# 任务

1. 安装zookeeper
2. 练习zookeeper命令

# 面试重点

## zookeeper是干什么的？

Zookeeper 是 分布式协调服务，分布式应用程序可以基于它实现同步服务，配置维护和命名服务等

## 说说zookeeper半数机制

举例说明：3台和4台的区别。

## zookeeper节点类型

Znode有两种类型：

短暂（ephemeral）（断开连接自己删除）

持久（persistent）（断开连接不删除）

Znode有四种形式的目录节点（默认是persistent ）

PERSISTENT

PERSISTENT\_SEQUENTIAL（持久序列/test0000000019 ）

EPHEMERAL

EPHEMERAL\_SEQUENTIAL

创建znode时设置顺序标识，znode名称后会附加一个值

顺序号是一个单调递增的计数器，由父节点维护

在分布式系统中，顺序号可以被用于为所有的事件进行全局排序，这样客户端可以通过顺序号推断事件的顺序

## zookeeper选举机制

### zookeeper的选举机制（全新集群）

以一个简单的例子来说明整个选举的过程.  
假设有五台服务器组成的zookeeper集群,它们的id从1-5,同时它们都是最新启动的,也就是没有历史数据,在存放数据量这一点上,都是一样的.假设这些服务器依序启动,来看看会发生什么.  
1) 服务器1启动,此时只有它一台服务器启动了,它发出去的报没有任何响应,所以它的选举状态一直是LOOKING状态  
2) 服务器2启动,它与最开始启动的服务器1进行通信,互相交换自己的选举结果,由于两者都没有历史数据,所以id值较大的服务器2胜出,但是由于没有达到超过半数以上的服务器都同意选举它(这个例子中的半数以上是3),所以服务器1,2还是继续保持LOOKING状态.  
3) 服务器3启动,根据前面的理论分析,服务器3成为服务器1,2,3中的老大,而与上面不同的是,此时有三台服务器选举了它,所以它成为了这次选举的leader.  
4) 服务器4启动,根据前面的分析,理论上服务器4应该是服务器1,2,3,4中最大的,但是由于前面已经有半数以上的服务器选举了服务器3,所以它只能接收当小弟的命了.  
5) 服务器5启动,同4一样,当小弟.

### 非全新集群的选举机制(数据恢复)

那么，初始化的时候，是按照上述的说明进行选举的，但是当zookeeper运行了一段时间之后，有机器down掉，重新选举时，选举过程就相对复杂了。

需要加入数据id、leader id和逻辑时钟。

数据id：数据新的id就大，数据每次更新都会更新id。

Leader id：就是我们配置的myid中的值，每个机器一个。

~~逻辑时钟：这个值从0开始递增,每次选举对应一个值,也就是说:  如果在同一次选举中,那么这个值应该是一致的 ;  逻辑时钟值越大,说明这一次选举leader的进程更新.~~

选举的标准就变成：

~~1、逻辑时钟小的选举结果被忽略，重新投票~~

2、统一逻辑时钟后，数据id大的胜出

3、数据id相同的情况下，leader id大的胜出

根据这个规则选出leader。

## 说说共享锁

共享锁在同一个进程中很容易实现，但是在跨进程或者在不同 Server 之间就不好实现了。

Zookeeper 却很容易实现这个功能，实现方式也是需要获得锁的 Server 创建一个 EPHEMERAL\_SEQUENTIAL 目录节点，然后调用 getChildren方法获取当前的目录节点列表中最小的目录节点是不是就是自己创建的目录节点，如果正是自己创建的，那么它就获得了这个锁，如果不是那么它就调用 exists(String path, boolean watch) 方法并监控 Zookeeper 上目录节点列表的变化，一直到自己创建的节点是列表中最小编号的目录节点，从而获得锁，释放锁很简单，只要删除前面它自己所创建的目录节点就行了。

应用于选主