Docker 搭建集群MongoDB

这次采用副本集的方式来搭建集群，三台服务器，一主、一副、一仲裁

基本概念

Replica Set 副本集：一个副本集就是一组 MongoDB 实例组成的集群，由一个主（Primary）服务器和多个备份（Secondary）服务器构成

- 主节点（master）：主节点接收所有写入操作。主节点将对其数据集所做的所有更改记录到其 oplog。
- 副节点（secondary）：复制主节点的 oplog 并将操作应用到其数据集，如果主节点不可用，一个合格的副节点将被选为新的主节点。
- 仲裁节点（arbiter）：负载选举，当主节点不可用，它将从副节点中选一个作为主节点。

Sharding 分片：

Master-slave 主备

- MongoDB 4.0 以上版本运行时提示：[main] Master/slave replication is no longer supported，也就是 MongoDB 4.0 后不在支持主从复制

一、环境准备

使用 CentOS 7.6 64bit 系统，安装 Docker、Docker-compose、Docker-Swarm

二、生成 KeyFile

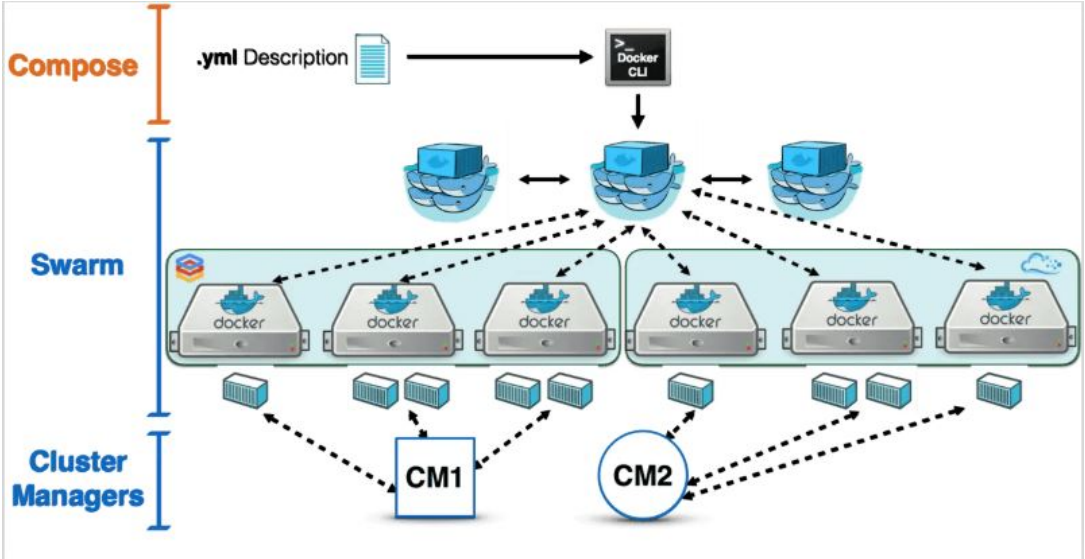
- MongoDB 使用 KeyFile 认证，副本集中的每个 MongoDB 实例使用 KeyFile 内容作为认证其他成员的共享密码。MongoDB 实例只有拥有正确的 KeyFile 才可以加入副本集。
- keyFile 的内容必须是 6 到 1024 个字符的长度，且副本集所有成员的 KeyFile 内容必须相同。
- 有一点要注意是的：在 UNIX 系统中，KeyFile 必须没有组权限或完全权限（也就是权限要设置成 X00 的形式）。Windows 系统中，keyFile 权限没有被检查。

- 可以使用任意方法生成 keyFile。例如，如下操作使用 openssl 生成复杂的随机的 1024 个字符串。然后使用 chmod 修改文件权限，只给文件拥有者提供读权限。
- 这是 MongoDB 官方推荐 keyFile 的生成方式：

```
# 400权限是要保证安全性，否则mongod启动会报错
openssl rand -base64 756 > mongodb.key
chmod 400 mongodb.key
```

二、创建跨主机网络

搭建集群我们肯定是跨主机通讯，要搭建 Overlay Network 网络，我们就要用到 Docker Swarm 这个工具了。Docker Swarm 是 Docker 内置的集群工具，它能够帮助我们更轻松地将服务部署到 Docker daemon 的集群之中。



既然要将 Docker 加入到集群，我们就必须先有一个集群，我们在任意一个 Docker 实例上都可以通过 docker swarm init 来初始化集群。

```
$ sudo docker swarm init

Swarm initialized: current node (t4ydh2o5mwp5io2netepcauyl) is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-4dvxvx4n7magy5zh0g0de0xoues9azekw308jlv6hlvqwpriwy-cb43z26n5jbadk024tx0cqz5r 192.168.1.5:2377
```

在集群初始化后，这个 Docker 实例就自动成为了集群的管理节点，而其他 Docker 实例可以通过运行这里所打印的 docker swarm join 命令来加入集群。

加入到集群的节点默认为普通节点，如果要以管理节点的身份加入到集群中，我们可以通过 docker swarm join-token 命令来获得管理节点的加入命令。

```
$ sudo docker swarm join-token manager
To add a manager to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-60am9y6axwot0angn1e5inxrpzrj5d6aa91gx72f8et94wztm1-7lz0dth35wywekjd1qn30jtes 192.168.1.5:2377
```

我们通过这些命令来建立用于我们服务开发的 Docker 集群，并将相关开发同事的 Docker 加入到这个集群里，就完成了搭建跨主机网络的第一步。

建立跨主机网络

接下来，我们就通过 `docker network create` 命令来建立 Overlay 网络。

```
$ sudo docker network create --driver overlay --attachable mongodbs
```

在创建 Overlay 网络时，我们要加入 --attachable 选项以便不同机器上的 Docker 容器能够正常使用到它。

在创建了这个网络之后，我们可以在任何一个加入到集群的 Docker 实例上使用 docker network ls 查看一下其下的网络列表。我们会发现这个网络定义已经同步到了所有集群中的节点上。

```
$ sudo docker network ls
NETWORK ID     NAME        DRIVER       SCOPE
## ......
y89bt74ld9l8   mongodbs        overlay      swarm
## ......
```

接下来我们要修改 Docker Compose 的定义，让它使用这个我们已经定义好的网络，而不是再重新创建网络。

我们只需要在 Docker Compose 配置文件的网络定义部分，将网络的 external 属性设置为 true，就可以让 Docker Compose 将其建立的容器都连接到这个不属于 Docker Compose 的项目上了。

```
networks:
 mesh:
  external: true
```

通过这个实现，我们在开发中就使整个服务都处于一个可以使用别名映射网络中，避免了要对不同功能联调时切换服务 IP 的烦琐流程。在这种结构下，我们只需要让我们开发的 Docker 退出和加入不同的集群，就能马上做到切换不同联调项目。

二、编写 docker-compose 文件

主节点

```
version: ”3”
services:
 master:
  image: mongo:4.1
  container_name: master
  environment:
   MONGO_INITDB_ROOT_USERNAME: root
   MONGO_INITDB_ROOT_PASSWORD: 123456
   TZ: ”Asia/Shanghai”
  volumes:
   # 挂载 MongoDB 数据目录
   – ”/data/docker/mongodb/data/mongo:/data/db:rw”
   # 挂载 KeyFile
   – ”/data/docker/mongodb/data/mongodb.key:/data/mongodb.key”
  ports:
   – ”27018:27017”
  networks:
   – mongodbs
  command:
   # 密码
   ––auth
   # 副本集名称
   ––replSet testSet
   ––oplogSize 128
   ––keyFile /data/mongodb.key
# Swarm 跨主机网络网络
networks:
 mongodbs:
  external: true
```

副节点

```
version: ”3”
services:
secondary:
 image: mongo:4.1
 container_name: secondary
 environment:
  MONGO_INITDB_ROOT_USERNAME: root
  MONGO_INITDB_ROOT_PASSWORD: 123456
  TZ: ”Asia/Shanghai”
 volumes:
  – ”/data/docker/mongodb/data/mongo:/data/db:rw”
  – ”/data/docker/mongodb/data/mongodb.key:/data/mongodb.key”
 ports:
  – ”27018:27017”
 networks:
  – mongodbs
 command:
  ––auth
  ––replSet testSet
  ––oplogSize 128
  ––keyFile /data/mongodb.key
networks:
mongodbs:
 external: true
```

仲裁节点，因为仲裁节点不需要存储数据，他只是用来当主节点挂掉后选举新的主节点，所以不需要密码、映射端口等操作

```
version: ”3”
services:
arbiter:
 image: mongo:4.1
 container_name: arbiter
 restart: always
 volumes:
  – ”/data/docker/mongodb/data/mongo:/data/db:rw”
  – ”/data/docker/mongodb/data/mongo_key:/mongo:rw”
 networks:
  – mongodbs
 command:
  mongod ––replSet testSet ––smallfiles ––oplogSize 128
networks:
mongodbs:
 external: true
```

三、启动容器

接下来我们分别在三台服务器中使用容器编排启动容器

```
docker-compose up -d
```

四、配置副本集

进入主节点容器内部

```
docker exec –it master mongo
```

在 mongo shell 里执行：

```
> rs.initiate()
{
   ”info2” : ”no configuration specified. Using a default configuration for the set”,
   ”me” : ”7abd89794aa7:27017”,
   ”ok” : 1
}
```

继续执行：

```
testSet:SECONDARY> rs.add('secondary:27017')
{
   ”ok” : 1,
   ”$clusterTime” : {
      ”clusterTime” : Timestamp(1599562800, 1),
      ”signature” : {
         ”hash” : BinData(0,”wrxMUlX/0bEyLgCVoQqdLvH59T0=”),
         ”keyId” : NumberLong(”6870069879538450434”)
      }
   },
   ”operationTime” : Timestamp(1599562800, 1)
}
```

继续执行，其中 true 表示这个节点是仲裁节点

```
testSet:PRIMARY> rs.add('arbiter:27017',true)
{
   ”ok” : 1,
   ”$clusterTime” : {
      ”clusterTime” : Timestamp(1599562838, 1),
      ”signature” : {
         ”hash” : BinData(0,”p9ub49lLD8ij8nkxpfu2l/AvRRY=”),
         ”keyId” : NumberLong(”6870069879538450434”)
      }
   },
   ”operationTime” : Timestamp(1599562838, 1)
}
```

查看配置

```
testSet:PRIMARY> rs.conf()
{
   ”_id” : ”testSet”,
   ”version” : 3,
   ”protocolVersion” : NumberLong(1),
   ”writeConcernMajorityJournalDefault” : true,
   ”members” : [
      {
         ”_id” : 0,
         ”host” : ”7abd89794aa7:27017”,
         ”arbiterOnly” : false,
```

```
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        },
        {
            "_id" : 1,
            "host" : "secondary:27017",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        },
        {
            "_id" : 2,
            "host" : "arbiter:27017",
            "arbiterOnly" : true,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 0,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        }
    ],
    "settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : -1,
        "catchUpTakeoverDelayMillis" : 30000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
            "w" : 1,
            "wtimeout" : 0
        },
        "replicaSetId" : ObjectId("5f576426fe90ef2dd8cd2700")
    }
}
```

查看状态

```
testSet:PRIMARY> rs.status()
{
```

```
"set" : "testSet",
"date" : ISODate("2020-09-08T11:45:12.096Z"),
"myState" : 1,
"term" : NumberLong(1),
"syncingTo" : "",
"syncSourceHost" : "",
"syncSourceId" : -1,
"heartbeatIntervalMillis" : NumberLong(2000),
"optimes" : {
    "lastCommittedOpTime" : {
        "ts" : Timestamp(1599565502, 1),
        "t" : NumberLong(1)
    },
    "lastCommittedWallTime" : ISODate("2020-09-08T11:45:02.775Z"),
    "readConcernMajorityOpTime" : {
        "ts" : Timestamp(1599565502, 1),
        "t" : NumberLong(1)
    },
    "readConcernMajorityWallTime" : ISODate("2020-09-08T11:45:02.775Z"),
    "appliedOpTime" : {
        "ts" : Timestamp(1599565502, 1),
        "t" : NumberLong(1)
    },
    "durableOpTime" : {
        "ts" : Timestamp(1599565502, 1),
        "t" : NumberLong(1)
    },
    "lastAppliedWallTime" : ISODate("2020-09-08T11:45:02.775Z"),
    "lastDurableWallTime" : ISODate("2020-09-08T11:45:02.775Z")
},
"lastStableRecoveryTimestamp" : Timestamp(1599565492, 1),
"lastStableCheckpointTimestamp" : Timestamp(1599565492, 1),
"members" : [
    {
        "_id" : 0,
        "name" : "7abd89794aa7:27017",
        "ip" : "10.0.1.41",
        "health" : 1,
        "state" : 1,
        "stateStr" : "PRIMARY",
        "uptime" : 2784,
        "optime" : {
            "ts" : Timestamp(1599565502, 1),
            "t" : NumberLong(1)
        },
        "optimeDate" : ISODate("2020-09-08T11:45:02Z"),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "electionTime" : Timestamp(1599562790, 2),
        "electionDate" : ISODate("2020-09-08T10:59:50Z"),
        "configVersion" : 3,
        "self" : true,
        "lastHeartbeatMessage" : ""
    },
    {
        "_id" : 1,
        "name" : "secondary:27017",
        "ip" : "10.0.1.233"
```

```
            "ip" : "10.0.1.233",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 2711,
            "optime" : {
                "ts" : Timestamp(1599565502, 1),
                "t" : NumberLong(1)
            },
            "optimeDurable" : {
                "ts" : Timestamp(1599565502, 1),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2020-09-08T11:45:02Z"),
            "optimeDurableDate" : ISODate("2020-09-08T11:45:02Z"),
            "lastHeartbeat" : ISODate("2020-09-08T11:45:11.494Z"),
            "lastHeartbeatRecv" : ISODate("2020-09-08T11:45:11.475Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "",
            "syncingTo" : "7abd89794aa7:27017",
            "syncSourceHost" : "7abd89794aa7:27017",
            "syncSourceId" : 0,
            "infoMessage" : "",
            "configVersion" : 3
        },
        {
            "_id" : 2,
            "name" : "arbiter:27017",
            "ip" : null,
            "health" : 0,
            "state" : 8,
            "stateStr" : "(not reachable/healthy)",
            "uptime" : 0,
            "lastHeartbeat" : ISODate("2020-09-08T11:45:10.463Z"),
            "lastHeartbeatRecv" : ISODate("1970-01-01T00:00:00Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "Error connecting to arbiter:27017 :: caused by
:: Could not find address for arbiter SocketException: Host not found
(authoritative)",
            "syncingTo" : "",
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "",
            "configVersion" : -1
        }
    ],
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1599565502, 1),
        "signature" : {
            "hash" : BinData(0,"7/ei+8UrhlpIny9zKeWuAFpn46c="),
            "keyId" : NumberLong("6870069879538450434")
        }
    },
    "operationTime" : Timestamp(1599565502, 1)
}
```


五、验证 MongoDB 可用性

先进入主节点服务器添加一条数据

```
docker exec –it master mongo
use admin
db.auth('root', '123456')
use test
db.test.insert({name:"muyang",age:20})
```

在来副节点服务器查看是否已经同步了这条数

```
[root@linux secondary] docker exec –it secondary mongo
testSet:SECONDARY> use admin
testSet:SECONDARY> db.auth('root', '123456')
testSet:SECONDARY> use test
testSet:SECONDARY> db.test.find()
2020–09–08T19:03:02.295+0800 E QUERY  [js] uncaught exception: Error:
listCollections failed: {
  "operationTime" : Timestamp(1599562972, 1),
  "ok" : 0,
  "errmsg" : "not master and slaveOk=false",
  "code" : 13435,
  "codeName" : "NotMasterNoSlaveOk",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1599562972, 1),
    "signature" : {
      "hash" : BinData(0,"mhsrpGHRl7qZg2QOjyS3RbBb/Yc="),
      "keyId" : NumberLong("6870069879538450434")
    }
  }
} :
testSet:SECONDARY> rs.slaveOk()
testSet:SECONDARY> db.users.find()
{ "_id" : ObjectId("5f5764b1f909544b783696c2"), "name" : "muyang", "age" :
20 }
```

在 secondary 查询时报如下错误：

> not master and slaveok=false

这是正常的，因为 secondary 是不允许读写的，如果非要解决，方法如下：

```
testSet:SECONDARY> rs.slaveOk()
```