实验五：基于 Docker 部署 MongoDB 集群

一、实验目的

1、了解 Docker；
2、了解 MongoDB 数据库；
3、掌握基于 Docker 部署 MongoDB 数据库服务；
4、掌握 MongoDB 数据库集群的实现；
5、掌握使用 MongoDB Compass 管理 MongoDB 数据库集群。

二、实验学时

2 学时

扫码看操作演示

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台。

2、软件

Windows 操作系统，安装 Oracle VM VirtualBox 软件，安装 MobaXterm 软件。
安装 MongoDB Compass 软件。

3、网络

无。

4、工具

无

五、实验任务

1、完成 MongoDB 的安装；
2、完成 MongoDB 数据库集群的部署；
3、完成使用 MongoDB Compass 管理 MongoDB 数据库集群，并进行副本集测试。

六、实验环境

1、本实验需要 VM 1 台；
2、本实验 VM 配置信息如下表所示；

虚拟机配置	操作系统配置
虚拟机名称：VM-Lab-05-Task-01-172.20.1.16 内存：4GB CPU：1 颗，4 核心 虚拟磁盘：100GB 网卡：1 块，桥接	主机名：Lab-05-Task-01 IP 地址：172.20.1.16 子网掩码：255.255.255.0 网关：172.20.1.1 DNS：8.8.8.8

3、本实验拓扑图。

无

4、本实验操作演示视频。

本实验为视频集的第 5 集：：https://www.bilibili.com/video/BV1h14y1k7Gc?p=5

七、实验内容步骤

1、安装 Docker 环境

（1）查看防火墙 Firewalld 服务状态（openEuler 操作系统默认安装 Firewalld 防火墙，并创建 firewalld 服务，该服务已开启且已配置为开机自启动）。

（2）使用 firewall-cmd 命令添加防火墙规则放行端口，并重新载入防火墙配置使其生效。

（3）使用 yum 命令安装 docker 环境，查看 docker 版本信息，启动 docker 服务，设置 docker 服务为开机自启动并查看 docker 服务状态。

```
# 查看防火墙 Firewalld 服务状态
[root@Lab-05-Task-01 ~]# systemctl status firewalld

# 添加本地客户端允许远程连接 MongoDB 数据库
[root@Lab-05-Task-01 ~]# firewall-cmd --zone=public --add-port=27017/tcp --per
manent
[root@Lab-05-Task-01 ~]# firewall-cmd --zone=public --add-port=27018/tcp --per
manent
[root@Lab-05-Task-01 ~]# firewall-cmd --zone=public --add-port=27019/tcp --per
manent
# 重新载入防火墙配置使其生效
[root@Lab-05-Task-01 ~]# firewall-cmd --reload

# 安装 dokcer
[root@Lab-05-Task-01 ~]# dnf -y install docker
# 查看 Docker 版本信息
[root@Lab-05-Task-01 ~]# docker --version
# 启动 docker 服务
[root@Lab-05-Task-01 ~]# systemctl start docker
# 设置 docker 服务为开机自启动
[root@Lab-05-Task-01 ~]# systemctl enable docker
# 查看 docker 服务状态
```

```
[root@Lab-05-Task-01 ~]# systemctl status docker
```

2、安装 MongoDB 数据库

（1）使用 docker pull 命令拉取 MongoDB 镜像。

（2）使用 docker run 命令创建并启动 3 个 MongoDB 容器。

```
# 拉取 MongoDB 镜像
[root@Lab-05-Task-01 ~]# docker pull mongo

# 创建并启动第 1 个 MongoDB 容器，将容器 27017 端口映射到宿主机的 27017 端口
[root@Lab-05-Task-01 ~]# docker run -d --name mongo1 -v /usr/local/mongodb/
datadb1:/data/db -v /usr/local/mongodb/key:/data/key -v /etc/localtime:/etc/localti
me -p 27017:27017 mongo --replSet mongodb-cluster

# 创建并启动第 2 个 MongoDB 容器，将容器 27017 端口映射到宿主机的 27018 端口
[root@Lab-05-Task-01 ~]# docker run -d --name mongo2 -v /usr/local/mongodb/
datadb2:/data/db -v /usr/local/mongodb/key:/data/key -v /etc/localtime:/etc/localti
me -p 27018:27017 mongo --replSet mongodb-cluster

# 创建并启动第 2 个 MongoDB 容器，将容器 27017 端口映射到宿主机的 27019 端口
[root@Lab-05-Task-01 ~]# docker run -d --name mongo3 -v /usr/local/mongodb/
datadb3:/data/db -v /usr/local/mongodb/key:/data/key -v /etc/localtime:/etc/localti
me -p 27019:27017 mongo --replSet mongodb-cluster
```

3、生成副本集密钥

进入到“/usr/local/mongodb/key”目录下，生成 MongoDB 的副本集密钥。

```
[root@Lab-05-Task-01 ~]# cd /usr/local/mongodb/key
[root@Lab-05-Task-01 key]# openssl rand -base64 756 > mongodb.key
```

4、在容器 mongo1 内配置副本集

（1）使用 docker exec 命令进入到容器名为“mongo1”的容器内部，并在容器内部安装 vim 编辑器。

（2）在容器名为“mongo1”的容器内部修改 mongdb 数据库配置文件。

（3）使用 exit 命令退出当前容器，返回到当前主机的命令行界面，使用 docker restart 重启容器名为“mongo1”的 docker 容器，确保配置生效。

```
# 进入到容器名为“mongo1”的容器内部
[root@Lab-05-Task-01 key]# docker exec -it mongo1 bash
# 安装 vim 编辑器
root@50080eb44de2:/# apt-get update
root@50080eb44de2:/# apt-get install -y vim

root@50080eb44de2:/# cp /etc/mongod.conf.orig /etc/mongod.conf
# 使用 vim 命令编辑/etc/mongod.conf 文件
root@50080eb44de2:/# vim /etc/mongod.conf
```

```
# ----------/etc/mongod.conf 文件----------
net:
  port: 27017
  bindIp: 0.0.0.0
security:
  keyFile: /data/key/mongodb.key
replication:
  replSetName: "mongodb-cluster"
# ----------/etc/mongod.conf 文件----------

# exit 命令退出当前容器
root@50080eb44de2:/# exit
# 重启 mongo1 容器
[root@Lab-05-Task-01 key]# docker restart mongo1
```

5、在容器 mongo2 内配置副本集

（1）使用 docker exec 命令进入到容器名为“mongo2”的容器内部，并在容器内部安装 vim 编辑器。

（2）在容器名为“mongo2”的容器内部修改 mongdb 数据库配置文件。

（3）使用 exit 命令退出当前容器，返回到当前主机的命令行界面，使用 docker restart 重启容器名为“mongo2”的 docker 容器，确保配置生效。

```
# 进入到容器名为“mongo2”的容器内部
[root@Lab-05-Task-01 key]# docker exec -it mongo2 bash
# 安装 vim 编辑器
root@9710b49a6a41:/# apt-get update
root@9710b49a6a41:/# apt-get install -y vim

root@9710b49a6a41:/# cp /etc/mongod.conf.orig /etc/mongod.conf
# 使用 vim 命令编辑/etc/mongod.conf 文件
root@9710b49a6a41:/# vim /etc/mongod.conf
# ----------/etc/mongod.conf 文件----------
net:
  port: 27017
  bindIp: 0.0.0.0
security:
  keyFile: /data/key/mongodb.key
replication:
  replSetName: "mongodb-cluster"
# ----------/etc/mongod.conf 文件----------

# exit 命令退出当前容器
root@9710b49a6a41:/# exit
# 重启 mongo2 容器
[root@Lab-05-Task-01 key]# docker restart mongo2
```

6、在容器 mongo3 内配置副本集

（1）使用 docker exec 命令进入到容器名为“mongo3”的容器内部，并在容器内部安装 vim 编辑器。

（2）在容器名为“mongo3”的容器内部修改 mongdb 数据库配置文件。

（3）使用 exit 命令退出当前容器，返回到当前主机的命令行界面，使用 docker restart 重启容器名为“mongo3”的 docker 容器，确保配置生效。

```
# 进入到容器名为“mongo3”的容器内部
[root@Lab-05-Task-01 key]# docker exec -it mongo3 bash
# 安装 vim 编辑器
root@30cefbae3722:/# apt-get update
root@30cefbae3722:/# apt-get install -y vim

root@30cefbae3722:/# cp /etc/mongod.conf.orig /etc/mongod.conf
# 使用 vim 命令编辑/etc/mongod.conf 文件
root@30cefbae3722:/# vim /etc/mongod.conf
# --------------------/etc/mongod.conf 文件--------------------
net:
  port: 27017
  bindIp: 0.0.0.0
security:
  keyFile: /data/key/mongodb.key
replication:
  replSetName: "mongodb-cluster"
# --------------------/etc/mongod.conf 文件--------------------

# exit 命令退出当前容器
root@30cefbae3722:/# exit
# 重启 mongo2 容器
[root@Lab-05-Task-01 key]# docker restart mongo3
```

7、在容器 mongo1 内初始化副本集

（1）使用 docker exec -it mongo1 mongosh 命令连接到容器名为“mongo1”的 MongoDB 客户端，初始化副本集，查看副本集状态并退出当前连接。

（2）使用 docker exec -it mongo1 mongosh 命令连接 MongoDB 客户端，为副本集创建用户，并退出当前连接。

```
# 使用 docker exec -it mongo1 mongosh 命令连接 MongoDB 客户端
[root@Lab-05-Task-01 ~]# docker exec -it mongo1 mongosh
# 初始化副本集
> rs.initiate( {
  _id : "mongodb-cluster",
  members: [
    { _id:0, priority:2, host:"172.20.1.16:27017"},
    { _id:1, host:"172.20.1.16:27018"},
    { _id:2, host:"172.20.1.16:27019"}
  ]
})
```

```
#查看副本集状态
mongodb-cluster [direct: primary] test> rs.status()
# 退出当前连接
mongodb-cluster [direct: primary] test> quit()

# 使用 docker exec -it mongo1 mongosh 命令连接 MongoDB 客户端
[root@Lab-05-Task-01 ~]# docker exec -it mongo1 mongosh
# 为副本集创建用户
mongodb-cluster [direct: primary] test> use admin
mongodb-cluster [direct: primary] admin> db.getSiblingDB("admin").createUser(
  {
   user: "mongodblab",
   pwd: "mongodblab#PWD",
   roles: [{role: "clusterAdmin",db: "admin"},"readWriteAnyDatabase"]
  }
)
# 退出当前连接
mongodb-cluster [direct: primary] admin> quit()
```

通过上述 rs.status()命令查看副本集状态，mongo1（172.20.1.16:27017）为主节点，mongo2（172.20.1.16:27018）、mongo3（172.20.1.16:27019）为副本节点。

8、使用 MongoDB Compass 管理 MongoDB 数据库集群

（1）从 MongoDB Compass 的官方网站（https://www.mongodb.com）获取可执行程序。

（2）配置 MongoDB Compass 完成 3 台 MongoDB 的连接。

（3）点击“Databases”查看 MongoDB 数据库列表。

9、测试 MongoDB 副本集

场景 1：主节点增加数据，从节点同步增加

（1）使用 MongoDB Compass 通过用户“mongodblab”连接到主节点（172.20.1.16:27017）的 MongoDB 数据库，创建数据库、集合，并添加数据。

（2）使用 MongoDB Compass 通过用户“mongodblab”连接到副本节点（172.20.1.16:27018）的 MongoDB 数据库，查看在主节点上创建的数据库、集合以及添加的数据，是否存在。

（3）使用 MongoDB Compass 通过用户“mongodblab”连接到副本节点（172.20.1.16:27019）的 MongoDB 数据库，查看在主节点上创建的数据库、集合以及添加的数据，是否存在。

场景 2：主节点删除数据，从节点同步删除

（1）使用 MongoDB Compass 通过用户“mongodblab”连接到主节点（172.20.1.16:27017）的 MongoDB 数据库，删除数据。

（2）使用 MongoDB Compass 通过用户“mongodblab”连接到副本节点（172.20.1.16:27018）的 MongoDB 数据库，查看在主节点上删除的数据，是否已不存在。

（3）使用 MongoDB Compass 通过用户“mongodblab”连接到副本节点（172.20.1.16:

27019）的 MongoDB 数据库，查看在主节点上删除的数据，是否已不存在。

场景 3：主节点宕机，业务不受影响

（1）使用 docker stop 命令，关闭主节点容器，以模拟主节点宕机故障。

（2）使用 docker exec -it mongo2 mongosh 命令连接副本节点（172.20.1.16:27018）的 MongoDB 数据库客户端，查看当前主节点所在的容器。

```
# # 停止主节点对应容器
[root@Lab-05-Task-01 ~]# docker stop mongo1

# 连接 MongoDB 客户端
[root@Lab-05-Task-01 ~]# docker exec -it mongo2 mongosh
# 查看当前主节点所在的容器
mongodb-cluster [direct: secondary] test> rs.status()
```

场景 4：原主节点恢复正常，业务不受影响

（1）使用 docker start 命令，启动原主节点对应容器，以模拟原主节点（172.20.1.19:27017）恢复正常，使用 docker exec -it mongo1 mongosh 命令连接原主节点，并查看副本集状态。

（2）使用 MongoDB Compass 通过用户“mongodblab”连接到原主节点（172.20.1.19:27017），查看宕机期间未同步的数据是否已同步。

```
# 启动原主节点对应容器
[root@Lab-05-Task-01 ~]# docker start mongo1
# 连接 MongoDB 客户端
[root@Lab-05-Task-01 ~]# docker exec -it mongo1 mongosh
# 查看副本集状态
mongodb-cluster [direct: secondary] test> rs.status()
```