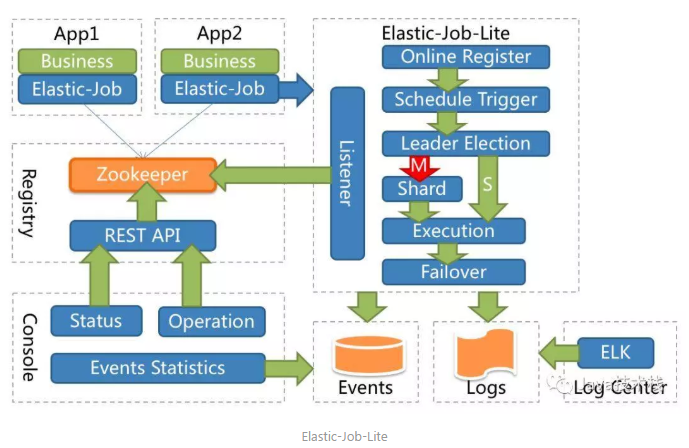
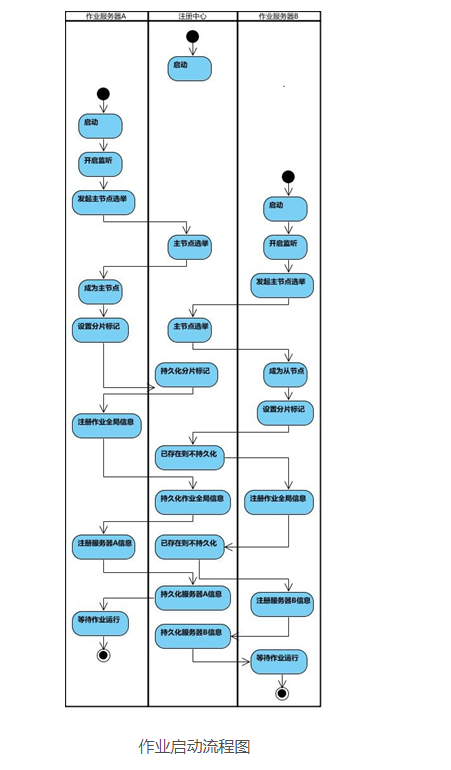
分布式调度：在分布式集群环境下，Quartz已经不再适应分布式集群的部署，在这种情况下分布式调度任务就应运而生。Elastic job主要的功能有支持弹性扩容，通过Zookepper集中管理和监控job，支持失效转移等。

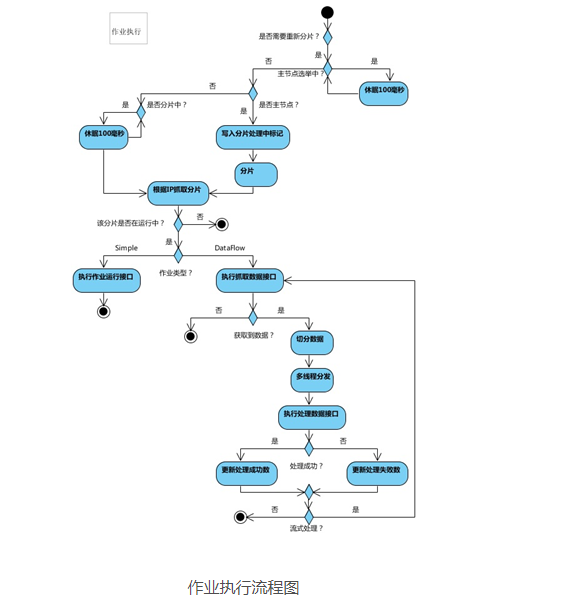
Elastic job是基于Zookeeper和Quartz开发的一个java分布式定时任务，由两个相互独立的子项目Elastic-Job-Lite和Elastic-Job-Cloud组成，在实际开发中应用比较多的是Elastic-Job-Lite

。Zookeeper在整个环境中起到一个服务注册，监听，失效转移的作用。

Zookeeper：注册中心在定义的命名空间下，创建作业名称节点，用于区分不同作业，所以作业一旦创建则不能修改作业名称，如果修改名称将视为新的作业。作业名称节点下又包含5个数据子节点，分别是config, instances, sharding, servers和leader。config节点保存作业配置信息，以JSON格式存储；instances节点保存作业运行实例信息，子节点是当前作业运行实例的主键；sharding节点保存作业分片信息，子节点是分片项序号，从零开始，至分片总数减一；servers节点保存作业服务器信息，子节点是作业服务器的IP地址；leader节点保存作业服务器主节点信息，分为election，sharding和failover三个子节点，分别用于主节点选举，分片和失效转移处理。

   Elastic-Job-Lite源码由两个入口，一个是JobSchedule类的init方法，当启动服务时，程序会进入该方法，进行任务的初始化等操作；另一个是LiteJob类，它实现了Quartz框架的Job接口，当定时任务启动时，会进入该实现类，完成失效转移项执行、重新分片、获取并执行本机任务项、错过任务重触发等操作。





  Elastic-Job-Lite执行器的入口是实现了Job接口的LiteJob类，当任务调度执行时，进入LiteJob类的execute方法。在这里完成一系列的操作，包括获取失效转移分片项，如果没有分配的失效转移项，则判断是否需要重新分片，然后获取分配给自己的分片项，然后判断当前分片项是否正在running，如果否，则执行任务项；如果是，则在sharding/[item]下添加misfire节点，标示该分片项错过执行，等待分片项执行结束后，再触发misfire的分片项执行。

总结：

1. Elastic-Job-Lite框架的任务处理、执行等都是针对的分片项，也就是说quartz调度执行面向的是定时任务的分片项，而不是定时任务本身。如果我们不打算对定时任务分片，那么可以把分片数设为1，这样在sharding节点下创建1个分片项0，0分片项将会被分配给一个instance，并启动执行。

2. Elastic-Job-Lite是提供失效转移功能的，即当正在执行的任务项遇到进程退出或机器宕机等故障，该任务项应该转移到某空闲服务器执行。但是，该功能存在bug：（1）失效转移只能在同一机器上不同实例间完成，跨机器无效；（2）没有判断机器宕机时任务项是否正在执行状态，而是只要遇到宕机，即使任务项没有开始执行，也被转移到其他机器上执行一遍，导致重复执行。

3. Elastic-Job-Lite是预分片，不是动态分片，即Elastic-Job-Lite是在服务启动时就完成分片项的创建和分配，并保存在zk节点上，而不是定时任务在每次启动时，根据机器的处理能力，重新分配任务项，例如：任务比较繁忙的机器不参与新一轮的任务项分配。这样做的目的是对zk弱依赖，如果进行真正的动态分片，对于秒级的定时任务将会产生很大影响，因为每次任务启动，应用服务器都要跟zookeeper进行通信，并执行重新分片的逻辑，频繁的通信，对于秒级定时任务，会错过很多次执行