CentOS7.4搭建基于用户认证的MongoDB4.0三节点副本集集群详细文档 - xshrim - 博客园

导语

不久前MongoDB发布了4.0正式版，4.0版本的最大特性是支持多文档事务，但这一特性只支持副本集或者分片集群，单节点MongoDB环境是无法使用此特性的.

本文将详细介绍在CentOS7.4操作系统上搭建基于用户认证的MongoDB4.0三节点的副本集集群的完整过程.

基础规划

软件环境

MongoDB官网下载中心　　MongoDB4.0.3下载地址

```
# 三节点均执行
# wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.0.3.tgz    #mongodb下载
# tar -zxvf mongodb-linux-x86_64-rhel70-4.0.3.tgz                                #mongodb解压
# mv mongodb-linux-x86_64-rhel70-4.0.3 /usr/local/share/mongodb                  #mongodb安装
# echo 'export PATH=/usr/local/share/mongodb/bin:$PATH' >> /etc/profile         #mongodb环境配置
# source /etc/profile                                                            #使配置生效
# mongo --version                                                                # 查看是否安装成功
```

环境规划

节点规划				
节点名	节点用途	节点IP	节点端口	集群名
mongo1	主节点(PRIMARY)	10.0.2.11	9927	myrs

mongo2	从节点(SECONDARY)	10.0.2.12	9927
mongo3	仲裁节点(ARBITER)	10.0.2.13	9927

[注]:

1. 本文的三节点环境使用VirtualBox创建的三台虚拟机. 配置集群的过程与真实服务器无异. 本文MongoDB服务使用的端口统一改为9927, 也可直接使用默认的27017端口.

2. 虚拟机网络选择NAT网络, 注意不是网络地址转换(NAT), 后者无法支持虚拟机之间的网络互通.

3. 建议为虚拟机可以开启端口映射(VirtualBox全局设定-网络), 开启端口映射后, 主机可以通过ssh访问虚拟机(映射22端口), 也可以在主机上连接虚拟机的MongoDB服务(映射9927端口).

4. 单服务器环境下同样可以配置MongoDB副本集集群, 即"三节点"部署在一台服务器上, 节点IP相同, 节点端口不同(如9927, 9928, 9929), 配置过程与三服务器大同小异(每个"节点"需分配一个单独的目录).

5. 为保证集群节点互通, 建议关闭虚拟机操作系统防火墙(systemctl disable firewalld, systemctl stop firewalld)或者为防火墙开放9927端口(firewall-cmd --zone=public --add-port=9927/tcp --permanent, firewall-cmd --reload).

目录规划		
目录	用途	备注
/mgrs/data	Mongo集群数据文件目录	
/mgrs/logs	Mongo集群系统日志目录	
/mgrs/conf	Mongo集群配置文件目录	

[注]:

1. 三节点均配置相同的目录环境. 目录环境并非MongoDB硬性规定, 可根据自身需求规划.

2. 如是单服务器下的Mongo集群, 则需为每个"节点"配置单独目录(/mgrs/node1/data, /mgrs/node1/logs, /mgrs/node1/conf, 依此类推).

搭建步骤

配置文件

MongoDB服务支持命令行配置参数和配置文件两种启动方式, 为便于管理, 推荐使用配置文件进行服务参数的配置. 自Mongo2.6版起, 官方推荐使用YAML格式的配置参数: MongoDB官方YAML配置文件参数说明; 旧版配置文件格式依然可用但不建议使用: MongoDB官方旧版配置文件参数说明.

下面是本集群使用的YAML配置文件mongod.conf:

```
systemLog:
    destination: file
    path: "/mgrs/logs/mongod.log"
    logAppend: true
storage:
    dbPath: "/mgrs/data"
    journal:
        enabled: true
processManagement:
    fork: true
    pidFilePath: "/mgrs/mongod.pid"
net:
    bindIpAll: true
    port: 9927
#security:
#    keyFile: "/mgrs/conf/access.key"
#    authorization: enabled
#setParameter:
#    authenticationMechanisms: SCRAM-SHA-1
replication:
    oplogSizeMB: 500
    replSetName: myrs
```

[注]:

1. 三节点的配置文件均相同. 配置文件存放位置建议为:/mrgs/conf/mongod.conf.
2. 如是单服务器环境, 需根据具体目录规划为三个配置文件配置相应的参数.
3. 配置文件参数可根据实际需要对照官方参数说明进行修改.
4. security部分的参数用于启用集群用户认证, 需要先注释掉, 等集群搭建完成再启用, setParameter部分可以去掉.
5. net部分直接启用bindIpAll参数将操作系统上配置的所有ip均绑定到mongodb服务上, 生产环境可使用bindIp: "10.0.2.11"参数代替.

启动服务

直接使用配置文件在三个节点上分别启动mongod后台进程.

```
# 三节点均执行
# mongod -f /mgrs/conf/mongod.conf
```

配置集群

进入其中一个节点(主节点)的mongo控制台, 配置集群(务必保证节点防火墙关闭或开放mongo服务端口).

```
# 仅在一个节点执行
# mongo 127.0.0.1:9927     #进入mongo控制台
> cfg = {_id: 'myrs', members: []}                                  #生成集群配置变量
> cfg.members.push({_id: 1, host: '10.0.2.11'})                     #变量中加入节点1
> cfg.members.push({_id: 2, host: '10.0.2.12'})                     #变量中加入节点2
> cfg.members.push({_id: 3, host: '10.0.2.13'}, arbiterOnly: true)  #变量中加入节点3(仲裁节点)
> rs.initiate(cfg)                                                  #根据变量配置集群
> rs.isMaster()                                                     #查看集群是否配置成功
> rs.status()
```

添加用户

集群配置完成后, 仍然在主节点的mongo控制台中添加三个用户:数据库管理员, 集群管理员和访问特定数据库的用户.

```
# 仅在一个节点执行
# mongo 127.0.0.1:9927     #进入mongo控制台
> use admin                #使用内置的admin库
> db.createUser(           #创建数据库管理员
{
```

```
user:"root",
pwd:"root",
roles:[{role:"readWriteAnyDatabase",db:"admin"},{role:"dbAdminAnyDatabase",db:"admin"},{role:"userAdminAnyDatabase",db:"admin"}]
}
)
> db.createUser(             #创建集群管理员
{
user:"suroot",
pwd:"suroot",
roles:[{role:"clusterAdmin",db:"admin"},{role:"clusterManager",db:"admin"},{role:"clusterMonitor",db:"admin"}]
}
)
> use testdb                  #切换到testdb数据库,不用事先创建
> db.createUser(
{
user:"test",
pwd:"test",
roles:[{role:"readWrite",db:"testdb"},{role:"dbAdmin",db:"testdb"},{role:"userAdmin",db:"testdb"}]
}
)
> use admin
> db.system.users.find()      #查看创建的用户
```

[注]:

1. MongoDB的用户和数据库是绑定的，必须指定某个用户归属于哪个数据库，即在roles字段的每个role中指定db字段.
2. 数据库管理员通常需要具有读写,管理任意数据库和管理任意用户的role，后续可以登录此用户进行数据库和用户的增删改查.
3. 集群管理员通常需要具有集群管理和集群监控的role，只有集群管理员可以关闭集群.
4. 普通用户根据用途不同可以对特定或者多个数据库拥有各种不同的role，本例中的test用户既可以读写testdb库，同时也是testdb库的管理员.
5. 主节点上添加的用户应该能够在从节点上查询到(需要先键入rs.slaveOk()命令).

开启用户认证

用户添加完成后需要关闭所有节点(先关闭仲裁和从节点，再关闭主节点，避免主节点切换):

```
# 在三个节点均执行
# mongo 127.0.0.1:9927      #节点关闭只能在本机操作
> use admin
> db.shutdownServer()       #关闭mongo后台进程
```

生成keyFile(keyFile的用途是作为所有mongod后台进程允许加入集群的凭证，所有集群中的节点共用一个keyFile，避免其他mongod非法加入集群):

```
# 仅在一个节点执行
# openssl rand -base64 756 > /mgrs/conf/access.key              #生成keyFile，keyFile的长度必须在6-1024个字符之间
# chmod 400 /mgrs/conf/access.key                               #设置keyFile文件为只读
# scp /mgrs/conf/access.key root@10.0.2.12:/mgrs/conf/          #将keyFile复制到其他节点
# scp /mgrs/conf/access.key root@10.0.2.13:/mgrs/conf/
```

取消三个节点mongod.conf文件中security部分的注释:

```
systemLog:
    destination: file
    path: "/mgrs/logs/mongod.log"
    logAppend: true
storage:
    dbPath: "/mgrs/data"
    journal:
        enabled: true
processManagement:
    fork: true
    pidFilePath: "/mgrs/mongod.pid"
net:
    bindIpAll: true
    port: 9927
security:
    keyFile: "/mgrs/conf/access.key"
    authorization: enabled
setParameter:
    authenticationMechanisms: SCRAM-SHA-1
replication:
    oplogSizeMB: 500
    replSetName: myrs
```

依次启动主节点，从节点和仲裁节点的mongod后台进程：

```
# 三个节点均执行
# mongod -f /mgrs/conf/mongod.conf
```

使用认证用户登录：

```
# mongo 127.0.0.1:9927
> use admin
> db.auth('root', 'root')          #使用数据库管理员认证
> rs.slaveOk()                     #默认读写操作均在主节点执行，从节点上需要执行此命令才能读取数据库数据
> db.system.users.find()
> use testdb
> db.auth('test', 'test')          #切换到test用户
> db.getCollectionNames()
> use admin                        #切换到集群管理员用户，关闭mongo服务
> db.auth('suroot', 'suroot')
```

至此，基于用户认证的MongoDB4.0三节点副本集集群环境已经搭建完成.

常用命令

结束