```
#db协调节点
var db= new Sdb("localhost",11850)
#db1数据节点  group1
var db1= new Sdb("localhost",11870)
#db2数据节点  group2
var db2= new Sdb("localhost",11810)

#演示无切分，手动切分
db.foo.createCL("bar0")
#插入数据
for(var i=0;i<1000;i++){db.foo.bar0.insert({id:i,name:"pkpm-"+i})}
db.foo.bar0.count()
db1.foo.bar0.count()
db2.foo.bar0.count()
#开启切分
db.foo.bar0.enableSharding( { ShardingKey: { id: 1 }, ShardingType: "hash" ,AutoSplit:true} )
db1.foo.bar0.count()
db2.foo.bar0.count()

#演示自动切分
db.foo.createCL("bar1",{ShardingKey:{id:1}, ShardingType:"hash",AutoSplit:true,Partition:4096, Compressed:true, ReplSize:1})
for(var i=0;i<100;i++){db.foo.bar1.insert({id:i,name:"pkpm-"+i})}
db.foo.bar0.count()
db1.foo.bar1.count()
db2.foo.bar1.count()

#查看hash
db.foo.bar1.find({id:10,name:"pkpm-"+10}).explain()
db.foo.bar1.find({id:11,name:"pkpm-"+11}).explain()
db.foo.bar1.find({id:12,name:"pkpm-"+12}).explain()

for(var i=1000;i<20000;i++){db.foo.bar1.insert({id:i,name:"pkpm-"+i})}
```

```
#演示水平范围切分
#更多用于数据分析
db.foo.createCL("exam",{ShardingKey:{score:1}, ShardingType:"range",Compressed:true, ReplSize:1})
for(var i=0;i<101;i++){db.foo.exam.insert({score:i,name:"user-"+i})}
#将50<=score<60的数据切分到group1,也许还有救
#'src'、'dst'分别表示“数据原本所在复制组”、“数据将要切分到的目标复制组”
#数据切分及分区上的数据范围皆遵循左闭右开原则。即：{a:10},{a:20} 代表迁移数据范围为[10, 20)
db.foo.exam.split( "group2", "group1", { score: 50 }, { score: 60 } )
db.foo.exam.split( "group2", "group1", 50 )
```

0328

db.foo.createCL("bar0328",{ShardingKey:{id:1}, ShardingType:"hash",AutoSplit:true,Partition:4096, Compressed:true, ReplSize:1})

for(var i=0;i<100;i++){db.foo.bar0328.insert({id:i,name:"pkpm-"+i})}

http://114.116.84.219:8000

for(var i=0;i<100000;i++){db.foo.bar0328.insert({id:i,name:"pkpm-"+i})}

114.116.84.219

49.4.55.173

```
db.foo.createCL("bar2")
for(var i=0;i<1000;i++){db.foo.bar2.insert({id:i,name:"pkpm-"+i})}
db.foo.bar0.count()
db1.foo.bar0.count()
db2.foo.bar0.count()
```

```
db.foo.bar.enableSharding( { ShardingKey: { id: 1 }, ShardingType: "hash" ,AutoSplit:true} )
```