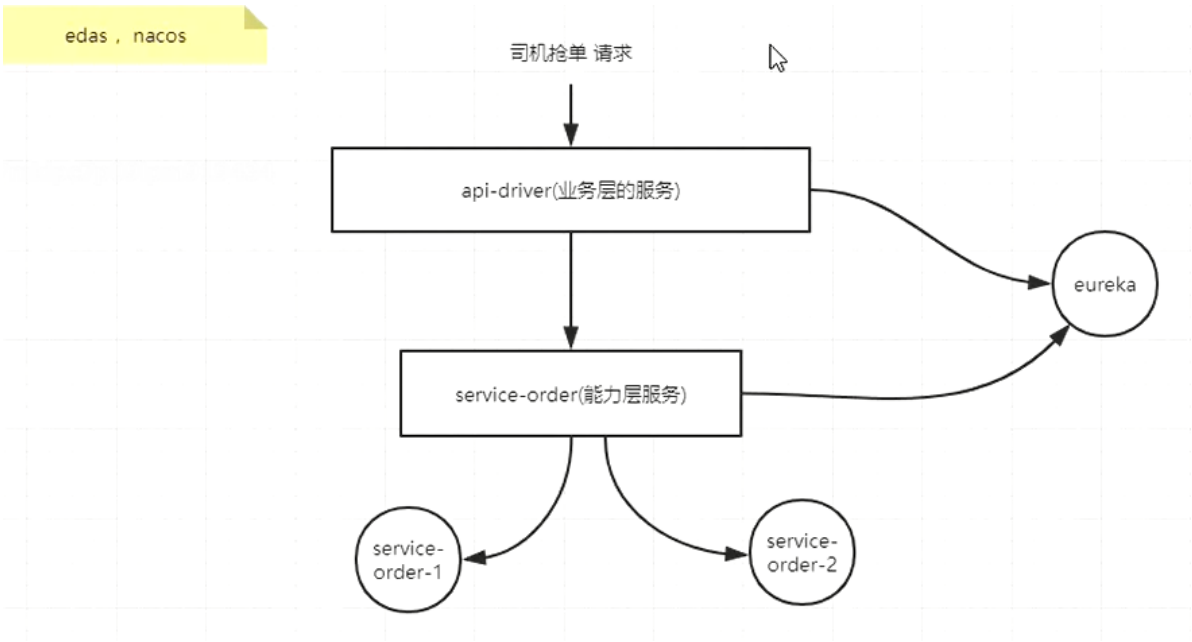
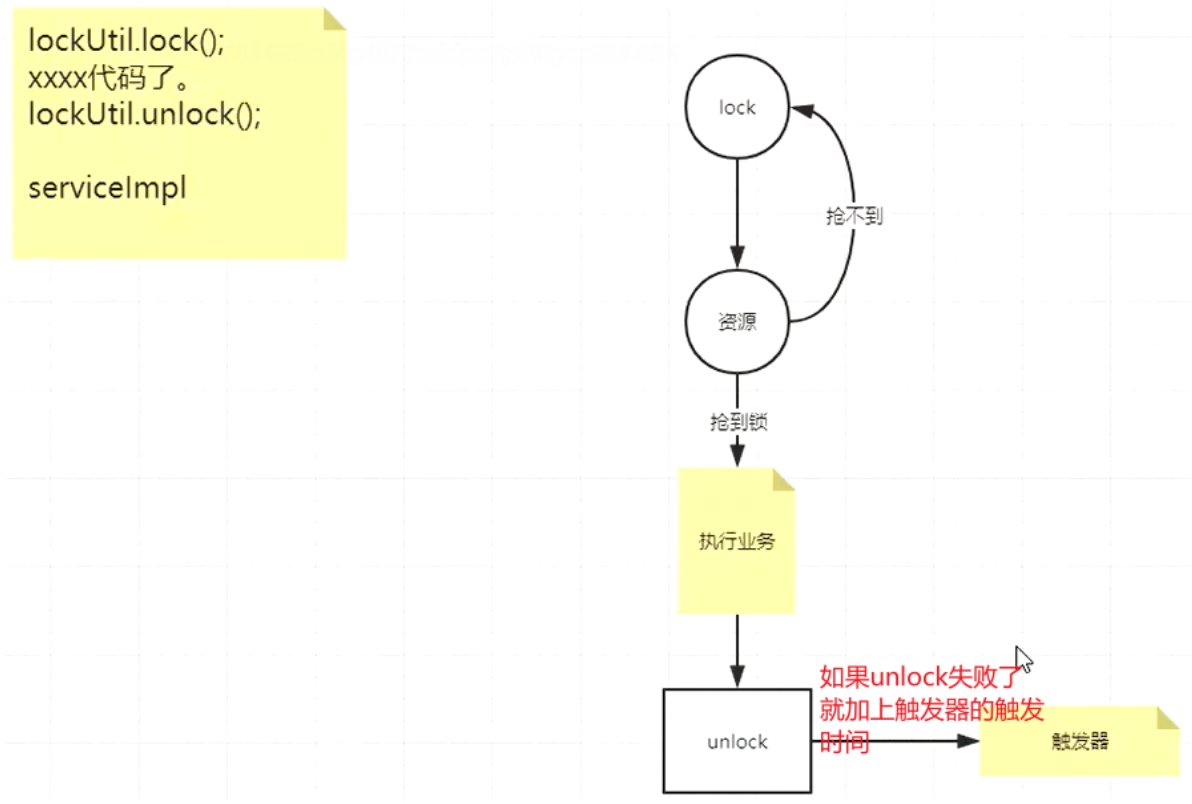
分布式锁

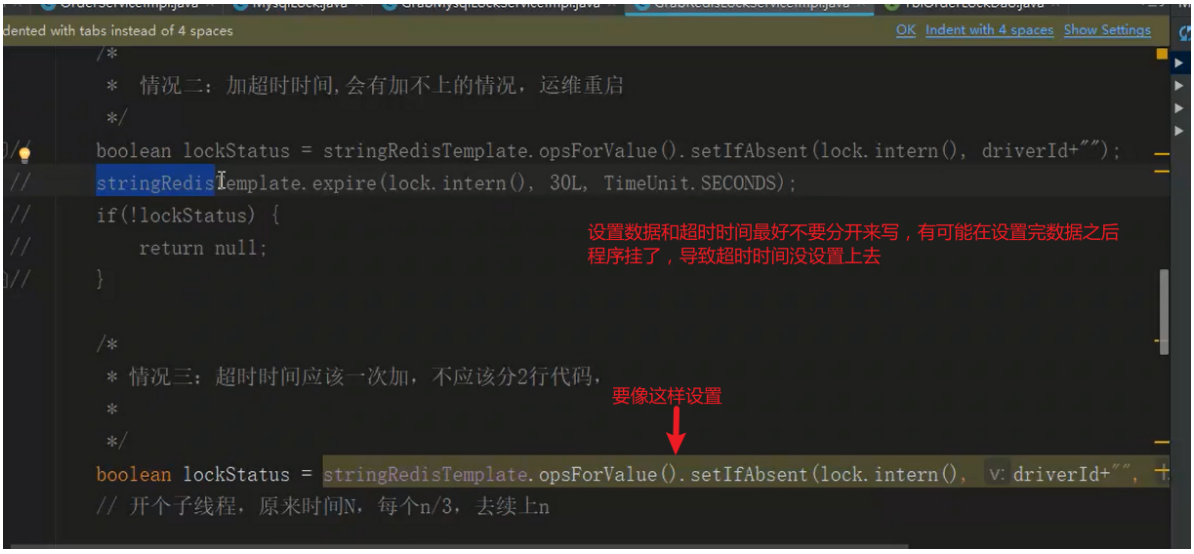
JVM是解决不了分布式集群里的锁的

分布式锁场景



如果不用分布式锁的控制的话就会产生两个司机抢一个订单的情况



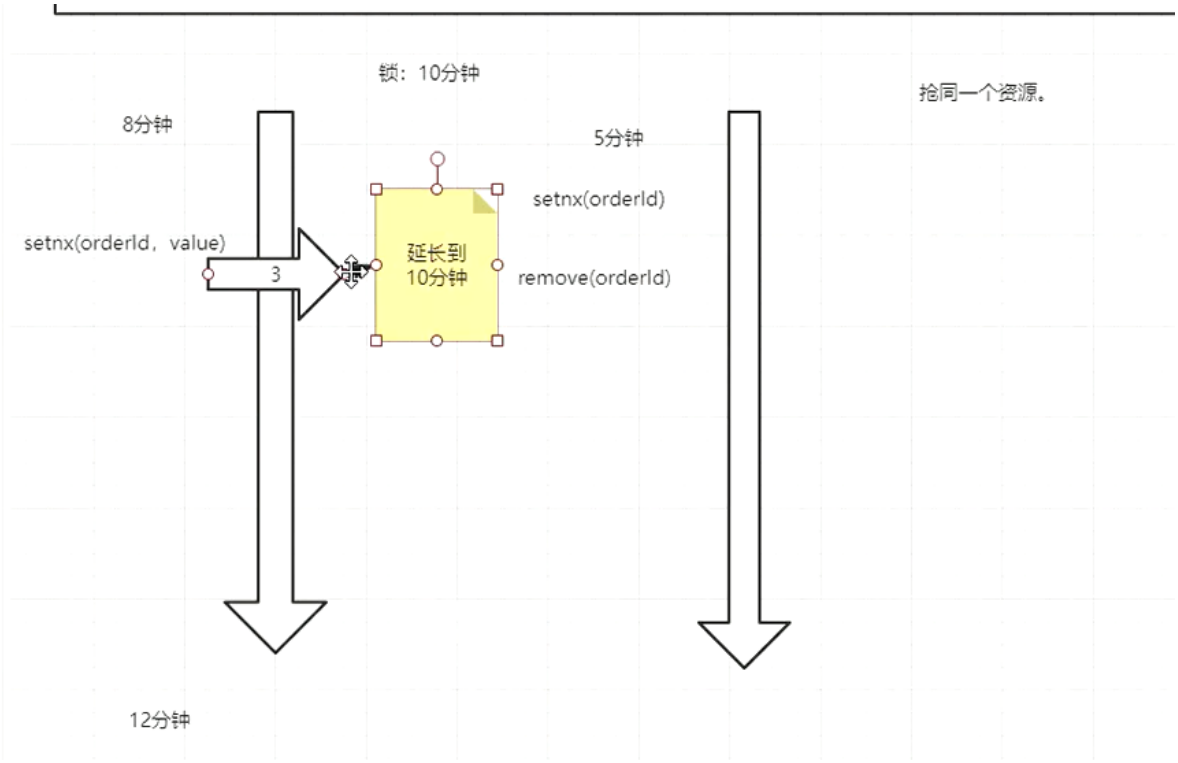


程序的执行是在锁的有效期内的， 如果锁的有效期是10分钟，程序8分钟执行之后会把锁放掉，这个锁就被删除了。如果8分钟的程序执行了12分钟，锁也过期了。如果这时候另外一个程序进来了也是锁同样的订单，他执行了5分钟，这时候12分钟的锁刚刚要去释放的时候，把才执行了5分钟的key给释放掉remove了，两个id为什么一样？因为要抢同一个订单，同一个资源，就有可能发生这个情况

谁加的锁谁释放自己的锁

可以在value中设置自己的值

还有一个解决办法就是起一个守护程序去续期，在业务没有执行完的时候不释放锁



如果锁的有效期是10分钟

一般做法：在程序执行了3分钟的时候去执行，如果key还在，则延时

启动了哨兵之后原来的redis就用不了啦

红锁

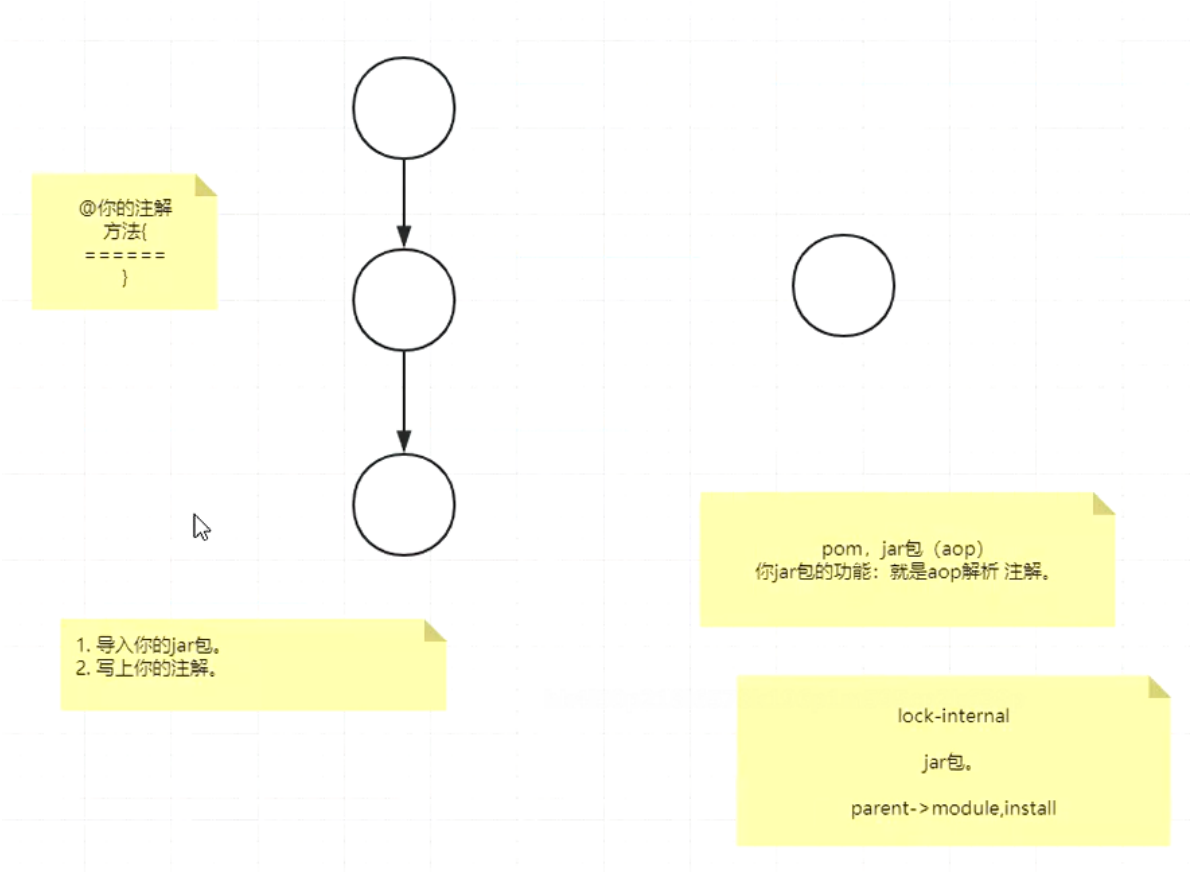
只有红锁才能解决redis的问题

弄三个独立的redis

```
@Bean
public RedissonClient redissonRed1(){
    Config config = new Config();
    config.useSingleServer().setAddress("127.0.0.1:6379").setDatabase(0);
    return Redisson.create(config);
}
@Bean
public RedissonClient redissonRed2(){
    Config config = new Config();
    config.useSingleServer().setAddress("127.0.0.1:6380").setDatabase(0);
    return Redisson.create(config);
}
@Bean
public RedissonClient redissonRed3(){
    Config config = new Config();
    config.useSingleServer().setAddress("127.0.0.1:6381").setDatabase(0);
    return Redisson.create(config);
}
```

```
$ ./redis-server.exe ./redis.windows6381.conf
```

RedLock

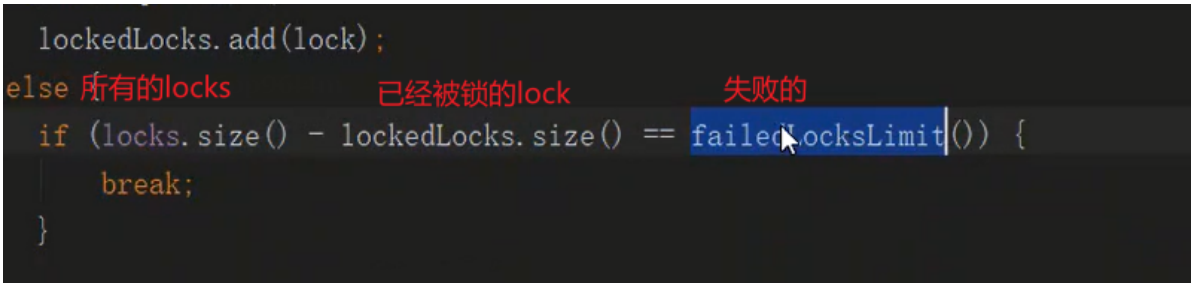


lua脚本

分布式锁的几个条件

- 互斥性，在同一时间只有一个能够拿到
- 防死锁
- 自己解自己的锁
- 容错性

红锁的流程



```
@Override
protected int failedLocksLimit() {
    return locks.size() - minLocksAmount(locks);
}

protected int minLocksAmount(final List<RLock> locks) {
    return locks.size()/2 + 1;
}
```

获取当前时间

按照一定的顺序逐个加锁

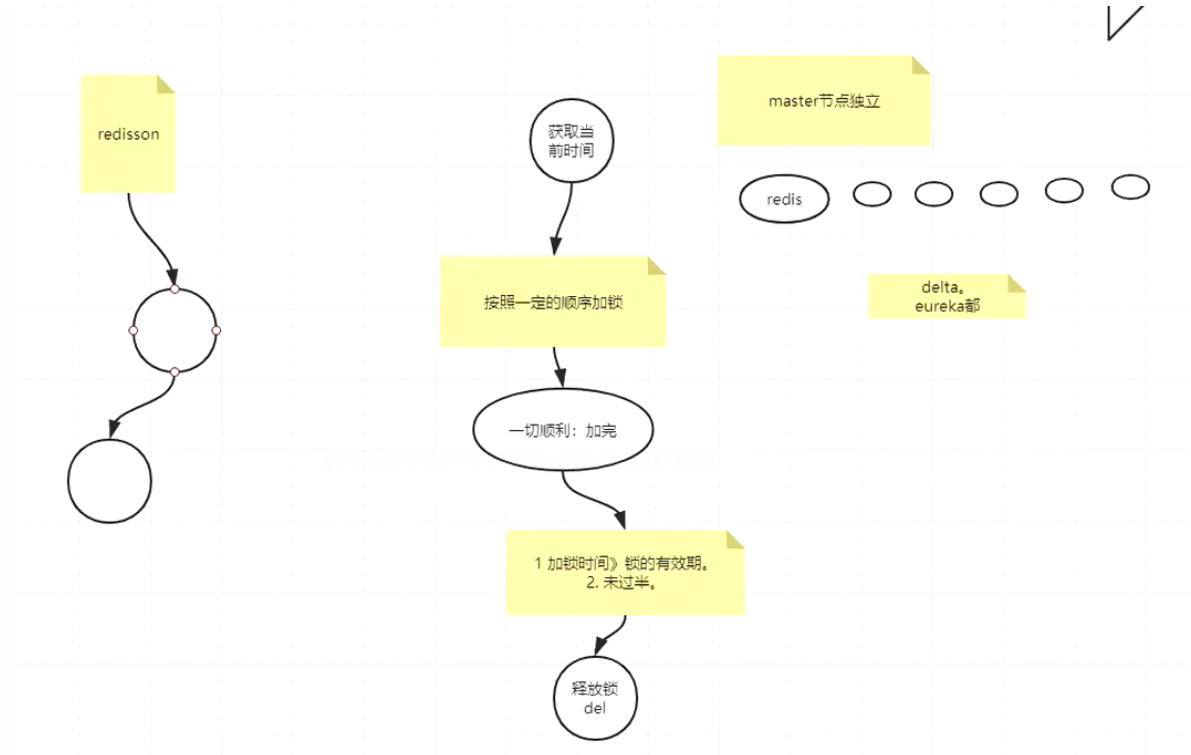
超时时间要比锁的过期时间短

加锁失败：

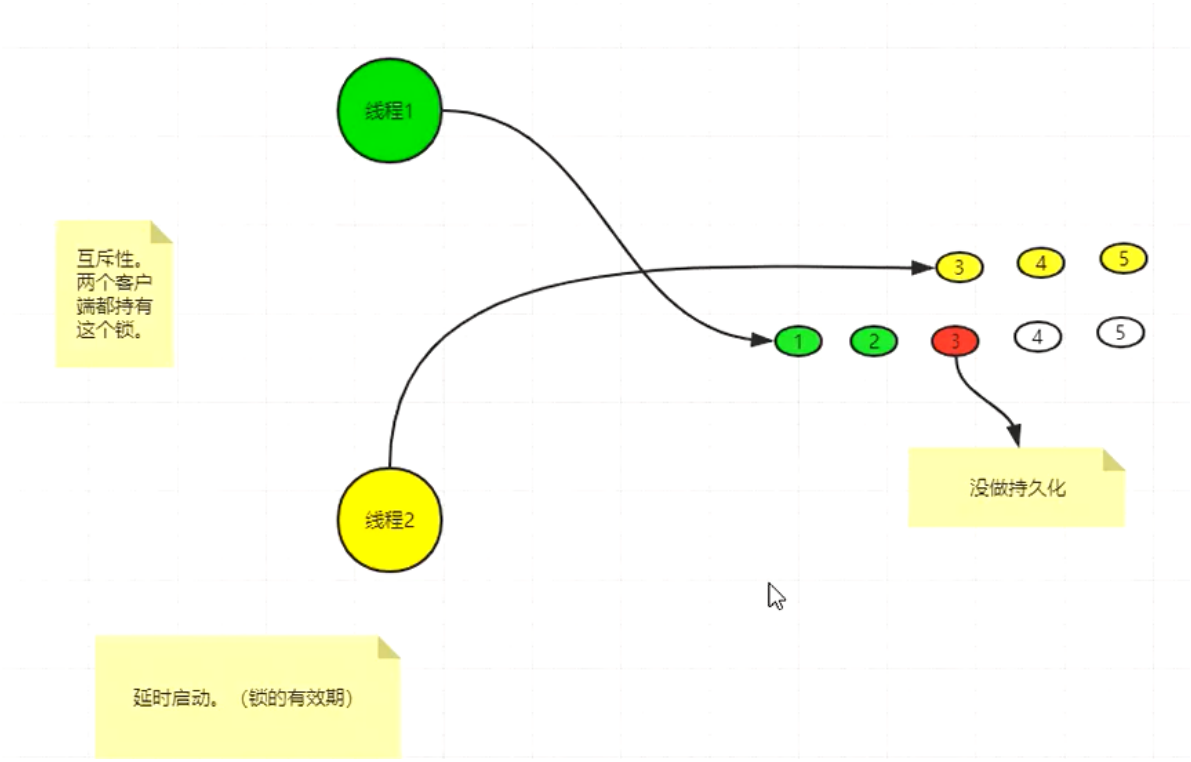
1、加锁的时间>锁的有效期

2、未过半

如果加锁失败那么就把所有的锁都释放，执行del



有可能线程1去加锁的时候，给1、2、3加锁的时候恰好3号挂了，然后恰好线程3没有做数据的持久化，那么当线程2进来的时候去给3、4、5加锁的时候会造成两个客户端都持有这个锁的情况，违反了互斥性



这时候则需要运维延时启动redis，延时的时间为锁的有效期。