[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY07/CASE/01/index.html" \l "page_top_case)

# NSD ARCHITECTURE DAY08

1. [案例1：Zookeeper安装](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY07/CASE/01/index.html" \l "case1)
2. [案例2：Kafka集群实验](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY07/CASE/01/index.html" \l "case2)
3. [案例3：Hadoop高可用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY07/CASE/01/index.html" \l "case3)
4. [案例4：高可用验证](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY07/CASE/01/index.html" \l "case4)

## 1 案例1：Zookeeper安装

### 1.1 问题

本案例要求：

* 搭建Zookeeper集群并查看各服务器的角色
* 停止Leader并查看各服务器的角色

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：安装Zookeeper

1）编辑/etc/hosts ,所有集群主机可以相互 ping 通（在nn01上面配置，同步到node1，node2，node3）

1. [root@nn01 hadoop]# vim /etc/hosts
2. 192.168.1.60 nn01
3. 192.168.1.61 node1
4. 192.168.1.62 node2
5. 192.168.1.63 node3
6. 192.168.1.66 node4
7. [root@nn01 hadoop]# for i in {62..64} \
8. do    \
9. scp /etc/hosts 192.168.1.$i:/etc/    \
10. done        //同步配置
11. hosts 100% 253 639.2KB/s 00:00
12. hosts 100% 253 497.7KB/s 00:00
13. hosts 100% 253 662.2KB/s 00:00

2）安装 java-1.8.0-openjdk-devel,由于之前的hadoop上面已经安装过，这里不再安装，若是新机器要安装

3）zookeeper 解压拷贝到 /usr/local/zookeeper

1. [root@nn01 ~]# tar -xf zookeeper-3.4.13.tar.gz
2. [root@nn01 ~]# mv zookeeper-3.4.13 /usr/local/zookeeper

4）配置文件改名，并在最后添加配置

1. [root@nn01 ~]# cd /usr/local/zookeeper/conf/
2. [root@nn01 conf]# ls
3. configuration.xsl log4j.properties zoo\_sample.cfg
4. [root@nn01 conf]# mv zoo\_sample.cfg zoo.cfg
5. [root@nn01 conf]# chown root.root zoo.cfg
6. [root@nn01 conf]# vim zoo.cfg
7. server.1=node1:2888:3888
8. server.2=node2:2888:3888
9. server.3=node3:2888:3888
10. server.4=nn01:2888:3888:observer

5）拷贝 /usr/local/zookeeper 到其他集群主机

1. [root@nn01 conf]# for i in {62..64}; do rsync -aSH --delete /usr/local/zookeeper/ 192.168.1.$i:/usr/local/zookeeper -e 'ssh' & done
2. [4] 4956
3. [5] 4957
4. [6] 4958

6）创建 mkdir /tmp/zookeeper，每一台都要

1. [root@nn01 conf]# mkdir /tmp/zookeeper
2. [root@nn01 conf]# ssh node1 mkdir /tmp/zookeeper
3. [root@nn01 conf]# ssh node2 mkdir /tmp/zookeeper
4. [root@nn01 conf]# ssh node3 mkdir /tmp/zookeeper

7）创建 myid 文件，id 必须与配置文件里主机名对应的 server.(id) 一致

1. [root@nn01 conf]# echo 4 >/tmp/zookeeper/myid
2. [root@nn01 conf]# ssh node1 'echo 1 >/tmp/zookeeper/myid'
3. [root@nn01 conf]# ssh node2 'echo 2 >/tmp/zookeeper/myid'
4. [root@nn01 conf]# ssh node3 'echo 3 >/tmp/zookeeper/myid'

8）启动服务，单启动一台无法查看状态，需要启动全部集群以后才能查看状态，每一台上面都要手工启动（以nn01为例子）

1. [root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start
2. ZooKeeper JMX enabled by default
3. Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
4. Starting zookeeper ... STARTED

注意：刚启动zookeeper查看状态的时候报错，启动的数量要保证半数以上，这时再去看就成功了

9）查看状态

1. [root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh status
2. ZooKeeper JMX enabled by default
3. Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
4. Mode: observe
5. [root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh stop
6. //关闭之后查看状态其他服务器的角色
7. ZooKeeper JMX enabled by default
8. Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
9. Stopping zookeeper ... STOPPED
10. [root@nn01 conf]# yum -y install telnet
11. [root@nn01 conf]# telnet node3 2181
12. Trying 192.168.1.24...
13. Connected to node3.
14. Escape character is '^]'.
15. ruok        //发送
16. imokConnection closed by foreign host.        //imok回应的结果

10）利用 api 查看状态（nn01上面操作）

1. [root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start
2. [root@nn01 conf]# vim api.sh
3. #!/bin/bash
4. function getstatus(){
5. exec 9<>/dev/tcp/$1/2181 2>/dev/null
6. echo stat >&9
7. MODE=$(cat <&9 |grep -Po "(?<=Mode:).\*")
8. exec 9<&-
9. echo ${MODE:-NULL}
10. }
11. for i in node{1..3} nn01;do
12. echo -ne "${i}\t"
13. getstatus ${i}
14. done
15. [root@nn01 conf]# chmod 755 api.sh
16. [root@nn01 conf]# ./api.sh
17. node1    follower
18. node2    leader
19. node3    follower
20. nn01    observer

## 2 案例2：Kafka集群实验

### 2.1 问题

本案例要求：

* 利用Zookeeper搭建一个Kafka集群
* 创建一个topic
* 模拟生产者发布消息
* 模拟消费者接收消息

### 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：搭建Kafka集群

1）解压 kafka 压缩包

Kafka在node1，node2，node3上面操作即可

1. [root@node1 hadoop]# tar -xf kafka\_2.12-2.1.0.tgz

2）把 kafka 拷贝到 /usr/local/kafka 下面

1. [root@node1 ~]# mv kafka\_2.12-2.1.0 /usr/local/kafka

3）修改配置文件 /usr/local/kafka/config/server.properties

1. [root@node1 ~]# cd /usr/local/kafka/config
2. [root@node1 config]# vim server.properties
3. broker.id=22
4. zookeeper.connect=node1:2181,node2:2181,node3:2181

4）拷贝 kafka 到其他主机，并修改 broker.id ,不能重复

1. [root@node1 config]# for i in 63 64; do rsync -aSH --delete /usr/local/kafka 192.168.1.$i:/usr/local/; done
2. [1] 27072
3. [2] 27073
4. [root@node2 ~]# vim /usr/local/kafka/config/server.properties
5. //node2主机修改
6. broker.id=23
7. [root@node3 ~]# vim /usr/local/kafka/config/server.properties
8. //node3主机修改
9. broker.id=24

5）启动 kafka 集群（node1，node2，node3启动）

1. [root@node1 local]# /usr/local/kafka/bin/kafka-server-start.sh -daemon /usr/local/kafka/config/server.properties
2. [root@node1 local]# jps        //出现kafka
3. 26483 DataNode
4. 27859 Jps
5. 27833 Kafka
6. 26895 QuorumPeerMain

6）验证配置，创建一个 topic

1. [root@node1 local]# /usr/local/kafka/bin/kafka-topics.sh --create --partitions 1 --replication-factor 1 --zookeeper node3:2181 --topic aa
2. Created topic "aa".

7) 模拟生产者，发布消息

1. [root@node2 ~]# /usr/local/kafka/bin/kafka-console-producer.sh \
2. --broker-list node2:9092 --topic aa        //写一个数据
3. ccc
4. ddd

9）模拟消费者，接收消息

1. [root@node3 ~]# /usr/local/kafka/bin/kafka-console-consumer.sh \
2. --bootstrap-server node1:9092 --topic aa        //这边会直接同步
3. ccc
4. ddd

注意：kafka比较吃内存，做完这个kafka的实验可以把它停了

## 3 案例3：Hadoop高可用

### 3.1 问题

本案例要求：

* 配置Hadoop的高可用
* 修改配置文件

### 3.2 方案

配置Hadoop的高可用，解决NameNode单点故障问题，使用之前搭建好的hadoop集群，新添加一台nn02，ip为192.168.1.66，具体要求如图-1所示：

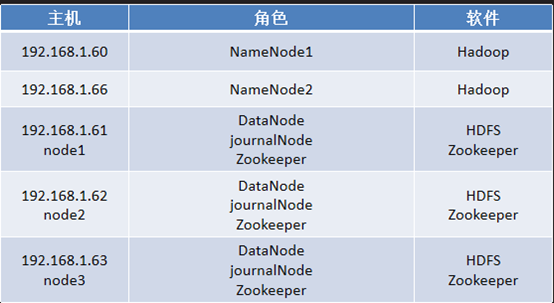


图-1

### 3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：hadoop的高可用

1）停止所有服务（由于 kafka的实验做完之后就已经停止，这里不在重复）

1. [root@nn01 ~]# cd /usr/local/hadoop/
2. [root@nn01 hadoop]# ./sbin/stop-all.sh //停止所有服务

2）启动zookeeper（需要一台一台的启动）这里以nn01为例子

1. [root@nn01 hadoop]# /usr/local/zookeeper/bin/zkServer.sh start
2. [root@nn01 hadoop]# sh /usr/local/zookeeper/conf/api.sh //利用之前写好的脚本查看
3. node1    follower
4. node2    leader
5. node3    follower
6. nn01    observer

3）新加一台机器nn02，这里之前有一台node4，可以用这个作为nn02

1. [root@node4 ~]# echo nn02 > /etc/hostname
2. [root@node4 ~]# hostname nn02

4）修改vim /etc/hosts

1. [root@nn01 hadoop]# vim /etc/hosts
2. 192.168.1.60 nn01
3. 192.168.1.66 nn02
4. 192.168.1.61 node1
5. 192.168.1.62 node2
6. 192.168.1.63 node3

5）同步到nn02，node1，node2，node3

1. [root@nn01 hadoop]# for i in {61..63} 66; do rsync -aSH --delete /etc/hosts 192.168.1.$i:/etc/hosts -e 'ssh' & done
2. [1] 14355
3. [2] 14356
4. [3] 14357
5. [4] 14358

6）配置SSH信任关系

注意：nn01和nn02互相连接不需要密码，nn02连接自己和node1，node2，node3同样不需要密码

1. [root@nn02 ~]# vim /etc/ssh/ssh\_config
2. Host \*
3. GSSAPIAuthentication yes
4. StrictHostKeyChecking no
5. [root@nn01 hadoop]# cd /root/.ssh/
6. [root@nn01 .ssh]# scp id\_rsa id\_rsa.pub nn02:/root/.ssh/
7. //把nn01的公钥私钥考给nn02

7）所有的主机删除/var/hadoop/\*

1. [root@nn01 .ssh]# rm -rf /var/hadoop/\*

8）配置 core-site

1. [root@nn01 .ssh]# vim /usr/local/hadoop/etc/hadoop/core-site.xml
2. <configuration>
3. <property>
4. <name>fs.defaultFS</name>
5. <value>hdfs://nsdcluster</value>
6. //nsdcluster是随便起的名。相当于一个组，访问的时候访问这个组
7. </property>
8. <property>
9. <name>hadoop.tmp.dir</name>
10. <value>/var/hadoop</value>
11. </property>
12. <property>
13. <name>ha.zookeeper.quorum</name>
14. <value>node1:2181,node2:2181,node3:2181</value>    //zookeepe的地址
15. </property>
16. <property>
17. <name>hadoop.proxyuser.nfs.groups</name>
18. <value>\*</value>
19. </property>
20. <property>
21. <name>hadoop.proxyuser.nfs.hosts</name>
22. <value>\*</value>
23. </property>
24. </configuration>

9）配置 hdfs-site

1. [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml
2. <configuration>
3. <property>
4. <name>dfs.replication</name>
5. <value>2</value>
6. </property>
7. <property>
8. <name>dfs.nameservices</name>
9. <value>nsdcluster</value>
10. </property>
11. <property>
12. <name>dfs.ha.namenodes.nsdcluster</name>
13. //nn1,nn2名称固定，是内置的变量，nsdcluster里面有nn1，nn2
14. <value>nn1,nn2</value>
15. </property>
16. <property>
17. <name>dfs.namenode.rpc-address.nsdcluster.nn1</name>
18. //声明nn1 8020为通讯端口，是nn01的rpc通讯端口
19. <value>nn01:8020</value>
20. </property>
21. <property>
22. <name>dfs.namenode.rpc-address.nsdcluster.nn2</name>
23. //声明nn2是谁，nn02的rpc通讯端口
24. <value>nn02:8020</value>
25. </property>
26. <property>
27. <name>dfs.namenode.http-address.nsdcluster.nn1</name>
28. //nn01的http通讯端口
29. <value>nn01:50070</value>
30. </property>
31. <property>
32. <name>dfs.namenode.http-address.nsdcluster.nn2</name>
33. //nn01和nn02的http通讯端口
34. <value>nn02:50070</value>
35. </property>
36. <property>
37. <name>dfs.namenode.shared.edits.dir</name>
38. //指定namenode元数据存储在journalnode中的路径
39. <value>qjournal://node1:8485;node2:8485;node3:8485/nsdcluster</value>
40. </property>
41. <property>
42. <name>dfs.journalnode.edits.dir</name>
43. //指定journalnode日志文件存储的路径
44. <value>/var/hadoop/journal</value>
45. </property>
46. <property>
47. <name>dfs.client.failover.proxy.provider.nsdcluster</name>
48. //指定HDFS客户端连接active namenode的java类
49. <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
50. </property>
51. <property>
52. <name>dfs.ha.fencing.methods</name>                    //配置隔离机制为ssh
53. <value>sshfence</value>
54. </property>
55. <property>
56. <name>dfs.ha.fencing.ssh.private-key-files</name>    //指定密钥的位置
57. <value>/root/.ssh/id\_rsa</value>
58. </property>
59. <property>
60. <name>dfs.ha.automatic-failover.enabled</name>        //开启自动故障转移
61. <value>true</value>
62. </property>
63. </configuration>

10）配置yarn-site

1. [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/yarn-site.xml
2. <configuration>
3. <!-- Site specific YARN configuration properties -->
4. <property>
5. <name>yarn.nodemanager.aux-services</name>
6. <value>mapreduce\_shuffle</value>
7. </property>
8. <property>
9. <name>yarn.resourcemanager.ha.enabled</name>
10. <value>true</value>
11. </property>
12. <property>
13. <name>yarn.resourcemanager.ha.rm-ids</name>        //rm1,rm2代表nn01和nn02
14. <value>rm1,rm2</value>
15. </property>
16. <property>
17. <name>yarn.resourcemanager.recovery.enabled</name>
18. <value>true</value>
19. </property>
20. <property>
21. <name>yarn.resourcemanager.store.class</name>
22. <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>
23. </property>
24. <property>
25. <name>yarn.resourcemanager.zk-address</name>
26. <value>node1:2181,node2:2181,node3:2181</value>
27. </property>
28. <property>
29. <name>yarn.resourcemanager.cluster-id</name>
30. <value>yarn-ha</value>
31. </property>
32. <property>
33. <name>yarn.resourcemanager.hostname.rm1</name>
34. <value>nn01</value>
35. </property>
36. <property>
37. <name>yarn.resourcemanager.hostname.rm2</name>
38. <value>nn02</value>
39. </property>
40. </configuration>

11）同步到nn02，node1，node2，node3

1. [root@nn01 ~]# for i in {61..63} 66; do rsync -aSH --delete /usr/local/hadoop/ 192.168.1.$i:/usr/local/hadoop -e 'ssh' & done
2. [1] 25411
3. [2] 25412
4. [3] 25413
5. [4] 25414

12）删除所有机器上面的/user/local/hadoop/logs，方便排错

1. [root@nn01 ~]# for i in {60..63} 66; do ssh 192.168.1.$i rm -rf /usr/local/hadoop/logs ; done

13）同步配置

1. [root@nn01 ~]# for i in {61..63} 66; do rsync -aSH --delete /usr/local/hadoop 192.168.1.$i:/usr/local/hadoop -e 'ssh' & done
2. [1] 28235
3. [2] 28236
4. [3] 28237
5. [4] 28238

## 4 案例4：高可用验证

### 4.1 问题

本案例要求：

* 初始化集群
* 验证集群

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：验证hadoop的高可用

1）初始化ZK集群

1. [root@nn01 ~]# /usr/local/hadoop/bin/hdfs zkfc -formatZK
2. ...
3. 18/09/11 15:43:35 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/nsdcluster in ZK //出现Successfully即为成功
4. ...

2）在node1，node2，node3上面启动journalnode服务（以node1为例子）

1. [root@node1 ~]# /usr/local/hadoop/sbin/hadoop-daemon.sh start journalnode
2. starting journalnode, logging to /usr/local/hadoop/logs/hadoop-root-journalnode-node1.out
3. [root@node1 ~]# jps
4. 29262 JournalNode
5. 26895 QuorumPeerMain
6. 29311 Jps

3）格式化，先在node1，node2，node3上面启动journalnode才能格式化

1. [root@nn01 ~]# /usr/local/hadoop//bin/hdfs namenode -format
2. //出现Successfully即为成功
3. [root@nn01 hadoop]# ls /var/hadoop/
4. dfs

4）nn02数据同步到本地 /var/hadoop/dfs

1. [root@nn02 ~]# cd /var/hadoop/
2. [root@nn02 hadoop]# ls
3. [root@nn02 hadoop]# rsync -aSH nn01:/var/hadoop/ /var/hadoop/
4. [root@nn02 hadoop]# ls
5. dfs

5）初始化 JNS

1. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs namenode -initializeSharedEdits
2. 18/09/11 16:26:15 INFO client.QuorumJournalManager: Successfully started new epoch 1        //出现Successfully，成功开启一个节点

6）停止 journalnode 服务（node1，node2，node3）

1. [root@node1 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh stop journalnode
2. stopping journalnode
3. [root@node1 hadoop]# jps
4. 29346 Jps
5. 26895 QuorumPeerMain

步骤二：启动集群

1）nn01上面操作

1. [root@nn01 hadoop]# /usr/local/hadoop/sbin/start-all.sh //启动所有集群
2. This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
3. Starting namenodes on [nn01 nn02]
4. nn01: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-nn01.out
5. nn02: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-nn02.out
6. node2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-node2.out
7. node3: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-node3.out
8. node1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-node1.out
9. Starting journal nodes [node1 node2 node3]
10. node1: starting journalnode, logging to /usr/local/hadoop/logs/hadoop-root-journalnode-node1.out
11. node3: starting journalnode, logging to /usr/local/hadoop/logs/hadoop-root-journalnode-node3.out
12. node2: starting journalnode, logging to /usr/local/hadoop/logs/hadoop-root-journalnode-node2.out
13. Starting ZK Failover Controllers on NN hosts [nn01 nn02]
14. nn01: starting zkfc, logging to /usr/local/hadoop/logs/hadoop-root-zkfc-nn01.out
15. nn02: starting zkfc, logging to /usr/local/hadoop/logs/hadoop-root-zkfc-nn02.out
16. starting yarn daemons
17. starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-root-resourcemanager-nn01.out
18. node2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-node2.out
19. node1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-node1.out
20. node3: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-node3.out

2）nn02上面操作

1. [root@nn02 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh start resourcemanager
2. starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-root-resourcemanager-nn02.out

3）查看集群状态

1. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
2. active
3. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2
4. standby
5. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
6. active
7. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2
8. standby

4）查看节点是否加入

1. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs dfsadmin -report
2. ...
3. Live datanodes (3): //会有三个节点
4. ...
5. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn node -list
6. Total Nodes:3
7. Node-Id     Node-State    Node-Http-Address    Number-of-Running-Containers
8. node2:43307     RUNNING     node2:8042     0
9. node1:34606     RUNNING     node1:8042     0
10. node3:36749     RUNNING     node3:8042     0

步骤三：访问集群

1）查看并创建

1. [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -ls /
2. [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -mkdir /aa //创建aa
3. [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -ls /        //再次查看
4. Found 1 items
5. drwxr-xr-x - root supergroup 0 2018-09-11 16:54 /aa
6. [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -put \*.txt /aa
7. [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -ls hdfs://nsdcluster/aa
8. //也可以这样查看
9. Found 3 items
10. -rw-r--r-- 2 root supergroup 86424 2018-09-11 17:00 hdfs://nsdcluster/aa/LICENSE.txt
11. -rw-r--r-- 2 root supergroup 14978 2018-09-11 17:00 hdfs://nsdcluster/aa/NOTICE.txt
12. -rw-r--r-- 2 root supergroup 1366 2018-09-11 17:00 hdfs://nsdcluster/aa/README.txt

2）验证高可用，关闭 active namenode

1. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
2. active
3. [root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh stop namenode
4. stopping namenode
5. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
6. //再次查看会报错
7. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2
8. //nn02由之前的standby变为active
9. active
10. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
11. active
12. [root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh stop resourcemanager
13. //停止resourcemanager
14. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2
15. active

3） 恢复节点

1. [root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh start namenode
2. //启动namenode
3. [root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh start resourcemanager
4. //启动resourcemanager
5. [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
6. //查看
7. [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
8. //查看