**MongoDB 集群安装部署**

# 文档说明

该文档用于说明如何基于docker安装搭建MongoDB集群，目前该自动化脚本只支持安（n分片m副本集群+1配置服务m副本集群+1mongos路由），默认安装的是（2分片3副本集群+1配置服务3副本集群+1mongos路由），不太理解这部分说明的使用者，可不必过多了解，按照后续步骤操作即可。目前脚本已测试和使用的环境是基于CentOS 7

# 前置条件准备

## 操作系统要求

一般的Linux系统即可，其次最好应该是网络连通的。

## 脚本文件介绍说明

脚本目录中应该包含如下文件：

|  |  |
| --- | --- |
| 文件名 | 文件说明 |
| mongo.properties | 包含集群基本配置信息，一般第一次需要修改 |
| sharding.sh | 包含需要在mongos中执行的分片脚本信息，一般第一次需要修改 |
| mongod.conf.yaml | mongod/mongos 服务启动依赖的配置脚本，一般不需要修改 |
| replset.yaml | sharding/configservice 副本集群 docker 形式定义模板，一般不需要修改 |
| stack.init.yaml | mongos 的 docker 形式定义模板，一般不需要修改 |
| prepare.sh | mongo集群前置环境搭建脚本，不需要修改，直接运行即可搭建好前置环境。可多次运行 |
| install.sh | 集群安装脚本，一般不需要修改 |
| Readme.md | 对集群搭建的简单介绍说明文件 |
| docker-ce-18.06.0.ce-3.el7.x86\_64.rpm | Docker rpm安装包 |
| docker-compose-Linux-x86\_64 | docker-compose 二进制可执行文件 |
| image.tar | MongoDB 4.0 版本都docker镜像 |

## 搭建前置环境

进入脚本文件prepare.sh所在目录，运行：

|  |
| --- |
| [root@node docker-shell]# ./prepare.sh |

看到输出结果包含类似如下信息：

|  |
| --- |
| $ docker --version  Docker version 18.06.0-ce, build 0ffa825 |

|  |
| --- |
| $ docker load  Loaded image: mongo:4.0  Loaded image: mongo-express:latest |

|  |
| --- |
| $ docker-compose --version  docker-compose version 1.22.0, build f46880fe |

以及最后输出如下结果：

|  |
| --- |
| ----------------------  mongo 集群安装依赖的 docker 和 docker-compose 都已成功安装  ---------------------- |

说明前置环境已经搭建完成！

# 集群搭建安装步骤

## 修改mongo.propertis

默认情况下不需要修改mongo.propertis文件中的信息，但是在某些情况下需要修改其中的参数信息，一般在第一次安装时修改，需要修改的场景有：

1. 服务器存在多个有效的IP，此时需要修改MANAGE\_IP的值为正确的 IP，否则执行安装脚本时会报出如下错误信息：

|  |
| --- |
| $ docker swarm init  Error response from daemon: could not choose an IP address to advertise since this system has multiple addresses on different interfaces (192.168.1.9 on ens33 and 192.168.10.9 on ens35) - specify one with --advertise-addr  ----------------------  docker swarm init 失败，当前主机有多个IP 请根据提示IP数据，修改mongo.propertis中的MANAGE\_IP值为特定IP，并再次执行该脚本  ---------------------- |

此时服务器有两个IP 192.168.1.9和192.168.10.9，需要将其中的一个指定给MANAGE\_IP

1. 修改mongo数据存放目录DATA\_ROOT，默认数据存放在/data目录下
2. 修改mongo集群规模ITEMS，参照脚本说明修改
3. 修改mongo-express的web ui中登录用户的账号密码

## 修改sharding.sh

默认情况下不需要修改sharding.sh文件中的信息，脚本中已经包含需要在mongos中执行的分片脚本信息，但是可能有额外脚本需要添加/修改时，可在此处添加/修改。如：默认使用的数据库是test，后续可能需要更改，可手动更改，一般在第一次安装时修改

## 运行安装脚本install.sh安装集群

在正确修改完所有需要修改的信息后，进入脚本文件所在目录，执行install.sh

|  |
| --- |
| [root@node docker-shell]# ./install.sh |

得到如下输出信息：

|  |
| --- |
| $ docker swarm init  Swarm initialized: current node (u4drkza6xiibo70zumg0b806m) is now a manager.  To add a worker to this swarm, run the following command:  docker swarm join --token SWMTKN-1-4o68w7ghoi89tv944xvtkxfcvg1hs98j55pbod88j29y2iwxo6-605kbrlx4u012647ljfvg2hfo 192.168.1.9:2377  To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions. |

|  |
| --- |
| $ docker stack deploy -c /data/mongo/stack.yaml mongo  Ignoring unsupported options: restart  Creating network mongo\_mongo  Creating service mongo\_shard1rs3  Creating service mongo\_shard2rs2  Creating service mongo\_shard1rs1  Creating service mongo\_mongo-express  Creating service mongo\_shard2rs3  Creating service mongo\_shard2rs1  Creating service mongo\_cfgrs2  Creating service mongo\_cfgrs3  Creating service mongo\_cfgrs1  Creating service mongo\_mongos  Creating service mongo\_shard1rs2  $ docker stack services mongo  ID NAME MODE REPLICAS IMAGE PORTS  5ko15obezg8q mongo\_shard2rs2 replicated 1/1 mongo:4.0  98qglo0fz3i0 mongo\_mongos replicated 1/1 mongo:4.0 \*:27017->27017/tcp  9aqtxxw27uhv mongo\_mongo-express replicated 0/1 mongo-express:latest \*:8081->8081/tcp  a235r1u660ys mongo\_shard2rs1 replicated 1/1 mongo:4.0  clrsphtj6thb mongo\_cfgrs2 replicated 1/1 mongo:4.0  gms5whhkrglx mongo\_shard1rs2 replicated 1/1 mongo:4.0  ohnu8of8sz7u mongo\_shard2rs3 replicated 1/1 mongo:4.0  ptist1lrzz16 mongo\_shard1rs1 replicated 1/1 mongo:4.0  rdnjyt8di1bl mongo\_shard1rs3 replicated 1/1 mongo:4.0  wfoynw0u3eck mongo\_cfgrs1 replicated 1/1 mongo:4.0  yyq4gm2ybr91 mongo\_cfgrs3 replicated 1/1 mongo:4.0 |

以及最后输出如下结果：

|  |
| --- |
| ----------------------  请分批次手动拷贝执行脚本文件/data/mongo/cluster.sh中的脚本  ---------------------- |

说明集群已经搭建完成！

# 初始化mongo集群

## 确认集群状态

运行如下命令，确认集群状态已经可以初始化

|  |
| --- |
| [root@node docker-shell]# docker stack services mongo  ID NAME MODE REPLICAS IMAGE PORTS  24rc7cty1o8b mongo\_shard1rs3 replicated 1/1 mongo:4.0  a1ocfq133elo mongo\_shard2rs1 replicated 1/1 mongo:4.0  dvbitmf7r39x mongo\_shard1rs2 replicated 1/1 mongo:4.0  iu74ug5xncxe mongo\_cfgrs3 replicated 1/1 mongo:4.0  jydteexlvcuv mongo\_cfgrs2 replicated 1/1 mongo:4.0  opyct658rnrf mongo\_shard2rs3 replicated 1/1 mongo:4.0  s6ivqczrejqh mongo\_mongo-express replicated 1/1 mongo-express:latest \*:8081->8081/tcp  t56ogm95dj61 mongo\_mongos replicated 1/1 mongo:4.0 \*:27017->27017/tcp  vf5eden9ijru mongo\_cfgrs1 replicated 1/1 mongo:4.0  vzsjajyq9p7i mongo\_shard2rs2 replicated 1/1 mongo:4.0  xiamthral71h mongo\_shard1rs1 replicated 1/1 mongo:4.0 |

其中REPLICAS列对应的结果都是 1/1，表示所有的应用都已经启动完成，即可开始执行初始化脚本，否则稍后执行该命令继续等待，知道状态已经正常。

## 分批次手动执行集群初始化脚本

按照install.sh集群初始化脚本安装成功后的最后说明

|  |
| --- |
| ----------------------  请分批次手动拷贝执行脚本文件/data/mongo/cluster.sh中的脚本  ---------------------- |

去分批次执行/data/mongo/cluster.sh中的脚本

首先进入脚本所在目录/data/mongo/，使用文档编辑器查看文件cluster.sh中的内容，分批次执行其中的脚本。

所有批次之间由以下内容行进行分割

|  |
| --- |
| #===================== 以上以等号划分区域脚本为一个脚本批次，请手动拷贝到linux命令行下执行 ======================== |

例如：

|  |
| --- |
| # shard1  docker exec -it $(docker ps | grep shard1rs1 | awk '{ print $1 }') mongo --port 27017  rs.initiate({\_id:'shard1',members:[{\_id:1,host:'shard1rs1:27017'},{\_id:2,host:'shard1rs2:27017'},{\_id:3,host:'shard1rs3:27017'}]})  rs.conf() //查看配置  rs.status() //查看副本级状态  quit()  #===================== 以上以等号划分区域脚本为一个脚本批次，请手动拷贝到linux命令行下执行 ========================  # shard2  docker exec -it $(docker ps | grep shard2rs1 | awk '{ print $1 }') mongo --port 27017  rs.initiate({\_id:'shard2',members:[{\_id:1,host:'shard2rs1:27017'},{\_id:2,host:'shard2rs2:27017'},{\_id:3,host:'shard2rs3:27017'}]})  rs.conf() //查看配置  rs.status() //查看副本级状态  quit()  #===================== 以上以等号划分区域脚本为一个脚本批次，请手动拷贝到linux命令行下执行 ======================== |

以上内容对应两个批次

第一个批次内容是

|  |
| --- |
| docker exec -it $(docker ps | grep shard1rs1 | awk '{ print $1 }') mongo --port 27017  rs.initiate({\_id:'shard1',members:[{\_id:1,host:'shard1rs1:27017'},{\_id:2,host:'shard1rs2:27017'},{\_id:3,host:'shard1rs3:27017'}]})  rs.conf() //查看配置  rs.status() //查看副本级状态  quit() |

第二个批次内容是

|  |
| --- |
| docker exec -it $(docker ps | grep shard2rs1 | awk '{ print $1 }') mongo --port 27017  rs.initiate({\_id:'shard2',members:[{\_id:1,host:'shard2rs1:27017'},{\_id:2,host:'shard2rs2:27017'},{\_id:3,host:'shard2rs3:27017'}]})  rs.conf() //查看配置  rs.status() //查看副本级状态  quit() |

请分配将对应脚本内容拷贝到Linux命令行执行，进行mongo集群初始化。

注意：所有mongo集群初始化命令rs.initiate()、sh.addShard()、sh.enableSharding()、sh.enableSharding()、sh.enableSharding()的输出结果中的”ok”，对应结果是1才表示运行正常，若结果是0，表示失败。

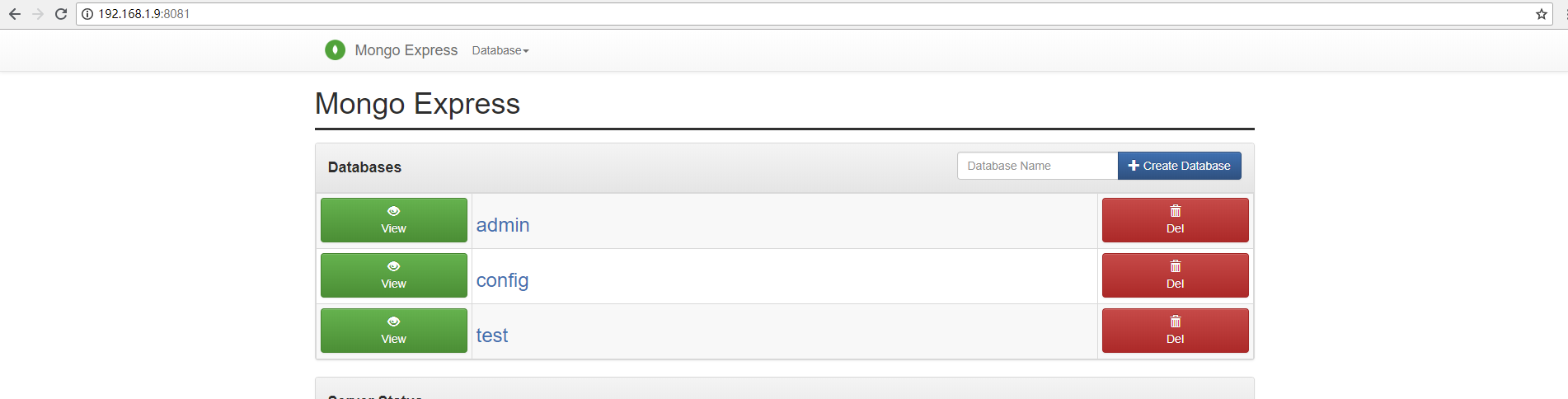
示例如下：

|  |
| --- |
| > rs.initiate({\_id:'shard1',members:[{\_id:1,host:'shard1rs1:27017'},{\_id:2,host:'shard1rs2:27017'},{\_id:3,host:'shard1rs3:27017'}]})  {  "ok" : 1,  "operationTime" : Timestamp(1537846047, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1537846047, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  } |

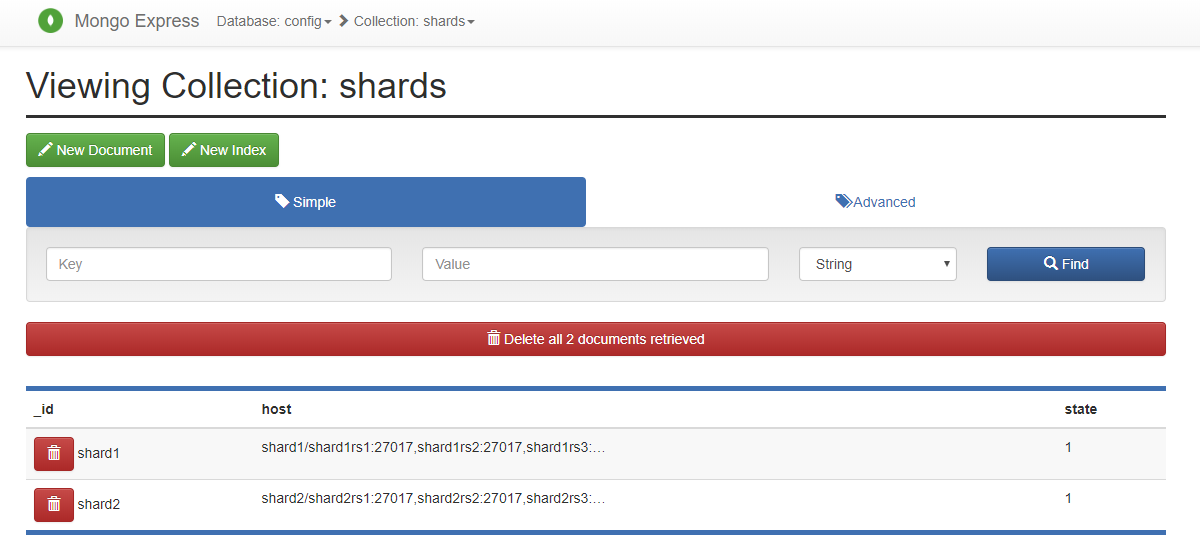
## 访问页面

集群初始化完成后，访问web页面：<http://ip:8081（ip>改为正确的ip，如：http://192.168.1.9:8081），账号密码: admin/123(mongo.properties中配置)，确认集群是否正常配置

登录成功后可看到如下页面：



点击config 前面的view，再点击shards前面的view，可看到如下页面：



其中对应的信息与cluster.sh中的信息是一致的。

恭喜，MongoDB集群搭建成功，且已经初始化完成