# zookeeper简介

zookeeper是一个开源的分面式协调服务，由知名互联网公司Yahoo创建，它是Chubby的开源实现；换句话讲，zk是一个典型的分布式数据一致性解决方案，分布式应用程序可以基于它实现数据的发布/订阅、负载均衡、名称服务、分布式协调/通知、集群管理、Master选举、分布式锁和分布式队列；

它是一个中间件，提供协调服务

两台电脑中，插入一台新的电脑，帮助其他两台来提供服务，提供计算、缓存、传递的作用。

作用于分布式系统，发挥其优势，可以为大数据服务，支持Java，提供给Java和C语言提供了API

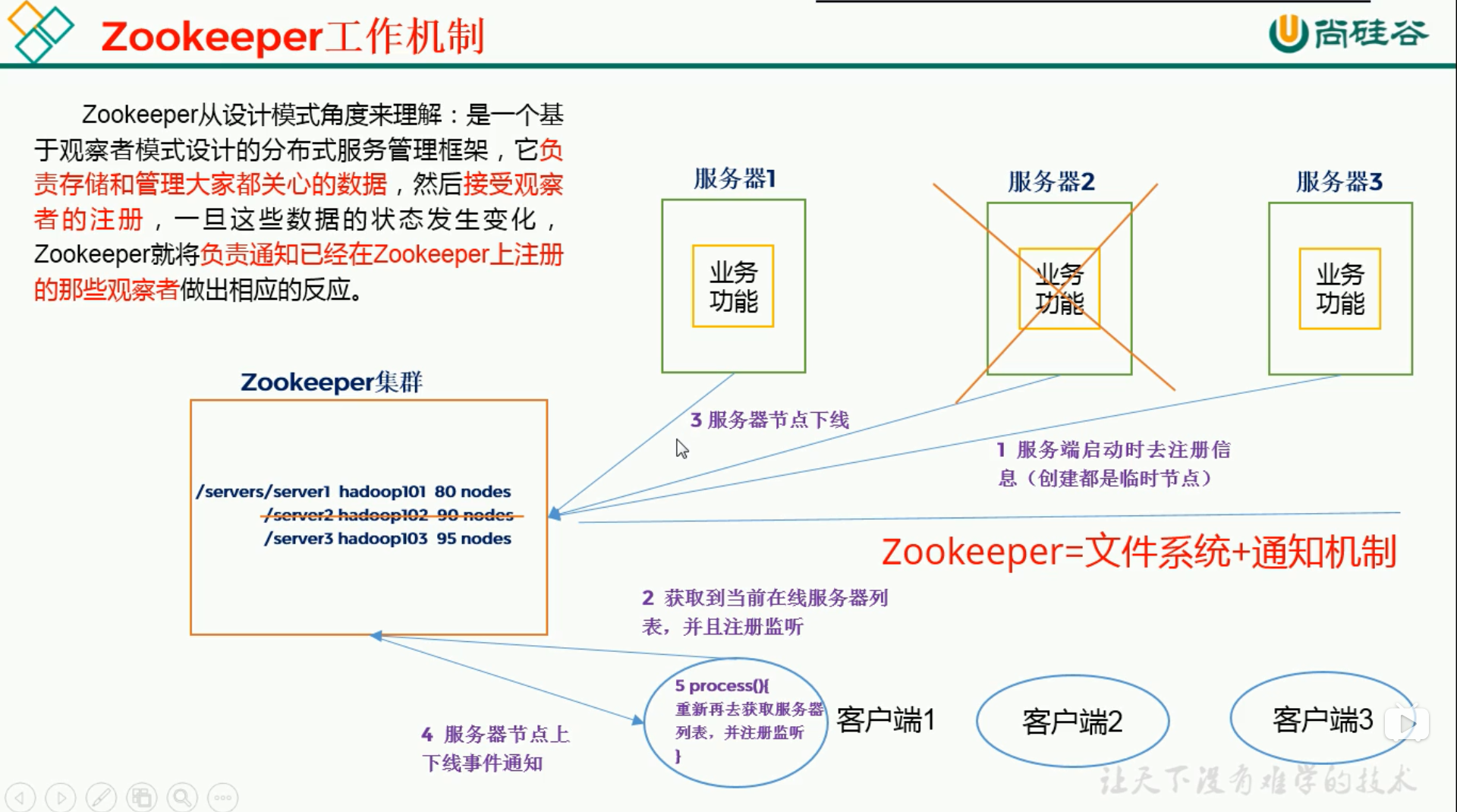
9zookeeper官网：https://zookeeper.apache.org/

zookeerper的特性

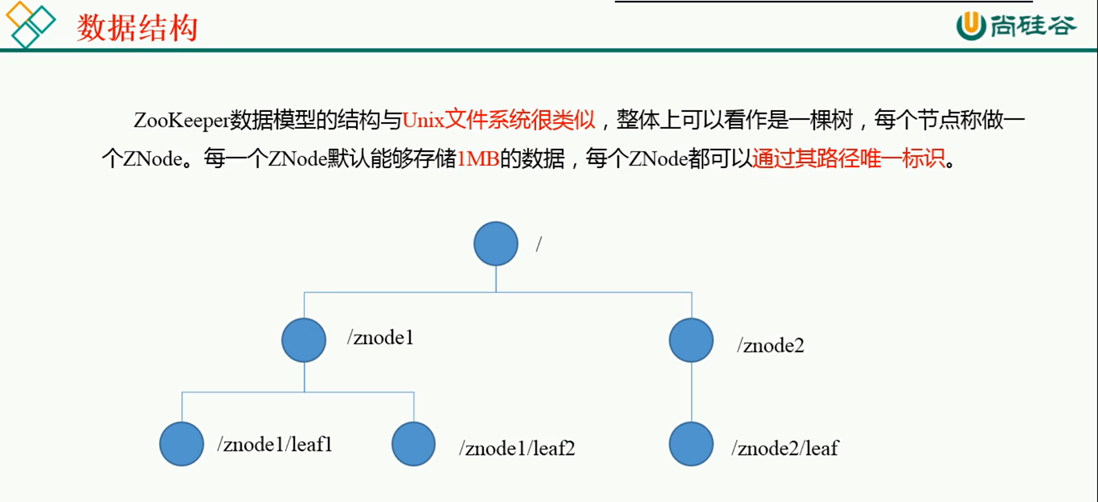
* 一致性：数据具有一致性，数据按照顺序分批入库
* 原子性：事务要么全部成功要么全部失败，不会出现数据不一致的情况
* 单一视图：客户端连接集群中的任意一个节点数据都是一致的
* 可靠性：每一次的操作的状态都可以保存在服务端
* 实时性：客户端可以读取到服务端的最新数据

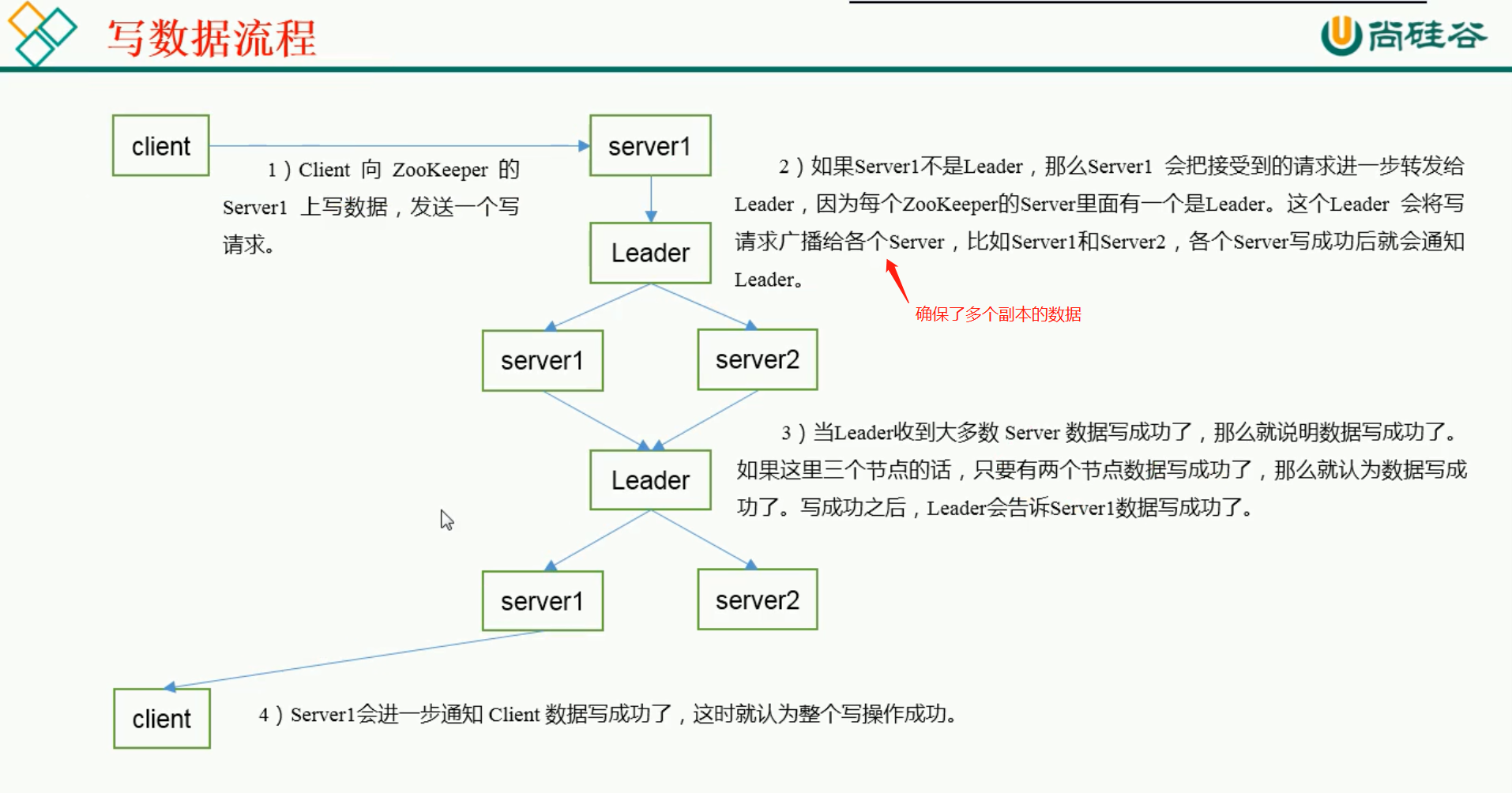
zookeeper的作用

* master节点选举，主节点挂了以后，从节点就会接手工作，并且保证这个节点是唯一的，这也是所谓首脑模式，从而保证我们的集群式高可用的
* 统一配置文件管理，只需要部署一台服务器，则可以把相同的配置文件同步更新到其他的所有服务器，云计算中使用较多（修改了就统一配置文件）









# 基本的概念

## 1.集群的角色

* Leader：领导者，选举产生，可读/可写
* Follower：跟随者，参与选举，可以被选举，只读服务
* Observer：观察者，参与选举，不可以被选举，只读服务。Observer 机器不参与 Leader 的选举过程，也不参与写操作的“过半写成功”策略，因此 Observer 机器可以在不影响写性能的情况下提升集群的读性能。

选举机制：

半数机制：集群中半数以上机器存活，集群可用，所以Zookeeper适合安装基数台服务器

Zookeeper虽然在配置文件中并没有指定leader和follower，但是在工作时，是有一个leader,其他为follower，leader是内部临时投票选举



## 2.会话（Session）的概念

Session 指的是 ZooKeeper 服务器与客户端会话。在 ZooKeeper 中，一个客户端连接是指客户端和服务器之间的一个 TCP 长连接。客户端启动的时候，首先会与服务器建立一个 TCP 连接，从第一次连接建立开始，客户端会话的生命周期也开始了。客户端 <—TCP长连接—> 服务端

通过这个连接，客户端能够通过心跳检测与服务器保持有效的会话，也能够向 Zookeeper 服务器发送请求并接受响应，同时还能够通过该连接接收来自服务器的 Watch 事件通知。

Session 的 sessionTimeout 值用来设置一个客户端会话的超时时间。当由于服务器压力太大、网络故障或是客户端主动断开连接等各种原因导致客户端连接断开时，只要在 sessionTimeout 规定的时间内能够重新连接上集群中任意一台服务器，那么之前创建的会话仍然有效。

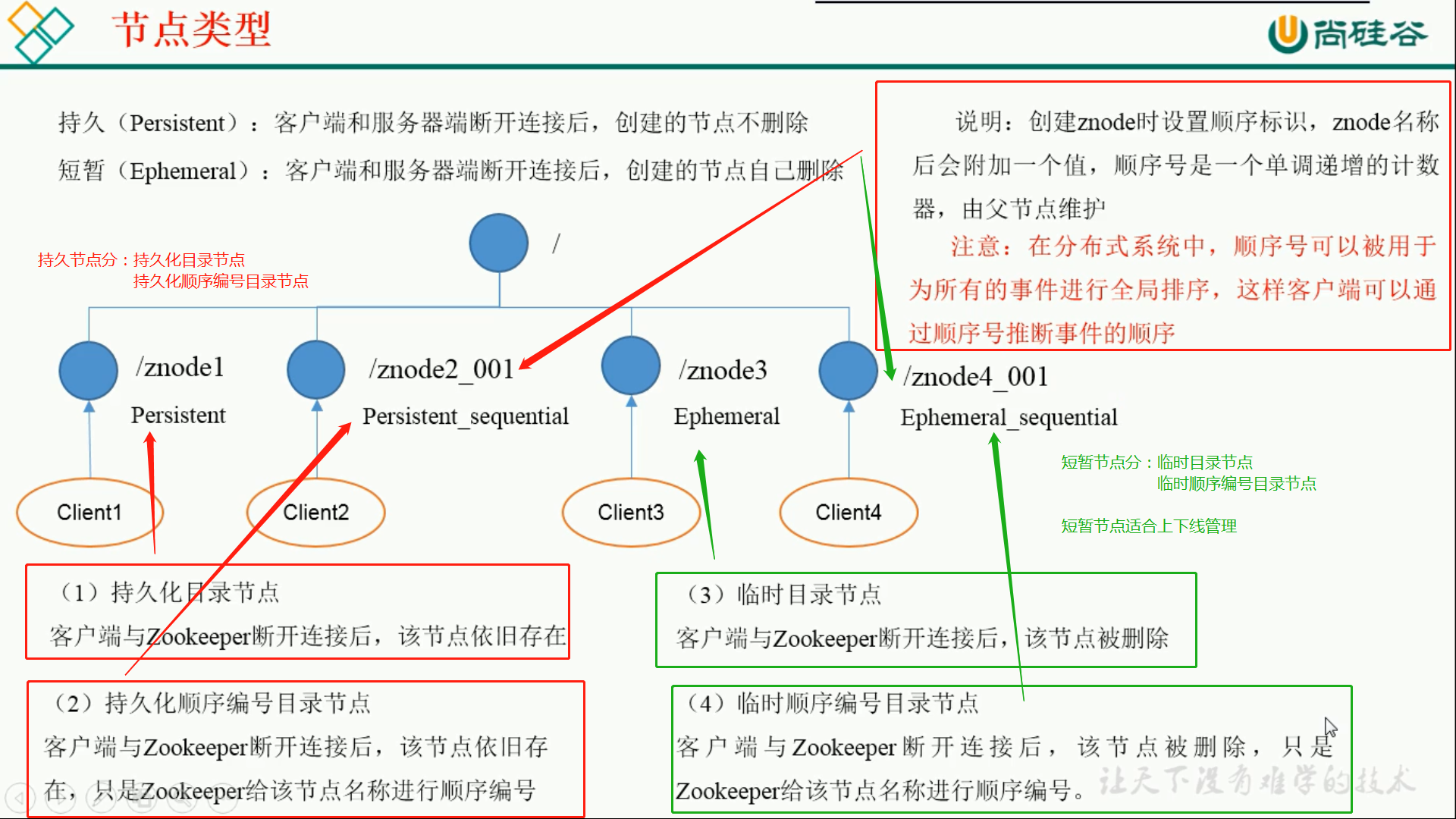
在为客户端创建会话之前，服务端首先会为每个客户端都分配一个 sessionID。由于 sessionID 是 Zookeeper 会话的一个重要标识，许多与会话相关的运行机制都是基于这个 sessionID 的。因此，无论是哪台服务器为客户端分配的 sessionID，都务必保证全局唯一。

## 3.数据节点

ZNode:zk数据模型中的数据单元，zk的数据都存储与内存之中，数据模型为树状结构（ZNode Tree），每一个ZNode都会保存自己的数据在内存中

持久节点：仅显示删除才消失，客户端和服务器断开连接后，创建的节点不删除

临时节点：会话终止自动消失，客户端和服务器端口连接后，创建的节点自己删除



## 4.版本

zk会为每一个ZNode维护一个称为stat的数据结构，记录了当前ZNode的三个数据版本

VERSION：当前的版本

CVERSION：当前ZNode的子节点的版本

AVERSION：当前ZNode的ACL版本

## 5.ACL

ZK使用ACL机制进行权限控制

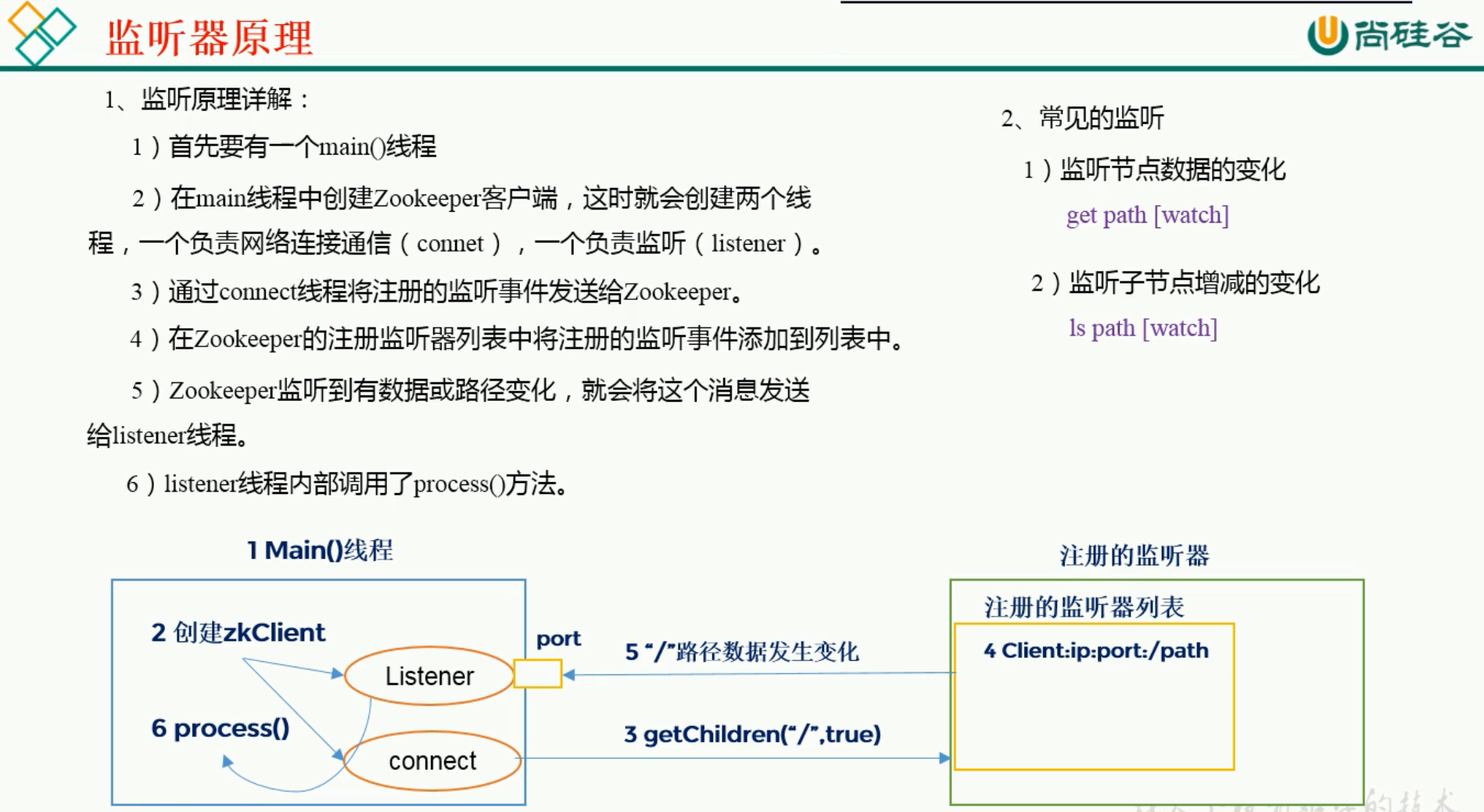
CREATE、READ、WRITE、DELETE、ADMIN

## 6.事件监听器(Watcher)

Watcher（事件监听器），是 ZooKeeper 中的一个很重要的特性。

ZooKeeper 允许用户在指定节点上注册一些 Watcher，并且在一些特定事件触发的时候，ZooKeeper 服务端会将事件通知到感兴趣的客户端上去，该机制是 ZooKeeper 实现分布式协调服务的重要特性。

ZK上，由用户指定的触发机制，在某一些事件产生时候，ZK能够将通知给相关的客户端



## 7.ZAB协议

Zookeeper Atomic Broadcast，ZK原子广播协议

ZAB协议中存在的三种状态

（1）Looking

（2）Following

（3）Leading

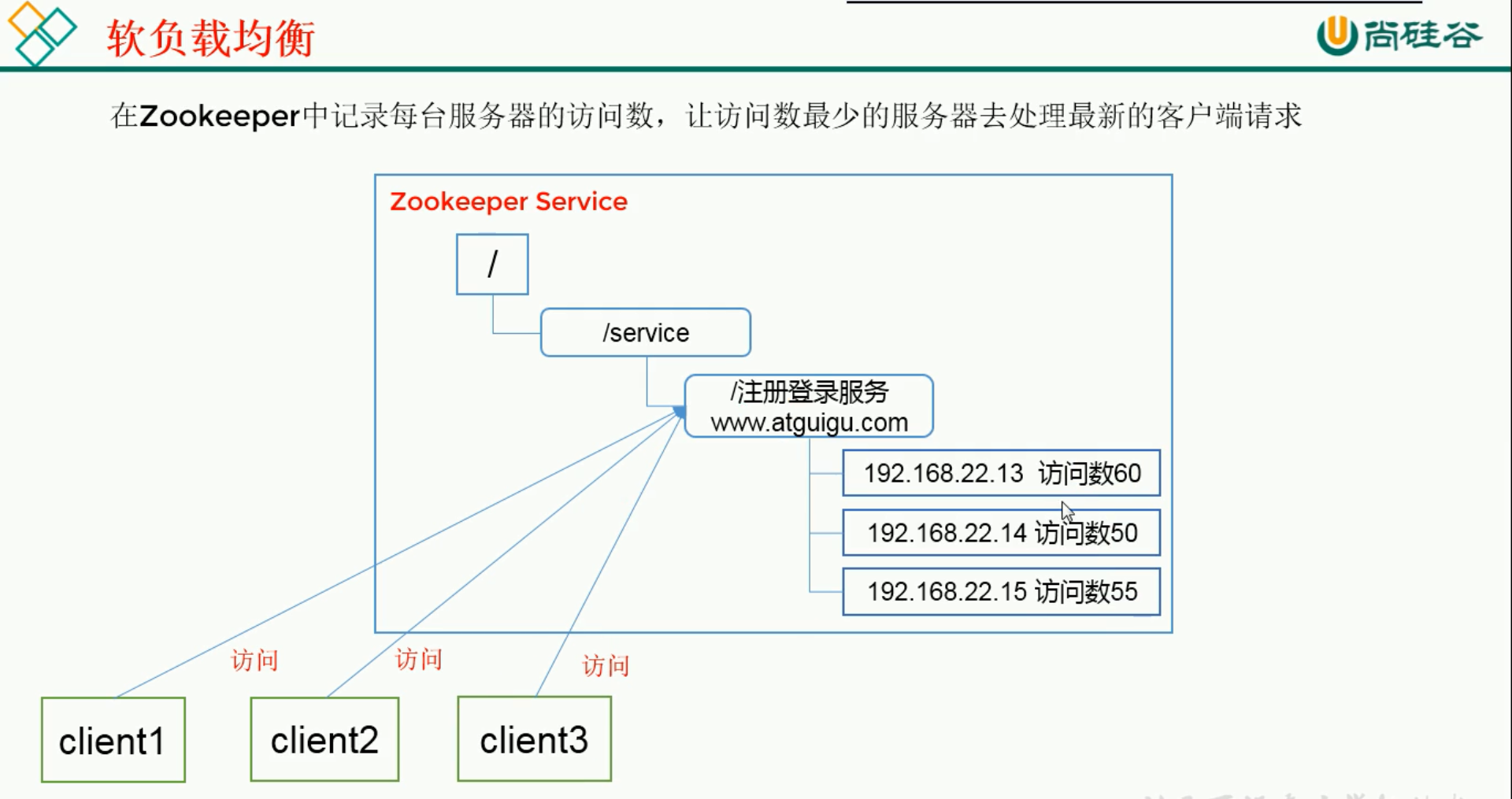
四个阶段：选举election、发现discovery、同步sync、广播Broadcast

ZAB 协议包括两种基本的模式，分别是崩溃恢复和消息广播。

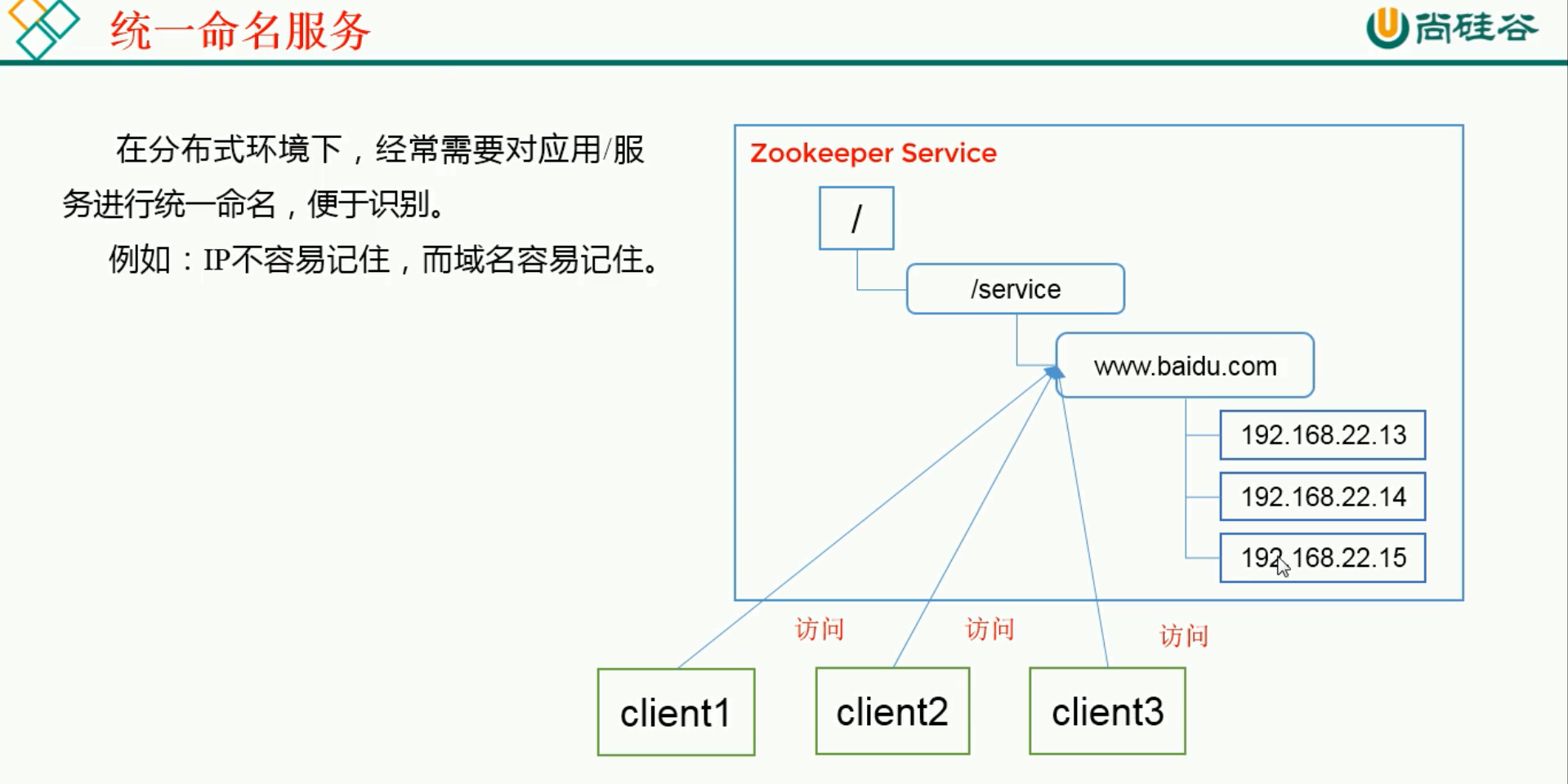
* 当整个服务框架在启动过程中，或是当 Leader 服务器出现网络中断、崩溃退出与重启等异常情况时，ZAB 协议就会进入恢复模式并选举产生新的 Leader 服务器。
* 当选举产生了新的 Leader 服务器，同时集群中已经有过半的机器与该 Leader 服务器完成了状态同步之后，ZAB 协议就会退出恢复模式。
* 其中，所谓的状态同步是指数据同步，用来保证集群中存在过半的机器能够和 Leader 服务器的数据状态保持一致。
* 当集群中已经有过半的 Follower 服务器完成了和 Leader 服务器的状态同步，那么整个服务框架就可以进人消息广播模式了。
* 当一台同样遵守 ZAB 协议的服务器启动后加入到集群中时，如果此时集群中已经存在一个 Leader 服务器在负责进行消息广播。那么新加入的服务器就会自觉地进人数据恢复模式：找到 Leader 所在的服务器，并与其进行数据同步，然后一起参与到消息广播流程中去。
* 正如上文介绍中所说的，ZooKeeper 设计成只允许唯一的一个 Leader 服务器来进行事务请求的处理。
* Leader 服务器在接收到客户端的事务请求后，会生成对应的事务提案并发起一轮广播协议。
* 而如果集群中的其他机器接收到客户端的事务请求，那么这些非 Leader 服务器会首先将这个事务请求转发给 Leader 服务器。

## 8. 典型应用场景

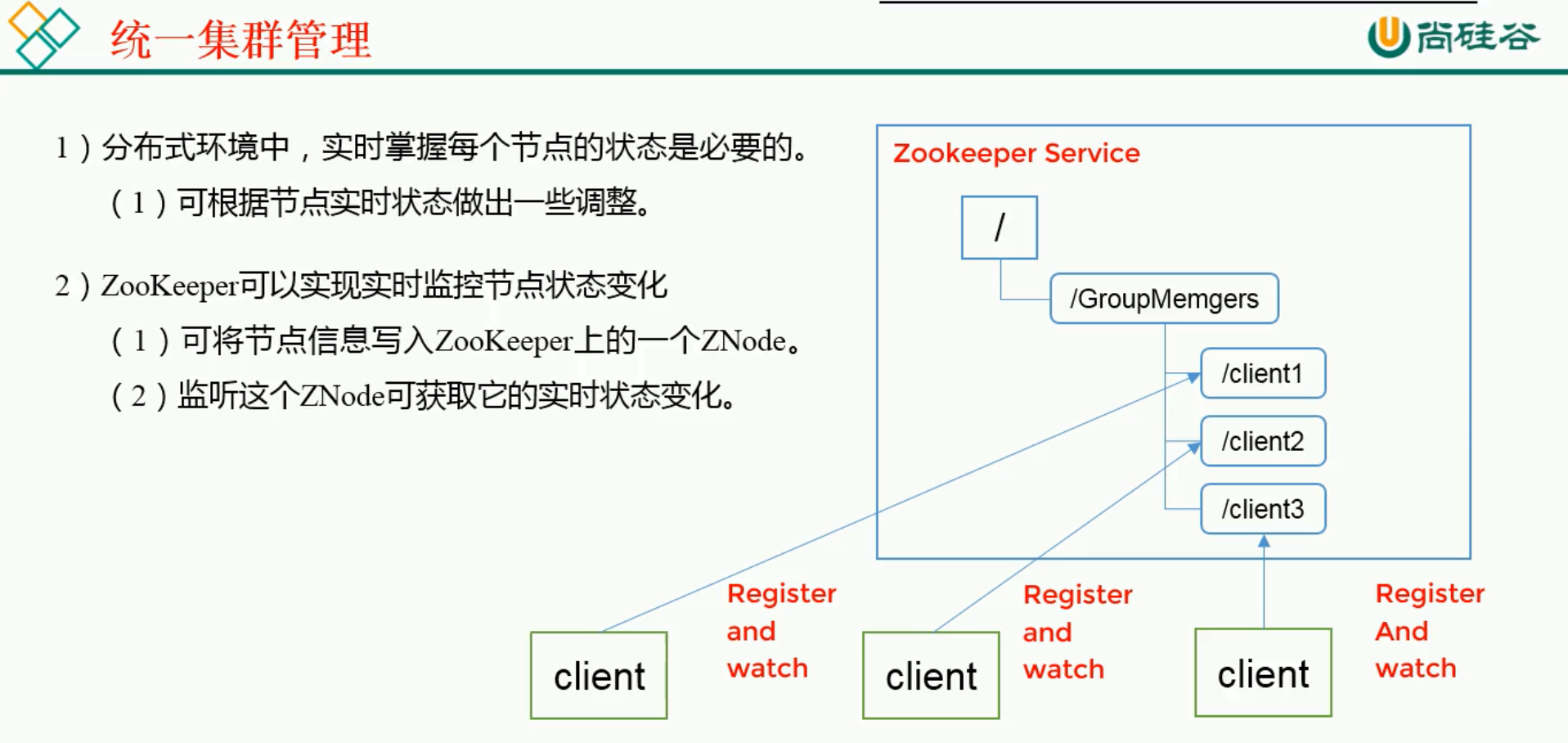
* 数据发布/订阅
* 分布式协调/通知
* Master选举
* 负载均衡



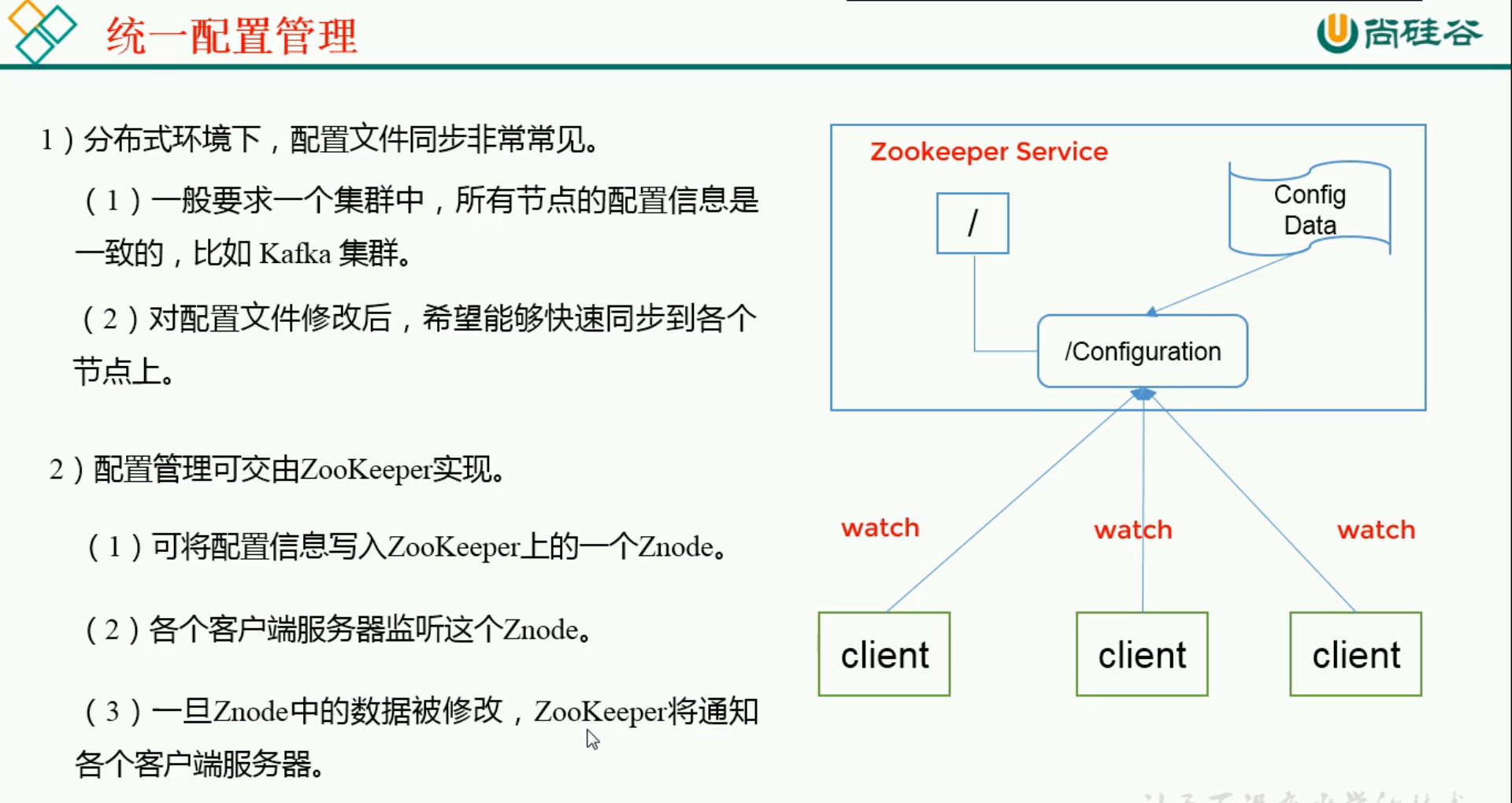
* 命名服务



* 集群管理

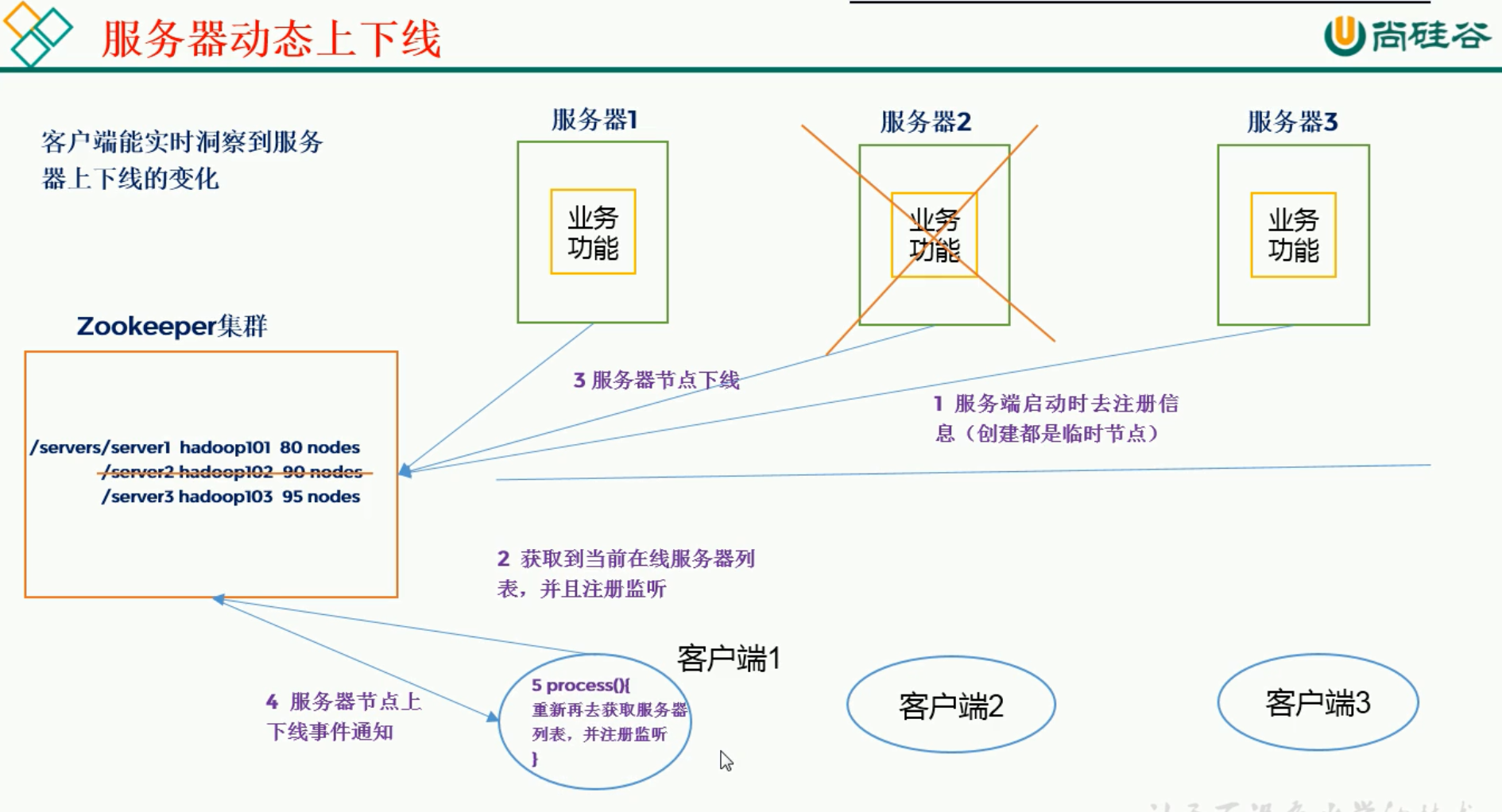


* 统一配置管理：



Kafka本身集成zookeeper

* 服务器动态上下线



## 9.ACL

ZooKeeper 采用 ACL（AccessControlLists）策略来进行权限控制，类似于 UNIX 文件系统的权限控制。

ZooKeeper 定义了 5 种权限，如下：

Create：创建子节点的权限。

Read：获取节点数据和子节点列表的权限。

Write：更新节点数据的权限。

Delete：删除子节点的权限

Admin：设置节点ACL的权限

其中尤其需要注意的是，CREATE 和 DELETE 这两种权限都是针对子节点的权限控制。

# Zookeeper实战安装

## 1-步骤一：安装Zookeeper

1. **编辑/etc/hosts ,**

所有集群主机可以相互 ping 通（在nn01上面配置，同步到node1，node2，node3）

[root@nn01 hadoop]# vim /etc/hosts

192.168.1.241 node1

192.168.1.242 node2

192.168.1.243 node3

[root@nn01 hadoop]# for i in {22..24} do \

scp /etc/hosts 192.168.1.$i:/etc/ done //同步配置

hosts 100% 253 639.2KB/s 00:00

hosts 100% 253 497.7KB/s 00:00

hosts 100% 253 662.2KB/s 00:00

**2）安装 java-1.8.0-openjdk-devel,**

由于之前的hadoop上面已经安装过，这里不再安装，若是新机器要安装

1. **zookeeper 解压拷贝到 安装路径**

/usr/local/zookeeper

[root@nn01 ~]# tar -xf zookeeper-3.4.10.tar.gz

[root@nn01 ~]# mv zookeeper-3.4.10 /usr/local/zookeeper

**4）配置文件改名，并在最后添加配置**

[root@nn01 ~]# cd /usr/local/zookeeper/conf/

[root@nn01 conf]# ls

configuration.xsl log4j.properties zoo\_sample.cfg

[root@nn01 conf]# mv zoo\_sample.cfg zoo.cfg

[root@nn01 conf]# chown root.root zoo.cfg #赋予权限

[root@nn01 conf]# vim zoo.cfg

-----------------------------------------------------------------------------------------------------------------------

## 2-zoo.cfg配置说明：

tickTime=2000 #2s -用于计算的时间单元，例如session超时：N\* tickTime

initLimit=10 #用于集群允许从节点连接并且同步到leader节点的初始化连接时间，以tickTime的倍数来表示

syncLimit=5 #用于集群，leader主节点与从节点之间发送消息，请求和应答的时间长度

dataDir=/usr/local/zookeeper/data #数据存储路径为/usr/local/zookeeper/data -必须配置

dataLogDir=/usr/local/zookeeper/data/log #日志存储路径，如果不配置就会和dataDir共用

clientPort=2181 #连接服务器的端口

server.1= 192.168.10.241:2888:3888

server.2= 192.168.10.242:2888:3888

server.3= 192.168.10.243:2888:3888

server.id=[hostname]:port:port[:observer] #集群配置

第一个port：follower与leader进行通信和数据同步时所使用端口；只有leader上才会开启

第二个port：leader选举时使用的端口；

observer：定义指定的服务器为observer；

-----------------------------------------------------------------------------------------------------------------------------

**5）拷贝 /usr/local/zookeeper 到其他集群主机**

[root@nn01 conf]# for i in {241..243}; do rsync -aSH --delete /usr/local/zookeeper/ 192.168.1.$i:/usr/local/zookeeper -e 'ssh' & done

[4] 4956

[5] 4957

[6] 4958

**6）创建数据目录**

每一台都要操作 mkdir /usr/local/zookeeper/data，

[root@nn01 conf]# mkdir /usr/local/zookeeper/data

[root@nn01 conf]# ssh node1 mkdir /usr/local/zookeeper/data

[root@nn01 conf]# ssh node2 mkdir /usr/local/zookeeper/data

[root@nn01 conf]# ssh node3 mkdir /usr/local/zookeeper/data

**7）创建 myid 文件**

id 必须与配置文件里主机名对应的 server.(id) 一致

[root@nn01 conf]# echo 4 >/tmp/zookeeper/myid

[root@nn01 conf]# ssh node1 'echo 1 >/usr/local/zookeeper/data/myid'

[root@nn01 conf]# ssh node2 'echo 2 >/usr/local/zookeeper/data/myid'

[root@nn01 conf]# ssh node3 'echo 3 >/usr/local/zookeeper/data/myid'

**8）启动服务**

单启动一台无法查看状态，需要启动全部集群以后才能查看状态，每一台上面都要手工启动（以nn01为例子）

[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

注意：刚启动zookeeper查看状态的时候报错，启动的数量要保证半数以上，这时再去看就成功了

**9）查看状态**

[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Mode: observe

[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh stop #关闭之后查看状态其他服务器的角色

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Stopping zookeeper ... STOPPED

[root@nn01 conf]# yum -y install telnet

[root@nn01 conf]# telnet node3 2181

Trying 192.168.1.24...

Connected to node3.

Escape character is '^]'.

ruok #发送

imokConnection closed by foreign host. #imok回应的结果

## 3-10）利用 api 查看状态（nn01上面操作）

[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start

[root@nn01 conf]# vim api.sh

#!/bin/bash

function getstatus(){

exec 9<>/dev/tcp/$1/2181 2>/dev/null

echo stat >&9

MODE=$(cat <&9 |grep -Po "(?<=Mode:).\*")

exec 9<&-

echo ${MODE:-NULL}

}

for i in node{1..3} nn01;do

echo -ne "${i}\t"

getstatus ${i}

done

[root@nn01 conf]# chmod 755 api.sh

[root@nn01 conf]# ./api.sh

node1 follower

node2 leader

node3 follower

nn01 observer

## 4-客户端登录操作：

[root@DB bin]# ./zkCli.sh #进入安装目录的bin目录下，执行./zkCli.sh

Connecting to localhost:2181

[zk: localhost:2181(CONNECTED) 0] help #help查看所有命令集合

ZooKeeper -server host:port cmd args

connect host:port

get path [watch]

ls path [watch]

set path data [version]

rmr path

delquota [-n|-b] path

quit

printwatches on|off

create [-s] [-e] path data acl

stat path [watch]

close

ls2 path [watch]

history

listquota path

setAcl path acl

getAcl path

sync path

redo cmdno

addauth scheme auth

delete path [version]

setquota -n|-b val path

[zk: localhost:2181(CONNECTED) 1] stat

[zk: localhost:2181(CONNECTED) 2] ls /

[dev-web, dev-rest, test-rest, chit-web1, dubbo, chit-rest, zookeeper, leader, chit-web, prod-rest, chit-rest1, chit, test-web, prod-web]