Apache Helix

在NoSQL和大数据领域，随着爆炸式的数据增长，分布式系统的数量有了显著增长。 根据业务发展需要多个分布式系统，这些系统运行在多服务器的集群上，集群在运维过程中需要处理分布式系统的遇到的问题：

1. Fault tolerance（容错性），当遇到服务器宕机或者网络问题保证系统的可用性
2. 水平扩展性，当负载增加时可以保证集群的无缝扩展

如果集群管理系统不提供这些能力，那么分布式平台的开发者需要为不同的分布式系统重复实现这些功能，带来不必要的开发。Apache Helix就是这样一个通用的集群资源管理框架，来简化多分布式系统为了容错及水平扩展的实现，其提供以下能力：

1. 实现集群节点资源的自动管理，包括分区和资源的replicated
2. 集群中软件及硬件资源的故障检测和恢复
3. 基于服务器能力和资源profile(partions size，访问策略等)的资源的smart placement实现负载均衡
4. 全局的配置管理和服务的自我发现，避免逐一节点的修改配置
5. 在集群扩展时实现容错和再分布
6. 实现节点的生命周期管理，在不停服的前提下支持节点的增加/启动/停止/启用/禁用等操作
7. 监控集群健康及SLA(服务等级协议，Service-Level Agreement)的问题的告警
8. 实现请求转发的服务发现机制

资源管理系统需要一定的机制来协调节点和系统服务组件，该机制通过系统感知集群的变化，并且在启动Tasks后将集群转换成稳定状态，启动的Task集合分配到集群的一个或者多个节点中，Helix同时起到管理集群不同的组件的目的。

# 基本概念

Apache Helix是一个通用的集群管理框架，可以做分布式task、分布式存储、服务发现及分布式锁等，可以简化分布式系统的设计及开发。分布式系统的设计，对于一个Task，有两个基本的属性：位置和状态，对于分布式存储，partition或者region也有位置和状态，因此可以将位置和状态属性抽象成资源。Apache Helix引入下面的术语来定义分布式系统及其操作：

1. **分布式系统组件（Distributed System Components）**，对于分布式系统集群具有以下组件和特性

instance: 主机节点，其一起构成集群

resouces: 数据库,主题topic,索引或者tasks等都可以映射为一个资源，资源可以跨越多个实例。

**partitions:** 资源的基本构成，每个资源可以分成多个partitions（子资源或者子任务）。分区概念一般在消息系统上比较常见，比如Kafka，每个topic有多个partition；对于分布式task来讲，Task可以映射为子任务，这种子任务，可以理解为storm作业的spout,bolt task等，这种子任务可以统一重新成partition，代表最小的资源颗粒。

"myTask\_0", "myTask\_1", "myTask\_2" 运行在N1, N2 和 N3 上，那么理想状态描述成下面这样，其实这里helix的理想状态就相当于storm里的每个topology的assignment，即任务分配细节。

*{*

*"id" : "myTask",*

*"simpleFields" : {*

*"NUM\_PARTITIONS" : "3",*

*}*

*"mapFields" : {*

*"myTask\_0" : {*

*"N1" : "ONLINE",*

*},*

*"myTask\_1" : {*

*"N2" : "ONLINE",*

*},*

*"myTask\_2" : {*

*"N3" : "ONLINE",*

*}}*

**replicas:**  副本，每个资源的副本，Partitioning将任务或者数据分成更细粒度的分片，然后通常为了可用性，每个分片还有多个副本，比如HDFS文件的block有3个副本。ES的索引文件，同样有多个副本。Helix统一将这种分片的副本或者备份，称为replica，资源的assignment的示例如下：

*{*

*"id" : "myIndex",*

*"simpleFields" : {*

*"NUM\_PARTITIONS" : "3",*

*"REPLICAS" : "2",*

*},*

*"mapFields" : {*

*"myIndex\_0" : {*

*"N1" : "ONLINE",*

*"N2" : "ONLINE"*

*},*

*"myIndex\_1" : {*

*"N2" : "ONLINE",*

*"N3" : "ONLINE"*

*},*

*"myIndex\_2" : {*

*"N3" : "ONLINE",*

*"N1" : "ONLINE"*

*}*

*}*

*}*

**state：** 每个副本的状态，例如Master,Slave,Leader,standby,online,offline等，比如Zookeeper中，每个节点分为leader和follow；Kafka的partition也分为master,slave；

hdfs在block recovery的时候，也会将持有副本的datanode选出master和slave，即解决一致性的问题，副本之间通常选出一个master负责接收请求，再同步给其他Slave。

比如使用Helix组织Mysql实例做一个Mysql集群，一个数据库做主备，那么assignments的idea state如下所示：

*{*

*"id" : "myDB",*

*"simpleFields" : {*

*"NUM\_PARTITIONS" : "1",*

*"REPLICAS" : "2",*

*},*

*"mapFields" : {*

*"myDB" : {*

*"N1" : "MASTER",*

*"N2" : "SLAVE",*

*}}*

1. **Roles，角色**

在分布式系统中不同的节点的功能是不同，例如部分节点获取请求，部分节点发送请求，可能其他节点用于管理集群中的节点。因此Helix将节点根据系统不同的功能实现以下几种：

Participant: 管理分布式资源的节点

Spectator: 监控每个Participant的状态，并转发请求。比较像Routers，知道每个partition在哪个instance及其状态，然后将请求分发到正确的endpoint

Controller: 该角色用于监控和控制每个Participant nodes，在集群进行变化时能够进行转换的协调，根据集群的限制条件达到稳定状态。

这些角色是逻辑概念，根据系统的需要进行部署。例如Controller可以以单独服务的形式占用一个节点，也可以和Participant一起部署，但是每个集群只能有一个Controller。

1. **Cluster State Metadata Store**

存储集群状态的分布式Store，当集群状态变换时能够通知相关系统。Helix使用ZK来实现该功能。ZK存储：

PERSISTENT状态，在被删除之前一直保存在ZK中

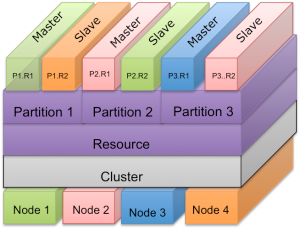
TRANSIENT/EPHEMERAL状态，当进程创建时初始化，进程停止时删除

通知机制，当PERSISTENT及EPHEMERAL状态变换时通知相关系统

ZK提供的namespace类似于标准的文件系统，使用slash(/)分割的路径表示一个名词。

1. **State Machine and Constraints**

Resources，Partitions和Replicas类似于大部分分布式系统中的概念，具体的不同分布式系统，功能和行为显著不同，例如搜索系统，用于支持查询的行为肯定和读取可变数据行为是不同的。分布式系统之间的不同是每个partition都会有有个状态及状态的限制，为了有效描述一个分布式系统的行为，需要定义资源和分区的各种状态，来捕捉各种有效状态和状态转换。例如：一个系统对外提供read-only数据，所有的partition副本相同，其状态是ONLINE或者OFFLINE如果一个系统同时支持读写，必须保证写时要同步所有的副本，STATE是MASTER,SLAVE及OFFLINE。将数据写到MASTER上，然后将数据复制到SLAVES上，数据读可以选择每个副本。结果如下图所示：



除了partition上定义状态，状态之间的转换路径也是application specific。例如，为了成为MASTER，必须要求首先成为SLAVE，保证该PARTITION不包含OFFLINE-SLAVE的数据，这样就可以从其他系统中将数据同步过来。

Helix提供一种方式来配置application-specific state Machine，状态机和状态转换，核心是failover，即容灾，Helix在一个集群中有一个controller角色，类似于storm的nimbus，或者HBase中的hbasemaster，负责状态转换，目的是为了让集群到达ideastate。每个状态上有相应的限制。Helix提供根据限制进行状态转换的矩阵，

Controller使用状态机来实现，用户提前定义好状态机，例如MasterSlave，最简单的状态模式示例如下：

*OFFLINE | SLAVE | MASTER*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*| | | |*

*OFFLINE | N/A | SLAVE | SLAVE |*

*|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|*

*| | | |*

*SLAVE | OFFLINE | N/A | MASTER |*

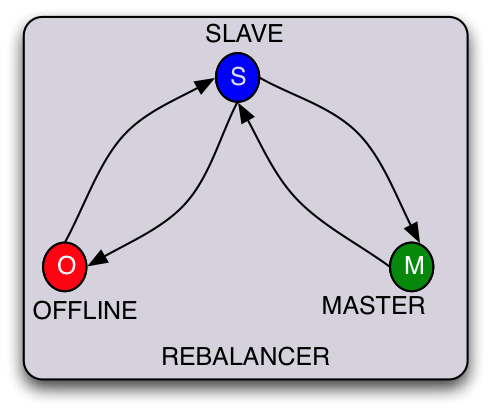
*|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|*

*| | | |*

*MASTER | SLAVE | SLAVE | N/A |*

*|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|*

Helix支持灵活的状态机配置，一种资源配一个状态机，同样在同一个集群中，可以有多种资源，多个状态机。转换图如下所示：



1. **State**

在状态机中来对resources进行建模的状态，包括：

IdealState: 在集群中所有的节点都状态运行，状态的限制被满足

CurrentState: 集群中每个节点的当前状态

ExternalView: 所有节点的当前状态的总览

**idealState**，也就是启动所有的节点，但是当遇到以下情况时状态会非Ideal：

部分或者所有的节点都被停掉

一个或者多个节点fail

增加新节点或者增加partitions（需要重新reassigned）

其中，Ideal State如下所示：

*"TASK\_NAME" : {*

*"LOCATION" : "STATE"*

*}*

如果想在N1上启动myTask，理想状态可以描述成这样：

*{*

*"id" : "MyTask",*

*"mapFields" : {*

*"myTask" : {*

*"N1" : "ONLINE",*

*}*

*}*

*}*

还有一种idealState称为Dynamic IdeaState

动态idealState其实是支持动态调整资源的状态，helix可以感知集群变化，然后动态调整ideastate,同时触发rebalance，helix支持三种方式实现动态调整ideastate：

FULL\_AUTO，全自动，根据约束条件，helix自动调整

SEMI\_AUTO，半自动，用于提供一个期望的idealstate信息，Helix会根据这个状态进行rebalance

CUSTOMIZED，完全用户定制，相当于jstorm中指定suprevisor调度。

**CurrentState**用于表示一个节点，即分布式系统的node的当前状态，比如storm中的supervisor心跳新，里面包含集群名，槽位及资源的状态。

*{*

*"id":"MyResource"*

*,"simpleFields":{*

*,"SESSION\_ID":"13d0e34675e0002"*

*,"INSTANCE\_NAME":"node1"*

*,"STATE\_MODEL\_DEF":"MasterSlave"*

*}*

*,"mapFields":{*

*"MyResource\_0":{*

*"CURRENT\_STATE":"SLAVE"*

*}*

*,"MyResource\_1":{*

*"CURRENT\_STATE":"MASTER"*

*}*

*,"MyResource\_2":{*

*"CURRENT\_STATE":"MASTER"*

*}*

*}*

*}*

集群的每个节点都有自己的currentState，节点在Helix中称为participants.

**External View**,集群的客户端称为spectators，相当于storm中的nimbusclient，比如Storm想要看作业的分配情况，只需要获取assignments，示例如下：

*{*

*"id":"MyResource",*

*"mapFields":{*

*"MyResource\_0":{*

*"N1":"SLAVE",*

*"N2":"MASTER",*

*"N3":"OFFLINE"*

*},*

*"MyResource\_1":{*

*"N1":"MASTER",*

*"N2":"SLAVE",*

*"N3":"ERROR"*

*},*

*"MyResource\_2":{*

*"N1":"MASTER",*

*"N2":"SLAVE",*

*"N3":"SLAVE"*

*}*

*}*

*}*

1. Rebalancer

集群的资源分布式不均，或者task分布不均，可能会导致热点问题，使用rebalance触发task或者资源的重新调度。在节点中的启动、stop、软硬件异常、集群新增节点、改变配置或者idea state变化，Helix Controller模块会触发rebalance，其算法步骤如下：

比较当前状态currentstate和ideastate

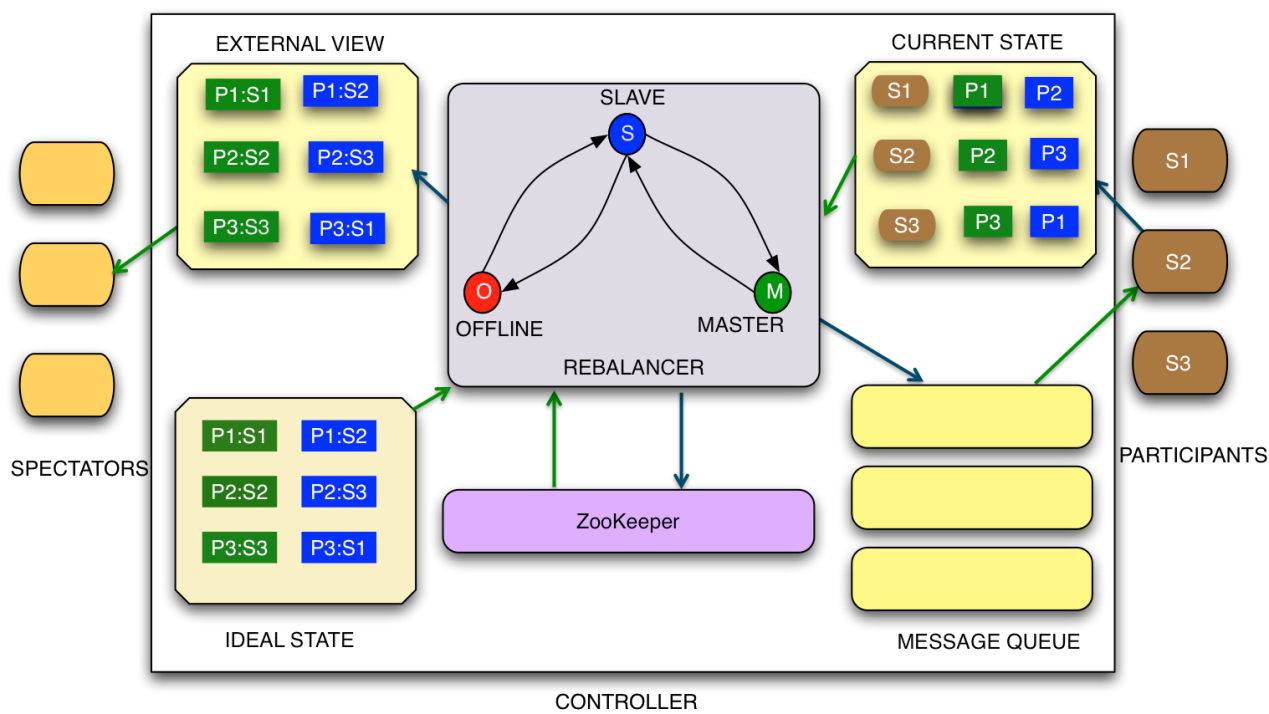
走状态机流程，计算怎样transition

通告节点执行transition

Rebalance的目标是达到ideastate = currentState =ExternalView这种状态。

# 系统架构

下图是Controllers,Participants和Specators相互之间如何交互：



集群中的不同角色节点可执行的流程如下：

1. Participant，进程将该节点注册到集群中，接收消息并更新当前状态，例如分布式数据库中的存储节点
2. Spectator，该进程仅关心ExternalView中的变化
3. Controller，根据集群状态变化来控制集群，并将状态变化消息发送给Participants。

下面介绍各角色节点的执行过程：

1. Participant Node Process

当Participant启动后，将本身注册到LiveInstances中

注册后，在消息队列中等待新的消息

当接收消息后，执行消息中需要进行的Tasks

Tasks执行完以后，根据执行的结果更新CurrentState

1. Controller Process

监控IdeaState

当某个Participant宕机，comes up或者增加，异常等操作，监控ephemeral liveInstance Znode及cluster中的currentState

触发正确的state转换，并将messages发送给Participants

1. Spectator Process

当该进程启动后，向Helix Agent请求，当ExternalView变化时发送Notified请求给该进程

接收到Notification后，读取ExternalView并执行需要的duties

1. Core Controller Algorithm

从ZK中获取IdealState和所有active storage nodes的CurrentState

计算所有partitipant nodes中每个partition replica中idealState和CurrentState之间的delta值

根据State Machine Table计算处理的partition Tasks，在状态转化过程中配置正确的权限。例如MasterSlave: 没有非法的限制进行mastership转换 => 执行partition Addition =>执行partition drop

对于每个storage node中对象的queue可以并行添加transition tasks，

如果transition task依赖其他的task的完成情况，不增加该task

通过一个Participant完成一个task，Controllers获取变化的notified，根据currentState和idealState之间的匹配进行重新计算

1. Helix Znode Layout

Helix使用Znode保存集群状态信息，Znode的目录结构如下所示：

|—test-cluster

|— PROPERTYSTORE，应用的attribute store

|— STATEMODELDEFS，状态模型的定义

|— INSTANCES,正在运行的instance信息，包括current state和messages

|—localhost\_8901

|—CURRENTSTATES

|—sessionId

|—resourceName

|—ERRORS

|—STATUSUPDATES

|—MESSAGES

|—HEALTHREPORT

|—localhost\_8902

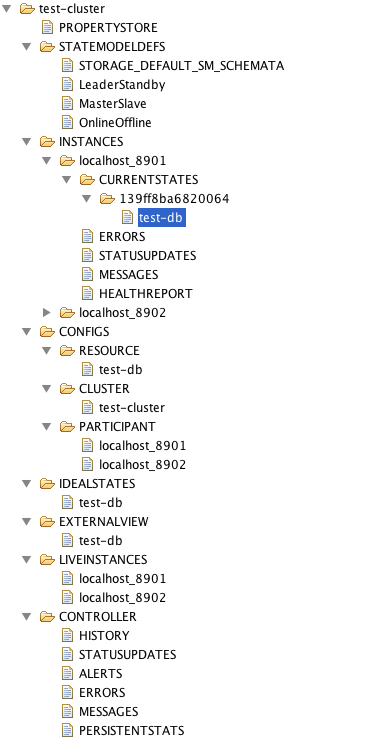
|—CONFIGS，有层次的配置

|—RESOURCE, 包括resource层次的配置

|—CLUSTER,包括cluster层次的配置

|—PARTICIPANT,包括participant层次的配置

下图是一个运行的例子：



# 3.安装和使用

## 3.1 下载安装包

*wget http://mirror.bit.edu.cn/apache/helix/0.6.8/binaries/helix-core-0.6.8-pkg.tar*

*wget http://mirror.bit.edu.cn/apache/helix/0.6.8/binaries/helix-admin-webapp-0.6.8-pkg.tar*

*wget http://mirror.bit.edu.cn/apache/helix/0.6.8/binaries/helix-agent-0.6.8-pkg.tar*

解压helix-core-0.6.8-pkg.tar，目录结构如下图所示：

*[root@fys1 helix-core-0.6.8]# ll*

*bin //执行脚本*

*conf //配置文件，当前仅有log4j*

*LICENSE*

*NOTICE*

*repo //程序运行所需要的jar包*

*[root@fys1 helix-core-0.6.8] cd bin/*

*helix-admin.sh*

*JmxDumper.sh*

*quickstart.sh*

*run-helix-controller.sh*

*start-helix-participant.sh*

*start-standalone-zookeeper.sh*

*test-util.sh*

*yaml-cluster-setup.sh*

*zkcopy.sh*

*zk-dumper.sh*

*zkgrep.sh*

*zk-log-csv-formatter.sh*

*zk-log-parser.sh*

## 3.2 Quickstart

执行命令如下所示：

*./quickstart.sh*

可以在界面上看到Helix使用过程

1. The Initial Setup

创建一个Helix集群，并将添加两个节点，并对partitions进行rebalanced。

*Creating cluster: HELIX\_QUICKSTART*

*Adding 2 participants to the cluster*

*Added participant: localhost\_12000*

*Added participant: localhost\_12001*

1. 配置状态机模型

*Configuring StateModel: MyStateModel with 1 Master and 1 Slave*

1. 创建资源，配置6个partions，并且2个副本

*Adding a resource MyResource: with 6 partitions and 2 replicas*

*Starting Participants*

*Started Participant: localhost\_12000*

*Started Participant: localhost\_12001*

1. 启动Helix Controller，并进行状态转换

*Starting Helix Controller*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_1*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_2*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_5*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_0*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_4*

*localhost\_12000 transitioning from OFFLINE to SLAVE for MyResource\_3*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_3*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_5*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_2*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_1*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_0*

*localhost\_12001 transitioning from OFFLINE to SLAVE for MyResource\_4*

*localhost\_12001 transitioning from SLAVE to MASTER for MyResource\_1*

*localhost\_12001 transitioning from SLAVE to MASTER for MyResource\_4*

*localhost\_12001 transitioning from SLAVE to MASTER for MyResource\_5*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_2*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_3*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_0*

5)至此，启动了两个节点，状态如下：

*CLUSTER STATE: After starting 2 nodes*

*localhost\_12000 localhost\_12001*

*MyResource\_0 M S*

*MyResource\_1 S M*

*MyResource\_2 M S*

*MyResource\_3 M S*

*MyResource\_4 S M*

*MyResource\_5 S M*

1. 增加节点，如下所示：

*###################################################################*

*ADDING NEW NODE :localhost\_12002. Partitions will move from old nodes to the new node.*

*localhost\_12000 transitioning from MASTER to SLAVE for MyResource\_0*

*localhost\_12001 transitioning from MASTER to SLAVE for MyResource\_1*

*localhost\_12001 transitioning from MASTER to SLAVE for MyResource\_4*

*localhost\_12001 transitioning from SLAVE to MASTER for MyResource\_0*

*localhost\_12000 transitioning from MASTER to SLAVE for MyResource\_3*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_4*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_0*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_4*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_5*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_2*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_3*

*localhost\_12002 transitioning from OFFLINE to SLAVE for MyResource\_1*

*localhost\_12002 transitioning from SLAVE to MASTER for MyResource\_3*

*localhost\_12002 transitioning from SLAVE to MASTER for MyResource\_1*

*CLUSTER STATE: After adding a third node*

*localhost\_12000 localhost\_12001 localhost\_12002*

*MyResource\_0 S M S*

*MyResource\_1 S S M*

*MyResource\_2 M S S*

*MyResource\_3 S S M*

*MyResource\_4 M S S*

*MyResource\_5 S M S*

每个Partition由1个Master和2两个Slave

7) 删除某个节点，如下所示：

*STOPPING localhost\_12002. Mastership will be transferred to the remaining nodes*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_1*

*localhost\_12000 transitioning from SLAVE to MASTER for MyResource\_3*

*CLUSTER STATE: After the 3rd node stops/crashes*

*localhost\_12000 localhost\_12001 localhost\_12002*

*MyResource\_0 S M -*

*MyResource\_1 M S -*

*MyResource\_2 M S -*

*MyResource\_3 M S -*

*MyResource\_4 M S -*

*MyResource\_5 S M -*

*###############################################################*

## 3.3 Long Version

可以通过手动的创建上步骤的实例，步骤如下：

### 3.3.1 安装和启动Zookeeper

Zookeer可以以Standalone模式或者replicated 模式的方式启动，不再介绍，Standalone的启动命令如下：

*./start-standalone-zookeeper.sh 2199 &*

### 3.3.2 Define the Cluster

helix-admin tool用于集群管理任务，在QuickStart例子中使用Command Line接口。Helix还支持REST 接口。zookeeper\_address的格式为host:port，例如localhost:2199。

创建集群MYCLUSTER，可以配置一下性质：

1）在localhost上使用端口：12913,12914,12915启动3个instance

2）数据MyDB，有6个partitions

3）每个Partition有3个副本，1个作为master，2个作为slave。

4）Zookeeper运行在localhost:2199

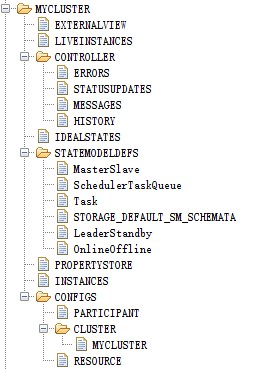
### 3.3.3 创建集群MYCLUSTER

命令格式如下

*[format]# ./helix-admin.sh --zkSvr <zk\_address> --addCluster <clustername>*

*./helix-admin.sh --zkSvr localhost:2199 --addCluster MYCLUSTER*

执行后，在zk中增加znode如下图所示：



### 3.3.4 向集群中添加节点

命令格式如下：

*# helix-admin.sh --zkSvr <zk\_address> --addNode <clustername> <host:port>*

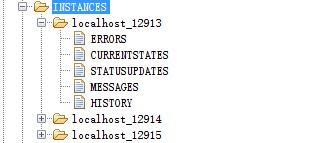
增加三个节点，localhost:12913,localhost:12914,localhost:12915，执行如下：

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12913*

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12914*

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12915*

在zk中的INSTANCES中创建的节点目录如下所示：



### 3.3.5 Define the Resource and Partitioning

在本例中，resource是一个数据库，分成6个partition，在生产系统中一般使用over-partition来实现更好的负载均衡。Helix已经在数十个物理节点上管理数百个数据库，每个数据库有数十至数百个partition。

1. 使用MasterSlave State Model来创建一个有6个Partitions的数据库myDB ,Helix保证每个Partition有一个master

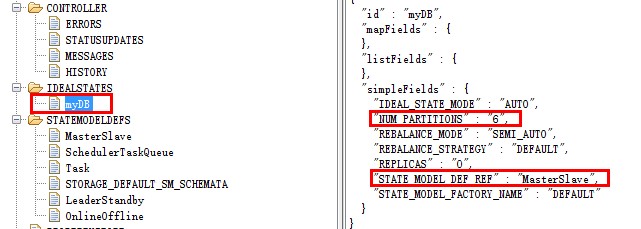
命令格式如下：

*# helix-admin.sh --zkSvr <zk\_address> --addResource <clustername> <resourceName> <numPartitions> <StateModelName>*

本示例的执行命令如下：

*./helix-admin.sh --zkSvr localhost:2199 --addResource MYCLUSTER myDB 6 MasterSlave*

在ZK中的IDEALSTATES的Znode下会增加myDB，具体如下图所示：



2.使用Helix 将Partitions分发到Node

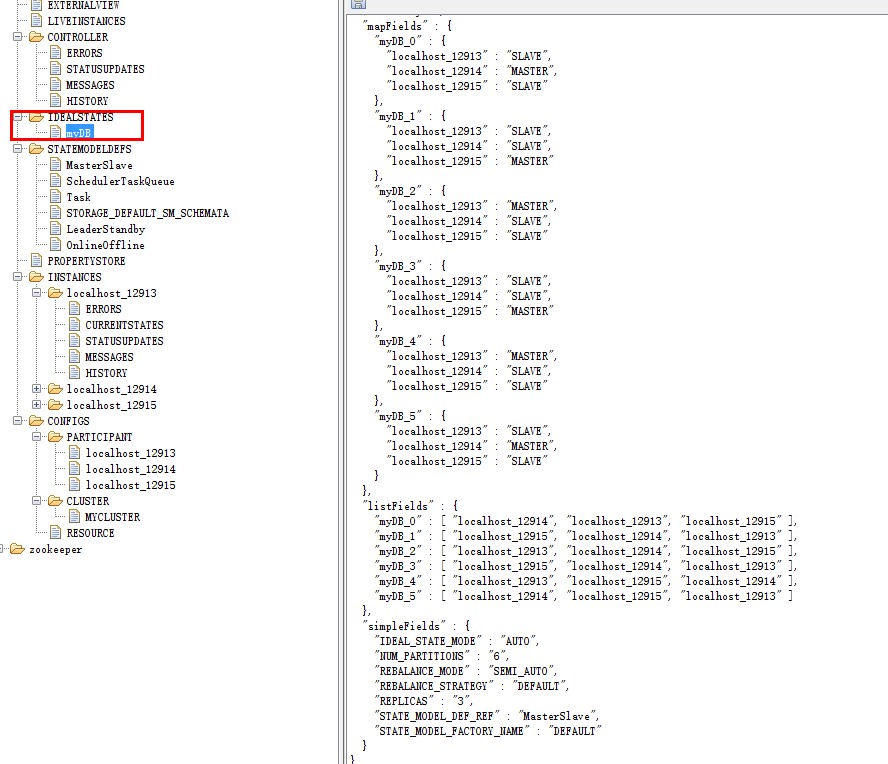
命令格式如下：

*# helix-admin.sh --zkSvr <zk\_address> --rebalance <clustername> <resourceName> <replication factor>*

本例中将myDB 的partition以三个副本的方式分发到cluster中。

*./helix-admin.sh --zkSvr localhost:2199 --rebalance MYCLUSTER myDB 3*

在IDEALSTATES的myDB中会存储partiton的放置，信息如下所示：



如上图所示，myDB resource的partition分成[myDB\_0,myDB\_5]。

至此，在zookeeper中定义了cluster，包含以下：

1. nodes, localhost:12913, localhost:12914, localhost:12915
2. resource, myDB，使用MasterSlave模型，共有6个partition
3. idealState，已经经过计算，副本数设置为3

### 3.3.6 启动Helix Controller

经过以下步骤，已经在zk中定义好了Cluster，Helix Controller用于管理Cluster，命令如下：

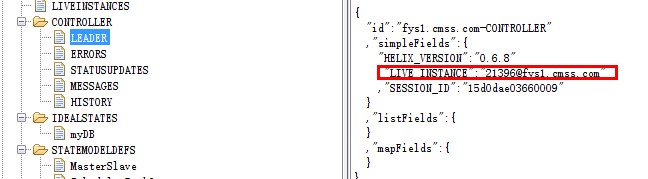
*./run-helix-controller.sh --zkSvr localhost:2199 --cluster MYCLUSTER 2>&1 > /tmp/controller.log &*

命令执行后，在节点启动进程：

*$ jps*

*21396 HelixControllerMain*

在zookeeper中修改的CONTROLLER znode，信息如下：



### 3.3.7 启动Cluster

以上已经完成cluster,resources,partitioning的定义及启动Helix Controller，现在启动管理的系统节点，每个node都是一个Participant，可以看做是系统组件的一个实例。Helix将Work分发给Participants，并实时获取其roles和health，当node fails，执行相应操作。

命令如下所示：

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12913 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12913.log*

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12914 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12914.log*

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12915 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12915.log*

命令执行后，启动进程如下：

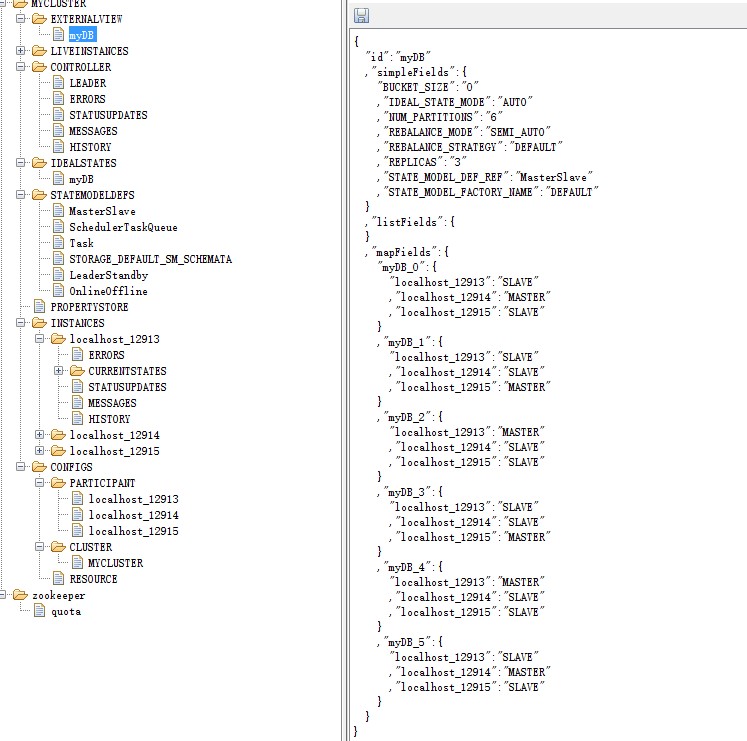
*$ jps*

*23380 ExampleProcess*

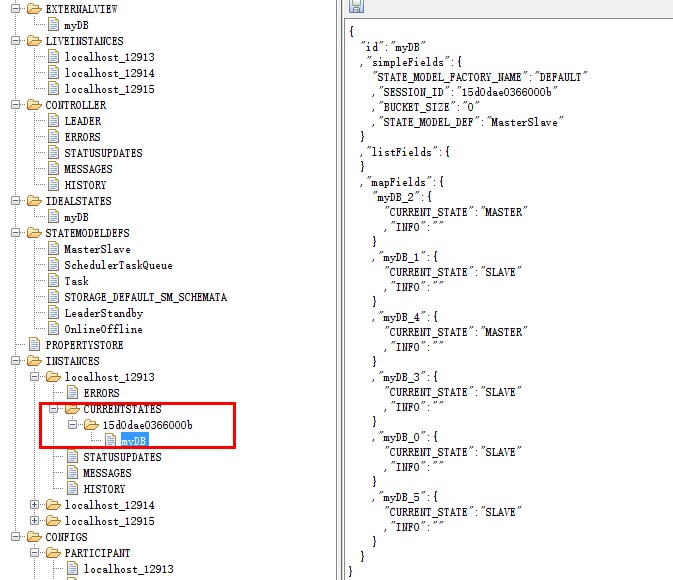
*23507 ExampleProcess*

*23554 ExampleProcess*

启动三个Participants后，Resource的partition分发到各自的node上，在Znode上的EXTERNALVIEW展示信息如下：



可以查看到每个partition的master和slave所在的Node。在INSTANCES中会有每个Node上的当前states，如下所示：



至此进行了启动Helix所有的步骤。

### 3.3.8 Inspect the Cluster

下面查看集群的Helix View ，查看的信息如下：

*Clusters -> MYCLUSTER -> instances -> instance detail*

*-> resources -> resource detail*

*-> partitions*

在Helix中Controller可以管理多个Cluster，通过下面的命令来查看Cluster列表：

*$helix-admin.sh --zkSvr localhost:2199 --listClusters*

*Existing clusters:*

*MYCLUSTER*

查看MYCLUSTER 的Helix View，命令如下所示：

*$ helix-admin.sh --zkSvr localhost:2199 --listClusterInfo MYCLUSTER*

*Existing resources in cluster MYCLUSTER:*

*myDB*

*Instances in cluster MYCLUSTER:localhost\_12913,localhost\_12914localhost\_12915*

展示了MYCLUSTER的resources和instances（node）。

下面查看每个instance的details，命令如下：

*$ helix-admin.sh --zkSvr localhost:2199 --listInstanceInfo MYCLUSTER localhost\_12913*

*InstanceConfig: {*

*"id" : "localhost\_12913",*

*"mapFields" : {*

*},*

*"listFields" : {*

*},*

*"simpleFields" : {*

*"HELIX\_ENABLED" : "true",*

*"HELIX\_HOST" : "localhost",*

*"HELIX\_PORT" : "12913"*

*}*

*}*

下面的命令是查询resouce的Query Information，命令如下所示：

*$ helix-admin.sh --zkSvr localhost:2199 --listResourceInfo MYCLUSTER myDB*

*IdealState for myDB:*

*{*

*"id" : "myDB",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_1" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "MASTER"*

*},*

*"myDB\_2" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_3" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "MASTER"*

*},*

*"myDB\_4" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_5" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*"myDB\_0" : [ "localhost\_12914", "localhost\_12913", "localhost\_12915" ],*

*"myDB\_1" : [ "localhost\_12915", "localhost\_12914", "localhost\_12913" ],*

*"myDB\_2" : [ "localhost\_12913", "localhost\_12914", "localhost\_12915" ],*

*"myDB\_3" : [ "localhost\_12915", "localhost\_12914", "localhost\_12913" ],*

*"myDB\_4" : [ "localhost\_12913", "localhost\_12915", "localhost\_12914" ],*

*"myDB\_5" : [ "localhost\_12914", "localhost\_12915", "localhost\_12913" ]*

*},*

*"simpleFields" : {*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

*ExternalView for myDB:*

*{*

*"id" : "myDB",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_1" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "MASTER"*

*},*

*"myDB\_2" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_3" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "MASTER"*

*},*

*"myDB\_4" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12915" : "SLAVE"*

*},*

*"myDB\_5" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*},*

*"simpleFields" : {*

*"BUCKET\_SIZE" : "0",*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

查看某个Partitions信息的命令如下：

$ helix-admin.sh --zkSvr localhost:2199 --listPartitionInfo MYCLUSTER myDB *myDB\_0*

*IdealState for myDB/myDB\_0:*

*{*

*"id" : "myDB/myDB\_0",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*"myDB\_0" : [ "localhost\_12914", "localhost\_12913", "localhost\_12915" ]*

*},*

*"simpleFields" : {*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

*ExternalView for myDB/myDB\_0:*

*{*

*"id" : "myDB/myDB\_0",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*},*

*"simpleFields" : {*

*"BUCKET\_SIZE" : "0",*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

### 3.3.9 Expand the Cluster

扩展Cluster，在集群运行过程中需要增加集群的能力，在上个例子中，启动了3个节点，6个partitions，而且集群时负载均衡的，每个节点上有2个Master和4个slaves。现在我们新增加3个nodes：localhost:12916, localhost:12917, localhost:12918，命令如下：

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12916*

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12917*

*./helix-admin.sh --zkSvr localhost:2199 --addNode MYCLUSTER localhost:12918*

上述命令，并不会启动instances进程，仅是在INSTANCES中增加了这些信息。启动这些instances的命令如下：

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12916 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12916.log &*

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12917 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12917.log &*

*./start-helix-participant.sh --zkSvr localhost:2199 --cluster MYCLUSTER --host localhost --port 12918 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12918.log &*

启动这些instance后，要对partitions进行负载均衡，命令如下：

*./helix-admin.sh --zkSvr localhost:2199 --rebalance MYCLUSTER myDB 3*

执行后，每个节点有1个master和2个slaves。具体的分布如下所示：

*"listFields" : {*

*"myDB\_0" : [ "localhost\_12917", "localhost\_12913", "localhost\_12914" ],*

*"myDB\_1" : [ "localhost\_12918", "localhost\_12917", "localhost\_12916" ],*

*"myDB\_2" : [ "localhost\_12913", "localhost\_12918", "localhost\_12917" ],*

*"myDB\_3" : [ "localhost\_12915", "localhost\_12917", "localhost\_12918" ],*

*"myDB\_4" : [ "localhost\_12916", "localhost\_12918", "localhost\_12917" ],*

*"myDB\_5" : [ "localhost\_12914", "localhost\_12913", "localhost\_12915" ]*

*}*

### 3.3.10 Failover

使用Helix，可以很容易实现一个容错系统，Helix可以检测到Failed instance，同时触发membership的转换。

在上例中，kill掉localhost:12918，myDB\_1失去了其Master Partition，Helix自动修复，将SLAVE，本例中将localhost:12917转换成MASTER，使用下面的命令进行查询：

*$ helix-admin.sh --zkSvr localhost:2199 --listResourceInfo MYCLUSTER myDB*

*IdealState for myDB:*

*{*

*"id" : "myDB",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12917" : "MASTER"*

*},*

*"myDB\_1" : {*

*"localhost\_12916" : "SLAVE",*

*"localhost\_12917" : "SLAVE",*

*"localhost\_12918" : "MASTER"*

*},*

*"myDB\_2" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12917" : "SLAVE",*

*"localhost\_12918" : "SLAVE"*

*},*

*"myDB\_3" : {*

*"localhost\_12915" : "MASTER",*

*"localhost\_12917" : "SLAVE",*

*"localhost\_12918" : "SLAVE"*

*},*

*"myDB\_4" : {*

*"localhost\_12916" : "MASTER",*

*"localhost\_12917" : "SLAVE",*

*"localhost\_12918" : "SLAVE"*

*},*

*"myDB\_5" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*"myDB\_0" : [ "localhost\_12917", "localhost\_12913", "localhost\_12914" ],*

*"myDB\_1" : [ "localhost\_12918", "localhost\_12917", "localhost\_12916" ],*

*"myDB\_2" : [ "localhost\_12913", "localhost\_12918", "localhost\_12917" ],*

*"myDB\_3" : [ "localhost\_12915", "localhost\_12917", "localhost\_12918" ],*

*"myDB\_4" : [ "localhost\_12916", "localhost\_12918", "localhost\_12917" ],*

*"myDB\_5" : [ "localhost\_12914", "localhost\_12913", "localhost\_12915" ]*

*},*

*"simpleFields" : {*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

*ExternalView for myDB:*

*{*

*"id" : "myDB",*

*"mapFields" : {*

*"myDB\_0" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "SLAVE",*

*"localhost\_12917" : "MASTER"*

*},*

*"myDB\_1" : {*

*"localhost\_12916" : "SLAVE",*

*"localhost\_12917" : "MASTER"*

*},*

*"myDB\_2" : {*

*"localhost\_12913" : "MASTER",*

*"localhost\_12917" : "SLAVE"*

*},*

*"myDB\_3" : {*

*"localhost\_12915" : "MASTER",*

*"localhost\_12917" : "SLAVE"*

*},*

*"myDB\_4" : {*

*"localhost\_12916" : "MASTER",*

*"localhost\_12917" : "SLAVE"*

*},*

*"myDB\_5" : {*

*"localhost\_12913" : "SLAVE",*

*"localhost\_12914" : "MASTER",*

*"localhost\_12915" : "SLAVE"*

*}*

*},*

*"listFields" : {*

*},*

*"simpleFields" : {*

*"BUCKET\_SIZE" : "0",*

*"IDEAL\_STATE\_MODE" : "AUTO",*

*"NUM\_PARTITIONS" : "6",*

*"REBALANCE\_MODE" : "SEMI\_AUTO",*

*"REBALANCE\_STRATEGY" : "DEFAULT",*

*"REPLICAS" : "3",*

*"STATE\_MODEL\_DEF\_REF" : "MasterSlave",*

*"STATE\_MODEL\_FACTORY\_NAME" : "DEFAULT"*

*}*

*}*

# Helix-admin-webapp

在第三部分介绍了Helix的命令行接口，下面介绍Helix的Rest API接口，启动命令如下所示：

*./run-rest-admin.sh --zkSvr <zookeeperAddress> --port <port>*

启动后如下所示：

*[helix@fys1 helix-admin-webapp-0.6.8]$ bin/run-rest-admin.sh --zkSvr fys1.cmss.com:2199 --port 8100*

*[helix@fys1 ~]$ jps*

*670 RestAdminApplication*

*[helix@fys1 ~]$ netstat -anp|grep 670*

*tcp 0 0 0.0.0.0:8670 0.0.0.0:\* LISTEN -*

*tcp 0 0 :::8100 :::\* LISTEN 670/java*

下面基于URL进行介绍：

## /clusters

列举所有的clusters

*GET http://10.139.9.123:8100/clusters*

*{*

*"id" : "Clusters Summary",*

*"simpleFields" : {*

*},*

*"listFields" : {*

*"clusters" : [ "MYCLUSTER" ]*

*},*

*"mapFields" : {*

*}}*

增加cluster

*curl -d 'jsonParameters={"command":"addCluster","clusterName":"TestClusters"}' -H "Content-Type: application/json" http://fys1.cmss.com:8100/clusters*

*{*

*"id" : "Clusters Summary",*

*"simpleFields" : {*

*},*

*"listFields" : {*

*"clusters" : [ "MyCluster", "MYCLUSTER", "TestClusters" ]*

*},*

*"mapFields" : {*

*}*

*}*

## /clusters/{clusterName}

查看cluster信息

*GET http://10.139.9.123:8100/clusters/MyCluster*

*{*

*"id" : "Cluster Summary",*

*"simpleFields" : {*

*"LEADER" : ""*

*},*

*"listFields" : {*

*"participants" : [ ],*

*"resources" : [ ],*

*"stateModelDefs" : [ "MasterSlave", "SchedulerTaskQueue", "Task", "STORAGE\_DEFAULT\_SM\_SCHEMATA", "LeaderStandby", "OnlineOffline" ]*

*},*

*"mapFields" : {*

*}*

*}*

在分布式controller模式下，启用或者禁用一个cluster

*curl -d 'jsonParameters={"command":"activateCluster","grandCluster":"MyControllerCluster","enabled":"true"}' -H "Content-Type: application/json" http://localhost:8100/clusters/MyCluster*

删除某个cluster

*curl -X DELETE http://localhost:8100/clusters/MyCluster*

## /clusters/{clusterName}/resourceGroups

列举所有的resources

*curl http://localhost:8100/clusters/MyCluster/resourceGroups*

详细Rest API见以下链接：

http://svn.apache.org/repos/asf/helix/site-content/0.6.8-docs/tutorial\_admin.html

http://blog.cloudera.com/blog/2013/09/distributed-systems-get-simpler-with-apache-helix/

http://www.jdon.com/45814

http://helix.apache.org/0.6.5-docs/Tutorial.html

http://helix.apache.org/0.6.8-docs/download.cgi#/a0.6.8\_Binaries

http://svn.apache.org/repos/asf/helix/site-content/0.6.8-docs/recipes/task\_dag\_execution.html