Zookeeper **安其内（quorum仲裁），攘其外。【CP原理 一致性和容错性】**

1角色 leader，follower , observer (观察者，需要在启动时进行配置，只进行查询和创建节点，不进行选举)

2 常用命令 zkServer.sh start / stop /status

Ls / set /delete /sync/ [watch 进行监控]

3 watcher 监视和通知 ，当节点信息变化时，能进行检测到变化，并通知客户端。（只运行一次回调函数）

4 创建节点大致流程，具体广播流程。【借助于消息队列数据】

4.1. 客户端发起一个写操作请求   
4.2. Leader服务器将客户端的request请求转化为事物proposql提案，同时为每个proposal分配一个全局唯一的ID，即ZXID。   
4.3. leader服务器与每个follower之间都有一个队列，leader将消息发送到该队列   
4.4. follower机器从队列中取出消息处理完(写入本地事物日志中)毕后，向leader服务器发送ACK确认。   
4.5. leader服务器收到半数以上的follower的ACK后，即认为可以发送commit   
4.6. leader向所有的follower服务器发送commit消息。

5 利用队列数据结构进行数据同步，每个flower 都有一个消息队列【操作事务】，flower消费消息队列里面的信息进行创建节点等操作来保持和主节点的一致性。

6 内部选主

1. 选主算法 先根据zxid进行选择，先选zxid大的，若zxid一致，再根据sid进行选择，sid大的获胜。

Zxid:【epoch(任期[iːpɒk])、counter】

1. 主节点和从节点都会进行心跳机制，当从节点（收、发）心跳时会重置时间闹钟，若在该规定时间内为收到心跳，则认为主节点失效，会重新进行选主。并更新新任选主的任期。
2. 假设老master恢复，会根据任期数进行判断，小于现有最大任期，则进行数据同步变为follower节点,根据uptodate进行差值化更新。

7 服务器为奇数的原因?

允许最大宕机的机器数在奇数节点和偶数节点是一致的，所以偶数个性价比不高。

8 zookeeper 实现分布式锁？

9 paxos -> raft -> zab（半数以上反馈） ？ 仲裁算法

Raft 协议首先提出了HW的概念。水位模型（确认commit）

Zab 保证只要有一台服务器提交了proposal，就要保证所有服务器最终都能正确提交proposal。实现最终一致性。

ZAB协议崩溃恢复要求满足如下2个要求：   
1. 确保已经被leader提交的proposal必须最终被所有的follower服务器提交。   
2. 确保丢弃已经被leader出的但是没有被提交的proposal。