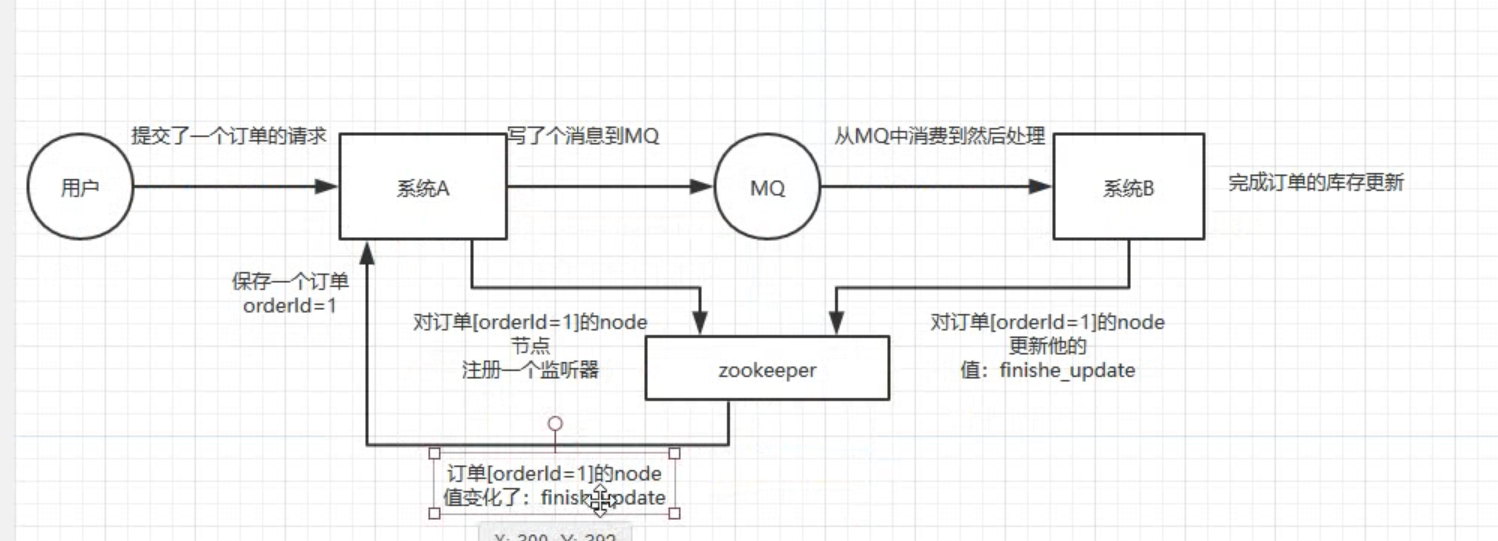
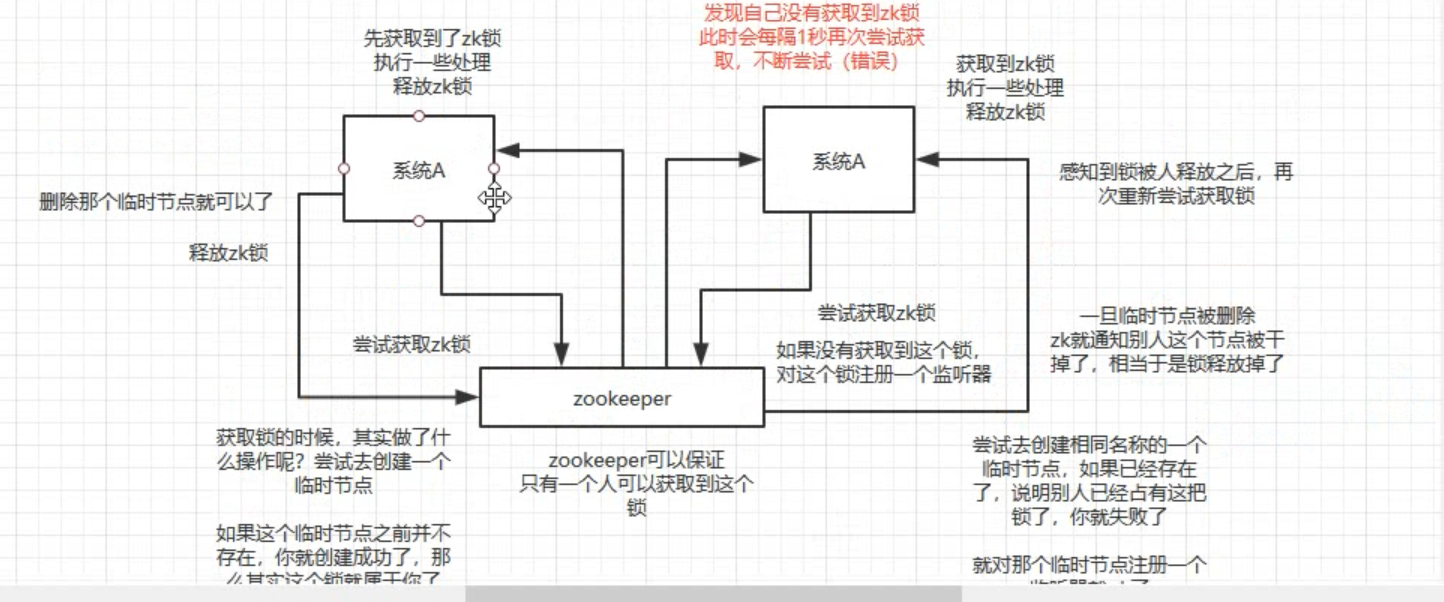
ZK的使用场景

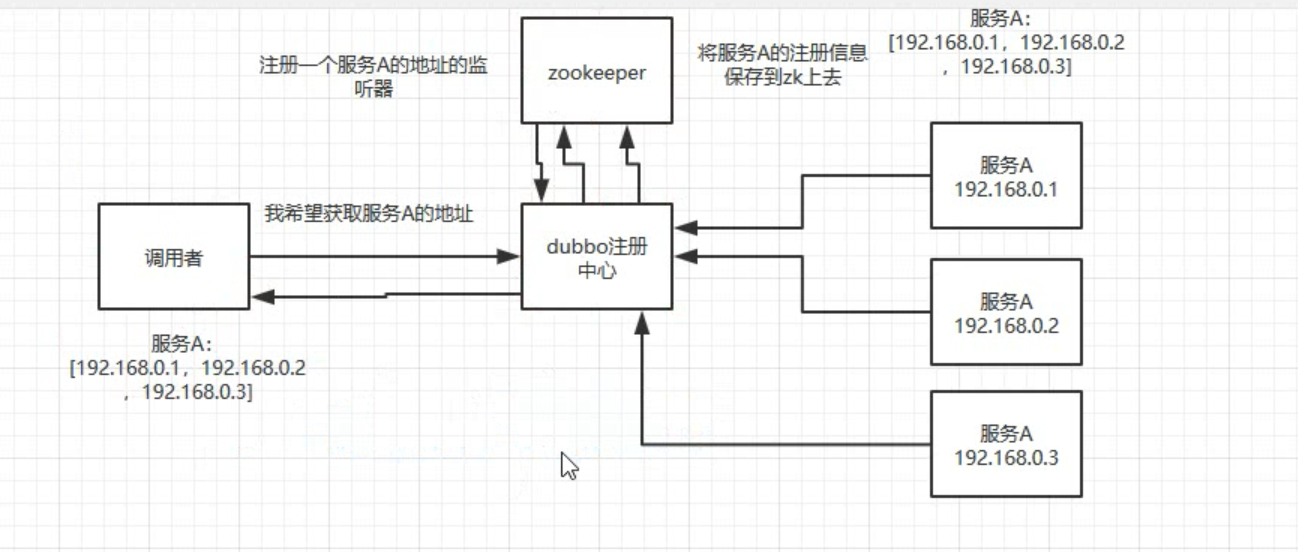
1. 分布式协调场景：



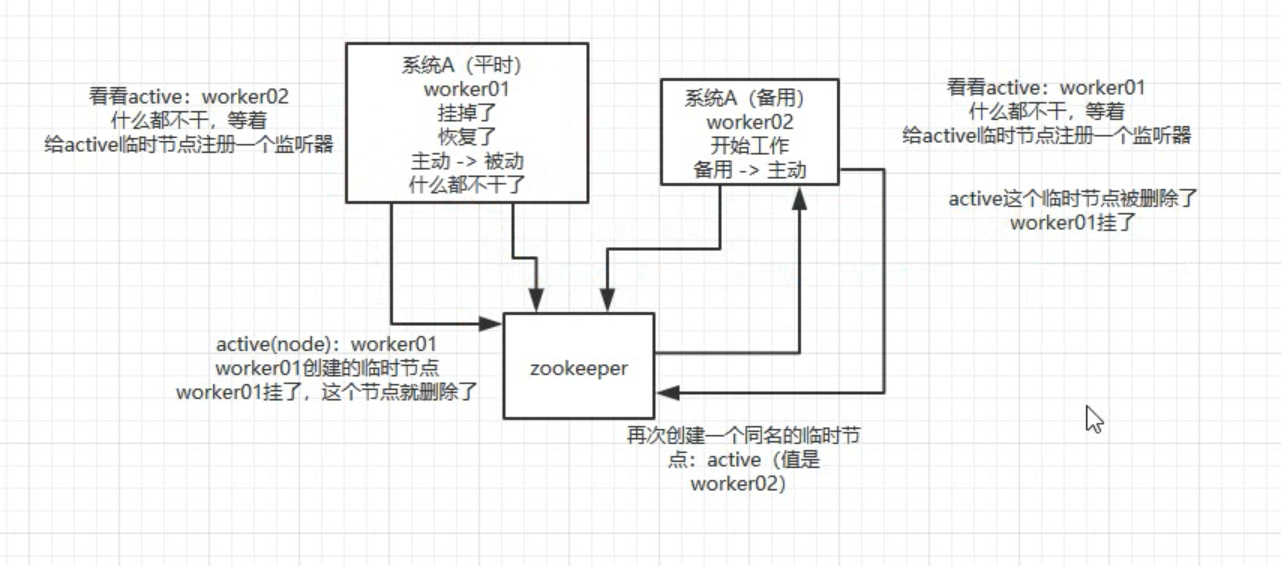
1. 分布式锁场景



1. 配置信息管理



1. HA高可用性

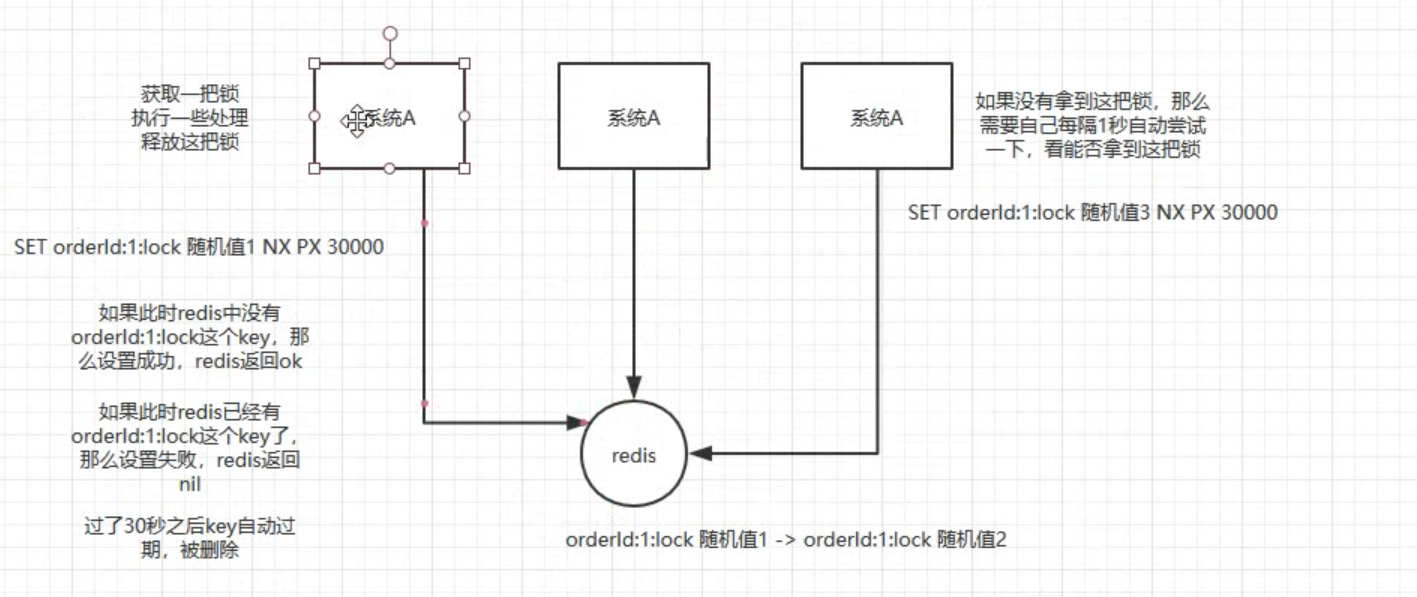


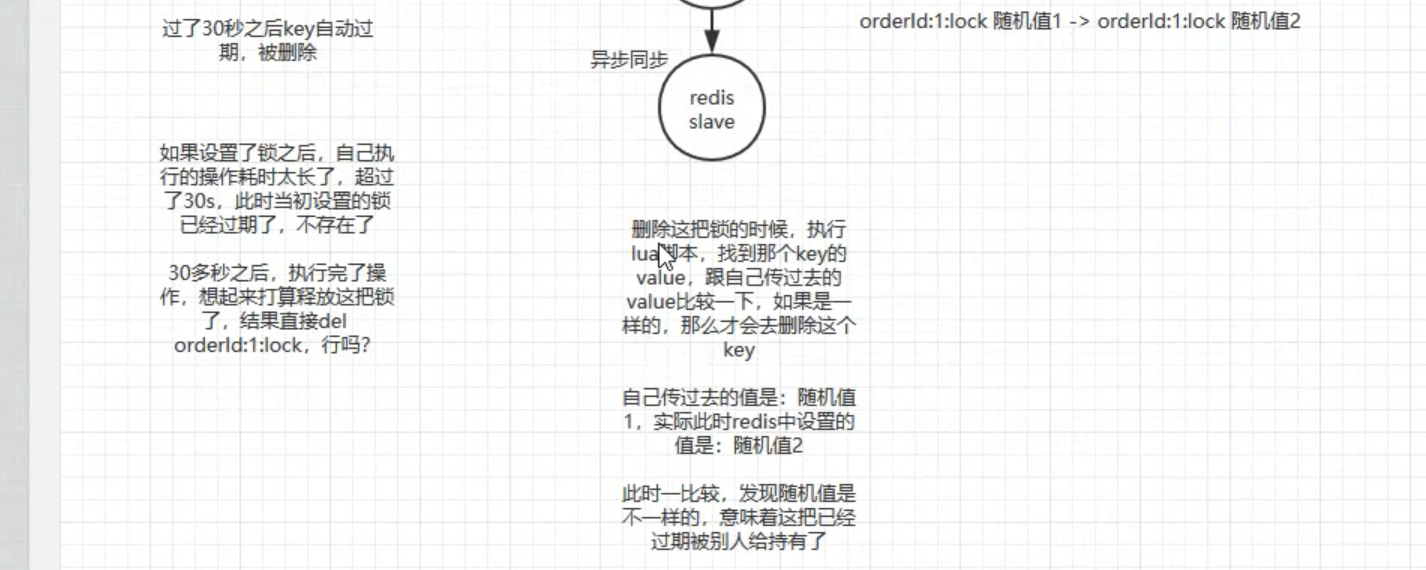
1. ZKdemo（重点）

分布式锁：

Redis：

最简单的SetNx px





Redlock(作用在redis集群中，一般不使用，麻烦而且不怎么严谨)

1. 获取当前时间戳 单位是毫秒
2. 轮流尝试在每个master上建立锁，过期时间较短
3. 在过半的节点上建锁成功即可，同时满足建锁时间小于超时时间
4. 如果建立失败 就使用LUA都删除

ZK分布式锁：

临时节点的名字

创建临时节点

结合监听器+countdownlunch

临时顺序节点实现

1. 多个系统调用同一个资源时，会生成多个临时的节点，根据编号会有一个排序
2. 判断是不是最小的节点 是的话就获取锁
3. 不是最小的节点的话 就将比自己小1的点 设置为监听节点
4. 某个节点释放锁 那么zk就会给排在自己后面的节点通知，同时countdown-1
5. Countdownlunch激活，继续执行下一步==获取锁

分布式session方案：(百度一下)

1、Tomcat+redis(优点：方便，缺点：与Tomcat绑的太死)

在Tomcat中配置redissessionmanager

Session写的时候放入redis 读的时候从redis中取

2、spring+redis 将配置转移到spring中了

3、springboot+redis 更简洁了