Apache Helix

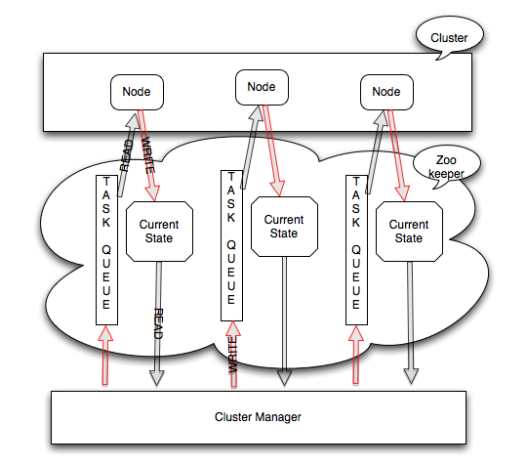
Apache Helix是一个通用的集群资源管理框架，其提供的功能包括：

* 自动给集群中的节点分配资源的partition及replica
* 节点的失败检测和恢复
* 动态的添加资源及节点，根据用户自定义的资源和分配的分配实现可插拔的负载均衡机制
* 集群的配置管理及自动发现机制

通过Helix可以将资源管理从核心的系统功能中分离处理，系统组件不需要管理整个全局资源，简化单节点的系统到可操作的分布式系统的转换难度。其系统架构如下所示：

# Apache Helix系统架构

Apache Helix是基于Zookeeper的通用的集群管理框架，其既能做分布式Task、分布式存储，还可以做服务发现、分布式锁等，简化分布式系统的设计。Apache Helix将分布式系统中的通用部分进行抽象实现，集群各个组件及概念如下图所示

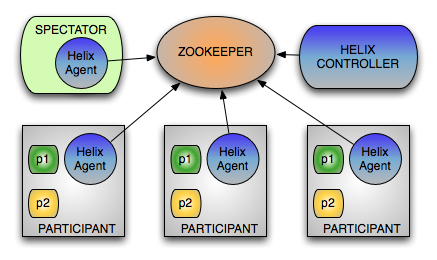


在分布式系统中，Helix提供不同节点和组件之间交互的机制，将Task分发集群中的多个节点上。Helix感知集群中组件、Task的状态变化，然后根据规则进行调整，使其达到稳定的状态。在Apache Helix中，可以将分布式系统抽象成以下组件：

1. instance，集群节点集合，每个节点都作为instances
2. resources，资源集合，资源可以是数据库、Lucene Index或者Tasks，只要有位置和状态属性的实体都可以抽象成资源
3. partitions，资源可以分成多个分区
4. replicas，每种资源的副本
5. state，副本的状态，例如Master、Slave、Leader、Standby、Online及Offline等

## 1.1 组件

在分布式系统中节点实现的功能不同，例如接收请求、发送请求及控制集群中的节点，Helix根据节点的不同角色分成以下逻辑组件，如下图所示：



* Participant，托管分布式资源
* Specator，该节点监控每个Participant的当前状态，并路由请求。例如路由器，需要感知每个节点托管了哪些分区及分区状态，并将客户端情况路由到相应的Endpoint上。
* Controller，该节点用于监控及管理所有的Participant节点，集群发生操作时时进行状态转换实现集群的稳定

根据系统的需要部署这些逻辑组件，例如Controller可以作为单独的服务来部署，也可以和Participant部署在一起，但是在集群中只能有一个Controller处于Active状态。

## 1.2 状态

Apache Helix使用状态（State）对资源进行建模，比如在Zookeeper中，节点分为leader和follower；Kafka的Partition也区分为Master,Slave。比如使用Helix组织Mysql实例构成Mysql集群，为数据库做主备，那么assignment(ideal state)的例子：

*{*

*"id" : "myDB",*

*"simpleFields" : {*

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

*"REPLICAS" : "2",*

*},*

*"mapFields" : {*

*"myDB" : {*

*"N1" : "MASTER",*

*"N2" : "SLAVE",*

*}*

*}*

*}*

**1）idealState**

在分布式系统中将资源分布式放置，需要有一种平衡的理想状态，Apache Helix让App定义资源的idealState，包括：

* partitions列表，例如64个partition
* 每个partition的replicas数目，例如3副本
* parititon分发的节点及副本的状态

例如在N1上启动myTask，理想状态可以描述为：

*{*

*"id" : "MyTask",*

*"mapFields" : {*

*"myTask" : {*

*"N1" : "ONLINE",*

*}*

*}*

*}*

Helix使用不同的算法将分区分发到节点上，默认的算法是新节点添加时分发的数目最少。

动态IdealState其实就是支持动态调整资源的状态，Helix感知集群变化后，动态调整idealState，同时出发rebalance操作，Helix支持的三种方式：

* FULL\_AUTO，全自动，根据约束条件自动调整
* SEMI\_AUTO, 半自动，用户提供一个期望的ideaState信息，Helix朝IdeaState进行调整
* CUSTOMIZED，完全用户定制，相当于jstorm中指定的supervisor调度

**2）CurrentState**

在集群中的每个Pariticipant托管资源的一个或者多个partitions，每个Partitions都有状态，比如storm中的supervisor的心跳信息，包括集群名，slot以及资源的状态等。

*{*

*"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。

3）ExternalView

客户端需要获取集群中每个分区的状态和托管分区的节点，在Helix中把集群客户端成为Specators，Helix使用ExternalView来获取所有节点的聚合信息，例如Storm的Nimbus Client，获取作业的分配情况

*{*

*"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"*

*}*

*}*

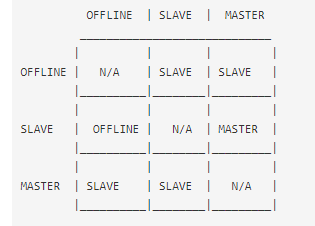
*}*

## 1.3 状态机和转换

在分布式系统中资源、分区和副本都是一致的概念，区别在于如何决定分区的分发（达到Ideal State），以及在State上的限制条件，例如Storm的assignment，调度分配槽位及执行Task等。在Helix中Controller负责状态转换，让集群中的资源达到idealState：

* 如果系统对外提供数据，所有分区的副本是相同的，状态为ONLINE或者OFFLINE
* 系统对外提供数据保存和读取操作，数据分区的状态可以为MASTER/SLAVE及OFFLINE，数据写入MASTER的分区，然后同步到SLAVE到副本上。读操作可以在任意分区上。

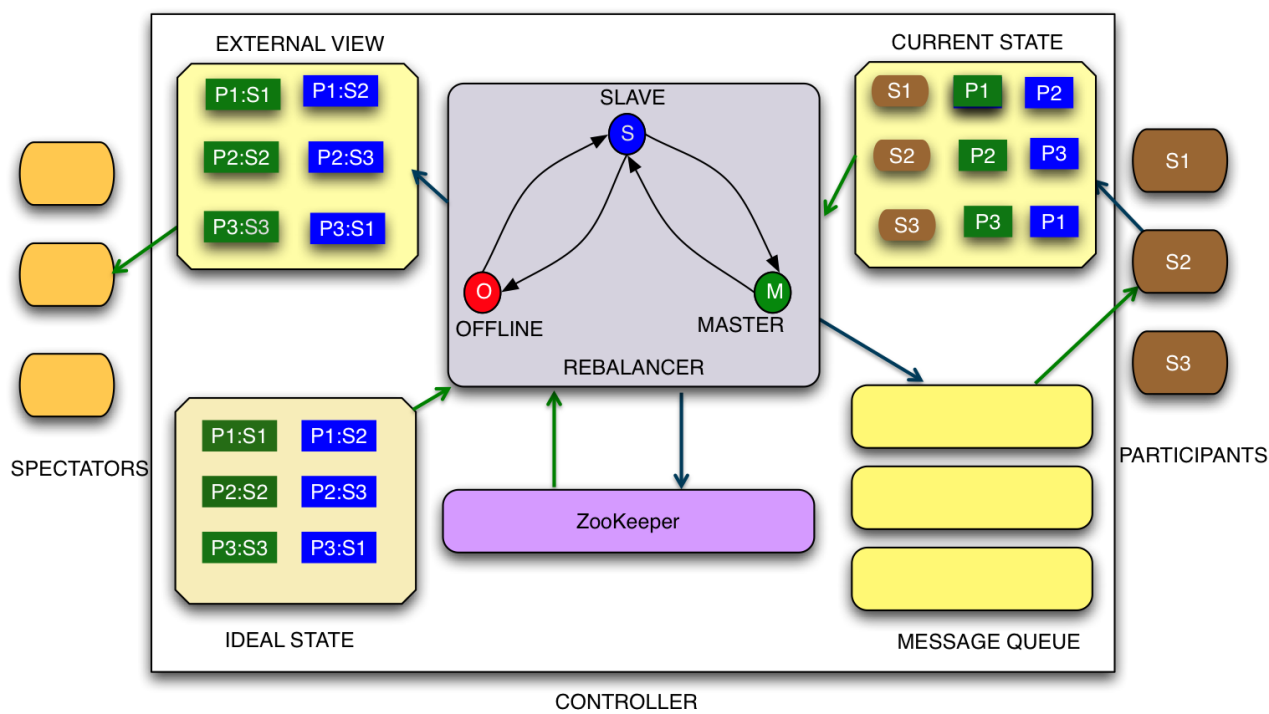
Helix使用状态机来实现状态的转换（Hadoop中的状态机），状态机模型如下：



Helix支持灵活的状态机配置，一种资源配置一个状态机，可以有多种资源配置多个状态机，例如集群管理一个数据库，同时也要管理一个索引存储系统。

## 1.4 组件间的交互

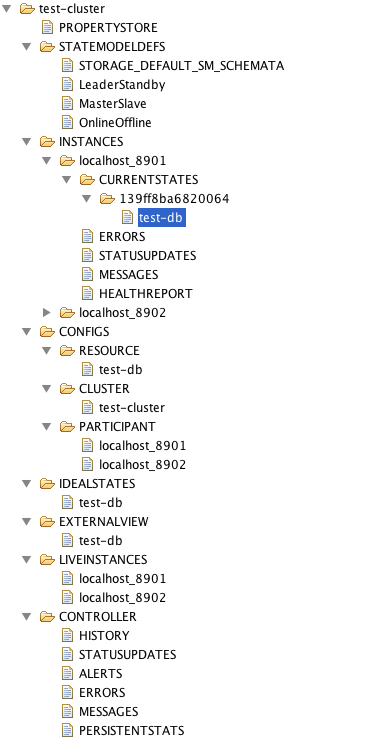
Helix集群中包含三种角色：participant，启动后注册到集群中，接收消息并更新当前的状态；Spectator，查看集群状态的变化，对客户端提供ExternalView; Controller，根据集群的状态变化控制集群，并将状态变化信息发送给participant。三者之间的交互如下图所示：



1. Participant Node Process，启动后将注册到LiveInstances，将注册消息发送到消息队列中，根据接收到的消息执行相应Task
2. Controller Process，监控idealState，当Participant宕机、启动等变化时触发操作。根据状态转换，发送消息给Participants
3. Spectator Process，当进程启动后，向Helix Agent获取ExternalViews。

## 1.5 ZNode Layout

Helix使用Zookeeper来记录集群中的运行状态，下图是test-cluster的Helix Znode结构：



其描述如下：

*Test-Cluster*

└──*PROPERTYSTORE*，应用的配置

└──*STATEMODELDEFES*，状态模型

└──*INSTANCES*，节点实例运行信息，包括当前状态和处理消息

├── *CURRENTSTATES*，

├──*SessionId*

├──r*esourceName*

├──*ERRORS*

├──*STATUSUPDATES*

├──*MESSAGES*

├──*HEALTHREPORT*

└──CONFIGS

├── RESOURCE，资源的配置

├── CLUSTER，集群配置

├── PARTICIPATN，PARTICIPANT配置

└──IDEASTATE, ideal states

└──LIVEINTANCE，live Instances

└──CONTROLLER，集群Controller的运行信息

# Apache Helix QuickStart

*git clone https://github.com/apache/helix.git*

*cd helix*

*mvn install package –DskipTests*

*chmod +x \**

通过以上命令生成Helix安装包。

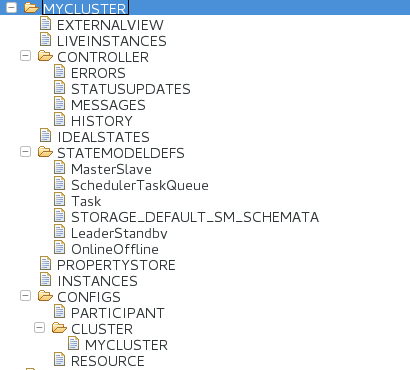
## 2.1 创建集群

使用helix-admin来执行集群管理Task，Helix也支持REST接口

1. 创建集群MYCLUSTERS

*bin/helix-admin.sh -zkSvr fys1.cmss.com:2181 --addCluster MYCLUSTER*

在ZK中创建了ZNODE目录，如下图所示：



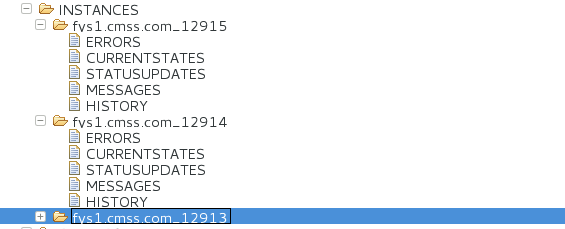
1. 向集群添加三个节点，命令如下：

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --addNode MYCLUSTER fys1.cmss.com:12913*

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --addNode MYCLUSTER fys1.cmss.com:12914*

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --addNode MYCLUSTER fys1.cmss.com:12915*

在INSTANCES中创建Znode，如下：

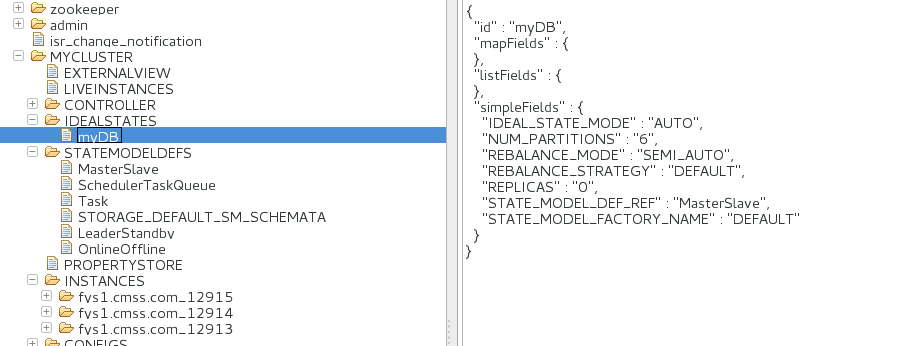


1. 定义资源及分区

该示例中资源是数据库（分成6个分区），使用MasterSlave状态模型来创建数据库的6个分区，命令如下：

*bin/helix-admin.sh -zkSvr fys1.cmss.com:2181 --addResource MYCLUSTER myDB 6 MasterSlave*

每个Partition都会有一个MASTER，创建后ZNODE信息如下：

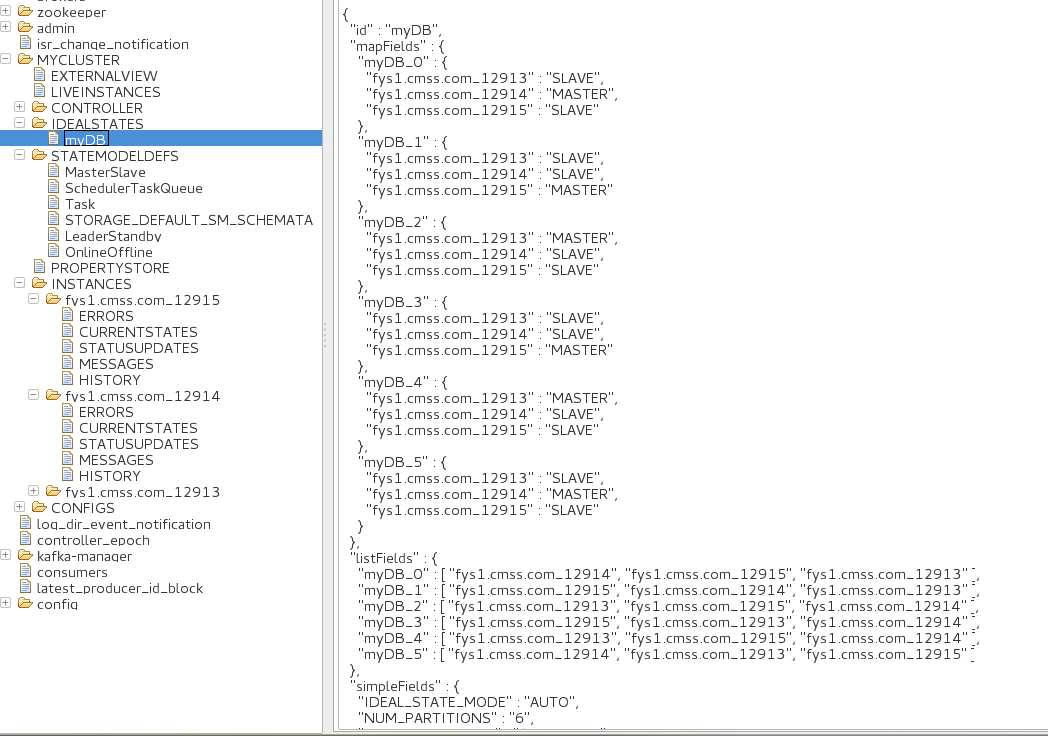


在IDEALSTATES下创建myDB，信息如图

1. 使用Helix分配分区到节点，命令如下：

*bin/helix-admin.sh -zkSvr fys1.cmss.com:2181 --rebalance MYCLUSTER myDB 3*

每个分区有3个副本，并将分区分发到集群的所有节点上，经过计算生成IdealState，信息如下：



至此已经创建好了集群。

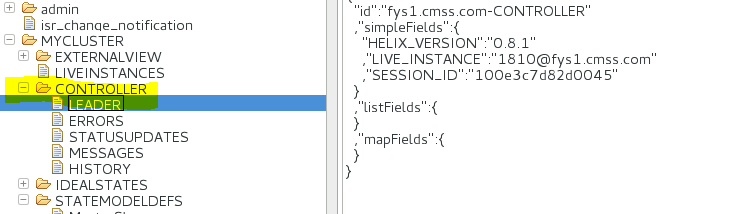
## 2.2 启动Helix Controller及Participants

在Helix 中Controller用于监控及管理所有的Participant节点，集群发生操作时时进行状态转换实现集群的稳定，

1）Controller的启动命令如下：

*bin/run-helix-controller.sh --zkSvr fys1.cmss.com:2181 --cluster MYCLUSTER 2>&1 > /tmp/controller.log &*

启动Controller组件，创建的Znode节点如下：



Controller进程如下：

*$ jps*

*1810 HelixControllerMain*

进程ID号为1810

2）启动要管理的集群节点

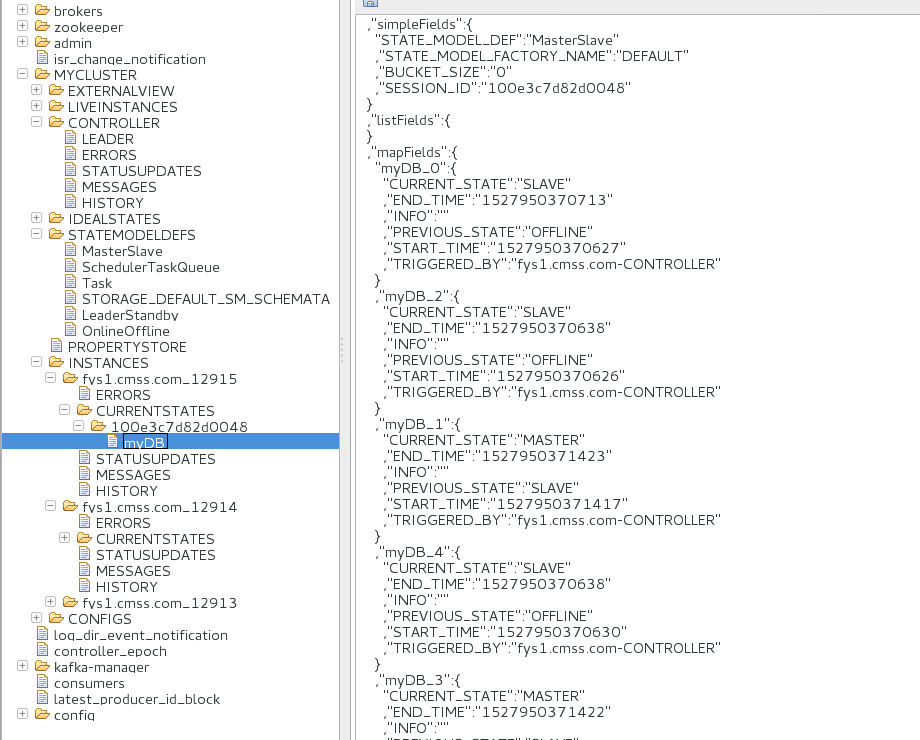
在Helix中每个节点都是Participant，Helix会将任务分发到Participants中，同时追踪节点角色和健康状况，当节点失败时也会进行相应的操作，节点的启动命令如下：

*bin/start-helix-participant.sh --zkSvr fys1.cmss.com:2181 --cluster MYCLUSTER --host fys1.cmss.com --port 12914 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12914.log &*

*bin/start-helix-participant.sh --zkSvr fys1.cmss.com:2181 --cluster MYCLUSTER --host fys1.cmss.com --port 12914 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12914.log &*

*bin/start-helix-participant.sh --zkSvr fys1.cmss.com:2181 --cluster MYCLUSTER --host fys1.cmss.com --port 12913 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12913.log &*

启动了3个节点，在Znode中节点信息如下：



在每个实例中会有当前节点状态的Znode，其中记录每个分区及状态。

## 2.3 查看集群资源信息

可以通过命令行来查看Helix集群的信息

1. 查看Cluster列表及Cluster信息，命令如下：

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 –listClusters*

*输出信息如下：*

*Existing clusters:*

*MYCLUSTER*

查询集群的详细信息：

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --listClusterInfo MYCLUSTER*

*输出信息如下：*

*Existing resources in cluster MYCLUSTER: //集群中的资源*

*myDB*

*Instances in cluster MYCLUSTER: //集群中的实例*

*fys1.cmss.com\_12915*

*fys1.cmss.com\_12914*

*fys1.cmss.com\_12913*

1. 查看Cluster Instance的详细信息

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --listInstanceInfo MYCLUSTER fys1.cmss.com\_12915*

*输出信息如下：*

*InstanceConfig: {*

*"id" : "fys1.cmss.com\_12915",*

*"mapFields" : {*

*},*

*"listFields" : {*

*},*

*"simpleFields" : {*

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

*"HELIX\_ENABLED\_TIMESTAMP" : "1527946639651",*

*"HELIX\_HOST" : "fys1.cmss.com",*

*"HELIX\_PORT" : "12915"*

*}*

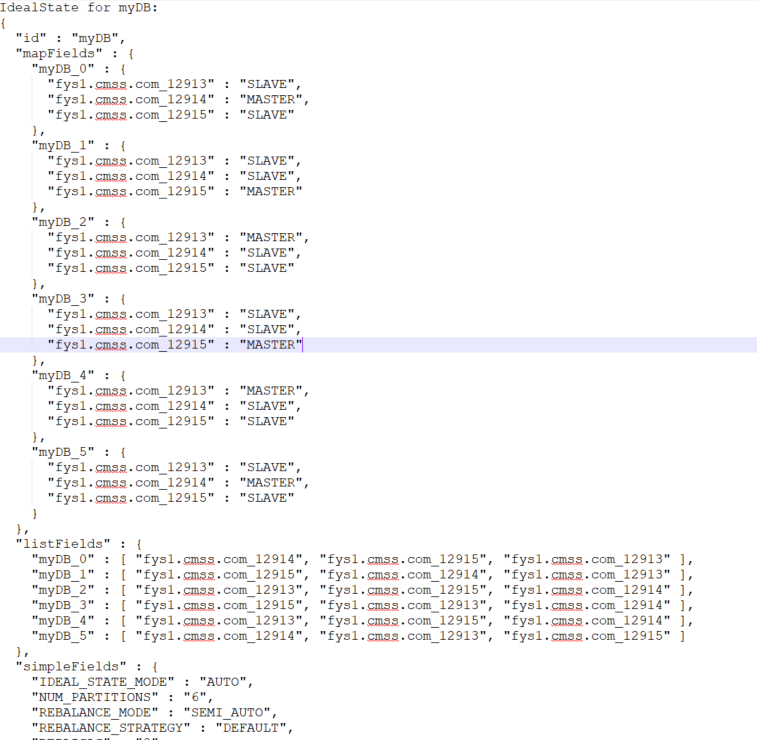
*}*

1. 查看集群资源信息，命令如下：

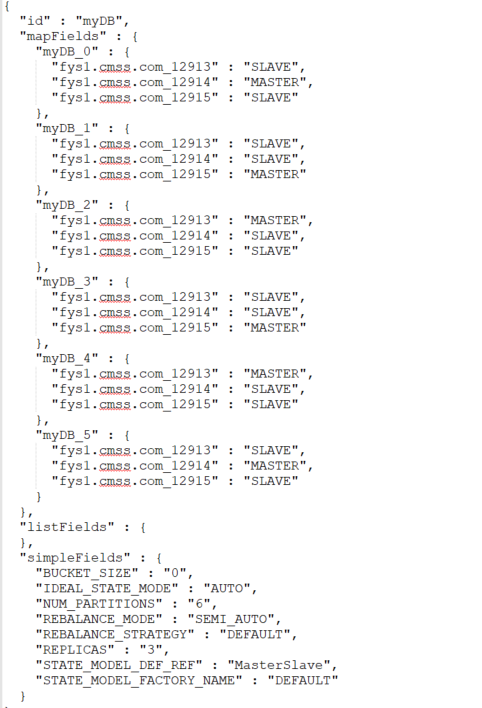
*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --listResourceInfo MYCLUSTER myDB*

信息输出如下：

* IdealState



2）资源的ExternalView



## 2.4 扩展集群

以上操作共有3个节点，每个节点上托管2个master和4个slave分区，可以通过添加节点来均衡分区，添加节点如下：

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --addNode MYCLUSTER fys1.cmss.com:12916*

*bin/start-helix-participant.sh --zkSvr fys1.cmss.com:2181 --cluster MYCLUSTER --host fys1.cmss.com --port 12916 --stateModelType MasterSlave 2>&1 > /tmp/participant\_12916.log &*

通过命令进行资源的rebalance，命令如下：

*bin/helix-admin.sh --zkSvr fys1.cmss.com:2181 --rebalance MYCLUSTER myDB 3*

均衡后资源中的分区如下：

*"listFields" : {*

*"myDB\_0" : [ "fys1.cmss.com\_12913", "fys1.cmss.com\_12916", "fys1.cmss.com\_12915" ],*

*"myDB\_1" : [ "fys1.cmss.com\_12914", "fys1.cmss.com\_12916", "fys1.cmss.com\_12913" ],*

*"myDB\_2" : [ "fys1.cmss.com\_12913", "fys1.cmss.com\_12915", "fys1.cmss.com\_12914" ],*

*"myDB\_3" : [ "fys1.cmss.com\_12915", "fys1.cmss.com\_12914", "fys1.cmss.com\_12916" ],*

*"myDB\_4" : [ "fys1.cmss.com\_12916", "fys1.cmss.com\_12914", "fys1.cmss.com\_12915" ],*

*"myDB\_5" : [ "fys1.cmss.com\_12914", "fys1.cmss.com\_12916", "fys1.cmss.com\_12915" ]*

*}*

Apache Helix提供Java API，可以通过API在实现将已有服务构建成分布式服务。

http://helix.apache.org/0.8.1-docs/Tutorial.html

http://helix.apache.org/0.8.1-docs/Quickstart.html

https://blog.csdn.net/tracymkgld/article/details/51089280

http://helix.apache.org/Architecture.html