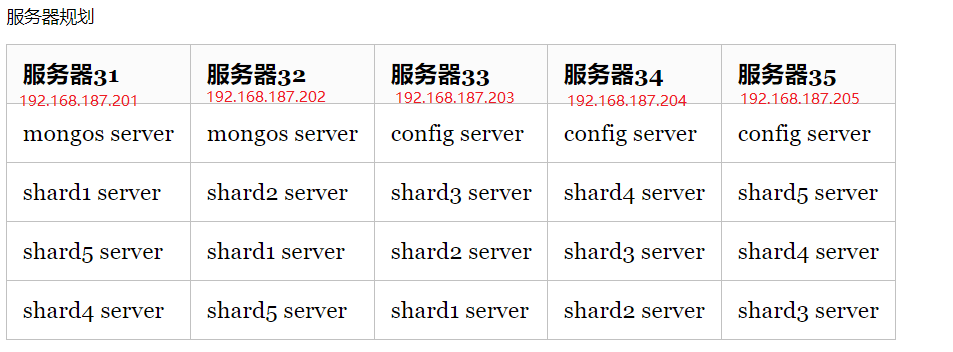
# Mongodb高可用搭建

# Mongodb节点分配规划

在Mongo的官网下载Linux版本安装包，然后解压到对应的目录下；由于资源有限，我们采用Replica Sets + Sharding方式来配置高可用。Mongodb部署节点图如下所示，当前部署在5台服务器上：



这里我说明下这个图所表达的意思：

* Shard服务器：使用Replica Sets确保每个数据节点都具有备份、自动容错转移、自动恢复的能力。
* 配置服务器：使用3个配置服务器确保元数据完整性。
* 路由进程：使用3个路由进程实现平衡，提高客户端接入性能
* 分片与副本集：Shard1、Shard2、Shard3、Shard4、Shard5组成5个分片，每个分片均为一个副本集
* 3个配置服务器进程和2个路由器进程。
* Arbiter仲裁者，是副本集中的一个MongoDB实例， 它并不保存数据。仲裁节点使用最小的资源并且不要求硬件设备。为了确保复制集中有奇数的投票成员(包括primary)，需要添加仲裁节点作为投票，否则primary不能运行时不会自动切换primary。

构建一个mongoDB Sharding Cluster需要三种角色：shard服务器(ShardServer)、配置服务器(config Server)、路由进程(Route Process)

**Shard 服务器 shard server**

Shard服务器即存储实际数据的分片，每个shard可以是一个mongod实例，也可以是一组mongod实例构成的Replica Sets。为了实现每个Shard内部的故障自动转换，MongoDB官方建议每个shard为一组Replica Sets。

**配置服务器config server**

为了将一个特定的collection存储在多个shard中，需要为该collection指定一个shard key，决定该条记录属于哪个chunk，配置服务器可以存储以下信息，每个shard节点的配置信息，每个chunk的shard key范围，chunk在各shard的分布情况，集群中所有DB和collection的sharding配置信息。

**路由进程mongos server**

它是一个前段路由，客户端由此接入，首先询问配置服务器需要到哪个shard上查询或保存记录，然后连接相应的shard执行操作，最后将结果返回给客户端，客户端只需要将原本发给mongod的查询或更新请求原封不动地发给路由进程，而不必关心所操作的记录存储在哪个shard上。

# 配置Shard + Replica Sets

下面的命令默认都使用root用户进行操作

## 2.1系统配置

1. **修改系统配置文件/etc/security/limits.conf和/etc/profile**

修改/etc/security/limits.conf 持久化设置允许用户/进程打开文件句柄数，这一步需要重启系统，不然不起作用

*\* soft nofile 1048576*

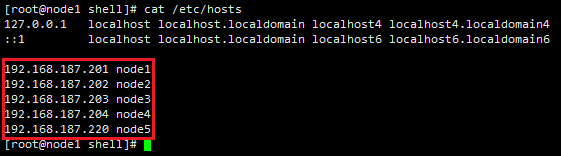
*\* hard nofile 1048576*

*\* soft nproc 524288*

*\* hard nproc 524288*

修改/etc/profile，在最后添加*ulimit -s 1024*，然后保存并*source /etc/profile*

1. **每台机器修改/etc/hosts和hostname，并在/etc/hosts添加集群所有的ip及对应的hostname，如图：**



1. **关闭每台机器的防火墙：**

chkconfig iptables off && service iptables stop

使用命令查看chkconfig –list 是否设置自动启动为关闭

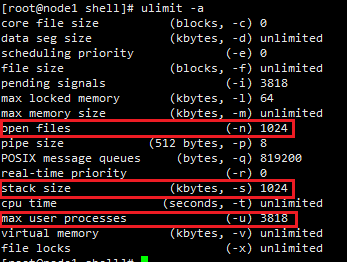


使用命令查看防火墙是否关闭service iptables status



注意：完成上面操作后重启每台机器

使用命令ulimit -a 查看



要保证open files对应的值为*1048576*

Stack size对应的值为1024

Max user processes对应的值为*524288*

然后接着执行下面命令：

1. **使用以下命令修改Linux操作系统参数**

系统全局允许分配的最大文件句柄数:

*sysctl -w fs.file-max=2097152*

*sysctl -w fs.nr\_open=2097152*

*echo 2097152 > /proc/sys/fs/nr\_open*

允许当前会话/进程打开文件句柄数:

*ulimit -n 1048576*

修改 ‘fs.file-max’ 设置到 /etc/sysctl.conf 文件:

*fs.file-max = 1048576*

1. **使用以下命令修改TCP 协议栈网络参数**

并发连接 backlog 设置:

*sysctl -w net.core.somaxconn=32768*

*sysctl -w net.ipv4.tcp\_max\_syn\_backlog=16384*

*sysctl -w net.core.netdev\_max\_backlog=16384*

可用知名端口范围:

*sysctl -w net.ipv4.ip\_local\_port\_range='80 65535'*

TCP Socket 读写 Buffer 设置:

*sysctl -w net.core.rmem\_default=262144*

*sysctl -w net.core.wmem\_default=262144*

*sysctl -w net.core.rmem\_max=16777216*

*sysctl -w net.core.wmem\_max=16777216*

*sysctl -w net.core.optmem\_max=16777216*

*#sysctl -w net.ipv4.tcp\_mem='16777216 16777216 16777216'*

*sysctl -w net.ipv4.tcp\_rmem='1024 4096 16777216'*

*sysctl -w net.ipv4.tcp\_wmem='1024 4096 16777216'*

修改系统内核参数：

*echo "never" > /sys/kernel/mm/transparent\_hugepage/enabled*

*echo "never" > /sys/kernel/mm/transparent\_hugepage/defrag*

TCP 连接追踪设置:

*sysctl -w net.nf\_conntrack\_max=1000000*

*sysctl -w net.netfilter.nf\_conntrack\_max=1000000*

*sysctl -w net.netfilter.nf\_conntrack\_tcp\_timeout\_time\_wait=30*

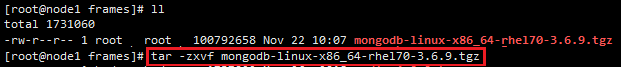
## 2.2 安装

统一分别在三台机器的/opt/app/mongoCluster369部署mongodb集群。

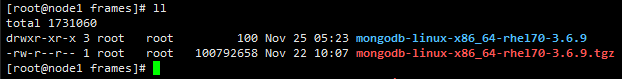
**开始安装：**

在linux官网下载mongodb的安装包，目前所用安装包为：mongodb-linux-x86\_64-rhel70-3.6.9.tgz

使用命令*tar -zxvf mongodb-linux-x86\_64-rhel70-3.6.9.tgz* 解压mongodb

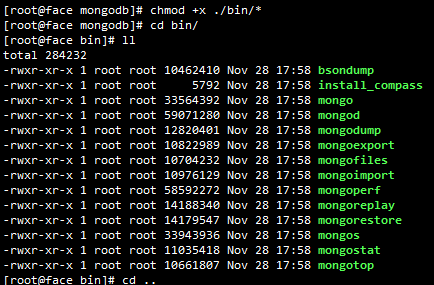


解压后查看如下：



把mongodb-linux-x86\_64-rhel70-3.6.9移动到指定目录，当前指定目录为/opt/app/，并改名为mongodb

给mongodb/bin目录下的所有mongodb命令加上x(执行权限)



在mongodbCluster目录中建立conf(配置文件)、mongos(路由)、config(配置)、shard1、shard2、shard3、shard4、shard5(三个切片)，因为mongos不存储数据，只需要建立日志文件即可。

由第一节点mongodb节点规划图所示：

在服务器31，即node1上安装mongos、shard1、shard4、shard5

*mkdir -p /opt/app/mongodbCluster/conf*

*mkdir -p /opt/app/mongodbCluster/mongos/log*

*mkdir -p /opt/app/mongodbCluster/mongos/pid*

*mkdir -p /opt/app/mongodbCluster/shard1/data*

*mkdir -p /opt/app/mongodbCluster/shard1/log*

*mkdir -p /opt/app/mongodbCluster/shard1/pid*

*mkdir -p /opt/app/mongodbCluster/shard4/data*

*mkdir -p /opt/app/mongodbCluster/shard4/log*

*mkdir -p /opt/app/mongodbCluster/shard4/pid*

*mkdir -p /opt/app/mongodbCluster/shard5/data*

*mkdir -p /opt/app/mongodbCluster/shard5/log*

*mkdir -p /opt/app/mongodbCluster/shard5/pid*

在服务器32，即node2上安装mongos、shard1、shard2、shard5

*mkdir -p /opt/app/mongodbCluster/conf*

*mkdir -p /opt/app/mongodbCluster/mongos/log*

*mkdir -p /opt/app/mongodbCluster/mongos/pid*

*mkdir -p /opt/app/mongodbCluster/shard1/data*

*mkdir -p /opt/app/mongodbCluster/shard1/log*

*mkdir -p /opt/app/mongodbCluster/shard1/pid*

*mkdir -p /opt/app/mongodbCluster/shard2/data*

*mkdir -p /opt/app/mongodbCluster/shard2/log*

*mkdir -p /opt/app/mongodbCluster/shard2/pid*

*mkdir -p /opt/app/mongodbCluster/shard5/data*

*mkdir -p /opt/app/mongodbCluster/shard5/log*

*mkdir -p /opt/app/mongodbCluster/shard5/pid*

在服务器33，即node3上安装config、shard1、shard2、shard3

*mkdir -p /opt/app/mongodbCluster/conf*

*mkdir -p /opt/app/mongodbCluster/config/data*

*mkdir -p /opt/app/mongodbCluster/config/log*

*mkdir -p /opt/app/mongodbCluster/config/pid*

*mkdir -p /opt/app/mongodbCluster/shard1/data*

*mkdir -p /opt/app/mongodbCluster/shard1/log*

*mkdir -p /opt/app/mongodbCluster/shard1/pid*

*mkdir -p /opt/app/mongodbCluster/shard2/data*

*mkdir -p /opt/app/mongodbCluster/shard2/log*

*mkdir -p /opt/app/mongodbCluster/shard2/pid*

*mkdir -p /opt/app/mongodbCluster/shard3/data*

*mkdir -p /opt/app/mongodbCluster/shard3/log*

*mkdir -p /opt/app/mongodbCluster/shard3/pid*

在服务器34，即node4上安装config、shard2、shard3、shard4

*mkdir -p /opt/app/mongodbCluster/conf*

*mkdir -p /opt/app/mongodbCluster/config/data*

*mkdir -p /opt/app/mongodbCluster/config/log*

*mkdir -p /opt/app/mongodbCluster/config/pid*

*mkdir -p /opt/app/mongodbCluster/shard2/data*

*mkdir -p /opt/app/mongodbCluster/shard2/log*

*mkdir -p /opt/app/mongodbCluster/shard2/pid*

*mkdir -p /opt/app/mongodbCluster/shard3/data*

*mkdir -p /opt/app/mongodbCluster/shard3/log*

*mkdir -p /opt/app/mongodbCluster/shard3/pid*

*mkdir -p /opt/app/mongodbCluster/shard4/data*

*mkdir -p /opt/app/mongodbCluster/shard4/log*

*mkdir -p /opt/app/mongodbCluster/shard4/pid*

在服务器35，即node5上安装config、shard3、shard4、shard5

*mkdir -p /opt/app/mongodbCluster/conf*

*mkdir -p /opt/app/mongodbCluster/config/data*

*mkdir -p /opt/app/mongodbCluster/config/log*

*mkdir -p /opt/app/mongodbCluster/config/pid*

*mkdir -p /opt/app/mongodbCluster/shard3/data*

*mkdir -p /opt/app/mongodbCluster/shard3/log*

*mkdir -p /opt/app/mongodbCluster/shard3/pid*

*mkdir -p /opt/app/mongodbCluster/shard4/data*

*mkdir -p /opt/app/mongodbCluster/shard4/log*

*mkdir -p /opt/app/mongodbCluster/shard4/pid*

*mkdir -p /opt/app/mongodbCluster/shard5/data*

*mkdir -p /opt/app/mongodbCluster/shard5/log*

*mkdir -p /opt/app/mongodbCluster/shard5/pid*

## 2.3 配置分片副本集

**在服务器31、32、33上设置第一个分片副本集**

这里以服务器31的配置文件为例

*vi /opt/app/mongodbCluster/conf/shard1.conf*

配置文件内容为：

#配置文件内容

dbpath=/opt/app/mongodbCluster/shard1/data

logpath=/opt/app/mongodbCluster/shard1/log/shard1.log

pidfilepath=/opt/app/mongodbCluster/shard1/pid/shard1.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.201

port=29060

oplogSize=10000

fork=true

noprealloc=true

#副本集名称

replSet=shard1

#declare this is a shard db of a cluster

shardsvr=true

#设置最大连接数

maxConns=20000

服务器32、33的shard1配置文件除了bind\_ip相应的修改为192.168.187.202、192.168.187.203，其他的配置属性一致

**在服务器32、33、34上设置第二个分片副本集**

这里以服务器32的配置文件为例：

*vi /opt/app/mongodbCluster/conf/shard2.conf:*

# 配置文件内容

dbpath=/opt/app/mongodbCluster/shard2/data

logpath=/opt/app/mongodbCluster/shard2/log/shard2.log

pidfilepath=/opt/app/mongodbCluster/shard2/pid/shard2.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.202

port=29070

oplogSize=10000

fork=true

noprealloc=true

#副本集名称

replSet=shard2

#declare this is a shard db of a cluster

shardsvr=true

#设置最大连接数

maxConns=20000

服务器33、34的shard2配置文件除了bind\_ip相应的修改为192.168.187.203、192.168.187.204，其他的配置属性一致

**在服务器33、34、35上设置第三个分片副本集**

这里以服务器33的配置文件为例：

*vi /opt/app/mongodbCluster/conf/shard3.conf*

# 配置文件内容

dbpath=/opt/app/mongodbCluster/shard3/data

logpath=/opt/app/mongodbCluster/shard3/log/shard3.log

pidfilepath=/opt/app/mongodbCluster/shard3/pid/shard3.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.203

port=29080

oplogSize=10000

fork=true

noprealloc=true

#副本集名称

replSet=shard3

#declare this is a shard db of a cluster

shardsvr=true

#设置最大连接数

maxConns=20000

服务器34、35的shard3配置文件除了bind\_ip相应的修改为192.168.187.204、192.168.187.205，其他的配置属性一致

**在服务器34、35、31上设置第四个分片副本集**

这里以服务器34的配置文件为例

*vi /opt/app/mongodbCluster/conf/shard4.conf*

# 配置文件内容

dbpath=/opt/app/mongodbCluster/shard4/data

logpath=/opt/app/mongodbCluster/shard4/log/shard4.log

pidfilepath=/opt/app/mongodbCluster/shard4/pid/shard4.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.204

port=29090

oplogSize=10000

fork=true

noprealloc=true

#副本集名称

replSet=shard4

#declare this is a shard db of a cluster

shardsvr=true

#设置最大连接数

maxConns=20000

服务器35、31的shard4配置文件除了bind\_ip相应的修改为192.168.187.205、192.168.187.201，其他的配置属性一致

**在服务器35、31、32上设置第五个分片副本集**

这里以服务器35的配置文件为例

vi /opt/app/mongodbCluster/conf/shard5.conf

# 配置文件内容

dbpath=/opt/app/mongodbCluster/shard5/data

logpath=/opt/app/mongodbCluster/shard5/log/shard5.log

pidfilepath=/opt/app/mongodbCluster/shard5/pid/shard5.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.205

port=29100

oplogSize=10000

fork=true

noprealloc=true

#副本集名称

replSet=shard5

#declare this is a shard db of a cluster

shardsvr=true

#设置最大连接数

maxConns=20000

服务器31、32的shard3配置文件除了bind\_ip相应的修改为192.168.187.201、192.168.187.202，其他的配置属性一致

## 2.4 config server配置服务器

在服务器33、34、35上设置config副本集

这里以服务器35的配置文件为例:

*vi /opt/app/mongodbCluster/conf/config.conf*

# 配置文件内容

dbpath=/opt/app/mongodbCluster/config/data

logpath=/opt/app/mongodbCluster/config/log/config.log

pidfilepath=/opt/app/mongodbCluster/config/pid/config.pid

directoryperdb=true

logappend=true

bind\_ip=192.168.187.203

port=29040

oplogSize=10000

fork=true

#副本集名称

replSet=configs

#declare this is a shard db of a cluster

configsvr=true

#设置最大连接数

maxConns=20000

服务器34、35的config配置文件除了bind\_ip相应的修改为192.168.187.204、192.168.187.205，其他的配置属性一致

## 2.5 配置路由服务器mongos

在服务器31、32上设置mongos

这里以服务器31的配置文件为例:

*vi /opt/app/mongodbCluster/conf/mongos.conf*

# 配置文件内容

logpath=/opt/app/mongodbCluster/mongos/log/mongos.log

pidfilepath=/opt/app/mongodbCluster/mongos/pid/mongos.pid

logappend=true

bind\_ip=192.168.187.201

port=29050

fork=true

#监听的配置服务器，只能有1个或3个 configs为配置服务器的副本集名字

configdb=configs/192.168.187.203:29040,192.168.187.204:29040,192.168.187.205:29040

#设置最大连接数

maxConns=20000

服务器32的mongos配置文件除了bind\_ip相应的修改为192.168.187.202，其他的配置属性一致

**参数说明：**

dbpath：数据存放目录

logpath：日志存放路径

pidfilepath：进程文件，方便停止mongodb

logappend：以追加的方式记录日志

directoryperdb：为每一个数据库按照数据库名建立文件夹

replSet：replica set的名字

bind\_ip：mongodb所绑定的ip地址

port：mongodb进程所使用的端口号，默认为27017

fork：以后台方式运行进程

oplogSize：mongodb操作日志文件的最大大小。单位为Mb，默认为硬盘剩余空间的5%

shardsvr：分片节点

configsvr：配置服务节点

configdb：配置config节点到route节点

journal：写日志

smallfiles：当提示空间不够时添加此参数

noprealloc：预分配方式，使用预分配方式来保证写入性能的稳定，预分配在后台运行，并且每个预分配的文件都用0进行填充。这会让MongoDB始终保持额外的空间和空余的数据文件，从而避免了数据增长过快而带来的分配磁盘空间引起的阻塞。设置noprealloc=true来禁用预分配的数据文件，会缩短启动时间，但在正常操作过程中，可能会导致性能显著下降。

## 2.7 启动mongodb集群

先启动config配置服务器和shard分片服务器，后启动mongos路由实例。

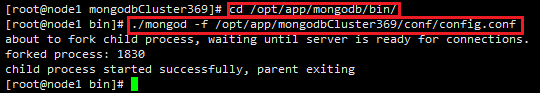
### 2.7.1启动配置服务器

**启动服务器33、34、35的config server**

33、34、35服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

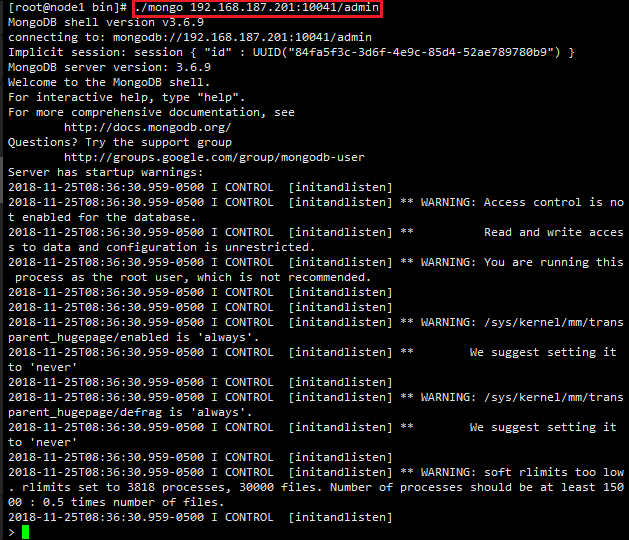
*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/config.conf*



登录任意一台服务器，初始化配置副本集

登录连接命令：./mongo 192.168.187.203:29040/admin



配置如下内容：

> config = {

... \_id : "configs",

... members : [

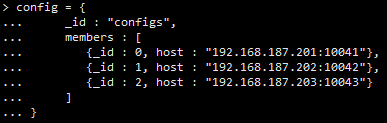
... {\_id : 0, host : "192.168.187.203:29040"},

... {\_id : 1, host : "192.168.187.204:29040"},

... {\_id : 2, host : "192.168.187.205:29040"}

... ]

... }



初始化副本集

> rs.initiate(config)

其中，“\_id” : “configs” 应与配置文件中配置的replSet一致，“members”中的“host”为三个节点的ip和port

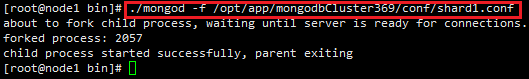
### 2.7.2 启动分片服务器

* **启动服务器31、32、33的shard1 server**

31、32、33服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

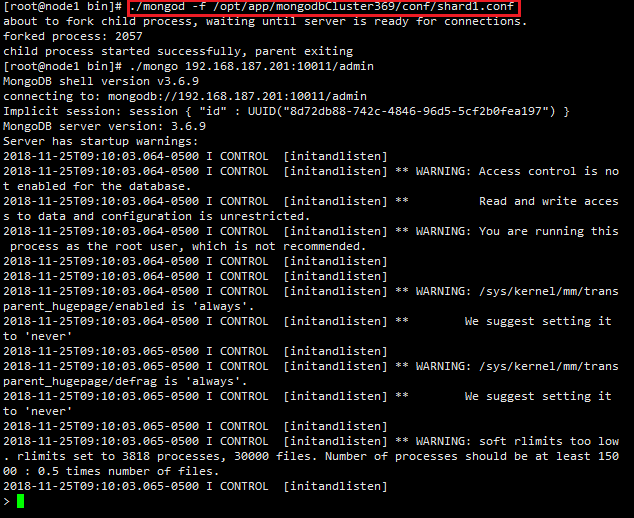
*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/shard1.conf*



登录192.168.187.201一台服务器，初始化分片副本集

登录连接命令：./mongo 192.168.187.201:29060/admin



配置如下内容：

> config = {

... \_id : "shard1",

... members : [

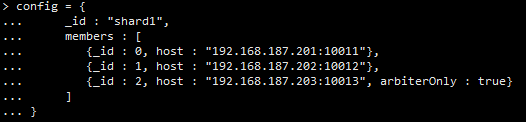
... {\_id : 0, host : "192.168.187.201:29060",priority:2},

... {\_id : 1, host : "192.168.187.202:29060",priority:1},

... {\_id : 2, host : "192.168.187.203:29060", arbiterOnly : true}

... ]

... }



初始化副本集

> rs.initiate(config)

第三个节点的“arbiterOnly”:true代表其为仲裁节点。

使用*exit*命令退出mongo的shell操作界面

* **启动服务器32、33、34的shard2 server**

32、33、34服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/shard2.conf*

登录192.168.187.202一台服务器，初始化分片副本集

登录连接命令：./mongo 192.168.187.202:29070/admin

配置如下内容：

> config = {

... \_id : "shard2",

... members : [

... {\_id : 0, host : "192.168.187.202:29070", priority:2},

... {\_id : 1, host : "192.168.187.203:29070", priority:1},

... {\_id : 2, host : "192.168.187.204:29070", arbiterOnly : true}

... ]

... }

初始化副本集

> rs.initiate(config)

使用*exit*命令退出mongo的shell操作界面

* **启动服务器33、34、35的shard3 server**

33、34、35服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/shard3.conf*

登录192.168.187.203一台服务器，初始化分片副本集

登录连接命令：./mongo 192.168.187.203:29080/admin

配置如下内容：

> config = {

... \_id : "shard3",

... members : [

... {\_id : 0, host : "192.168.187.203:29080",priority:2},

... {\_id : 1, host : "192.168.187.204:29080", priority:1},

... {\_id : 2, host : "192.168.187.205:29080", arbiterOnly : true}

... ]

... }

初始化副本集

> rs.initiate(config)

使用*exit*命令退出mongo的shell操作界面

* **启动服务器34、35、31的shard4 server**

34、35、31服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/shard4.conf*

登录192.168.187.204一台服务器，初始化分片副本集

登录连接命令：./mongo 192.168.187.204:29090/admin

配置如下内容：

> config = {

... \_id : "shard4",

... members : [

... {\_id : 0, host : "192.168.187.204:29090", priority:2},

... {\_id : 1, host : "192.168.187.205:29090", priority:1},

... {\_id : 2, host : "192.168.187.201:29090", arbiterOnly : true}

... ]

... }

初始化副本集

> rs.initiate(config)

使用*exit*命令退出mongo的shell操作界面

* **启动服务器35、31、32的shard5 server**

35、31、32服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

*cd /opt/app/mongodb/bin*

*./mongod -f /opt/app/mongodbCluster/conf/shard5.conf*

登录192.168.187.205一台服务器，初始化分片副本集

登录连接命令：./mongo 192.168.187.205:29100/admin

配置如下内容：

> config = {

... \_id : "shard3",

... members : [

... {\_id : 0, host : "192.168.187.205:29080",priority:2},

... {\_id : 1, host : "192.168.187.201:29080", priority:1},

... {\_id : 2, host : "192.168.187.202:29080", arbiterOnly : true}

... ]

... }

初始化副本集

> rs.initiate(config)

使用*exit*命令退出mongo的shell操作界面

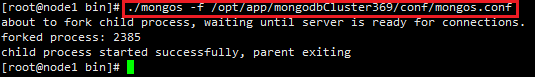
### 2.7.3启动路由实例

**启动服务器31、32的mongos server**

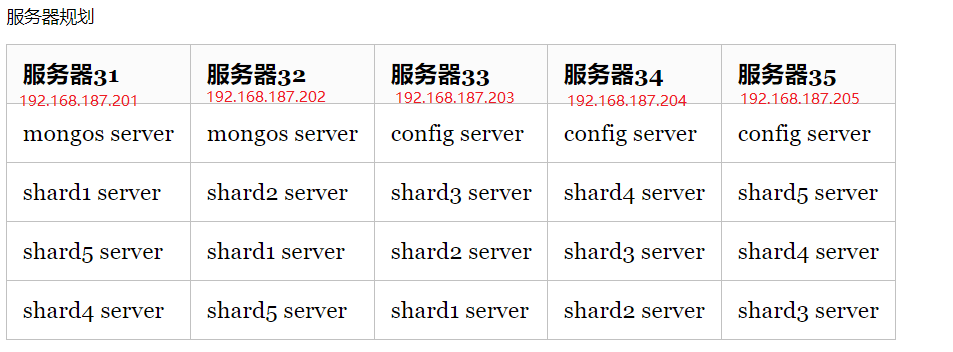
31、32服务器分别进入mongodb的安装包目录/opt/app/mongodb，使用如下命令启动：

*cd /opt/app/mongodb/bin*

*./mongos -f /opt/app/mongodbCluster/conf/mongos.conf*

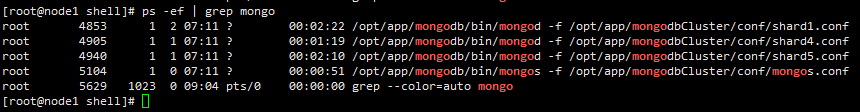


### 2.7.4 查看每台机器启动的mongod实例是否正确

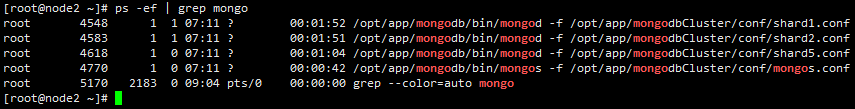


对比上面每台机器应该启动的mongo实例，查看每台机器启动的是否正确：

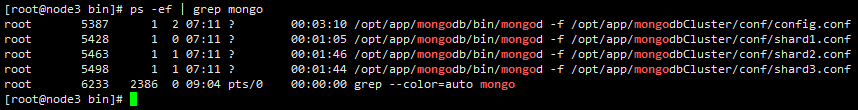
服务器31（192.168.187.201）：



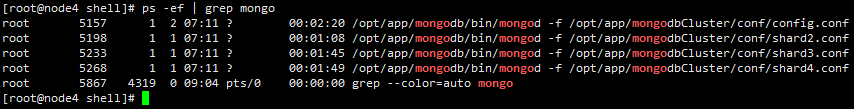
服务器32（192.168.187.202）：



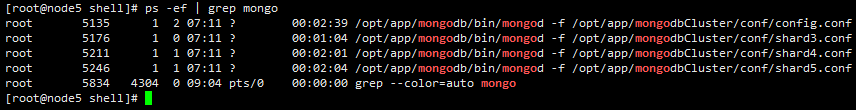
服务器33（192.168.187.203）：



服务器34（192.168.187.204）：



服务器35（192.168.187.205）：

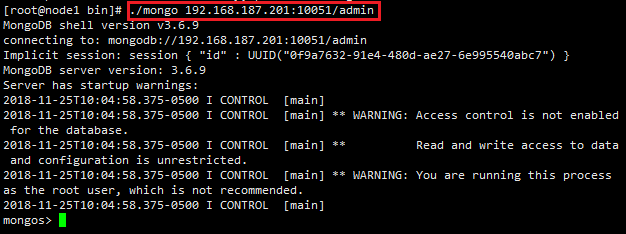


## 2.8 启用分片

目前搭建了mongodb配置服务器、路由服务器、各个分片服务器，不过应用程序连接到mongos路由服务器并不能使用分片机制，还需要在程序里设置分片配置，让分片生效。

登录任意一台mongos，这里以192.168.187.201:29050为例：

登录连接命令：./mongo 192.168.187.201:29050/admin



配置如下内容，串联路由服务器与切片副本集：

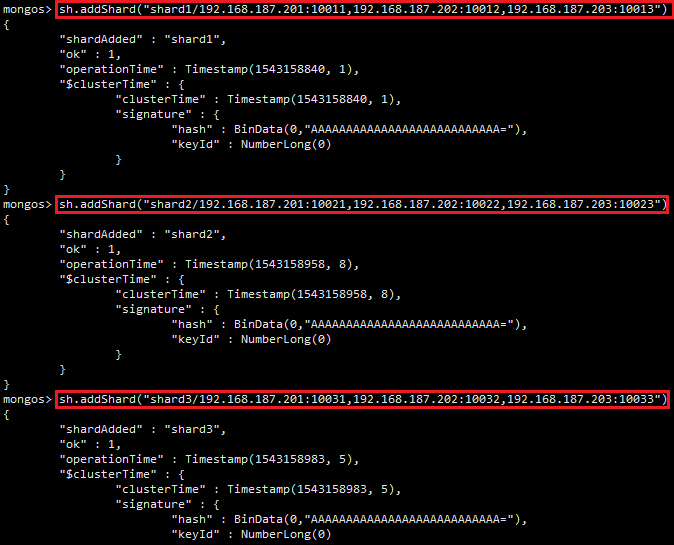
*sh.addShard("shard1/192.168.187.201:29060,192.168.187.202:29060,192.168.187.203:29060")*

*sh.addShard("shard2/192.168.187.202:29070,192.168.187.203:29070,192.168.187.204:29070")*

*sh.addShard("shard3/192.168.187.203:29080,192.168.187.204:29080,192.168.187.205:29080")*

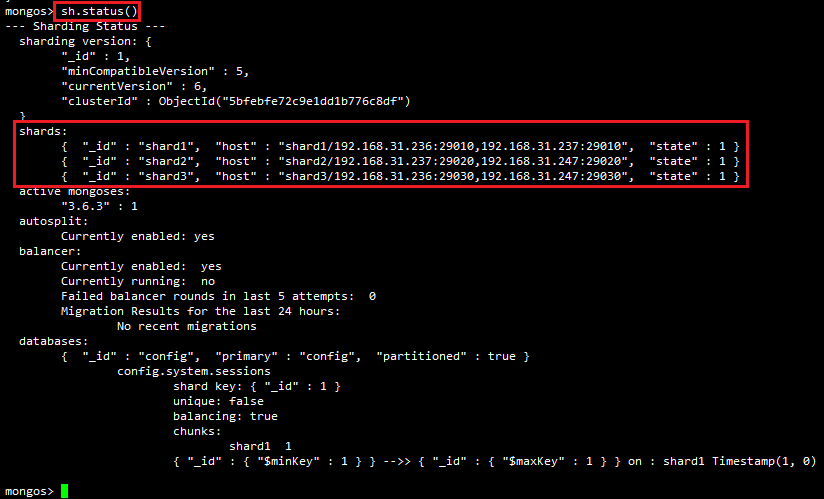
*sh.addShard("shard4/192.168.187.204:29090,192.168.187.205:29090,192.168.187.201:29090")*

*sh.addShard("shard5/192.168.187.205:29100,192.168.187.201:29100,192.168.187.202:29100")*



查看集群状态，如图所示：

*sh.status()*



## 2.9 指定数据库与集合分片生效

目前配置服务、路由服务、分片服务、副本集服务都已经串联起来了，但是我们的目的是希望插入数据、数据能够自动分片。连接在mongos上，准备让指定的数据库、指定的集合分片生效。

接着上面2.6的步骤，不用退出mongos的操作界面

指定数据库meerkat-test-v104分片生效：

db.runCommand({enablesharding : "meerkat-test-v104"});

db.runCommand({shardcollection : "meerkat-test-v104.account", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.alarm", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.blackClass", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.camera", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.case", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.collisionStatic", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.collisionTask", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.dispositionTask", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.face", key : {imsi : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceCollisionRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceCollsionAnalyze", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceFollowRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceFollowAnalyze", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceImage", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceImageExpires", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceImageTemp", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceImsiStatic", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.faceTrace", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.fileUploadHistory", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.followTask", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.ftpGroupConfig", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.hotSpotStatic", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imageSearchTask", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiCollisionRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiCollsionAnalyze", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiDetail", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiDevice", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiFaceRec", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiFollowAnalyze", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiFollowRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiPersonList", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiRecordExpires", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.imsiRecordTemp", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.mobile", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.person", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.place", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sparkJob", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sysGroup", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sysPermission", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sysPermission\_copy", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sysRole", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.sysUser", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.translateImsiRecord", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.userOperateLog", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.warning", key : {\_id : "hashed"}});

db.runCommand({shardcollection : "meerkat-test-v104.warningStatic", key : {\_id : "hashed"}});

指定数据库*face-manager*分片生效：

db.runCommand({enablesharding : " *face-manager* "});

db.runCommand({shardcollection : " *face-manager*.face", key : {\_id : "hashed"}});

db.runCommand({shardcollection : " *face-manager*.imsiFaceRel", key : {\_id : "hashed"}});

指定数据库*fusion*分片生效：

db.runCommand({enablesharding : " *fusion* "});

db.runCommand({shardcollection : " *fusion*.terminate", key : {\_id : "hashed"}});

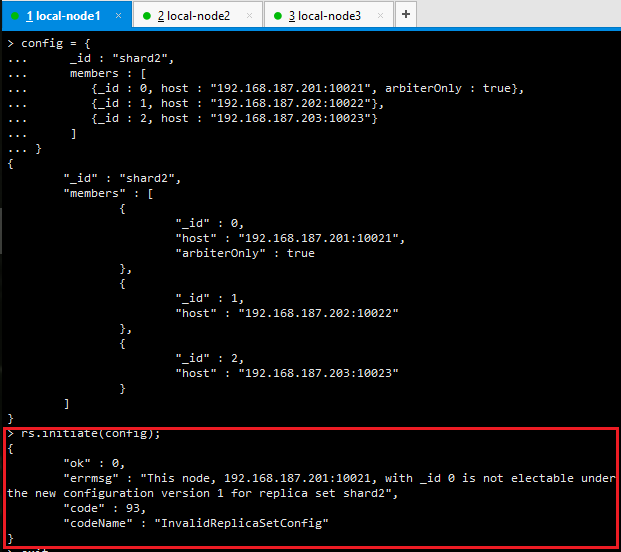
# 问题发现：



这与linux默认进程能打开最大文件数有关，可以通过ulimit解决.mongodb最大连接是20000

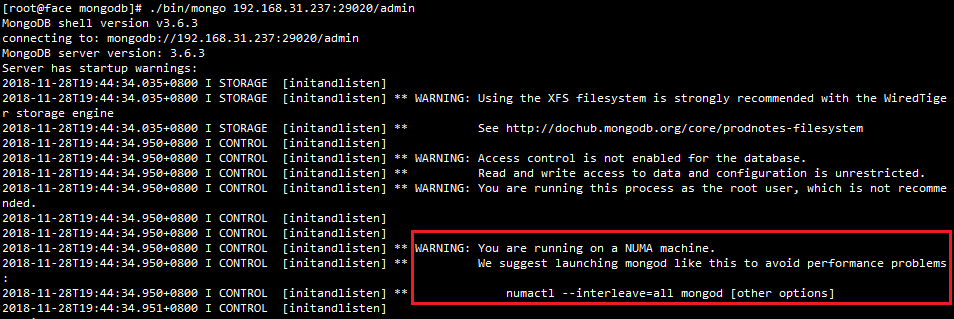
解决：ulimit -n 30000

1. 报错：is not electable under the new configuration version 1



解决：如果你设置的第一个节点是仲裁节点的话,那么设置登录设置节点状态的哪个客户端不能是仲裁节点,简单做法 换一个节点

3．Mongodb的NUMA问题



解决方案：

修改内核参数：echo 0 > /proc/sys/vm/zone\_reclaim\_mode

在原启动命令前面加numactl –interleave=all

numactl --interleave=all mongod --config /mongodb/mongod.conf

而mongos不用加numactl –interleave=all

参考资料：

http://www.ityouknow.com/mongodb/2017/08/05/mongodb-cluster-setup.html

https://www.cnblogs.com/smartloli/p/4305739.html

https://blog.csdn.net/caofeiliju/article/details/80193997