ~]# cat /proc/sys/fs/file-max  
185213  
  
[root@cdh1 ~]#
```

---

## 增加全局文件句柄数

- 查看: `cat /etc/sysctl.conf`
- 配置文件 `/etc/sysctl.conf`  
`vim /etc/sysctl.conf`
- 在文件末尾加上: `fs.file-max=1000000`

```
fs.file-max = 1500000
```

- 使配置文件生效: `sysctl -p`

## 总体的效果

```
cat /etc/sysctl.conf
fs.nr_open=2000000
fs.file-max=10000000
```

file-max是所有进程最大的文件数

nr\_open是单个进程可分配的最大文件数

### 特别注意:

局部文件句柄数一定不能超过全局文件句柄数!!!

把局部文件句柄数设置超过了全局文件句柄数, 可能导致无法开机后无法登陆的现象!!!

---

# 借助shell脚本\_极速安装基础组件

---

## mysql的安装

### 连接一个虚拟机

在可用区c上找一个实例, 改名为, mysql-to-eureka, 并且启动

## 实例

创建实例 选择实例属性搜索, 或者输入关键字识别搜索 高级搜索 创建诊断

搜索项: 可用区ID: cn-wulanchabu-c 清除

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-0jldyt0g07g2fibr0wu mysql-to-eureka			乌兰察布 可用区C	172.26.9.107 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 11:47 创建	管理 更改实例规格   更多
i-0jldyt0g07g2fibr0wx launch-advisor-20210904			乌兰察布 可用区C	172.26.9.109 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 11:47 创建	管理 更改实例规格   更多
i-0jldyt0g07g2fibr0wt launch-advisor-20210904			乌兰察布 可用区C	172.26.9.108 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 11:47 创建	管理 更改实例规格   更多
i-0jldyt0g07g2fibr0vw launch-advisor-20210904			乌兰察布 可用区C	172.26.9.106 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 11:47 创建	管理 更改实例规格   更多
i-0jldyt0g07g2fibr0ww launch-advisor-20210904			乌兰察布 可用区C	172.26.9.110 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 11:47 创建	管理 更改实例规格   更多
i-0jldyt0g07g1fzrq1aj seckill-ginx-20210904			乌兰察布 可用区C	172.26.9.105 (私有)	已停止	专有网络	8 vCPU 32 GiB (I/O优化) ecs.g7.2xlarge 1Mbps (峰值)	按量 2021年9月4日 10:32 创建	管理 更改实例规格   更多

共有6条, 每页显示: 20 条

## 连接服务器

### 会话设置



#### 基本 SSH 设置

远程主机\* 8.130.24.72  指定用户名 root 端口 22

#### 高级 SSH 设置

会话名称: mysql-to-eureka  锁定终端标题 会话图标

启动会话从 普通标签  在会话结束时显示重新连接消息

自定义标签颜色 注释:

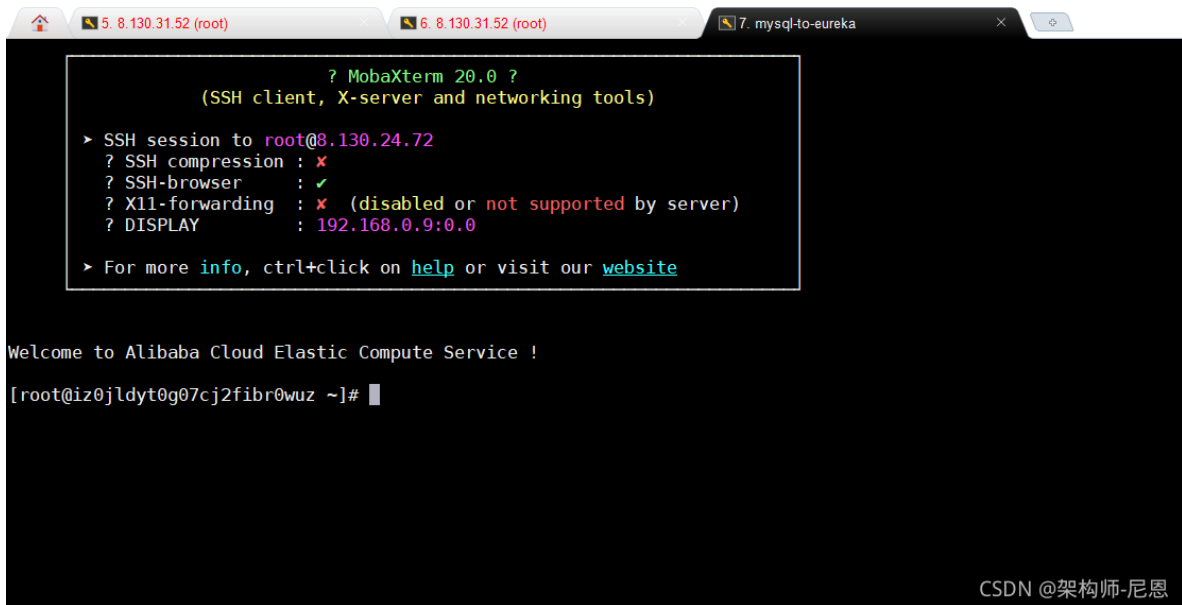
创建此会话的桌面快捷方式

好的

取消

CSDN @架构师-尼恩

## 使用密码登录



## 一键安装Mysql

### 参考链接的入口地址

<https://www.cnblogs.com/crazymakercircle/p/9904544.html>



### 创建数据库

```
mysql -uroot -p'wTJJJJZZZ@SSDDC'
```

```
CREATE DATABASE store DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;

root@localhost [(none)]>show databases;
+-----+
| Database          |
+-----+
| information_schema |
| mysql             |
| performance_schema |
| store             |
| sys               |
+-----+

GRANT ALL PRIVILEGES ON store.* TO "root" @'%' IDENTIFIED BY 'WTJJJZZZ@SSDDC';

flush privileges;
```

## 一键安装jdk

可以从百度网盘下载jdk1.8U212

百度网盘

暑期大促, SVIP拼团买更省钱! wqlinf11 企业认证 会员中心

我的文件

上传 新建文件夹 新建在线文档 高线下载

我的网盘 > ... 书和视频配套的中间件与工具、开发环境 > 搜索我的网盘文件

文件名	修改时间	类型	大小
mysql-5.7.28-linux-glibc2.12-x86_64.tar.gz	2021-09-04 12:15	gz文件	691.10MB
classLoaderDemo.zip	2021-09-04 09:12	zip文件	370KB
jdk-8u121-linux-x64.tar.gz			
MobaXterm_20.0汉化.rar	2021-09-04 09:12	rar文件	27.07MB
lombok-plugin-0.33-2019.2.zip	2021-09-04 09:12	zip文件	553KB
nacos-server-1.1.3.zip	2021-09-04 09:12	zip文件	43.69MB
openresty-1.15.8.2-win64.zip	2021-09-04 09:12	zip文件	11.67MB
openresty-1.13.6.2-win64.zip	2021-09-04 09:12	zip文件	11.27MB
redis-desktop-manager-0.8.8.384.exe	2021-09-04 09:12	exe文件	27.18MB
JDK8Source.zip	2021-09-04 09:12	zip文件	28.31MB
fiddlersetup.exe	2021-09-04 09:12	exe文件	4.06MB

4.06MB @架构师-尼恩

上传到 /work

然后执行下面的脚本

```
mkdir -p /usr/local/java
tar -zxvf /work/jdk-8u121-linux-x64.tar.gz -C /usr/local/java

cat >> /etc/profile <<EOF
export JAVA_HOME=/usr/local/java/jdk1.8.0_121
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH
EOF

source /etc/profile
```

```
[root@iz0jldyt0g07cj2fibr0wuz jdk1.8.0_121]# java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

## 一键安装zk的安装

伪装的集群模式

## 参考链接的入口地址

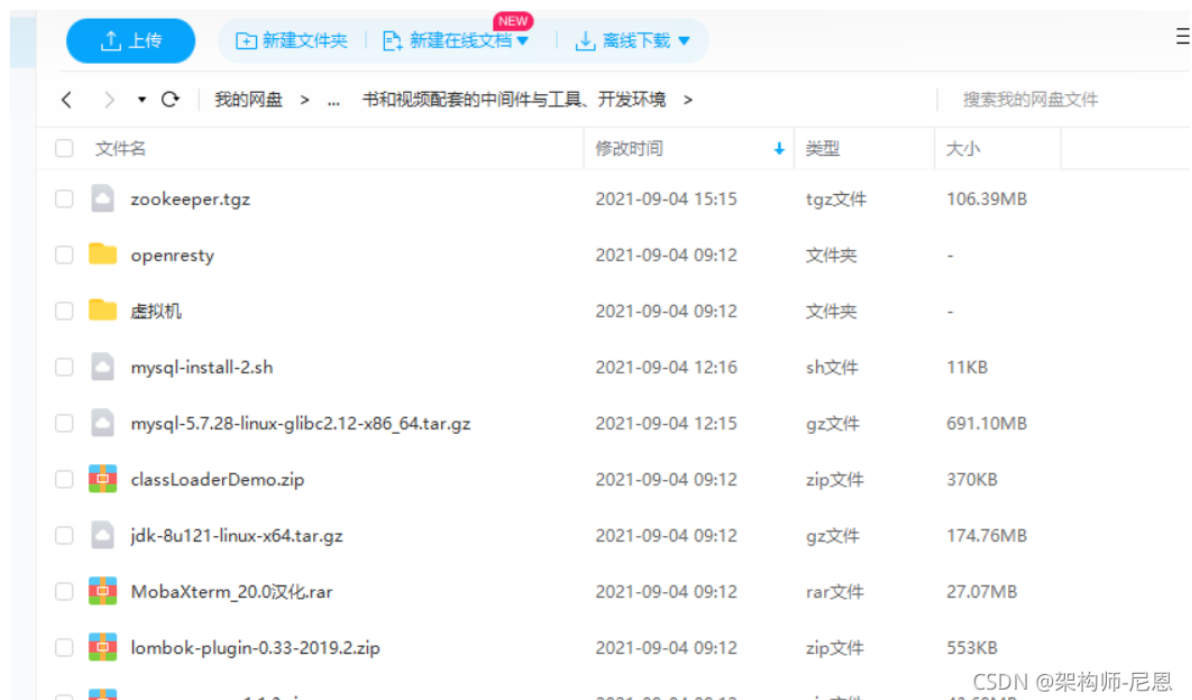
<https://www.cnblogs.com/crazymakercircle/p/9904544.html>

The screenshot shows a web browser window with the address bar displaying `cnblogs.com/crazymakercircle/p/9904544.html`. The page content includes a header with the title "地表最强 PPT 小工具：爆炸天，像与代码一样与PPT" and a sub-header "无编程不创客，无编程不创客，一大波编程高手正在疯狂创客圈交流、学习中! 找组织，GO". Below this is a table with two columns: "Java 分布式 高并发 开发环境搭建" and "链接地址". The table lists various installation guides for different environments and technologies.

Java 分布式 高并发 开发环境搭建	链接地址
windows centos 虚拟机 安装&排坑	vagrant+java+springcloud+redis+zookeeper镜像下载(&制作详解)
centos mysql 安装&排坑	centos mysql 笔记 (内含vagrant mysql 镜像)
linux kafka安装&排坑	kafka springboot (或 springcloud ) 整合
Linux openresty 安装	Linux openresty 安装
【必须】Linux Redis 安装 (带视频)	Linux Redis 安装 (带视频)
	redis集群架构知识 (史上最强，面试必看)
【必须】Linux Zookeeper 安装 (带视频)	Linux Zookeeper 安装,带视频
	ZooKeeper 客户端: GUI+命令行 (史上最全)
Windows Redis 安装 (带视频)	Windows Redis 安装 (带视频)
RabbitMQ 离线安装 (带视频)	RabbitMQ 离线安装 (带视频)

CSDN @架构师-尼恩

## 测试环境的一键安装包



CSDN @架构师-尼恩

## 上传一键安装包，开始安装

```
[root@ work]# cd /work  
  
[root@ work]# tar -zxvf zookeeper.tgz
```

## 开机启动脚本：

```
[root@ work]# /usr/bin/su - root -c  
"/work/zookeeper/zookeeper_01/bin/zkServer.sh start"  
ZooKeeper JMX enabled by default  
Using config: /work/zookeeper/zookeeper_01/bin/./conf/zoo.cfg  
Starting zookeeper ... STARTED  
[root@ work]# /usr/bin/su - root -c  
"/work/zookeeper/zookeeper_02/bin/zkServer.sh start"  
ZooKeeper JMX enabled by default  
Using config: /work/zookeeper/zookeeper_02/bin/./conf/zoo.cfg  
Starting zookeeper ... STARTED  
  
[root@ work]# jps
```

```
6672 QuorumPeerMain
26672 QuorumPeerMain
```

查看状态

```
[root@ work]# /usr/bin/su - root -c
"/work/zookeeper/zookeeper_01/bin/zkServer.sh status"
Zookeeper JMX enabled by default
Using config: /work/zookeeper/zookeeper_01/bin/../conf/zoo.cfg
Mode: follower
```

```
[root@ work]# /usr/bin/su - root -c
"/work/zookeeper/zookeeper_02/bin/zkServer.sh status"
Zookeeper JMX enabled by default
Using config: /work/zookeeper/zookeeper_02/bin/../conf/zoo.cfg
Mode: leader
```

## redis安装

### 参考链接的入口地址

<https://www.cnblogs.com/crazymakercircle/p/9904544.html>

The screenshot shows a web browser window with the address bar containing the URL <https://www.cnblogs.com/crazymakercircle/p/9904544.html>. The page content includes a navigation menu on the left and a main content area with a table of links. The table has two columns: 'Java 分布式 高并发 开发环境搭建' and '链接地址'. The table lists various installation guides for different operating systems and technologies, including Windows, CentOS, Linux, and RabbitMQ. The links are categorized as '必须' (must) or '可选' (optional). The page also features a search bar and a sidebar with a hamburger menu icon.

Java 分布式 高并发 开发环境搭建	链接地址
windows centos 虚拟机 安装&排坑	<a href="#">vagrant+java+springcloud+redis+zookeeper镜像下载(&amp;制作详解)</a>
centos mysql 安装&排坑	<a href="#">centos mysql 笔记 (内含vagrant mysql 镜像)</a>
linux kafka安装&排坑	<a href="#">kafka springboot (或 springcloud ) 整合</a>
Linux openresty 安装	<a href="#">Linux openresty 安装</a>
【必须】Linux Redis 安装 (带视频)	<a href="#">Linux Redis 安装 (带视频)</a>
	<a href="#">redis集群架构知识 (史上最强, 面试必看)</a>
【必须】Linux Zookeeper 安装 (带视频)	<a href="#">Linux Zookeeper 安装, 带视频</a>
	<a href="#">ZooKeeper 客户端: GUI+命令行 (史上最全)</a>
Windows Redis 安装 (带视频)	<a href="#">Windows Redis 安装 (带视频)</a>
RabbitMQ 离线安装 (带视频)	<a href="#">RabbitMQ 离线安装 (带视频)</a>

CSDN @架构师-尼恩

### 启动

```
/usr/local/redis/bin/redis-server /usr/local/redis/redis.conf
```

## 测试是否可以访问

```
/usr/local/redis/bin/redis-cli -h 127.0.0.1 -p 6379 -a "123456"
```

```
/usr/local/redis/bin/redis-cli -h cdh1 -p 6379 -a "123456"
```

```
[root@iz0j1dyt0g07cj2fibr0wuz bin]# /usr/local/redis/bin/redis-cli -h cdh1 -p 6379 -a "123456"  
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.  
cdh1:6379> quit
```

## rocketmq的安装

---

### 参考链接的入口地址

<https://www.cnblogs.com/crazymakercircle/p/9904544.html>

Linux openresty 安装	Linux openresty 安装
【必须】Linux Redis 安装 (带视频)	Linux Redis 安装 (带视频)
	redis集群架构知识 (史上最强, 面试必看)
【必须】Linux Zookeeper 安装 (带视频)	Linux Zookeeper 安装, 带视频
	ZooKeeper 客户端: GUI+命令行 (史上最全)
Windows Redis 安装 (带视频)	Windows Redis 安装 (带视频)
RabbitMQ 离线安装 (带视频)	RabbitMQ 离线安装 (带视频)
RocketMQ 原理 - 部署 - 入门	RocketMQ 原理 - 部署 - 入门 (图解)
ElasticSearch 安装, 带视频	ElasticSearch 安装, 带视频
Nacos 安装 (带视频)	Nacos 安装 (带视频)
【必须】Eureka	Eureka 入门, 带视频
【必须】springcloud Config 入门, 带视频	springcloud Config 入门, 带视频
【必须】SpringCloud 脚手架打包与启动	SpringCloud脚手架打包与启动
Linux 自启动 假死自启动 定时自启	Linux 自启动 假死启动

CSDN @架构师-尼恩

## 启动:

```
nohup mqnamesrv -n 172.26.9.107 &
```

```
nohup sh mqbroker -n 172.26.9.107:9876 autoCreateTopicEnable=true -c /usr/local/rocketmq/rocketmq-all-4.4.0-bin-release/conf/broker.conf &
```

```
mqbroker -m
```

## 部署服务层

### 配置环境变量

```
cat /etc/profile
```

```
export SCAFFOLD_DB_HOST=cdh1
export SCAFFOLD_DB_USER=root
export SCAFFOLD_DB_PSW=wTJJJZZZ@SSDDC
export SCAFFOLD_REDIS_HOST=cdh1
export SCAFFOLD_REDIS_PSW=123456
export SCAFFOLD_EUREKA_ZONE_HOSTS=http://cdh1:7777/eureka/
```

```
export SCAFFOLD_ZOOKEEPER_HOSTS=cdh1:2181
export LC_ALL=en_US.UTF-8
export ROCKETMQ_HOME=/usr/local/rocketmq/rocketmq-all-4.4.0-bin-release
export JAVA_HOME=/usr/local/java/jdk1.8.0_121
export
CLASSPATH=./usr/local/java/jdk1.8.0_121/lib/dt.jar:/usr/local/java/jdk1.8.0_121
/lib/tools.jar
export PATH=/usr/local/rocketmq/rocketmq-all-4.4.0-bin-
release/bin:/usr/local/java/jdk1.8.0_121/bin:/usr/bin:/usr/local/mysql/bin:/root
/bin

export SCAFFOLD_PREFERRED_NETWORKS=172.26.9.
export SCAFFOLD_DB_PWD=WTJJJJZZZ@SSDDC
export SCAFFOLD_ROCKETMQ_HOSTS=172.26.9.107:9876
```

cat /etc/hosts

```
[root@iz0j1dyt0g07cj2fibr0wuz tester-node-1.0-SNAPSHOT]# cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
172.26.9.107 cdh1
192.168.56.122 cdh2
192.168.56.123 cdh3
```

## 注册中心与配置中心

### eureka

### springcloud config

```
[root@iz0j1dyt0g07cj2fibr0wuz ~]# sh /work/cloud-eureka-1.0-
SNAPSHOT/bin/deploy.sh start
PORT:7777
JVM:-server -Xms64m -Xmx256m
nohup: appending output to 'nohup.out'
cloud-eureka-1.0-SNAPSHOT.jar is running,pid is 3292
[root@iz0j1dyt0g07cj2fibr0wuz ~]# sh /work/cloud-config-1.0-
SNAPSHOT/bin/deploy.sh start
PORT:7788
JVM:-server -Xms64m -Xmx256m
cloud-config-1.0-SNAPSHOT.jar is running,pid is 3536
```

## 网关zuul

```
sh /work/cloud-zuul-1.0-SNAPSHOT/bin/deploy.sh start
```

## 库存服务与秒杀服务

stockprovider

seckillprovider

```
sh /work/stock-provider-1.0-SNAPSHOT/bin/deploy.sh start
```

```
sh /work/seckill-provider-1.0-SNAPSHOT/bin/deploy.sh start
```

```
tail -f /work/logs/stock-provider-1.0-SNAPSHOT/output.log
```

## 开通网关的端口，可以访问swagger页面

### 秒杀库存操作

<http://8.130.31.52:8080/stock-provider/swagger-ui.html#>

← → ↻ (⚠ 不安全) 8.130.31.52:8080/stock-provider/swagger-ui.html

swagger Select a spec

### 疯狂创客圈 springcloud + Nginx 高并发核心编程 <sup>1.0</sup>

[ Base URL: zuul/stock-provider ]  
<http://8.130.31.52:8080/stock-provider/v2/api-docs>

Zuul+Swagger2 构建 RESTful APIs

[Terms of service](#)  
[疯狂创客圈 - Website](#)

---

**Rockmq消息Demo** Rockmq Message Controller

---

**商品库存** Seckill Sku Stock Controller

---

Models

CSDN @架构师-尼恩

# 部署接入层

另外找一台机器，

## Nginx的安装

### 参考链接的入口地址

<https://www.cnblogs.com/crazymakercircle/p/9904544.html>



修改主机名称

```
cat >> /etc/hosts <<EOF
172.26.9.107 cdh1
172.26.9.105 cdh2
192.168.56.123 cdh3
EOF
```

### 上传前端工程:

<https://gitee.com/crazymaker/LuaDemoProject.git>

```
mkdir -p /vagrant/LuaDemoProject/src/proxy_temp/
```

查看ng的work进程的执行用户组

```
[root@seckill-ginx-20210904 ~]# ps aux | grep "nginx: worker process" | awk
'{print $1}'
nobody
nobody
nobody
nobody
nobody
nobody
nobody
nobody
nobody
root
```

```
chown -R root:root /vagrant/LuaDemoProject
```

```
chown -R nobody:nobody /vagrant/LuaDemoProject
```

```
sh /vagrant/LuaDemoProject/sh/linux/openresty-restart.sh
```

```
curl http://localhost:8080/echo
```

## 特意说说配置文件

---

### 接入层秒杀

<http://8.130.31.52:8080/views/seckill.html>

请填写秒杀信息

用户ID 1

用户信息 点击右边设置用户,获取用户

秒杀ID 1157197244718385152

商品名称

库存数量 -

原始库存

CSDN @架构师-尼恩

## 获取商品详情

输入秒杀商品id

1238353441793769472

8.130.31.52:8080/views/seckill.html

用户信息 点击右边设置用户,获取用户

秒杀ID 1238353441793769472

商品名称 秒杀商品-1

库存数量 10000

原始库存 10000

暴露地址 9b6a0e51e295ebb19bcd37d415bb9b0

秒杀令牌 请获取秒杀令牌

2 获取商品详情

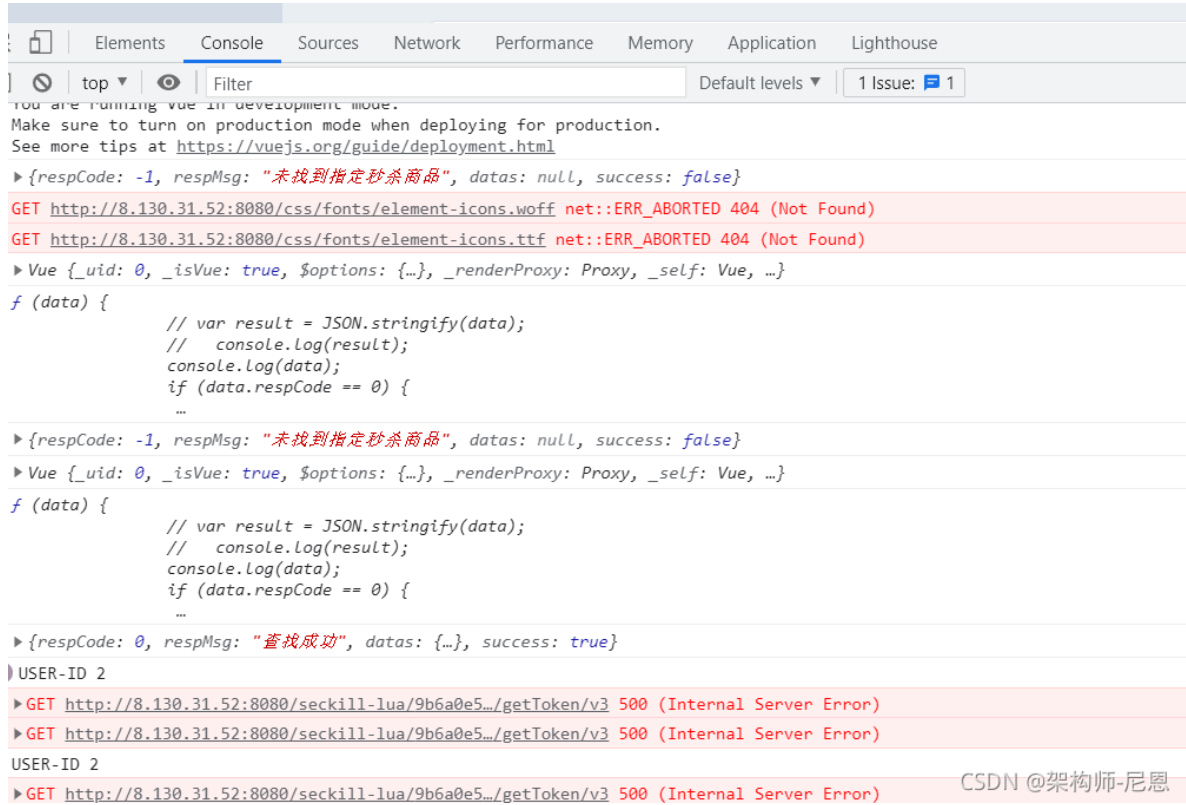
Lua获取令牌

```
log(aata);
respCode == 0) {
"未找到指定秒杀商品", datas: null, success: false}
```

CSDN @架构师-尼恩

## 获取秒杀令牌

## 有错误，看日志

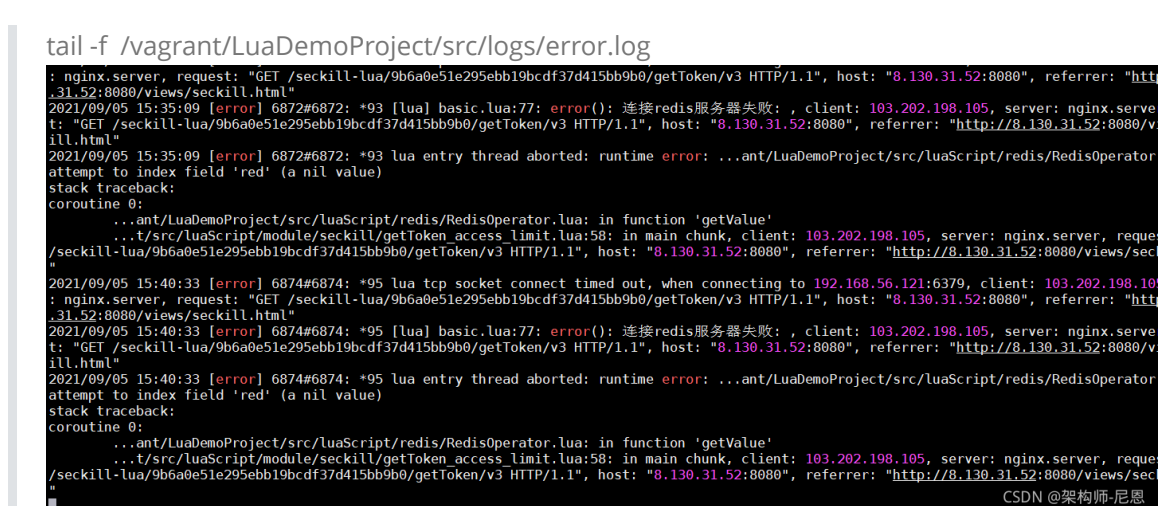


The screenshot shows the Chrome DevTools Console with the following content:

- Navigation bar: Elements, Console, Sources, Network, Performance, Memory, Application, Lighthouse.
- Filter: top, Filter, Default levels, 1 Issue: 1.
- Message: "You are running vue in development mode. Make sure to turn on production mode when deploying for production. See more tips at <https://vuejs.org/guide/deployment.html>"
- JavaScript log: `{respCode: -1, respMsg: "未找到指定秒杀商品", datas: null, success: false}`
- Network error: `GET http://8.130.31.52:8080/css/fonts/element-icons.woff net::ERR_ABORTED 404 (Not Found)`
- Network error: `GET http://8.130.31.52:8080/css/fonts/element-icons.ttf net::ERR_ABORTED 404 (Not Found)`
- JavaScript log: `Vue { _uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...}`
- JavaScript log: `f (data) { // var result = JSON.stringify(data); // console.log(result); console.log(data); if (data.respCode == 0) { ... }`
- JavaScript log: `{respCode: -1, respMsg: "未找到指定秒杀商品", datas: null, success: false}`
- JavaScript log: `Vue { _uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...}`
- JavaScript log: `f (data) { // var result = JSON.stringify(data); // console.log(result); console.log(data); if (data.respCode == 0) { ... }`
- JavaScript log: `{respCode: 0, respMsg: "查找成功", datas: {...}, success: true}`
- USER-ID 2
- Network error: `GET http://8.130.31.52:8080/seckill-lua/9b6a0e5.../getToken/v3 500 (Internal Server Error)`
- Network error: `GET http://8.130.31.52:8080/seckill-lua/9b6a0e5.../getToken/v3 500 (Internal Server Error)`
- USER-ID 2
- Network error: `GET http://8.130.31.52:8080/seckill-lua/9b6a0e5.../getToken/v3 500 (Internal Server Error)`

CSDN @架构师-尼恩

## 有错误，看日志



The screenshot shows the terminal output of the command `tail -f /vagrant/LuaDemoProject/src/logs/error.log`. The logs contain the following information:

- nginx server request: `GET /seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1`, host: `8.130.31.52:8080`, referer: `http://8.130.31.52:8080/views/seckill.html`
- 2021/09/05 15:35:09 [error] 6872#6872: \*93 [lua] basic.lua:77: error(): 连接redis服务器失败: , client: 103.202.198.105, server: nginx.server, request: "GET /seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1", host: "8.130.31.52:8080", referer: "http://8.130.31.52:8080/views/seckill.html"
- 2021/09/05 15:35:09 [error] 6872#6872: \*93 lua entry thread aborted: runtime error: ...ant/LuaDemoProject/src/luaScript/redis/RedisOperator attempt to index field 'red' (a nil value)
- stack traceback: coroutines:0: ...ant/LuaDemoProject/src/luaScript/redis/RedisOperator.lua: in function 'getValue' ...t/src/luaScript/module/seckill/getToken\_access\_limit.lua:58: in main chunk, client: 103.202.198.105, server: nginx.server, request: "/seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1", host: "8.130.31.52:8080", referer: "http://8.130.31.52:8080/views/seckill.html"
- 2021/09/05 15:40:33 [error] 6874#6874: \*95 lua tcp socket connect timed out, when connecting to 192.168.56.121:6379, client: 103.202.198.105, server: nginx.server, request: "GET /seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1", host: "8.130.31.52:8080", referer: "http://8.130.31.52:8080/views/seckill.html"
- 2021/09/05 15:40:33 [error] 6874#6874: \*95 [lua] basic.lua:77: error(): 连接redis服务器失败: , client: 103.202.198.105, server: nginx.server, request: "GET /seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1", host: "8.130.31.52:8080", referer: "http://8.130.31.52:8080/views/seckill.html"
- 2021/09/05 15:40:33 [error] 6874#6874: \*95 lua entry thread aborted: runtime error: ...ant/LuaDemoProject/src/luaScript/redis/RedisOperator attempt to index field 'red' (a nil value)
- stack traceback: coroutines:0: ...ant/LuaDemoProject/src/luaScript/redis/RedisOperator.lua: in function 'getValue' ...t/src/luaScript/module/seckill/getToken\_access\_limit.lua:58: in main chunk, client: 103.202.198.105, server: nginx.server, request: "/seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3 HTTP/1.1", host: "8.130.31.52:8080", referer: "http://8.130.31.52:8080/views/seckill.html"

CSDN @架构师-尼恩

找到ip: 172.26.9.107

云服务器 ECS / 实例

## 实例

创建实例  搜索 标签

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置
<input type="checkbox"/> i-0jldyt0g07cjj3yw65tng-zone-b			乌兰察布 可用区B	39.101.66.58 (公) 172.27.236.247 (私有)	运行中	专有网络	8 vCPU 32 G ecs.g6.2xlarge
<input type="checkbox"/> i-0jl6iyk3i7d253yhhy9a ng-zone-a			乌兰察布 可用区A	39.101.70.150 (公) 172.21.148.18 (私有)	运行中	专有网络	8 vCPU 32 G ecs.g6e.2xlarge
<input type="checkbox"/> i-0jldyt0g07cj2fibr0wu mysql-to-eureka			乌兰察布 可用区C	8.130.24.72 (公) 172.26.9.107 (私有)	运行中	专有网络	8 vCPU 32 G ecs.g7.2xlarge
<input type="checkbox"/> i-0jldyt0g07cj1fzrq1aj seckill-ginx-20210904			乌兰察布 可用区C	8.130.31.52 (公)	运行中	专有网络	8 vCPU 32 G ecs.g7.2xlarge

请选择按量的实例, 且实例没有被锁定

CSDN @架构师-尼恩

## 修改配置文件

```
1 -- 定义一个统一的redis 配置模块
2
3 -- 统一的模块对象
4 local _Module = {
5 -- redis 服务器的地址
6 host_name = "172.26.9.107";
7 --host_name = "192.168.56.121";
8 --host_name = "cdh1";
9
10 -- redis 服务器的端口
11 port = "6379";
12
13 -- redis 服务器的数据库
14 db = "0";
15
16 -- redis 服务器的密码
17 password = '123456';
18
19 --连接超时时长
20 timeout = 20000;
21
22 -- 线程池的连接数量
23 pool_size = 100;
24
25 -- 最大的空闲时间 ,单位: 毫秒
```

CSDN @架构师-尼恩

上传并且, 重启ng

▲ 不安全 | 8.130.31.52:8080/views/seckill.html

用户ID	2
用户信息	点击右边设置用户,获取用户
秒杀ID	1238353441793769472
商品名称	秒杀商品-1
库存数量	10000
原始库存	10000
暴露地址	9b6a0e51e295ebb19bcdcf37d415bb9b0
秒杀令牌	35c3d04e-eb42-4391-8b22-b75eeeeabd055

CSDN @架构师-尼恩

## 云主机使用的问题

### ssh不能登录

可能是sshd无法启动呢，然后输入这两行命令：

可采取以下两步解决

```
chown -R root.root /var/empty/sshd
chmod 744 /var/empty/sshd
systemctl start sshd.service
systemctl status sshd.service
```

```
netstat -ntlp | grep 22
```

就可以解决上述的问题。

如果还提示错误，查看empty的权限是否被修改

### 修改配置文件

按照实际情况，增加连接池的数量，关闭日志

```
#user nobody;
#worker_processes 1;
worker_processes 8; #这个根据硬件有多少核CPU而定
```

```

#开发环境
#error_log logs/error.log debug;
#生产环境
error_log logs/error.log;

pid logs/nginx.pid;

events {
    worker_connections 1024000;
}

http {
    default_type 'text/html';
    charset utf-8;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log logs/access_main.log main;
    access_log off; #日志功能要关闭
    #sendfile: 设置为on表示启动高效传输文件的模式。
    # sendfile可以让Nginx在传输文件时直接在磁盘和tcp socket之间传输数据。
    # 开启这个参数后可以让数据不用经过用户buffer。表示开启零复制，建议生产环境使用
    sendfile off;
    #sendfile on;
    #tcp_nopush on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    #gzip on;
    gzip off;
    #gzip最小长度 一般设置成1K就行，小于1K的就不压缩了 不然会越压越大
    gzip_min_length 1024;
    #gzip压缩比，1 压缩比最小处理速度最快，9 压缩比最大但处理最慢（传输快但比较消耗cpu）。
    gzip_comp_level 1;
    #匹配MIME类型进行压缩
    gzip_types text/plain application/json application/javascript application/x-
    javascript text/css application/xml text/javascript application/x-httpd-php
    image/jpeg image/gif image/png application/vnd.ms-fontobject font/ttf
    font/opentype font/x-woff image/svg+xml font/woff;
    gzip_vary on;
    gzip_disable "MSIE [1-6]\.";
    #设置系统获取几个单位的缓存用于存储gzip的压缩结果数据流。 例如 4 4k 代表以4k为单位，按照原
    始数据大小以4k为单位的4倍申请内存。 4 8k 代表以8k为单位，按照原始数据大小以8k为单位的4倍申请内
    存。
    #如果没有设置，默认值是申请跟原始数据相同大小的内存空间去存储gzip压缩结果。
    gzip_buffers 4 8k;
    #gzip默认最低支持1.1现在改成最低支持1.0 近代浏览器基本不设置
    gzip_http_version 1.0;
    #Nginx作为反向代理的时候启用，开启或者关闭后端服务器返回的结果，匹配的前提是后端服务器必须要
    返回包含"Via"的 header头。
    #off - 关闭所有的代理结果数据的压缩
    #expired - 启用压缩，如果header头中包含 "Expires" 头信息

```

```

#no-cache - 启用压缩, 如果header头中包含 "Cache-Control:no-cache" 头信息
#no-store - 启用压缩, 如果header头中包含 "Cache-Control:no-store" 头信息
#private - 启用压缩, 如果header头中包含 "Cache-Control:private" 头信息
#no_last_modified - 启用压缩, 如果header头中不包含 "Last-Modified" 头信息
#no_etag - 启用压缩, 如果header头中不包含 "ETag" 头信息
#auth - 启用压缩, 如果header头中包含 "Authorization" 头信息
#any - 无条件启用压缩
gzip_proxied expired no-cache no-store private auth;

#指定缓存信息
lua_shared_dict ngx_cache 128m;
#指定缓存信息
lua_shared_dict seckill_cache 128m;
#保证只有一个线程去访问redis或是mysql-lock for cache
lua_shared_dict cache_lock 100k;
#lua扩展加载

# for linux
# lua_package_path
"./?.lua;/vagrant/LuaDemoProject/src/?.lua;/usr/local/ZeroBraneStudio-
1.80/?.lua;/usr/local/ZeroBraneStudio-1.80/?.lua;";
# lua_package_cpath "/usr/local/ZeroBraneStudio-1.80/bin/clibs/?.so;";
lua_package_path
"./?.lua;/vagrant/LuaDemoProject/src/?.lua;/vagrant/LuaDemoProject/vendor/templa
te/?.lua;/vagrant/LuaDemoProject/src/?.lua;/usr/local/openresty/lualib/?.lua
;/usr/local/openresty/lualib/?.lua;";
lua_package_cpath
"/usr/local/openresty/lualib/?.so;/usr/local/openresty/lualib/?.so;";

# for windows
# lua_package_path
"./?.lua;c:/dev/refer/LuaDemoProject/src/vendor/jwt/?.lua;c:/dev/refer/LuaDemoPr
oject/src/?.lua;E:/tool/ZeroBraneStudio-
1.80/lualibs/?.lua;E:/tool/ZeroBraneStudio-
1.80/lualibs/?.lua;E:/tool/openresty-1.13.6.2-win32/lualib/?.lua;";
# lua_package_cpath "E:/tool/ZeroBraneStudio-
1.80/bin/clibs/?.dll;E:/tool/openresty-1.13.6.2-win32/lualib/?.dll;";

# 初始化项目
init_by_lua_file luaScript/initial/loading_config.lua;

#调试模式 (即关闭lua脚本缓存)
lua_code_cache on;
#lua_code_cache off;

#内部网关的代理, 内部网关带有 token 认证
upstream zuul {
# idea 开发环境
# server 192.168.56.121:7799;
# centos 自验证环境
server "cdh1:8888";
keepalive 1000;
}

underscores_in_headers on;

```

```

limit_req_zone $arg_sku_id zone=skuzone:10m rate=6r/m;
limit_req_zone $http_user_id zone=userzone:10m rate=6r/m;
limit_req_zone $binary_remote_addr zone=perip:10m rate=6r/m;
limit_req_zone $server_name zone=perserver:1m rate=10r/s;
limit_req_zone $server_name zone=seckill_server:1m rate=50000r/s;

server {
    listen 80;
    server_name admin.nginx.server;
    default_type 'text/html';
    charset utf-8;

    limit_req zone=perip;
    limit_req zone=perserver;

    #管理控制台的web页面
    location / {
        if ($request_method = 'OPTIONS') {
            add_header Access-Control-Max-Age 600000;
            add_header Access-Control-Allow-Origin *;
            add_header Access-Control-Allow-Credentials true;
            add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'DNT, X-Mx-ReqToken, Keep-Alive, User-Agent, X-Requested-With, If-Modified-Since, Cache-Control, Content-Type, token';
            return 204;
        }
        # IDEA 管理控制台的web页面
        proxy_pass http://192.168.56.121:8066/ ;
        # proxy_pass http://zuul;
    }
}

server {
    listen 80 default;
    server_name nginx.server *.nginx.server;
    default_type 'text/html';
    charset utf-8;

    # 转发 zuul
    location / {
        if ($request_method = 'OPTIONS') {
            add_header Access-Control-Max-Age 600000;
            add_header Access-Control-Allow-Origin *;
            add_header Access-Control-Allow-Credentials true;
            add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'DNT, X-Mx-ReqToken, Keep-Alive, User-Agent, X-Requested-With, If-Modified-Since, Cache-Control, Content-Type, token';
            return 204;
        }

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Nginx-Proxy true;
    }
}

```

```

    proxy_pass http://zuul;
}

# 开发调试: 用户服务
location ^~ /uaa-provider/ {
    # idea 开发环境
    # proxy_pass http://192.168.56.121:7702/;
    # centos 自验证环境
    proxy_pass http://192.168.56.121:7702/uaa-provider/ ;
}

# 开发调试: 秒杀服务
location ^~ /seckill-provider/ {
    proxy_pass http://192.168.56.121:7701/seckill-provider/ ;
}

# 开发调试: 管理服务
location ^~ /backend-provider/ {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header backend 'true'; # 给后台的请求加上自定义头, 方便session区分
    proxy_set_header X-Nginx-Proxy true;
    # 指向微服务
    proxy_pass http://192.168.56.121:6600/backend-provider/ ;

    #指向gate way
    # proxy_pass http://zuul;
}

# 反向代理: 秒杀web页面
location ^~ /seckill-web/ {
    proxy_pass http://192.168.56.121:6601/seckill-web/ ;
}

# Nginx+lua 秒杀: 获取秒杀 token
location = /seckill-provider/api/seckill/redis/token/v2 {
    default_type 'application/json';
    charset utf-8;
    # 限流的 lua 脚本
    access_by_lua_file luaScript/module/seckill/getToken_access_limit.lua;
    # 获取秒杀token lua 脚本
    content_by_lua_file luaScript/module/seckill/getToken.lua;
}

# ratelimit by sku id
location = /ratelimit/sku {
    limit_req zone=skuzone;
    echo "正常的响应";
}

# ratelimit by user id
location = /ratelimit/demo {
    limit_req zone=userzone;
    echo "正常的响应";
}

```

```

}

location = /50x.html{
    echo "限流后的降级内容";
}

error_page 502 503 =200 /50x.html;

# 访问的路径 http://cdh1/ratelimit/demo2?seckillskuId=3
# ratelimit by sku id
location = /ratelimit/demo2 {
    # 限流的 lua 脚本
    access_by_lua_file luaScript/module/seckill/getToken_access_limit.lua;

    echo "正常的响应";
}

}

server {
    listen      8080 default;
    server_name  nginx.server *.nginx.server ;
    limit_req   zone=seckill_server;

    #第一个lua脚本 hellword
    location /helloworld {
        default_type 'text/html';
        charset utf-8;
        content_by_lua_file luaScript/module/demo/helloworld.lua;
    }

    location / {          # 自动匹配到(htm|html)格式
        ## 开发阶段，配置页面不缓存html和htm结尾的文件
        add_header Cache-Control "private, no-store, no-cache, must-revalidate,
proxy-revalidate";
        index index.html;
        root /vagrant/LuaDemoProject/src/www/static; #服务器路径
        default_type 'text/html';
    }

    location ~ .*\. (htm|html)$ {          # 自动匹配到(htm|html)格式
        ## 开发阶段，配置页面不缓存html和htm结尾的文件
        add_header Cache-Control "private, no-store, no-cache, must-revalidate,
proxy-revalidate";
        root /vagrant/LuaDemoProject/src/www/static; #服务器路径
        default_type 'text/html';
    }

    location ~ .*\. (js|script)$ {          # 自动匹配到(jpg|gif|png)格式
        add_header Cache-Control "private, no-store, no-cache, must-revalidate,
proxy-revalidate";
        root /vagrant/LuaDemoProject/src/www/static; #服务器路径
    }

```

```

    default_type 'application/javascript';
}

location ~ .*\.css$ {          # 自动匹配到(css)格式
    root /vagrant/LuaDemoProject/src/www/static; #服务器路径
    default_type 'text/css';
}

# 开发调试: 库存服务
location ^~ /stock-provider/ {
    proxy_pass http://zuul/stock-provider/ ;
}

# 开发调试: 秒杀服务
location ^~ /seckill-provider/ {
    # proxy_pass http://localhost:7701/seckill-provider/ ;
    #指向gate way
    proxy_pass http://zuul/seckill-provider/;
}

# Nginx+lua : 获取 商品信息
location = /stock-lua/gooddetail {
    default_type 'application/json';
    charset utf-8;
    # 限流的 lua 脚本
    #access_by_lua_file luaScript/module/seckill/getToken_access_limit.lua;
    # 获取秒杀token lua 脚本
    content_by_lua_file luaScript/module/seckill/good_detail.lua;
}

# Nginx+lua 秒杀: 获取秒杀 token
location ~ /seckill-lua/(.*)/getToken/v3 {
    default_type 'application/json';
    charset utf-8;
    set $skuId $1;
    limit_req zone=userzone;

    # 限流的 lua 脚本
    access_by_lua_file luaScript/module/seckill/getToken_access_limit.lua;
    # 获取秒杀token lua 脚本
    content_by_lua_file luaScript/module/seckill/getToken_v3.lua;
}
}
}

```

## 更改启动脚本

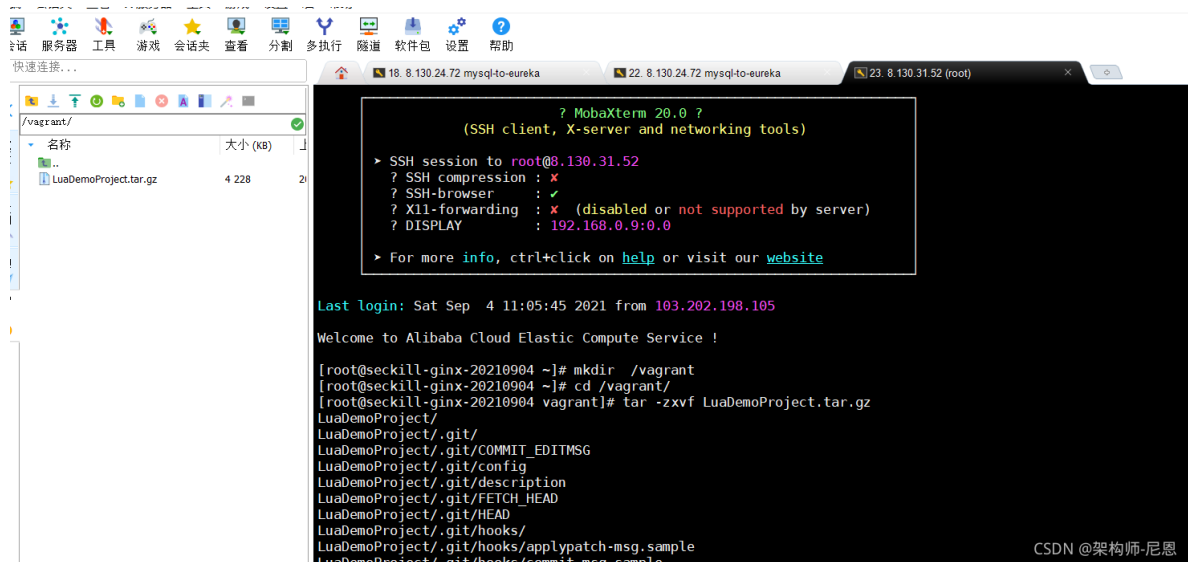
```
# PROJECT_CONF="nginx-seckill.conf"
PROJECT_CONF="nginx-seckill-sit.conf"
```

## 打包

```
MINGW64 /e/virtual/workcluster
$ tar -zcvf LuaDemoProject.tar.gz LuaDemoProject
```

## 创建目录并且启动

```
[root@seckill-ginx-20210904 ~]# mkdir /vagrant
[root@seckill-ginx-20210904 ~]# cd /vagrant/
[root@seckill-ginx-20210904 vagrant]# tar -zxvf LuaDemoProject.tar.gz
```



增加域名解析，然后启动

```
[root@seckill-ginx-20210904 vagrant]# cat >> /etc/hosts <<EOF
> 172.26.9.107 cdh1
> 192.168.56.122 cdh2
> 192.168.56.123 cdh3
> EOF
[root@seckill-ginx-20210904 vagrant]# sh
/vagrant/LuaDemoProject/sh/linux/openresty-restart.sh
shell dir is: /vagrant/LuaDemoProject/sh/linux
openrestry/nginx is not running!
OPENRESTRY_PATH:/usr/local/openresty
PROJECT_PATH:/vagrant/LuaDemoProject/src
openrestry/nginx starting succeeded!
pid is 3161 3162 3163 3164 3165 3166 3167 3168 3169
```

```
cat >> /etc/hosts <<EOF
172.26.9.107 cdh1
172.26.9.105 cdh2
192.168.56.123 cdh3
EOF
```

```
sh /vagrant/LuaDemoProject/sh/linux/openresty-restart.sh
```

测试一下访问的端口：

```
[root@seckill-ginx-20210904 vagrant]# curl http://cdh2:8080/echo
echo

[root@seckill-ginx-20210904 ~]# curl http://cdh2:8080/echo -X POST
```

## 开放端口，然后访问

---

### 开放端口

实例详情

# ← seckill-ginx-20210904 ▾

购买相同配置 刷新 全

- 实例详情
- 监控
- 安全组
- 云盘
- 快照
- 实例快照
- 弹性网卡
- 远程命令/文件
- 操作记录
- 健康诊断
- 事件

## 基本信息

诊断健康状态 new | 启动 | 重启 | 停止 | 配置安全组规则 | 重置实例密码

seckill-ginx-20210904 运行中

实例ID	i-0jldyt0g07cj1fzrq1aj	<a href="#">远程连接</a>	地域	华北6 (乌兰察布)
公网IP	8.130.31.52	<a href="#">转换为弹性公网IP</a>	所在可用区	乌兰察布 可用区C
安全组	sg-0jl12lxqxx06yg25e6jr	<a href="#">加入安全组</a>	主机名	seckill-ginx-20210904 <a href="#">修改实例主机名</a>
标签	-	<a href="#">编辑标签</a>	创建时间	2021年9月4日 10:32:00
描述	-	<a href="#">修改实例描述</a>	自动释放时间	- <a href="#">释放设置</a>

CPU&内存	8核 32 GiB	云盘	1	<a href="#">重新初始化云盘</a>
操作系统	CentOS 7.4 64位	<a href="#">更换操作系统</a>	快照	0
实例规格	ecs.g7.2xlarge	<a href="#">更改实例规格</a>	镜像ID	centos_7_04_64_20G_alibase_201... <a href="#">创建自定义镜像</a>
实例规格族	ecs.g7		当前使用带宽	1Mbps (峰值) <a href="#">按量付费实例更改带宽</a>

安全防护状态 健康状

## 重要事件告警

暂无重要事件告警信息

CSDN @架构师-尼恩

## 安全组

云服务器 ECS

# ← seckill-ginx-20210904 ▾

购买相同配置 刷新

- 实例详情
- 监控
- 安全组
- 云盘
- 快照
- 实例快照
- 弹性网卡
- 远程命令/文件
- 操作记录
- 健康诊断
- 事件

内网入方向全部规则 内网出方向全部规则 [安全组列表](#)

[加入安全组](#) [替换安全组](#)

安全组ID/名称	描述	所属专有网络	安全组类型	操作
sg-0jl12lxqxx06yg25e6jr sg-0jl12lxqxx06yg25e6j...	System created securit...	vpc-0jloI728kajniwxau0c07	普通安全组	<a href="#">配置规则</a>   <a href="#">移出</a>

## 配置规则

# ← sg-0jl12lxqxx06yg25e6j... ▾ / vpc-0jloI728kajniwxau0c07

## 基本信息

安全组ID/名称	sg-0jl12lxqxx06yg25e6jr/ sg-0jl12lxqxx06yg25e6jr	网络	vpc-0jloI728kajniwxau0c07
安全组类型	普通安全组	描述	System created security group.
标签		资源组	

访问规则 [导入安全组规则](#) [导出](#)

[入方向](#) [出方向](#)

[手动添加](#) [快速添加](#) [全部编辑](#)

授权策略	优先级	协议类型	端口范围	授权对象	描述	创建时间
<input type="checkbox"/> 允许	100	自定义 TCP	目的: 3389/3389	源: 0.0.0.0/0	System created rule.	2021年9月4日 10:32:11
<input type="checkbox"/> 允许	100	全部 ICMP(Ipv4)	目的: -1/-1	源: 0.0.0.0/0	System created rule.	2021年9月4日 10:32:11
<input type="checkbox"/> 允许	100	自定义 TCP	目的: 22/22	源: 0.0.0.0/0	System created rule.	2021年9月4日 10:32:11

[删除](#)

CSDN @架构师-尼恩

## 端口范围

入方向 出方向

手动添加 快速添加 全部编辑

授权策略	优先级	协议类型	端口范围	授权对象	描述	操作
允许	1	自定义 TCP	*目的: 8080/8080	*源: 0.0.0.0	秒杀端口	保存 预览 删除
允许	100	自定义 TCP	目的: 3389/3389	源: 0.0.0.0/0	System created rule.	编辑 复制 删除
允许	100	全部 ICMP(Iv4)	目的: -1/-1	源: 0.0.0.0/0	System created rule.	编辑 复制 删除
允许	100	自定义 TCP	目的: 22/22	源: 0.0.0.0/0	System created rule.	编辑 复制 删除

## 然后访问

```
curl http://8.130.31.52:8080/echo
```

浏览器访问库存服务:

```
http://8.130.31.52:8080/stock-provider/swagger-ui.html
```

```
http://8.130.31.52:8080/stock-provider/swagger-ui.html
```

## 秒杀商品添加

→ 不安全 | 8.130.31.52:8080/stock-provider/swagger-ui.html#/商品库存

### swagger

Base URL: 8.130.31.52:8080/stock-provider  
<http://8.130.31.52:8080/stock-provider/v2/api-docs>

Zuul+Swagger2 构建 RESTful APIs

[Terms of service](#)  
[疯狂创客圈 - Website](#)

#### Rockmq消息Demo

Rockmq Message Controller

#### 商品库存

Seckill Sku Stock Controller

- POST /api/seckill/sku/add/v1 增加秒杀商品
- POST /api/seckill/sku/delete/v1 删除商品信息
- POST /api/seckill/sku/detail/v1 秒杀商品详情
- POST /api/seckill/sku/expose/v1 暴露商品秒杀

CSDN @架构师-尼恩

## 增加秒杀的商品

Rockmq消息Demo Rockmq Message Controller

商品库存 Seckill Sku Stock Controller

**POST** /api/seckill/sku/add/v1 增加秒杀商品

Parameters

Name	Description
USER-ID string (header)	user-id <input type="text" value="USER-ID - user-id"/>
costPrice * required number(\$float) (query)	秒杀价格 <input type="text" value="1000"/>
price * required number(\$float) (query)	原始价格 <input type="text" value="1000"/>

Request URL

http://8.130.31.52:8080/stock-provider/api/seckill/sku/add/v1?costPrice=1000&price=1000&stockCount=10000&title=%E7%A7%92%E6%9D%80%E5%95%86%E5%93%81-1

Server response

Code	Details
200	<p>Response body</p> <pre>{   "respCode": 0,   "respMsg": "增加秒杀的商品成功",   "datas": {     "id": "1238353441793769472",     "title": "秒杀商品-1",     "image": null,     "price": 1000,     "costPrice": 1000,     "createTime": "2021-09-05 14:27:13",     "startTime": "2021-08-05 14:27:13",     "endTime": "2021-10-05 14:27:13",     "stockCount": 10000,     "rawStockCount": 10000,     "exposed": false,     "exposedKey": null   },   "success": true }</pre> <p>Response headers</p> <pre>access-control-allow-credentials: true access-control-allow-headers: Content-Type,x-requested-with,X-Custom-Header,Authorization,token,backend access-control-allow-methods: POST, GET, OPTIONS,HEAD,DELETE access-control-allow-origin: http://8.130.31.52:8080</pre>

CSDN @架构师-尼恩

id: 1238353441793769472

添加成功, 记住id

1238353441793769472

Request URL

```
http://8.130.31.52:8080/stock-provider/api/seckill/sku/add/v1?costPrice=1000&price=1000&stockCount=10000
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "respCode": 0,   "respMsg": "增加秒杀的商品成功",   "datas": {     "id": "1238379241972696064",     "title": "秒杀商品-1",     "image": null,     "price": 1000,     "costPrice": 1000,     "createTime": "2021-09-05 15:18:29",     "startTime": "2021-08-05 15:18:29",     "endTime": "2021-10-05 15:18:29",     "stockCount": 10000,     "rawStockCount": 10000,     "exposed": false,     "exposedKey": null   },   "success": true }</pre> <p>Response headers</p> <pre>access-control-allow-credentials: true access-control-allow-headers: Content-Type,x-requested-with,X-Custom-Header,Authorization access-control-allow-methods: POST, GET, OPTIONS,HEAD,DELETE access-control-allow-origin: http://8.130.31.52:8080</pre>

CSDN @架构师-尼恩

## 秒杀暴露

Name	Description
USER-ID string (header)	user-id <input type="text" value="USER-ID - user-id"/>
<b>dto</b> * required (body)	dto Example Value   Model <pre>{   "exposedKey": 1156308907673518000,   "newStockNum": 10000,   "seckillSkuId": 123835344179376947,   "seckillToken": 1156308907673518000,   "userId": 1 }</pre> <p>Cancel</p> <p>Parameter content type <input type="text" value="application/json"/></p>

CSDN @架构师-尼恩

## 暴露成功

记住 秒杀的暴露key: 9b6a0e51e295ebb19bcdf37d415bb9b0

Curl

```
curl -X POST "http://8.130.31.52:8080/stock-provider/api/seckill/sku/expose/v1" -H "accept: */*" -H "Content-Type: application/json" --data '{"newStockNum": 10000, "seckillSkuId": 1238353441793769472, "seckillToken": "1156308907673518000, "userId": "1"}'
```

Request URL

```
http://8.130.31.52:8080/stock-provider/api/seckill/sku/expose/v1
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "respCode": 0,   "respMsg": "秒杀开启成功",   "datas": {     "id": "1238353441793769472",     "title": "秒杀商品-1",     "image": null,     "price": 1000,     "costPrice": 1000,     "createTime": "2021-09-05 14:27:14",     "startTime": "2021-08-05 14:27:14",     "endTime": "2021-10-05 14:27:14",     "stockCount": 10000,     "rawStockCount": 10000,     "exposed": true,     "exposedKey": "9b6a0e51e295ebb19bcdf37d415bb9b0"   },   "success": true }</pre> <p>Response headers</p>

CSDN @架构师-尼恩

## 获取秒杀令牌

<http://8.130.31.52:8080/seckill-provider/swagger-ui.html#/>

→ 不安全 | 8.130.31.52:8080/seckill-provider/swagger-ui.html#/秒杀练习\_分段锁\_版本

Zuul+Swagger2 构建 RESTful APIs

[Terms of service](#)

[疯狂创客圈 - Website](#)

---

**Rockmq消息Demo** Rockmq Message Controller

---

**秒杀练习 RedisLock 版本** Seckill By Redis Lock Controller

---

**秒杀练习 分段锁 版本** Seckill By Segment Lock Controller

---

**POST** /api/seckill/seglock/doSeckill/v1 秒杀

---

**POST** /api/seckill/seglock/getSeckillResult/v1 获取秒杀的结果

---

**GET** /api/seckill/seglock/token/{exposedKey}/v1 排队获取秒杀令牌

---

**秒杀练习 ZkLock 版本** Seckill By Zk Lock Controller

---

**秒杀练习 订单管理** Seckill Order Controller

CSDN @架构师-尼恩

## 获取令牌成功

> 不安全 | 8.130.31.52:8080/seckill-provider/swagger-ui.html#/秒杀练习%20%20分段锁%20版本/getSeckillTokenUsingGET\_1

```
curl -X GET "http://8.130.31.52:8080/seckill-provider/api/seckill/seglock/token/9b6a0e51e295ebb19bcdf37d415bb9b0/v1" -H "accept: application/json"
```

Request URL

```
http://8.130.31.52:8080/seckill-provider/api/seckill/seglock/token/9b6a0e51e295ebb19bcdf37d415bb9b0/v1
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "respCode": 0,   "respMsg": "这是获取的结果",   "datas": "66b7dfe3b41c467db5757644b86e34d5",   "success": true }</pre> <p>Response headers</p> <pre>access-control-allow-credentials: true access-control-allow-headers: Content-Type,x-requested-with,X-Custom-Header,Authorization,token,backend access-control-allow-methods: POST, GET, OPTIONS,HEAD,DELETE access-control-allow-origin: * access-control-max-age: 3600 connection: keep-alive content-type: application/json;charset=UTF-8 date: Sun, 05 Sep 2021 07:15:18 GMT server: openresty/1.19.9.1 transfer-encoding: chunked</pre>

Responses CSDN @架构师-尼恩

# 高并发场景slb的使用实操

## SLB概念

负载均衡（Server Load Balancer）是将访问流量根据转发策略分发到后端多台云服务器（Elastic Compute Service，简称 ECS）的流量分发控制服务。

负载均衡服务通过设置虚拟服务地址，将位于同一地域的多台ECS实例虚拟成一个高性能、高可用的应用服务池；再根据应用指定的方式，将来自客户端的网络请求分发到云服务器池中。负载均衡服务是ECS面向多机方案的一个配套服务，需要同ECS结合使用。

负载均衡服务会检查云服务器池中ECS实例的健康状态，自动隔离异常状态的ECS实例，从而解决了单台ECS实例的单点问题，提高了应用的整体服务能力。

在标准的负载均衡功能之外，负载均衡服务还具备TCP与HTTP抗DDoS攻击的特性，增强了应用服务的防护能力。

## 参考文档

负载均衡（阿里云帮助与文档）：

<https://developer.aliyun.com/article/697165>

[https://help.aliyun.com/document\\_detail/86451.html?source=5176.11533457&userCode=ffsbbyn0](https://help.aliyun.com/document_detail/86451.html?source=5176.11533457&userCode=ffsbbyn0)

负载均衡 (SLB) 使用最佳实践

<https://developer.aliyun.com/article/80055>

## 一个slb至少需要两个可用区的机器

## 购买slb的服务

The screenshot shows the Alibaba Cloud product page. The URL is <https://aliyun.com/?spm=5176.2020520101.top-nav.dlogo.44a64df5cetzRh>. The page features a search bar and a navigation menu. The '阿里云' logo is visible. The main content area is divided into three columns: '热门推荐' (Recommended), '新晋畅销' (Newly Popular), and '新品发布' (New Products). The '负载均衡 SLB' product is highlighted with a red circle in the '热门推荐' section. The '新品发布' section includes items like '物联网数据存储发布会', '云网管正式商业化发布', '阿里云企业采购数字化产品公测发布', '微服务引擎 MSE 专业版发布', '云安全访问服务', '智能外呼机器人', '事件总线', 'Databricks 数据洞察', '云数据库专属集群 MyBase', '相册与网盘服务', '服务网格 ASM', 'Prometheus 监控服务 (公测中)', and '移动安全加固'.

查看全部产品 >

热门产品

- 弹性计算
- 存储
- 数据库
- 安全
- 大数据
- 人工智能
- 网络与CDN
- 视频服务
- 容器与中间件
- 开发与运维
- 物联网IoT
- 混合云
- 企业应用与云通信

Q 搜索云产品

热门推荐

- 云服务器 ECS
- 域名注册
- 对象存储 OSS
- 短信服务
- 云数据库 RDS MySQL 版
- 商标服务
- 负载均衡 SLB**
- 日志服务 SLS
- CDN
- 弹性公网 IP

新晋畅销

- 无影云桌面
- 文字识别
- 智能语音交互
- 云监控
- 云·速成美站
- 工商财税
- 语音服务
- 媒体处理
- 邮件推送
- 堡垒机

新品发布 >

- 物联网数据存储发布会
- 云网管正式商业化发布
- 阿里云企业采购数字化产品公测发布
- 微服务引擎 MSE 专业版发布
- 云安全访问服务
- 智能外呼机器人
- 事件总线
- Databricks 数据洞察 NEW
- 云数据库专属集群 MyBase
- 相册与网盘服务
- 服务网格 ASM
- Prometheus 监控服务 (公测中)
- 移动安全加固

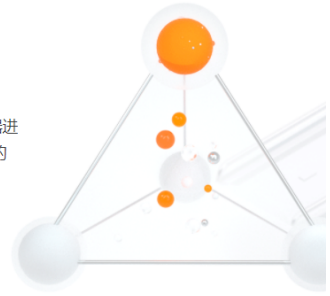
CSDN @架构师-尼恩

# 负载均衡 SLB 播放视频

阿里云负载均衡 (Server Load Balancer, 简称SLB) 是云原生时代应用高可用的基本要素。通过对多台云服务器进行均衡的流量分发调度, 消除单点故障提升应用系统的可靠性与吞吐力。阿里云SLB包含面向4层 (TCP/UDP) 的传统型负载均衡CLB和面向7层 (HTTP/HTTPS/QUIC) 的应用型负载均衡ALB, 是阿里云官方云原生网关。

[天猫淘宝同款高性能高可靠负载均衡CLB。邀您参与免费试用活动](#)

[ALB立即购买](#) [CLB立即购买](#) [产品控制台](#) [产品价格](#) [产品文档](#) [售前咨询](#)



[产品规格](#) [产品类型选择](#) [产品优势](#) [产品功能](#) [应用场景](#) [ALB使用步骤](#) [客户案例](#) [产品动态](#) [更多产品与服务](#)

## 最新动态

- 重磅发布** ALB负载均衡支持WAF透明接入模式, 控制台一键开启应用安全防护
- 产品家族** 阿里云负载均衡SLB产品家族介绍
- 免费试用** CLB负载均衡实名首购用户免费试用 (仅限包年包月简约型实例1个月)
- 重磅发布** ALB应用型负载均衡产品重磅发布, 了解详情

CSDN @架构师-尼恩

## 选择可用区

地域 中国 亚太 欧洲与美洲

华东1 (杭州) 华东2 (上海) 华南1 (深圳) 西南1 (成都) 华北2 (北京) 华北3 (张家口) 华北6 (乌兰察布)

中国 (香港)

VPC

[创建一个VPC](#)

可用区  乌兰察布 可用区A  乌兰察布 可用区B

应用型负载均衡支持多可用区部署, 为保障业务高可用, 请选择2个或以上可用区

IP模式  固定IP  动态IP

此模式下每个可用区只有一个IP, 且IP固定保持不变, 此模式下弹性能力受限, 最大支持10W qps

功能版本 (实例费)  基础版  标准版

基础版包含应用型负载均衡的基本功能, 可支持基于域名、URL、HTTP Header等路由转发, 详情参见[应用型负载均衡基础版和标准版功能差异](#)

实例网络类型  私网  公网

功能版本 (实例费) : ¥0.049 元/小时 容量单位: ¥0.049 元/CU

[立即购买](#)  
CSDN @架构师-尼恩

## slb配置

### 三个部分组成

负载均衡服务由负载均衡实例、监听和后端服务器三个部分组成。

## 负载均衡实例 (Server Load Balancer Instance)

如果您想使用负载均衡服务，必须先创建一个负载均衡实例。一个负载均衡实例可以添加多个监听和后端服务器。

### 监听 (Listener)

在使用负载均衡服务前，您必须为负载均衡实例添加一个监听，指定监听规则和转发策略，并配置健康检查。

针对不同的需求，您可以单独配置四层 (TCP/UDP) 或七层 (HTTP/HTTPS) 监听。

### 后端服务器 (Backend Server)

一组接收前端请求的ECS实例。您可以单独添加ECS实例到服务器池，也可以通过虚拟服务器组或主备服务器组来批量添加和管理。

默认后端服务器是在实例维度上维护的，即负载均衡实例下的所有监听都只能将请求转发到相同ECS实例的相同端口上。虚拟服务器组功能实现了监听维度的转发。您可以针对不同的监听创建不同的虚拟服务器组，即负载均衡实例中的不同监听可以将请求转发到不同端口的后端服务器上。

此外，七层负载均衡服务支持域名、URL转发策略，可以将来自不同域名或者URL的请求转发给不同的后端服务器处理。

## 负载均衡实例配置

### 配置端口监听

负载均衡 SLB / 实例 / 负载均衡业务配置向导

### ← 负载均衡业务配置向导

1 配置监听 2 选择服务器组 3 配置审核

选择负载均衡协议

HTTP

\* 监听端口

8080

监听名称

秒杀服务

高级配置 [修改](#)

连接空闲超时时间	连接请求超时时间	Gzip数据压缩	附加HTTP头字段
15 秒	60 秒	已开启	开启通过X-Forwarded-For头字段获取来访者真实IP。

CSDN @架构师-尼恩

### 配置服务器组

← 负载均衡业务配置向导

1 配置监听 2 选择服务器组 3 配置审核

添加后端服务器用于处理负载均衡接收到的访问请求

选择服务器组

服务器类型 nginx | sgp-up5c70zyulqxm1xkgd

后端协议: HTTP, 调度算法: 加权轮询, 会话保持: 未开启, 健康检查: 已开启

已添加的后端服务器

实例ID	地域	运行状态	专有网络ID	IP地址	端口	权重	描述
没有数据							

每页显示 20 总共 0 个 < 上一页

CSDN @架构师-尼恩

## 创建服务器组



\* 服务器组类型

服务器类型

\* 服务器组名称

nginx

\* VPC

vpc-0j1ol728kajniwxau0c07

选择后端协议

HTTP

选择调度算法

加权轮询

\* 选择资源组

default resource group

开启会话保持



配置健康检查

创建 取消

## 配置审核

负载均衡 SLB / 实例 / 负载均衡业务配置向导

监听

### ← 负载均衡业务配置向导

配置监听 选择服务器组 3 配置审核

#### 配置监听

负载均衡协议	监听端口	监听名称	连接空闲超时时间
HTTP	8080	秒杀服务	15 秒
连接请求超时时间	Gzip数据压缩	附加HTTP头字段	
60 秒	已开启	开启通过X-Forwarded-For头字段获取来访者真实 IP。	

#### 选择服务器组

服务器组  
sgp-up5c70zyulqxm1xkgd

已添加的后端服务器

实例ID	地域	运行状态	专有网络 ID	IP地址	端口	权重	描述
没有数据							

## 后端服务器

### 启动可用区A和B上的ng

连接上两个ecs，安装ng，复制vagrant目录过去，然后启动

```
mkdir -p /vagrant
```

```
scp -r /vagrant/ root@172.27.236.247:/
```

```
scp -r /vagrant/ root@172.21.148.18:/
```

```
sh /vagrant/LuaDemoProject/sh/linux/openresty-restart.sh
```

```
curl http://localhost:8080/echo
```

## slb服务器组管理

负载均衡 SLB / 服务器组

创建服务器组 服务器组名称 请输入服务器组名称进行查询

服务器组名称/ID	VPC	后端协议	关联的实例	类型	状态	标签	操作
nginx sgp-up5c70zyulqxm1xkgd	vpc-0j0l728kajniwxau0c07	HTTP	alb-jvoaudpfhgy7l3d1t	服务器类型	可用		<a href="#">编辑后端服务器</a> <a href="#">编辑基本编排健康检查</a>

批量释放 设置标签 每页显示 20 总共1条 < 上一页 下一页 >

CSDN @架构师-尼恩

## 添加后端服务器

负载均衡 SLB / 服务器组 / sgp-up5c70zyulqxm1xkgd

← sgp-up5c70zyulqxm1xkgd

详细信息 后端服务器

添加后端服务器 服务器ID 请输入服务器ID进行查询

实例ID	地域	运行状态	专有网络ID	IP地址	端口	权重	描述
没有数据							

移除 批量设置相同端口 批量设置相同权重 每页显示 20 总共0个 < 上一页 下一页 >

CSDN @架构师-尼恩

## 选择两接入层服务器

The screenshot shows the '添加后端服务器' (Add Backend Server) dialog in Step 1: '选择服务器' (Select Servers). The dialog is titled '添加后端服务器' and has a close button 'X'. It features two progress steps: '1 选择服务器' (selected) and '2 配置端口和权重' (Configure ports and weights). Below the steps, there are filters for '服务器类型' (Server type) set to '云服务器ECS/弹性网卡ENI', a search bar for '搜索服务器名称、ID或IP地址', and a '购买云服务器' link. The '网络类型' (Network type) is set to '专有网络' (VPC) with 'vpc-0j1cl728kajniwxau0c07' selected. There are checkboxes for '只显示可添加的实例' and '高级模式'. A table lists available servers with columns for '云服务器ID/名称', '可用区', '私网IP', '公网IP/专有网络属性', and '状态'. Two servers are selected with blue checkmarks: 'ng-zone-b' and 'ng-zone-a'. The 'mysql-to-eureka' and 'seckill-ginx' servers are not selected. At the bottom right, there is a watermark 'CSDN @架构师-尼恩'.

云服务器ID/名称	可用区	私网IP	公网IP/专有网络属性	状态
<input checked="" type="checkbox"/> ng-zone-b i-0jldyt0g07cjj3yw65t	乌兰察布 可用区B	172.27.236.247	39.101.66.58(公) vpc-0j1cl728kajniwxau0c07 vsw-0jl85p0q0xl37lysu4rqt	运行中
<input checked="" type="checkbox"/> ng-zone-a i-0j6iyk3i7d253yhhy9a	乌兰察布 可用区A	172.21.148.18	39.101.70.150(公) vpc-0j1cl728kajniwxau0c07 vsw-0jlv8qj7wdb12zjjkwa	运行中
<input type="checkbox"/> mysql-to-eureka i-0jldyt0g07c2fibr0wu	乌兰察布 可用区C	172.26.9.107	8.130.24.72(公) vpc-0j1cl728kajniwxau0c07 vsw-0j1fptkks66xqlm3ri7	运行中
<input type="checkbox"/> seckill-ginx-20210904 i-0jldyt0g07c1fzrq1aj	乌兰察布 可用区C	172.26.9.105	8.130.31.52(公) vpc-0j1cl728kajniwxau0c07 vsw-0j1fptkks66xqlm3ri7	运行中

## 填写端口

The screenshot shows the '添加后端服务器' (Add Backend Server) dialog in Step 2: '配置端口和权重' (Configure ports and weights). The dialog is titled '添加后端服务器' and has a close button 'X'. It features two progress steps: '1 选择服务器' (completed) and '2 配置端口和权重' (selected). Below the steps, there is a checkbox for '展示服务器组内全部内容'. A table lists the selected servers with columns for '云服务器ID/名称', '地域', '私网IP', '端口', '权重', '备注', and '操作'. Two servers are listed: 'ng-zone-b' and 'ng-zone-a'. The '端口' (port) is set to '8080' and the '权重' (weight) is set to '100'. There are '提交' (Submit) buttons for each server. At the bottom, there are checkboxes for '设置相同权重' (Set same weight) and '删除' (Delete). At the bottom right, there is a watermark 'CSDN @架构师-尼恩'.

云服务器ID/名称	地域	私网IP	端口	权重	备注	操作
<input checked="" type="checkbox"/> ng-zone-b i-0jldyt0g07cjj3yw65t	乌兰察布 可用区B	39.101.66.58(公) 172.27.236.247(私)	8080	100	备注	添加端口   移除
<input checked="" type="checkbox"/> ng-zone-a i-0j6iyk3i7d253yhhy9a	乌兰察布 可用区A	39.101.70.150(公) 172.21.148.18(私)	8080	100	备注	添加端口   移除

## 提交成功

The screenshot shows the '添加后端服务器' (Add Backend Server) dialog with a success message. The dialog is titled '添加后端服务器'. A green checkmark icon is followed by the text '添加成功!' (Added successfully!) and a sub-message: '您的本次申请已完成, 所有内容已成功添加。' (Your application is complete, all content has been successfully added.). At the bottom right, there is a watermark 'CSDN @架构师-尼恩'.

## 服务器组内的服务

负载均衡 SLB / 服务器组 / sgp-up5c70zyulqxm1xkgd

### ← sgp-up5c70zyulqxm1xkgd

详细信息	后端服务器							
添加后端服务器	服务器ID	请输入服务器ID进行查询	Q					
<input type="checkbox"/>	实例ID	地域	运行状态	专有网络 ID	IP地址	端口	权重	描述
<input type="checkbox"/>	ng-zone-a i-0j6iyk3i7d253yhhy9a	乌兰察布 可用区A	✓ 可用	vpc-0jloj728kajniwxau0c07	172.21.148.18	8080	100	-
<input type="checkbox"/>	ng-zone-b i-0jdyt0g07cjj3yw65t	乌兰察布 可用区B	✓ 可用	vpc-0jloj728kajniwxau0c07	172.27.236.247	8080	100	-
<input type="checkbox"/>	移除	批量设置相同端口	批量设置相同权重				每页显示 20	总共 2 个 < 上-

CSDN @架构师-尼恩

## 测试

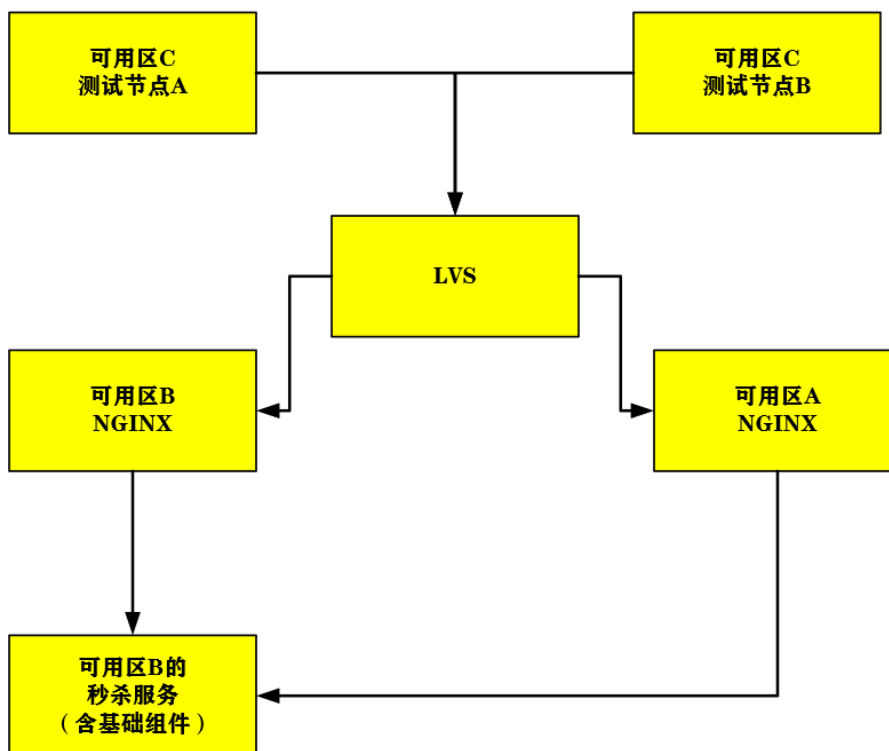
```
[root@nginx-zone-a vagrant]# ping alb-jvoaudpfrhgx713d1t.internal.cn-wulanchabu.alb.aliyuncs.com
PING alb-jvoaudpfrhgx713d1t.internal.cn-wulanchabu.alb.aliyuncs.com (172.21.148.19) 56(84) bytes of data.
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=1 ttl=102 time=0.191 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=2 ttl=102 time=0.126 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=3 ttl=102 time=0.099 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=4 ttl=102 time=0.097 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=5 ttl=102 time=0.100 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=6 ttl=102 time=0.099 ms
64 bytes from 172.21.148.19 (172.21.148.19): icmp_seq=7 ttl=102 time=0.095 ms
^C
--- alb-jvoaudpfrhgx713d1t.internal.cn-wulanchabu.alb.aliyuncs.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6000ms
rtt min/avg/max/mdev = 0.095/0.115/0.191/0.033 ms
```

## 测试接口

curl <http://alb-jvoaudpfrhgx713d1t.internal.cn-wulanchabu.alb.aliyuncs.com:8080/echo>

```
[root@nginx-zone-a vagrant]# curl http://alb-jvoaudpfrhgx713d1t.internal.cn-wulanchabu.alb.aliyuncs.com:8080/echo
echo
[root@nginx-zone-a vagrant]#
```

# 10WQPS压力测试实操



CSDN @架构师-尼恩

## 工具的源码路径

<https://gitee.com/crazymaker/d-hp-tester.git>

The screenshot shows an IDE window with the project 'd-hp-tester' open. The file explorer on the left shows the project structure, including 'src/main/java/com.crazymakercircle'. The main editor displays the 'report()' method in 'ServerConstants.java', which prints a detailed test report. The report includes fields like START\_TIME, END\_TIME, TAKE\_TIME, THREAD\_NUM, CLIENT\_NUM, TEST\_TIME, TOTAL\_REQ, TOTAL\_RES, TOTAL\_ERR, BELOW\_10, BT\_10\_20, BT\_20\_30, OVER\_30, AVG\_LATENCY, and QPS.

```
public void report ()
{
    System.out.println("----- TEST REPORT BEGIN-----");
    System.out.println("START_TIME      : " + this.getStart_time());
    System.out.println("END_TIME        : " + this.getEnd_time());
    System.out.println("TAKE_TIME       : " + this.getTakeTime());
    System.out.println("THREAD_NUM      : " + this.getThread_num());
    System.out.println("CLIENT_NUM      : " + this.getClient_num());
    System.out.println("TEST_TIME       : " + this.getTest_time() + " min");
    System.out.println("TOTAL_REQ       : " + this.getTotal_req());
    System.out.println("TOTAL_RES       : " + this.getTotal_res());
    System.out.println("TOTAL_ERR       : " + this.getTotal_err());
    System.out.println("BELOW_10        : " + this.getBelow_10());
    System.out.println("BT_10_20        : " + this.getBetween_10_20());
    System.out.println("BT_20_30        : " + this.getBetween_20_30());
    System.out.println("OVER_30         : " + this.getOver_30());
    System.out.println("AVG_LATENCY     : " + this.getAvg_latency() + " ms");
    System.out.println("QPS             : " + this.getQps());
    System.out.println("----- TEST REPORT END-----");
}
```

CSDN @架构师-尼恩

# 本地验证一下 压力测试的配置文件

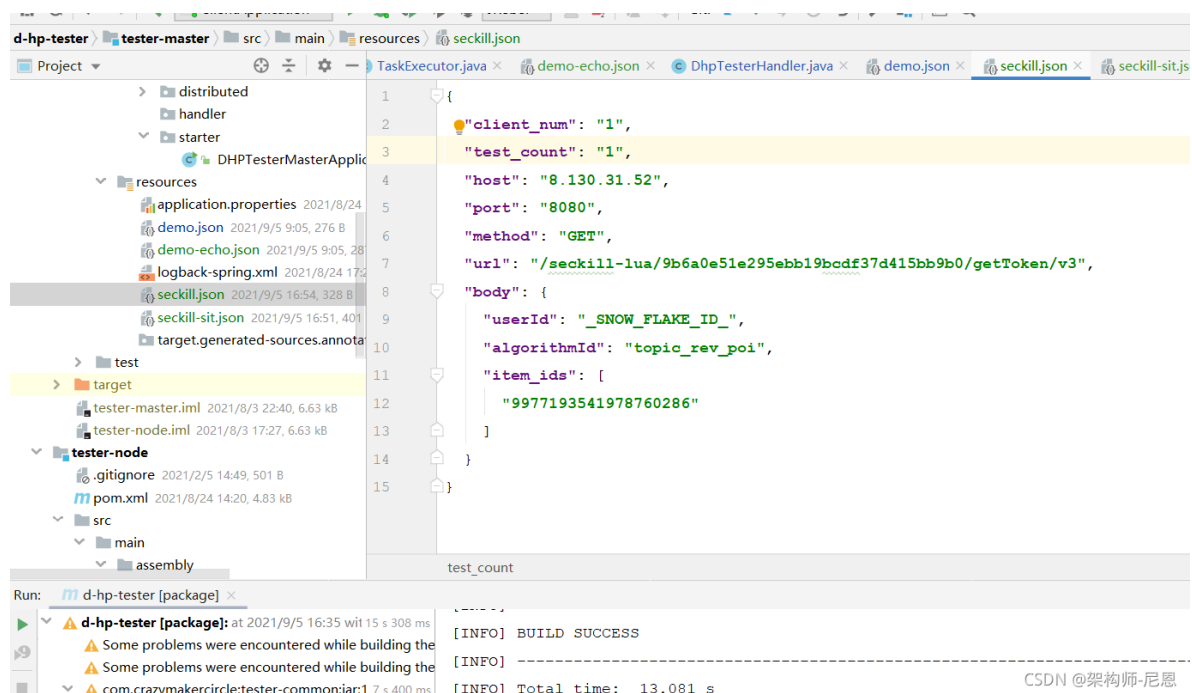
## 修改高并发的用例配置文件

, 修改主机、端口、url

改一下连接数 1,测试次数1

修改ip, 这里用公网ip (8.130.31.52), 而不是私网ip

修改获取令牌的url, 替换商品exposedkey

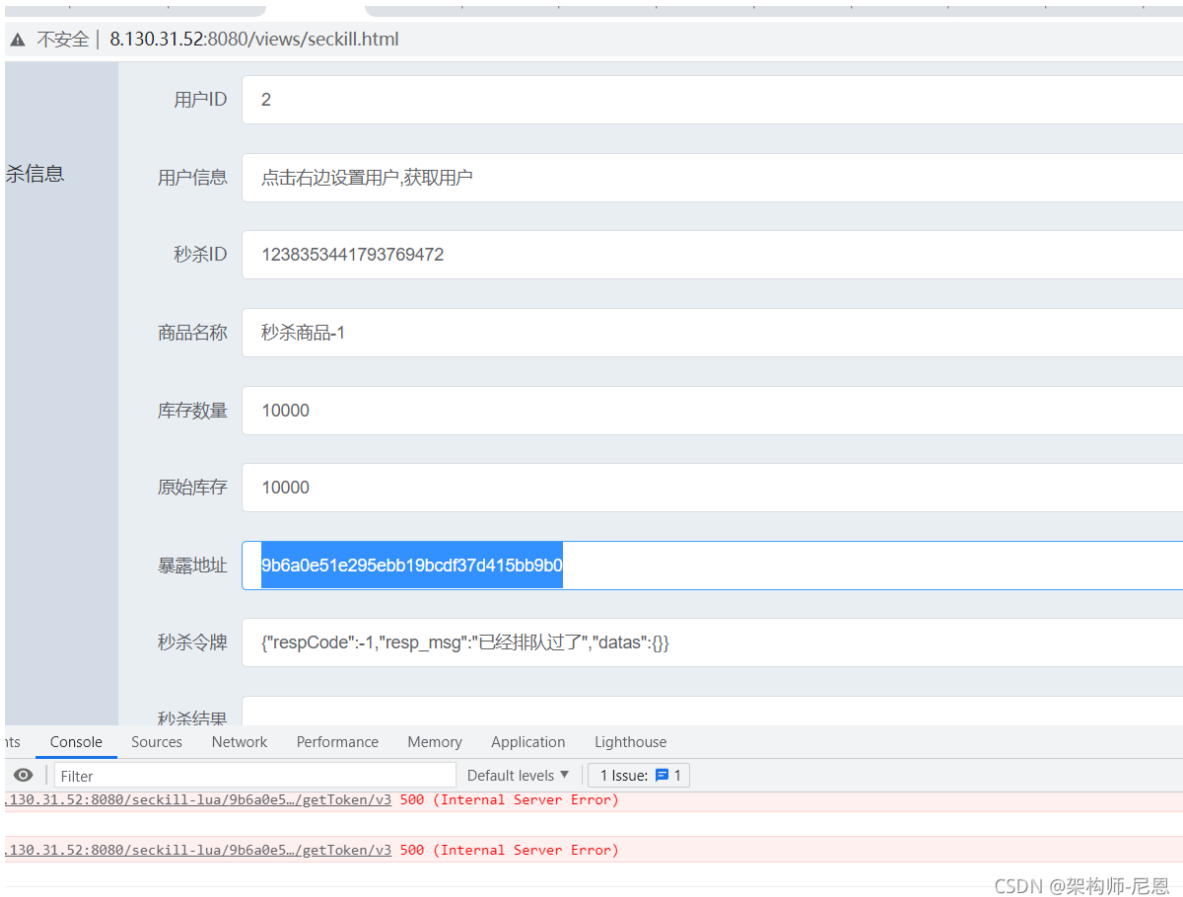


The screenshot shows an IDE window with the following content:

- Project Explorer:** Shows the project structure for 'd-hp-tester'. The 'resources' folder is expanded, showing files like 'application.properties', 'demo.json', 'demo-echo.json', 'logback-spring.xml', 'seckill.json', and 'seckill-sit.json'. The 'test' folder is also visible.
- Code Editor:** Displays the content of 'seckill.json'. The JSON configuration is as follows:

```
1 {
2   "client_num": "1",
3   "test_count": "1",
4   "host": "8.130.31.52",
5   "port": "8080",
6   "method": "GET",
7   "url": "/seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3",
8   "body": {
9     "userId": "_SNOW_FLAKE_ID_",
10    "algorithmId": "topic_rev_poi",
11    "item_ids": [
12      "9977193541978760286"
13    ]
14  }
15 }
```
- Run Console:** Shows the build output for the 'd-hp-tester [package]' target. The output includes:

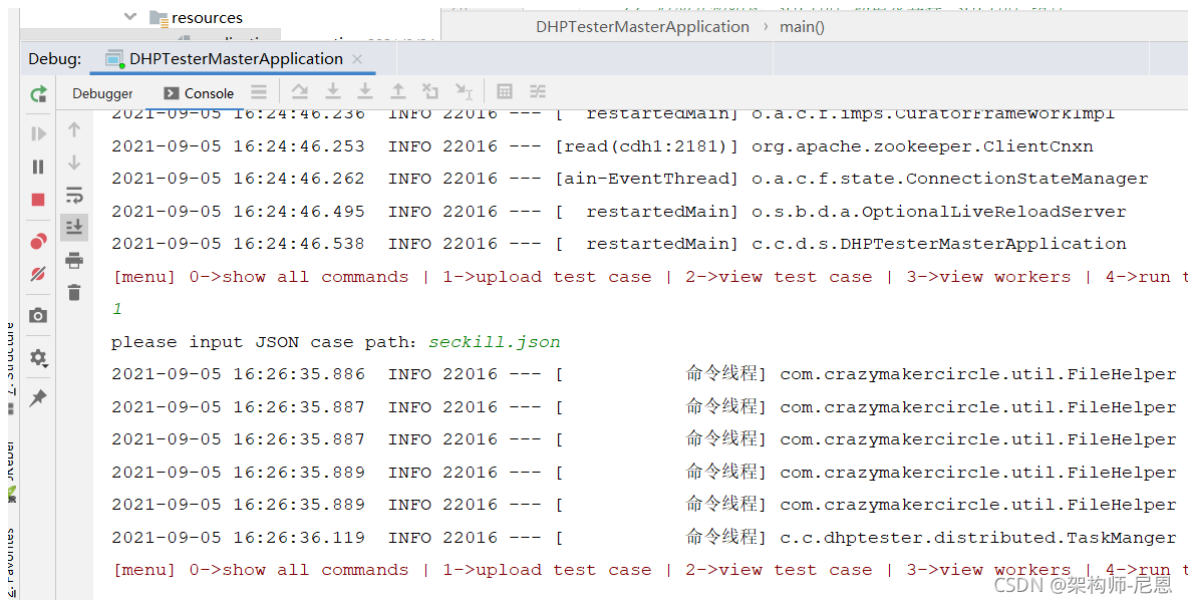
```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 13.081 s
```



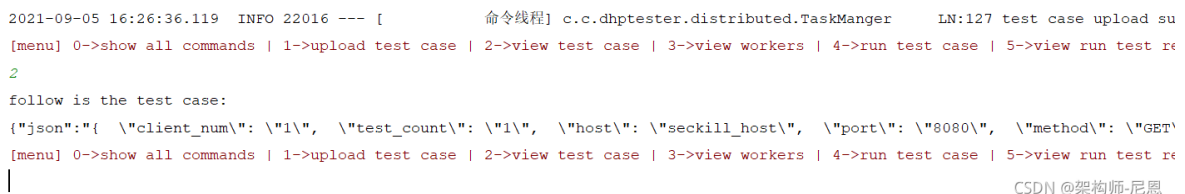
CSDN @架构师-尼恩

## 本地开始用例验证

启动master, 上传用例

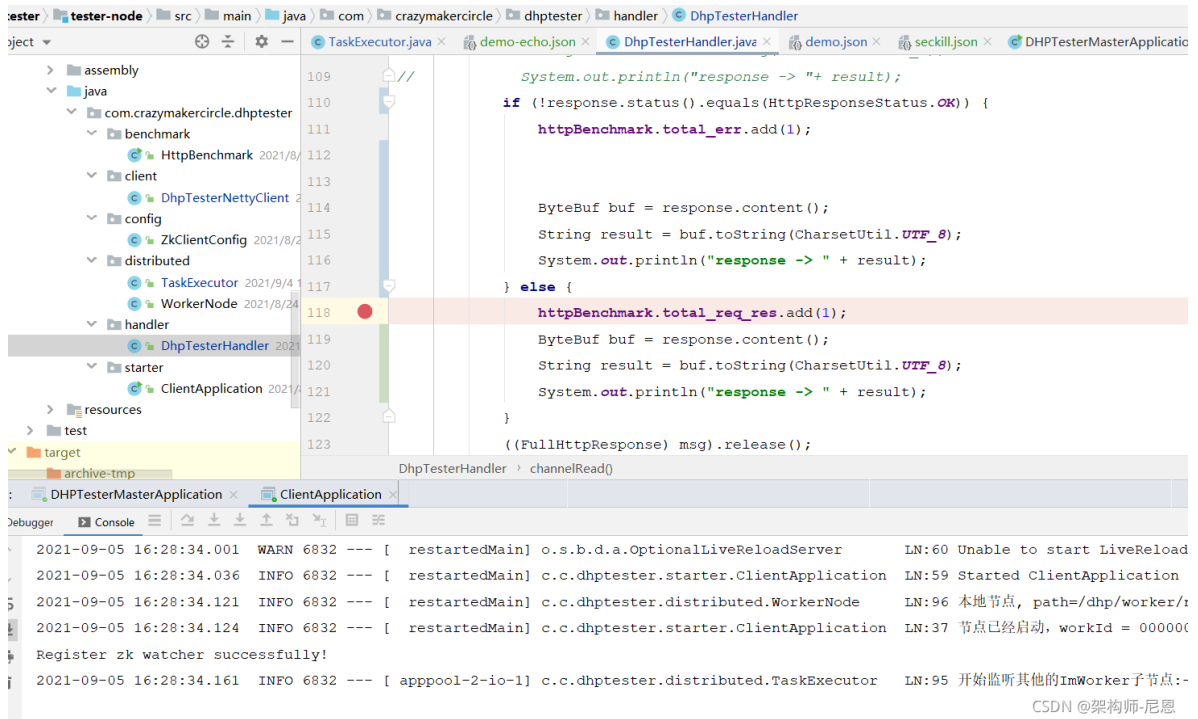


查看用例

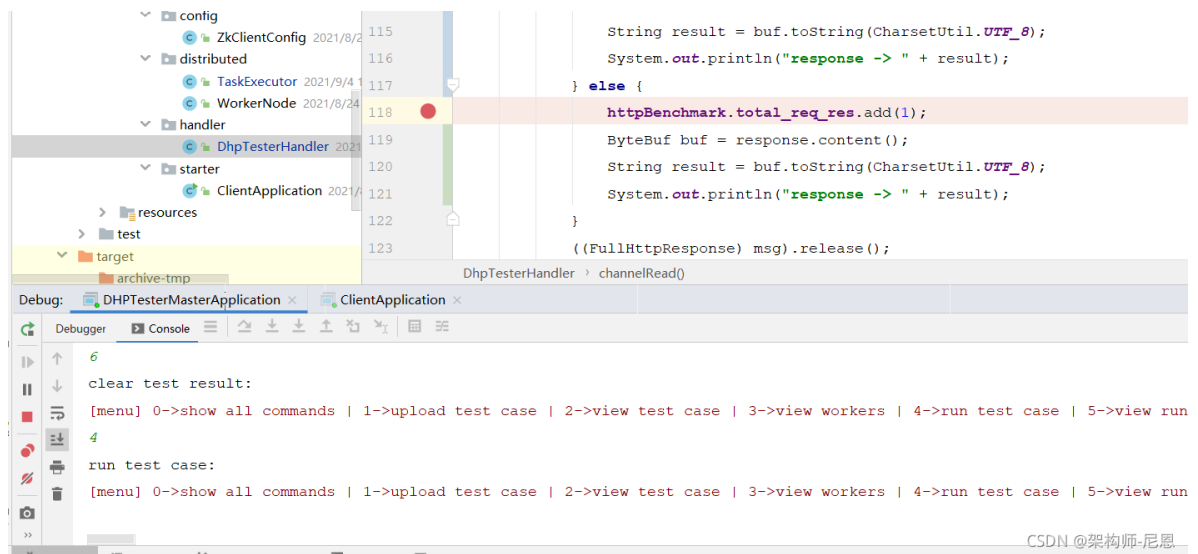


CSDN @架构师-尼恩

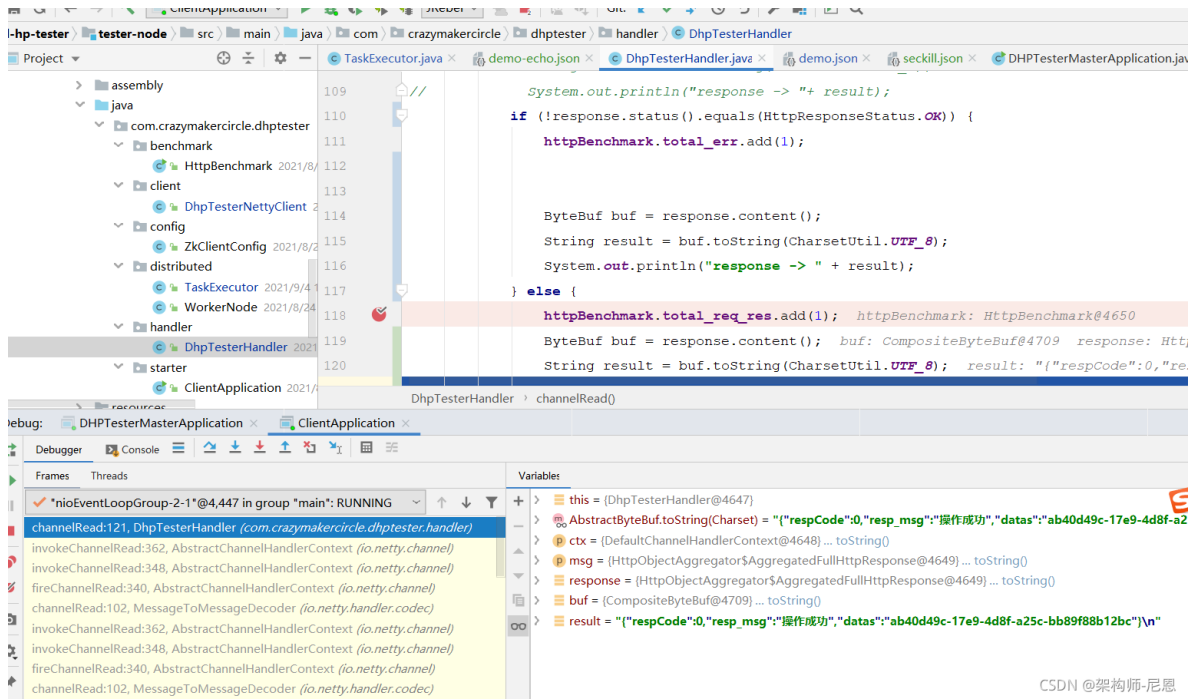
启动工作节点



## master执行命令



## worker请求发送成功

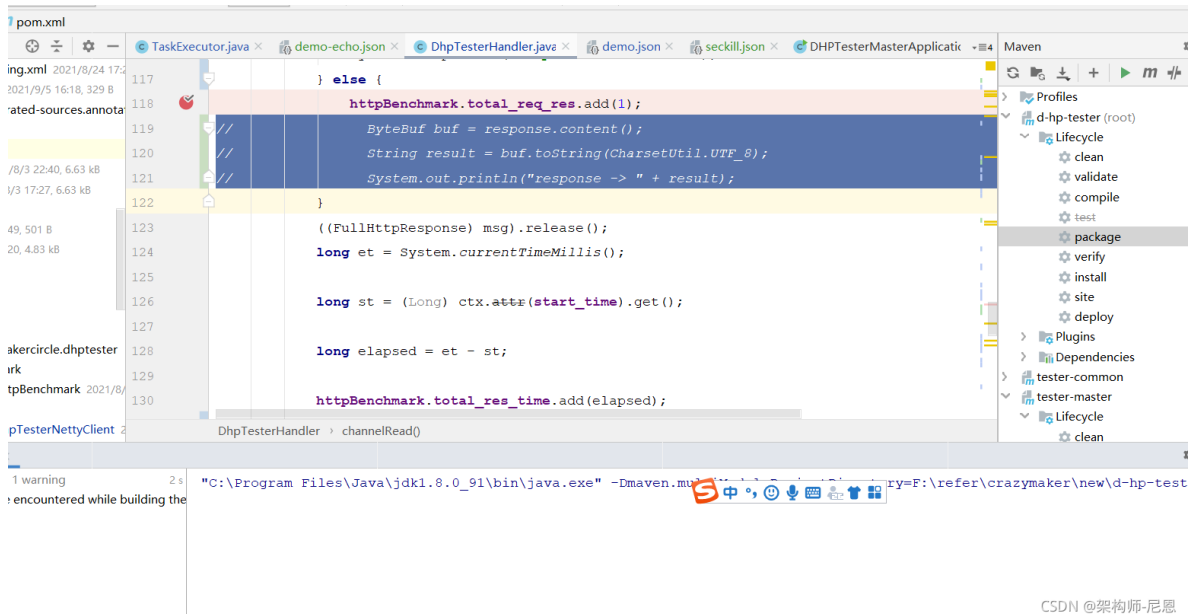


CSDN @架构师-尼恩

本地测试通过

## 工具打包

打包之前，去掉worker中刚才的结果打印代码

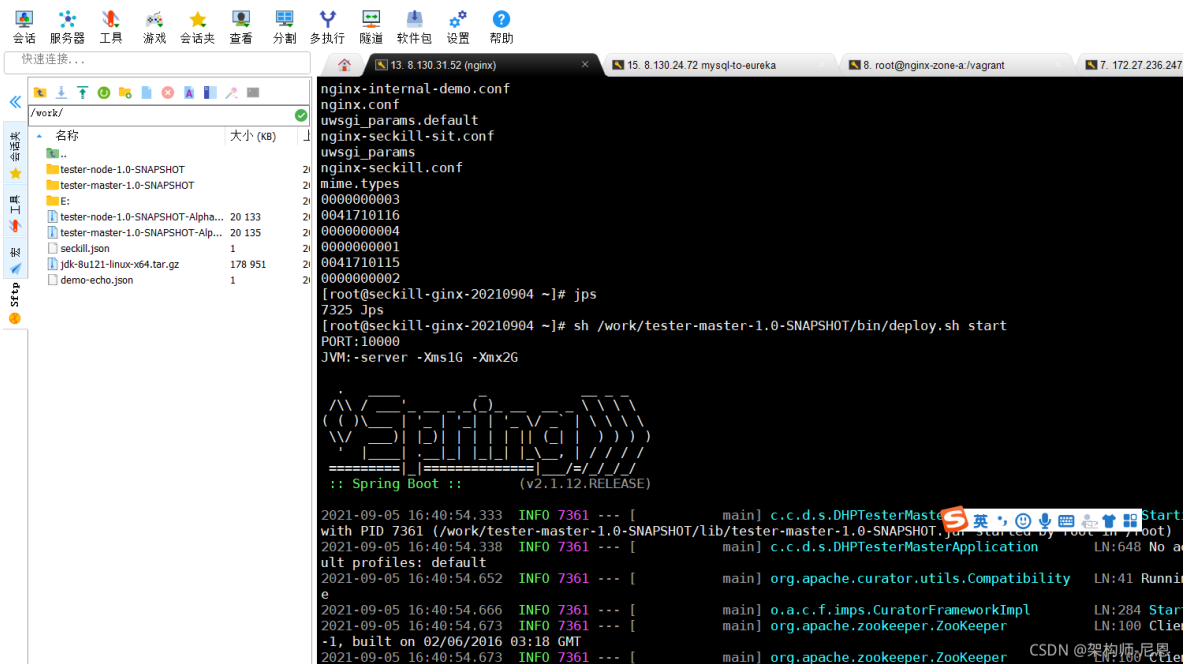


CSDN @架构师-尼恩

## 部署master

上传到/work 解压缩，然后启动

```
sh /work/tester-master-1.0-SNAPSHOT/bin/deploy.sh start
```



```

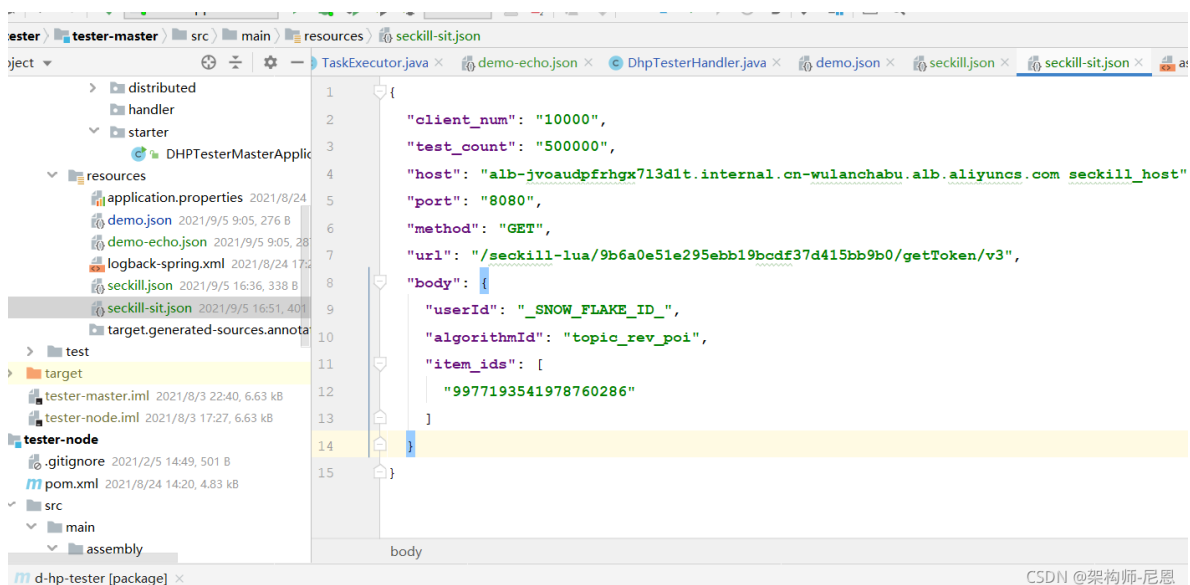
[root@seckill-ginx-20210904 work]# tar -zxvf tester-master-1.0-SNAPSHOT-
Alpha.tar.gz
tester-master-1.0-SNAPSHOT/bin/
tester-master-1.0-SNAPSHOT/bin/deploy.sh
tester-master-1.0-SNAPSHOT/lib/tester-master-1.0-SNAPSHOT.jar
tester-master-1.0-SNAPSHOT/logs/
[root@seckill-ginx-20210904 work]# sh /work/tester-master-1.0-
SNAPSHOT/bin/deploy.sh start

```

## 上传测试用例json

修改一下连接数 10000,测试次数50000

修改成slb的域名



上传

/root/demo-echo.json

/root/seckill-sit.json

## 加载测试用例json

```
[menu] 0->show all commands | 1->upload test case | 2->view test case | 3->view workers | 4->run test case | 5->view run test result | 6->
result |
1
please input JSON case path: /work/seckill.json
2021-09-05 16:43:02.950 INFO 7361 --- [命令线程] com.crazymakercircle.util.FileHelper LN:24 filepath 1=/work/seckill.json
2021-09-05 16:43:02.950 INFO 7361 --- [命令线程] com.crazymakercircle.util.FileHelper LN:37 filepath 3=/work/seckill.json
2021-09-05 16:43:02.979 INFO 7361 --- [命令线程] c.c.dhptester.distributed.TaskManger LN:127 test case upload succeed!
[menu] 0->show all commands | 1->upload test case | 2->view test case | 3->view workers | 4->run test case | 5->view run test result | 6->c
result |
2
follow is the test case:
{"json":{"client_num": "10000", "test_count": "500000", "host": "seckill_host", "port": "8080", "method": "GET",
"/seckill-lua/9b6a0e51e295ebb19bcd37d415bb9b0/getToken/v3", "body": { "userId": "SNOW_FLAKE_ID", "algorithmId": "top
i", "item_ids": [ "9977193541978760286" ] }}, "runId": "un-known", "uploadTime": "2021-09-05 16:43:02"}
[menu] 0->show all commands | 1->upload test case | 2->view test case | 3->view workers | 4->run test case | 5->view run test result | 6->c
result |
CSDN @架构师-尼恩
```

## 上传、解压、启动worker

```
sh /work/tester-node-1.0-SNAPSHOT/bin/deploy.sh start
```

打印日志

```
tail -f /work/logs/tester-node-1.0-SNAPSHOT/output.log
```

```
Welcome to Alibaba Cloud Elastic Compute Service !
[root@seckill-ginx-20210904 ~]# sh /work/tester-node-1.0-SNAPSHOT/bin/deploy.sh start
PORT:10002
jar:/work/tester-node-1.0-SNAPSHOT/lib/tester-node-1.0-SNAPSHOT.jar
JVM:-server -Xms4G -Xmx12G
cmd:nohup java -server -Xms4G -Xmx12G -Dserver.port=10002 -jar /work/tester-node-1.0-SNAPSHOT/lib/tester-node-1.0-SNAPSHOT.jar com
e.dhptester.starter.ClientApplication >> /work/tester-node-1.0-SNAPSHOT/logs/console.log 2>&1 &
tester-node-1.0-SNAPSHOT.jar is running,pid is 7427
[root@seckill-ginx-20210904 ~]# jps
7361 tester-master-1.0-SNAPSHOT.jar
7427 tester-node-1.0-SNAPSHOT.jar
7462 Jps
[root@seckill-ginx-20210904 ~]# tail -f /work/logs/tester-node-1.0-SNAPSHOT/output.log
2021-09-05 16:44:40.913 [main-SendThread(cdh1:2181)] INFO org.apache.zookeeper.ClientCnxn:876 - Socket connection established to c
2181, initiating session
2021-09-05 16:44:40.919 [main-SendThread(cdh1:2181)] INFO org.apache.zookeeper.ClientCnxn:1299 - Session establishment complete on
26.9.107:2181, sessionId = 0x10000012268000c, negotiated timeout = 40000
2021-09-05 16:44:40.924 [main-EventThread] INFO o.a.curator.framework.state.ConnectionStateManager:237 - State change: CONNECTED
2021-09-05 16:44:41.015 [main] INFO c.c.dhptester.starter.ClientApplication:59 - Started ClientApplication in 0.836 seconds (JVM r
2021-09-05 16:44:41.057 [main] INFO c.c.dhptester.distributed.WorkerNode:96 - 本地节点, path=/dhp/worker/node-0000000012, id=00000
2021-09-05 16:44:41.058 [main] INFO c.c.dhptester.starter.ClientApplication:37 - 节点已经启动, workId = 0000000012
2021-09-05 16:44:41.077 [appool-2-io-1] INFO c.c.dhptester.distributed.TaskExecutor:95 - 开始监听其他的ImWorker子节点:-----
2021-09-05 16:44:41.077 [appool-2-io-2] INFO c.c.dhptester.distributed.TaskExecutor:95 - 开始监听其他的ImWorker子节点:-----
2021-09-05 16:44:41.078 [appool-2-io-1] INFO c.c.dhptester.distributed.TaskExecutor:100 - CHILD ADDED : /dhp/task/run-0000000013
2021-09-05 16:44:41.080 [appool-2-io-1] INFO c.c.dhptester.distributed.TaskExecutor:161 - [TreeCache]节点更新端口, path=/dhp/task
data={"json":{"client_num": "10000", "test_count": "500000", "host": "172.26.9.105", "port": "8080", "metho
url": "/echo", "body": { "userId": "Random", "algorithmId": "topic_rev_poi", "item_ids": [ "99771
] }}, "runId": "0000000013", "uploadTime": "2021-09-04 19:15:33"}
CSDN @架构师-尼恩
```

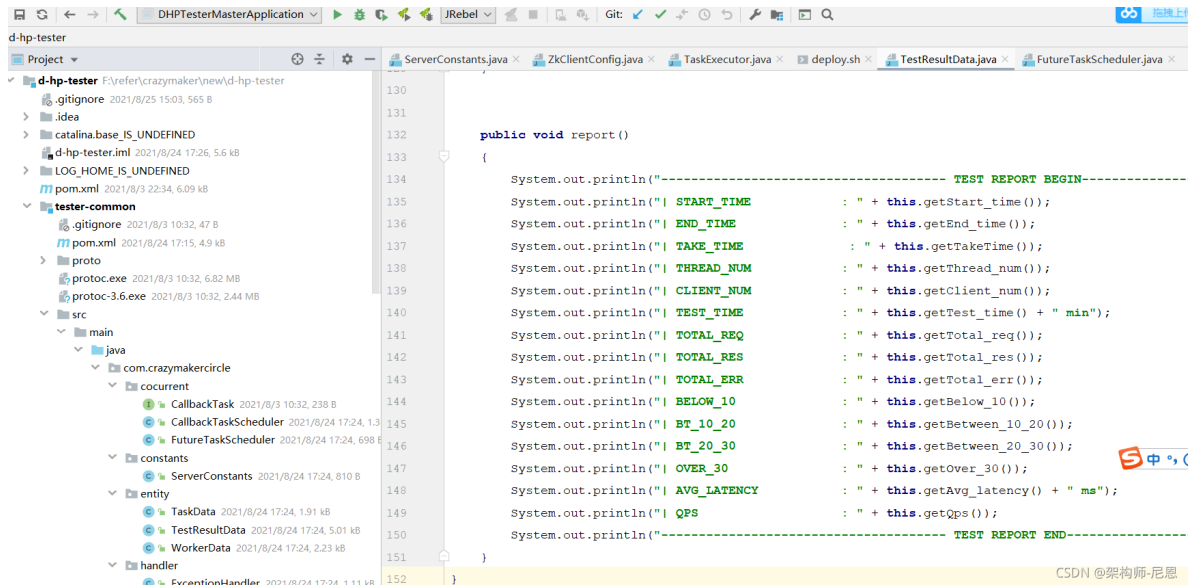
## 启动测试

master 使用4就可以了

# 手写基于ZOOKeeper+Netty的分布式测试工 具：工作节点原理与实战

# 工具的源码路径

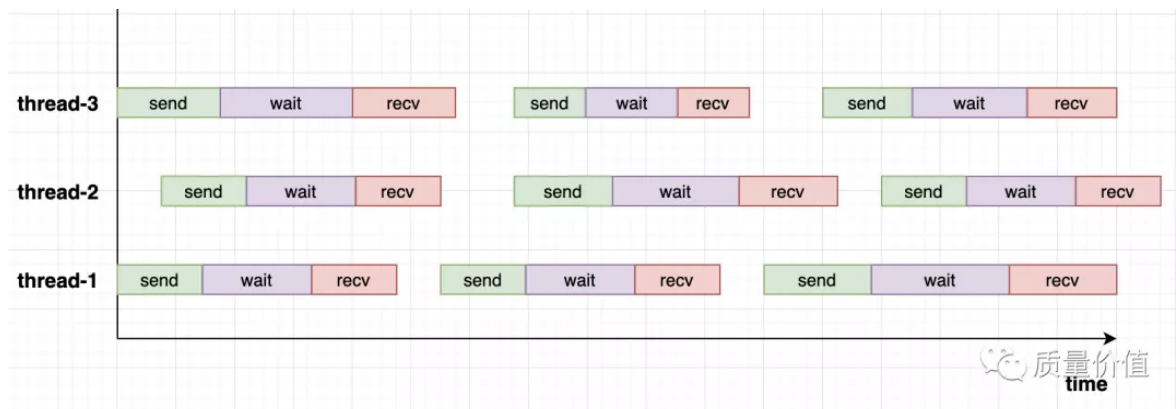
<https://gitee.com/crazymaker/d-hp-tester.git>



## 基于ZOOKeeper+Netty手写分布式测试工具：系统架构设计

### 基于多线程并发的ab、JMeter 性能差的根因

ab、JMeter分别是用C、Java开发的、基于多线程并发模型的压测工具，也是目前最流行的开源压测工具，两者的工作原理类似，如下图：



多线程并发

- 不管ab还是JMeter，其所谓的虚拟用户(vuser)就是对应一个线程
- 在单个线程中，每个请求（query）都是同步调用的，下一个请求要等待前一个请求完成才能进行
- 一个请求（query）分成三部分：
  - send - 施压端发送开始，直到承压端接收完成
  - wait - 承压端接收完成开始，直至业务处理结束
  - recv - 承压端返回数据，直至施压端接收完成
- 同一线程中连续的两个请求之间存在**等待时间**这种概念，即图中的空白处

在多线程并发模型下，是不是可以通过不断增加线程数量生产出更大的压力？

答案是否定的。

事实上一个进程在一个时间点只能执行一个线程，而所谓的**并发**是指在进程里不断切换线程实现了看上去的多个任务的**并发**，但是线程上下文切换有很高的成本，过多的线程数反而会造成性能的严重下滑。

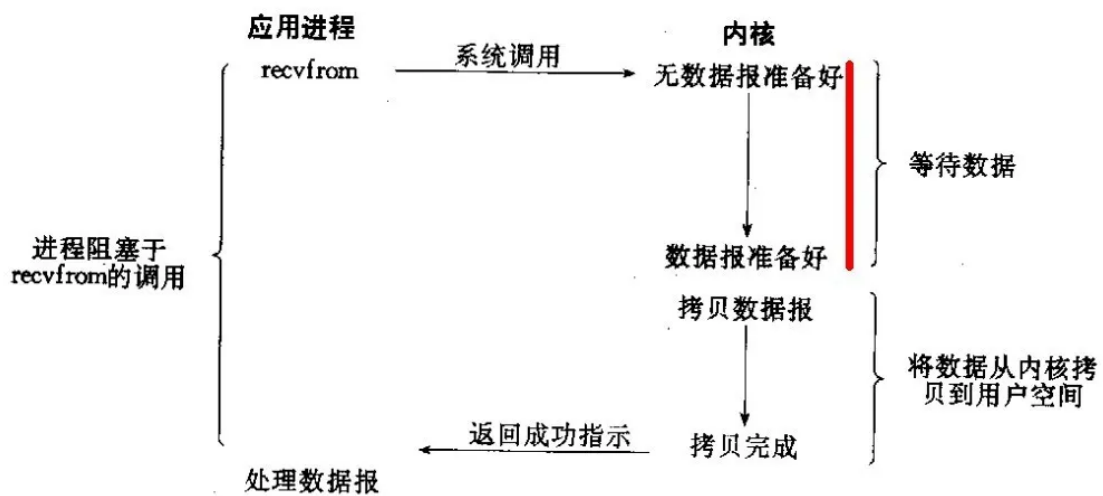


图 6.1 阻塞 I/O 模型

质量价值

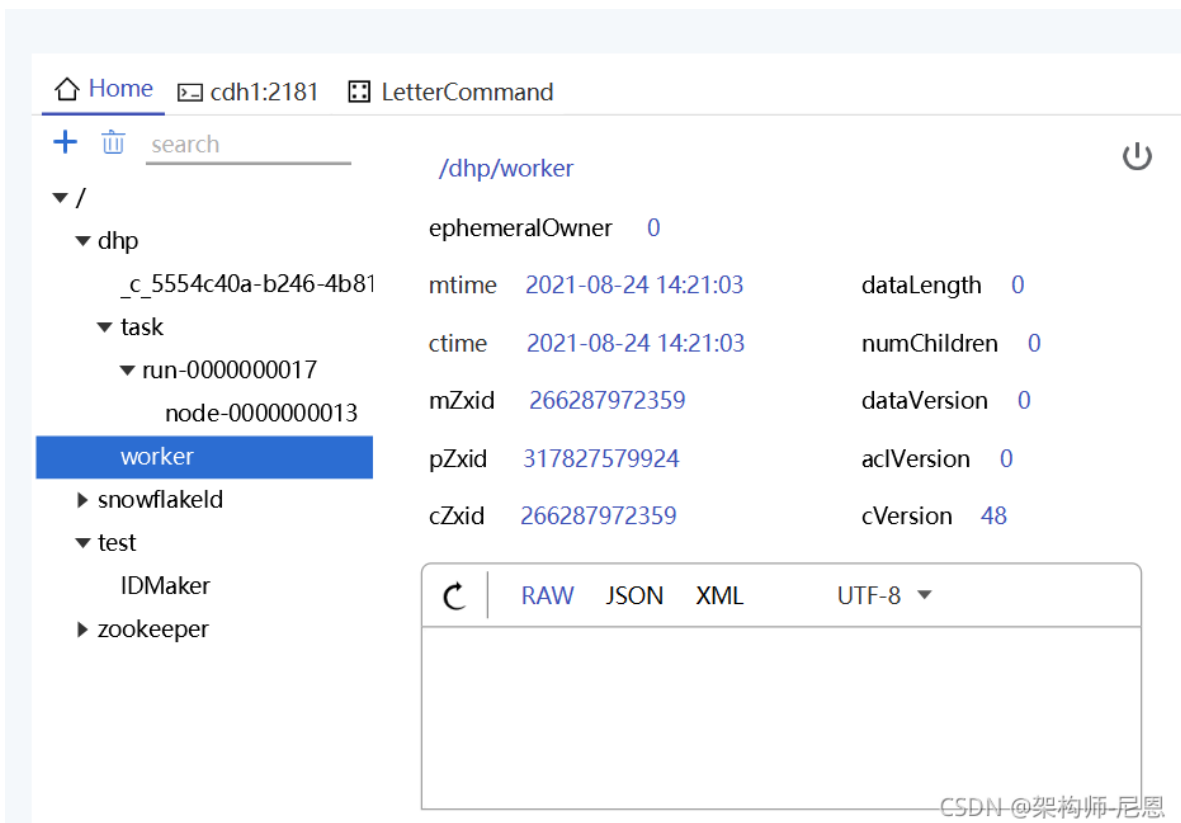
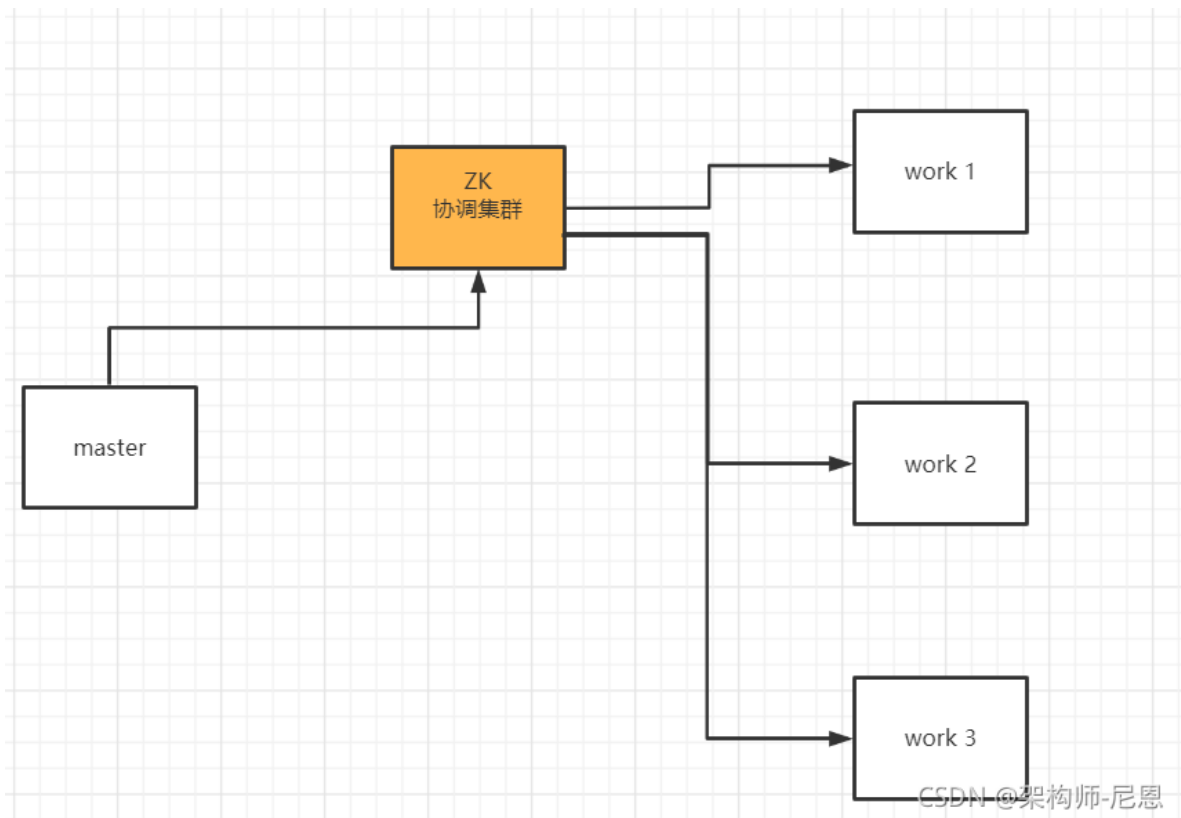
BIO

从应用角度来看，基于多线程的并发模型，往往需要设置**最大并发数**参数，而如果压测场景需要不断往上加压，那这类工具其实挺难应付的。

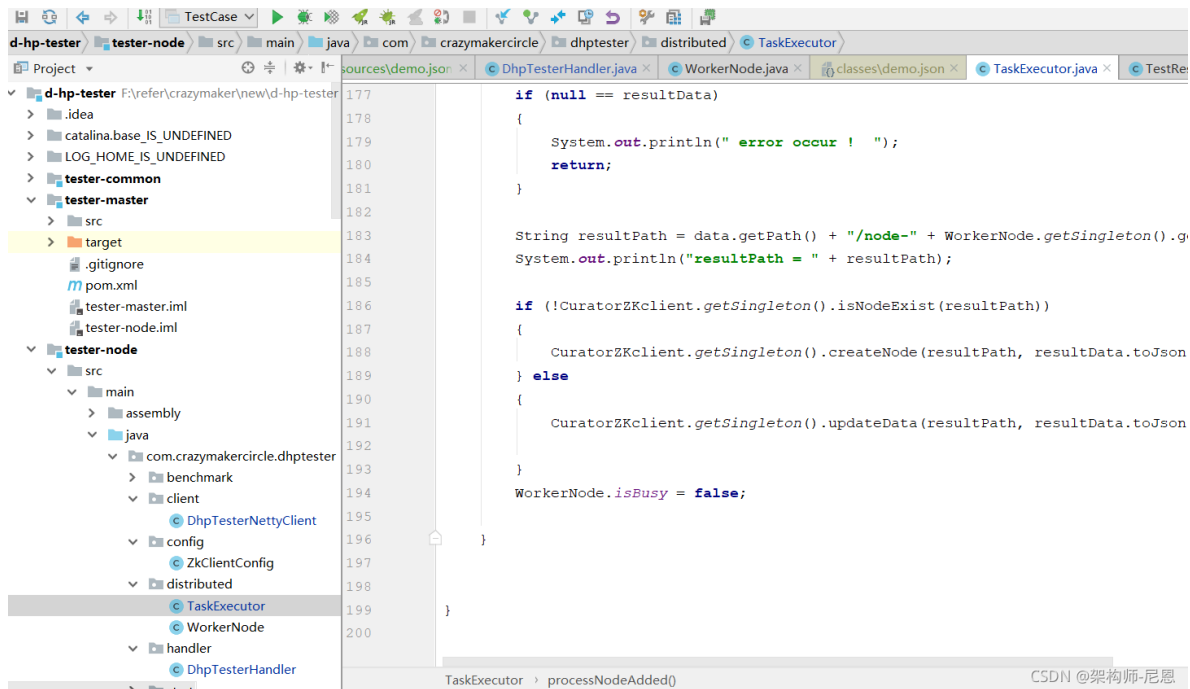
**根因：**

- 1 一个线程对应一个用户，线程数量大的时候，性能低
- 2 bio模型，性能低

## 基于ZOOKeeper+Netty的分布式测试工具的系统架构



## 基于ZOOKeeper+Netty手写分布式测试工具：工作节点的设计与实现



## 体验：执行一下工作节点的 请求测试

### 准备工作

准备好测试接口

```
[root@cdh1 ~]# /vagrant/LuaDemoProject/sh/linux/openresty-restart.sh
shell dir is: /vagrant/LuaDemoProject/sh/linux
openresty/nginx is not running!
OPENRESTY_PATH:/usr/local/openresty
PROJECT_PATH:/vagrant/LuaDemoProject/src
nginx: [alert] lua_code_cache is off; this will hurt performance in
/vagrant/LuaDemoProject/src/conf/nginx-seckill.conf:90
openresty/nginx starting succeeded!
pid is 11777
```

```
[root@cdh1 ~]# curl http://cdh1:8080/echo
echo
```

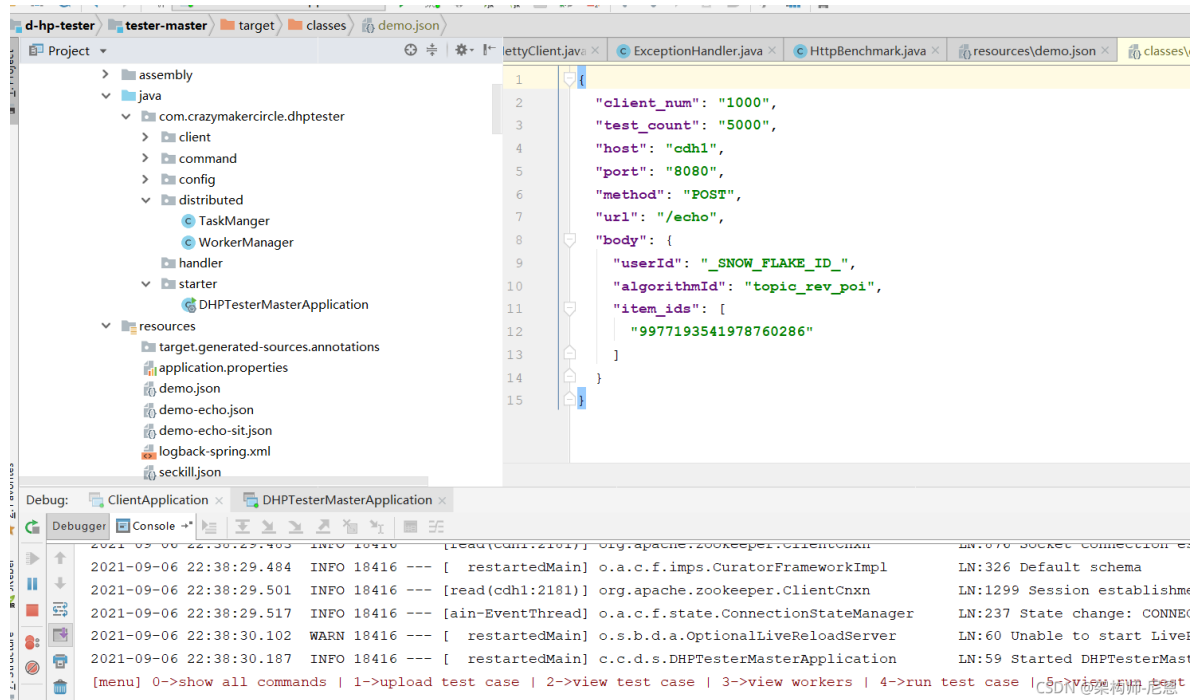
### 准备好测试脚本

```
{
```

```
"client_num": "1000",
"test_count": "10000",
"host": "cdh1",
"port": "80",
"method": "POST",
"url": "/echo",
"body": {
  "userId": "_SNOW_FLAKE_ID_",
  "algorithmId": "topic_rev_poi",
  "item_ids": [
    "9977193541978760286"
  ]
}
}
```

## 执行用例，查看结果

# 基于ZOOKeeper+Netty手写分布式测试工具：Master节点的设计与实现



## 功能设计

1 管理用例

上传、查看

2 查看工作节点

3 启动任务

4 汇总结果

## 功能实现

---