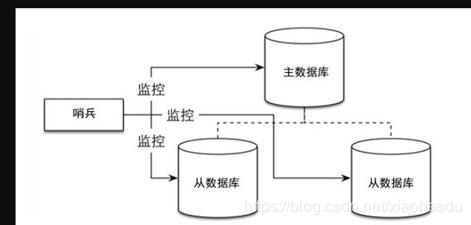
### Redis的高可用有2种方式

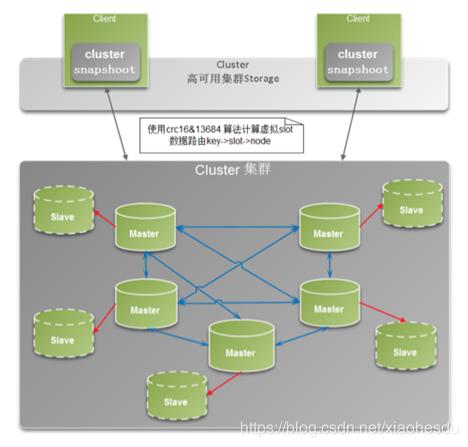
#### 1）哨兵模式

哨兵模式的基础依然是主从同步，只不过通过哨兵来做了监控和故障下的主从切换，从而保证高可用。

哨兵模式只是单纯的高可用，无法做到水平扩容。单机内存容量就是整个存储的瓶颈。  
例如3个redis实例；每个redis存储的内容都相同，通过哨兵实时监控主服务器的状态，当主服务器发送故障时，通过MASTER和SLAVE的切换来保证系统的高可用。  
  
哨兵进程使用流言协议(gossipprotocols)来接收关于Master主服务器是否下线的信息，并使用投票协议(Agreement Protocols)来决定是否执行自动故障迁移,以及选择哪个Slave作为新的Master。

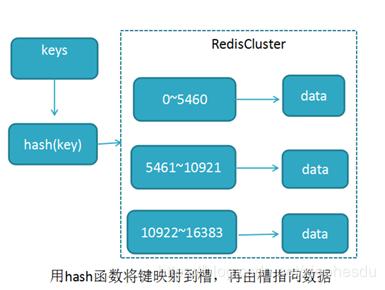
哨兵的作用就是监控redis主、从数据库是否正常运行，主出现故障自动将从数据库转换为主数据库。

#### 2）集群模式

集群模式是redis 3.0后提供的，它实现了redis存储的水平扩容，从而解决了单机内存容量的瓶颈。  
它的典型架构图如下所示  
  
在集群模式下，至少需要三主三从，通过rubygmes工具来创建redis集群。它的基础是由主从同步来保证高可用，由hash分区来实现分布式存储解决水平扩容问题。  
**常见的hash分区方法有如下三种方式：**

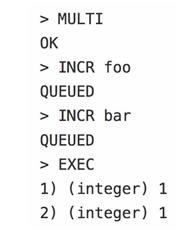
1.简单节点取余  
2.一致性哈希分区：一个环状结构，解决了节点动态扩容时的存储分配问题。

3.虚拟槽分区

Redis采用了虚拟槽分区技术。它把整个集群分成2的14次方个槽（0~16383），redis中的每个实例都只存储一部分的槽位以及槽中映射的数值。  
具体来说，对于一个key，redis基于如下公式计算槽位  
槽位 hash(key) = CRC16(key) & 16383  
  
这里通过hash(key)把任意的key映射到某个槽，然后就知道去哪个redis实例下存储K-V  
Redis通过虚拟槽分区方式，解耦了数据与节点的关系，每个redis节点自身要维护槽映射关系。

Redis集群的缺陷

* 键的批量操作支持有限，比如mset和mget，如果多个键映射到不同的槽，就不支持了
* 键的事务支持有限，当多个key分布在不同redis节点时就无法使用事务，只有相同节点内才支持事务
* 键是数据分区的最小粒度，容易因为热点数据导致单个节点负载较高
* 不支持多数据库

补充redis的事务：  
Redis事务的基础是redis本身是通过串行化的方式来顺序执行命令。而且redis还能通过MULTI命令开启事务，然后执行相应的命令，通过EXEC命令来表示命令结束。Redis可以保证上述事务中的命令依次被执行。而且，它还内置了一个检测器，如果在检测阶段就发现某个命令不能执行（比如语法错误、内存不足等），那么整个一组命令都不能被执行；如果检测通过，整个一组命令都会依次执行。  
  
但是它和常规的事务不同，它只能简单的执行，不具备回滚功能。当检测通过后，即使某个命令执行失败了，其他命令还是会继续执行，而不是回滚。

另外，补充redis的LUA脚本。  
https://redisbook.readthedocs.io/en/latest/feature/scripting.html

Lua 脚本功能是 Reids 2.6 版本的最大亮点， 通过内嵌对 Lua 环境的支持， Redis 解决了长久以来不能高效地处理 CAS （check-and-set）命令的缺点， 并且可以通过组合使用多个命令。我们可以定义一个lua脚本，然后用EVAL来执行该脚本。  
EVAL 命令的执行可以分为以下步骤：

1. 为输入脚本定义一个 Lua 函数。
2. 执行这个 Lua 函数。

实际上，就是把输入的lua脚本定义为一个lua函数，然后redis会把整个lua函数当做一个单条的命令来执行，不需要分开执行其中的每个命令。