目录

1. 实验一： 高可用 ZooKeeper 集群部署

1.1. 实验目的

完成本实验，您应该能够：

- 掌握 ZooKeeper 集群的安装部署
- 掌握 ZooKeeper 集群的文件参数配置
- 掌握 ZooKeeper 集群的启动

1.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 熟悉 ZooKeeper 集群规划部署
- 熟悉 ZooKeeper 文件参数含义
- 熟悉 ZooKeeper 常用操作命令

1.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个节点，节点间网络互通，各节点最低配置：X 核 CPU、XGB 内存、XG 硬盘
运行环境	CentOS 7.4
用户名/密码	root/password hadoop/password
服务和组件	ZooKeeper，其他服务根据实验需求安装

1.4. 实验视图

高可用 ZooKeeper 集群部署实验部署流程如图 1-1

图 1-1 部署流程

1.5. 实验过程

1.5.1. 实验任务一： ZooKeeper 安装部署

1.5.1.1. 步骤一：解压安装 jdk(第 4 章已安装)

[root@master ~]# tar -zxvf /opt/software/jdk-8u152-linux-x64.tar.gz -C /usr/local/src

更改 jdk 的名称

[root@master ~]# mv /usr/local/src/jdk1.8.0_152/ /usr/local/src/java

1.5.1.2. 步骤二：安装 ZooKeeper

解压并安装 zookeeper 到 apps 下

[root@master ~]# tar -zxvf /opt/software/zookeeper-3.4.8.tar.gz -C /usr/local/src/

[root@master ~]# cd /usr/local/src/

[root@master src]# mv zookeeper-3.4.8 zookeeper

1.5.1.3. 步骤二：创建 ZooKeeper 数据目录

data 是用来传输数据的，logs 是用来记录日志的

[root@master src]# mkdir /usr/local/src/zookeeper/data

[root@master src]# mkdir /usr/local/src/zookeeper/logs

1.5.2. 实验任务二：ZooKeeper 文件参数配置

1.5.2.1. 步骤一：配置 ZooKeeper 环境变量

[root@master src]# cd ./zookeeper

[root@master zookeeper]# vi /etc/profile

添加如下配置：

#java environment(已配置)

export JAVA_HOME=/usr/local/src/java #JAVA_HOME 指向 JAVA 安装目录

export PATH=$PATH:$JAVA_HOME/bin#将 JAVA 安装目录加入 PATH 路径

#zookeeper environment

export ZK_HOME=/usr/local/src/zookeeper

export PATH=$PATH:$ZK_HOME/bin

保存并退出

1.5.2.2. 步骤二：修改 zoo.cfg 配置文件

首先先进入到 conf 目录下将 zoo.cfg 文件拷贝过来

[root@master zookeeper]# cd conf/

[root@master conf]# ls

configuration.xsl log4j.properties zoo_sample.cfg

[root@master conf]# cp zoo_sample.cfg zoo.cfg

[root@master conf]# vi zoo.cfg

添加并更改如下配置：

#修改

dataDir=/usr/local/src/zookeeper/data

#增加

dataLogDir=/usr/local/src/zookeeper/logs

server.1=master:2888:3888

server.2=slave1:2888:3888

server.3=slave2:2888:3888

//上面的 IP 可以换成自己的主机地址，或者换成主机名，一般我们换成主机名

保存并退出

1.5.2.3. 步骤三：创建 myid 配置文件

[root@master conf]# cd ..
[root@master zookeeper]# cd data/
[root@master data]# echo "1" > myid

1.5.3. 实验任务三：ZooKeeper 集群启动

1.5.3.1. 步骤一：分发 ZooKeeper 集群

[root@master data]# scp -r /usr/local/src/zookeeper/ root@slave1:/usr/local/src/
[root@master data]# scp -r /usr/local/src/zookeeper/ root@slave2:/usr/local/src/
分发环境变量并使其生效
[root@master data]# scp /etc/profile root@slave1:/etc/
[root@master data]# scp /etc/profile root@slave2:/etc/

[root@master data]# source /etc/profile
[root@slave1 ~]# source /etc/profile
[root@slave2 ~]# source /etc/profile

1.5.3.2. 步骤二：修改 myid 配置

master 对应 1,slave1 对应 2，slave2 对应 3
[root@master data]# cat myid
1
[root@slave1 ~]# cd /usr/local/src/zookeeper/data/
[root@slave1 data]# echo "2">myid
[root@slave1 data]# cat myid
2
[root@slave2 ~]# cd /usr/local/src/zookeeper/data/
[root@slave2 data]# echo "3">myid
[root@slave2 data]# cat myid
3

1.5.3.3. 步骤三：修改 ZooKeeper 安装目录的归属用户为 hadoop 用户。

[root@master data]# chown -R hadoop:hadoop /usr/local/src/zookeeper
[root@slave1 data] # chown -R hadoop:hadoop /usr/local/src/zookeeper
[root@slave2 data] # chown -R hadoop:hadoop /usr/local/src/zookeeper

1.5.3.4. 步骤四：启动 ZooKeeper 集群

关闭防火墙
[root@master data]# systemctl stop firewalld.service
[root@slave1 data]# systemctl stop firewalld.service
[root@slave2 data]# systemctl stop firewalld.service
关闭防火墙自启
[root@master data]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.

Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@slave1 data]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@slave2 data]# systemctl disable firewalld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
同时启动三个节点的 zookeeper
[root@master data]# su hadoop
[hadoop@master root]$ cd
[hadoop@master ~]$ source /etc/profile
[hadoop@master ~]$ zkServer.sh start # ZooKeeper 启动
JMX enabled by default
Jsing. config:' /usr/local/src/zookeeper/bin/ ../conf/zoo.cfg
starting zookeeper . .STARTED
[root@slave1 data]# su hadoop
[hadoop@slave1 root]$ cd
[hadoop@slave1 ~]$ source /etc/profile
[hadoop@slave1 ~]$ zkServer.sh start # ZooKeeper 启动
JMX enabled by default
Jsing. config:' /usr/local/src/zookeeper/bin/ ../conf/zoo.cfg
starting zookeeper . .STARTED
[root@slave2 data]# su hadoop
[hadoop@slave2 root]$ cd
[hadoop@slave2 ~]$ source /etc/profile
[hadoop@slave2 ~]$ zkServer.sh start # ZooKeeper 启动
JMX enabled by default
Jsing. config:' /usr/local/src/zookeeper/bin/ ../conf/zoo.cfg
starting zookeeper . .STARTED
查看状态
[hadoop@master ~]$ zkServer.sh status
JMX enabled by default
Using config: /usr/local/src/zookeeper/bin/ ../conf/zoo.cfg
Mode: follower # follower 状态

#slave1 节点状态
[hadoop@slave1 ~]$ zkServer.sh status
JMX enab1ed by default
Using config:' /usr/local/src/zookeeper/bin/ . ./conf/zoo.cfg
Mode:leader # leader 状态

#slave2 节点状态
[hadoop@slave2 ~]$ zkServer.sh status
JMX enabled by default

```
Using config:' /usr/local/src/zookeeper/bin/ . ./conf/zoo.cfg
Mode: follower              # follower 状态
```

2. 实验二 Hadoop HA 集群部署

2.1. 实验目的

完成本实验，您应该能够：

- 掌握 Hadoop HA 集群的文件参数配置
- 掌握 JournalNode 初始化和启动

2.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 熟悉 Hadoop HA 集群规划部署
- 熟悉 Hadoop 文件参数含义
- 熟悉 Hadoop 常用操作命令

2.3. 实验环境

本实验所需之主要资源环境如表 2-1 所示。

表 2-1 资源环境

服务器集群	X 个节点，节点间网络互通，各节点最低配置：X 核 CPU、XGB 内存、XG 硬盘
运行环境	CentOS XX
大数据平台	H3C Hadoop
服务和组件	HDFS、Yarn 等，其他服务根据实验需求安装

2.4. 实验视图

高可用 Hadoop 集群部署实验部署流程如图 2-1

图 2-1 部署流程

2.5. 实验过程

2.5.1. 实验任务一：ssh 免密配置(第四章已配置)

2.5.1.1. 步骤一：创建免密（三个主机同时进行）

```
[root@master ~]# su - hadoop
[hadoop@master ~]$ ssh-keygen -t rsa -P   ""
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:12nHZiHa8tK+hgmqid/Vvq5lAxpMZFFoY4lmnkoU7s4 root@master
The key's randomart image is:
+---[RSA 2048]----+
|    .. .+=.              |
| .. +o*                  |
| ..+ +..       . .     |
| .. oo         + + . |
| ...    o S + = =    |
| o.       o.+ = +      |
|    E     ....*oo       |
|      . o.. +o+.        |
|    ..+.. .o+oo.      |
+----[SHA256]-----+
[root@slave1 ~]# su - hadoop
[hadoop@slave1 ~]$ ssh-keygen -t rsa -P    ""
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:iOZ1zxts951cUelPDLPXGY0OPrFi+HDbcOLchhKf2v0 root@slave1
The key's randomart image is:
+---[RSA 2048]----+
|                         |
|                        .o|
|                    o +oo|
|         . . . . =.==|
|       o o S * = o+=|
|      o . . & @ . oo|
|        .      . # =      o|
|               = * ...o|
|              . o ..Eo.|
+----[SHA256]-----+
[root@slave2 ~]# su - hadoop
[hadoop@slave2 ~]$ ssh-keygen -t rsa -P    ""
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

```
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LxeFYtxLWfV9ZKhh9M3vM6vwuLJl6sVoJbezjeHHf1w root@slave2
The key's randomart image is:
+---[RSA 2048]----+
|                 .o...o|
|            . . +o..*.|
|             + =..o. *|
|            . o o.    o|
|             S.oo      .|
|              .*..   .E|
|             .ooX.   +o|
|             .+* Xo   *|
|             .+o*o+oo.|
+----[SHA256]-----+
```

2.5.1.2. 步骤二：创建公钥

[hadoop@master ~]$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
[hadoop@master ~]$ chmod 700 ~/.ssh/authorized_keys

2.5.1.3. 步骤三：将 masterr 创建的公钥发给 slave1

[hadoop@master ~]$ scp ~/.ssh/authorized_keys root@slave1:~/.ssh/

```
The authenticity of host 'slave1 (192.168.1.7)' can't be established.
ECDSA key fingerprint is SHA256:Nnk2MJS3KmUzmXXzgE0DTgnq990XctFMFUV82UdgFnQ.
ECDSA key fingerprint is MD5:f3:fa:be:c7:52:1e:96:ee:1b:7d:1a:26:23:a9:66:ec.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,192.168.1.7' (ECDSA) to the list of known hosts.
root@slave1's password:
authorized_keys
100%   393     319.0KB/s   00:00
```

2.5.1.4. 步骤四：将 slave1 的私钥加到公钥里

[hadoop@slave1 ~]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

2.5.1.5. 步骤五：将公钥发给 slave2

[hadoop@slave1 ~]$ scp ~/.ssh/authorized_keys root@slave2:~/.ssh/

```
The authenticity of host 'slave2 (192.168.1.8)' can't be established.
ECDSA key fingerprint is SHA256:Nnk2MJS3KmUzmXXzgE0DTgnq990XctFMFUV82UdgFnQ.
ECDSA key fingerprint is MD5:f3:fa:be:c7:52:1e:96:ee:1b:7d:1a:26:23:a9:66:ec.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave2,192.168.1.8' (ECDSA) to the list of known hosts.
root@slave2's password:
authorized_keys
100%   786     525.0KB/s   00:00
```

2.5.1.6. 步骤六：登陆测试

[hadoop@master ~]$ ssh slave1
Last login: Wed Jun 24 16:41:46 2020 from 192.168.1.7
[hadoop@slave1 ~]$ ssh slave2
Last login: Wed Jun 24 16:35:46 2020 from 192.168.1.1
[hadoop@slave2 ~]$ exit
登出
Connection to slave2 closed.
[hadoop@slave1 ~]$ exit
登出
Connection to slave1 closed.
[hadoop@master ~]$

2.5.2. 实验任务二：Hadoop HA 文件参数配置

2.5.2.1. 步骤一：解压安装 Hadoop

[hadoop@master ~]$ stop-all.sh
[hadoop@master ~]$ su root
删除第 4 章安装的 hadoop
[root@master ~]# rm -r -f /usr/local/src/hadoop
[root@slave1 ~]# rm -r -f /usr/local/src/hadoop
[root@slave2 ~]# rm -r -f /usr/local/src/hadoop

[root@master ~]# tar -zxvf /opt/software/hadoop-2.7.1.tar.gz -C /usr/local/src/

2.5.2.2. 步骤二：更改 hadoop 文件名

[root@master ~]# mv /usr/local/src/hadoop-2.7.1 /usr/local/src/hadoop

2.5.2.3. 步骤三：配置 hadoop 环境变量

[root@master ~]# vi /etc/profile
进行如下配置(此处需先删除第四章配置的环境变量)
#hadoop enviroment
export HADOOP_HOME=/usr/local/src/hadoop #HADOOP_HOME 指向 JAVA 安装目录
export HADOOP_PREFIX=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib:$HADOOP_COMMON_LIB_NATIVE_DIR"

```
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

#java environment
export JAVA_HOME=/usr/local/src/java      #JAVA_HOME 指向 JAVA 安装目录
export PATH=$PATH:$JAVA_HOME/bin#将 JAVA 安装目录加入 PATH 路径
#zookeeper environment
export ZK_HOME=/usr/local/src/zookeeper
export PATH=$PATH:$ZK_HOME/bin
```

保存并退出

2.5.2.4. 步骤四：配置 hadoop-env.sh 配置文件

进入到 hadoop/etc/hadoop 下

```
[root@master ~]# cd /usr/local/src/hadoop/etc/hadoop
[root@master hadoop]# vi hadoop-env.sh
```

在最下面添加如下配置：

```
export JAVA_HOME=/usr/local/src/java
```

保存并退出

2.5.2.5. 步骤五：配置 core-site.xml 配置文件

```
[root@master hadoop]# vi core-site.xml
```

添加如下配置：

```
<!-- 指定 hdfs 的 nameservice 为 mycluster -->
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://mycluster</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>file:/usr/local/src/hadoop/tmp</value>
    </property>
    <!-- 指定 zookeeper 地址 -->
    <property>
        <name>ha.zookeeper.quorum</name>
        <value>master:2181,slave1:2181,slave2:2181</value>
    </property>
    <!-- hadoop 链接 zookeeper 的超时时长设置 -->
    <property>
        <name>ha.zookeeper.session-timeout.ms</name>
        <value>30000</value>
        <description>ms</description>
    </property>
    <property>
        <name>fs.trash.interval</name>
        <value>1440</value>
```

```
    </property>
```

保存并退出

2.5.2.6. 步骤六：配置 hdfs-site.xml 配置文件

[root@master hadoop]# vi hdfs-site.xml

进行如下配置：

```
<!-- journalnode 集群之间通信的超时时间 -->
    <property>
        <name>dfs.qjournal.start-segment.timeout.ms</name>
        <value>60000</value>
    </property>
    <!--指定 hdfs 的 nameservice 为 mycluster，需要和 core-site.xml 中的保持一致 dfs.ha.namenodes.[nameservice id]为在 nameservice 中的每一个 NameNode 设置唯一标示符。配置一个逗号分隔的 NameNode ID 列表。这将是被 DataNode 识别为所有的 NameNode。如果使用"mycluster"作为 nameservice ID，并且使用"master"和"slave1"作为 NameNodes 标示符 -->
    <property>
        <name>dfs.nameservices</name>
        <value>mycluster</value>
    </property>
        <!-- mycluster 下面有两个 NameNode，分别是 master，slave1 -->
    <property>
        <name>dfs.ha.namenodes.mycluster</name>
        <value>master,slave1</value>
    </property>
    <!-- master 的 RPC 通信地址 -->
    <property>
        <name>dfs.namenode.rpc-address.mycluster.master</name>
        <value>master:8020</value>
    </property>
    <!-- slave1 的 RPC 通信地址 -->
    <property>
        <name>dfs.namenode.rpc-address.mycluster.slave1</name>
        <value>slave1:8020</value>
    </property>
     <!-- master 的 http 通信地址 -->
    <property>
        <name>dfs.namenode.http-address.mycluster.master</name>
        <value>master:50070</value>
    </property>
    <!-- slave1 的 http 通信地址 -->
    <property>
        <name>dfs.namenode.http-address.mycluster.slave1</name>
        <value>slave1:50070</value>
    </property>
```

```
<!-- 指定 NameNode 的 edits 元数据的共享存储位置。也就是 JournalNode 列表
     该 url 的配置格式：qjournal://host1:port1;host2:port2;host3:port3/journalId
     journalId 推荐使用 nameservice，默认端口号是：8485 -->
<property>
    <name>dfs.namenode.shared.edits.dir</name>
    <value>qjournal://master:8485;slave1:8485;slave2:8485/mycluster</value>
</property>
<!-- 配置失败自动切换实现方式 -->
<property>
    <name>dfs.client.failover.proxy.provider.mycluster</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
<!-- 配置隔离机制方法，多个机制用换行分割，即每个机制暂用一行 -->
<property>
    <name>dfs.ha.fencing.methods</name>
    <value>
        sshfence
    shell(/bin/true)
    </value>
</property>
<property>
   <name>dfs.permissions.enabled</name>
   <value>false</value>
</property>
<property>
    <name>dfs.support.append</name>
    <value>true</value>
</property>
<!-- 使用 sshfence 隔离机制时需要 ssh 免登陆 -->
<property>
    <name>dfs.ha.fencing.ssh.private-key-files</name>
    <value>/root/.ssh/id_rsa</value>
</property>
<!-- 指定副本数 -->
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/src/hadoop/tmp/hdfs/nn</value>
</property>
<property>
```

```
        <name>dfs.datanode.data.dir</name>
        <value>/usr/local/src/hadoop/tmp/hdfs/dn</value>
    </property>
    <!-- 指定 JournalNode 在本地磁盘存放数据的位置 -->
    <property>
        <name>dfs.journalnode.edits.dir</name>
        <value>/usr/local/src/hadoop/tmp/hdfs/jn</value>
    </property>
    <!-- 开启 NameNode 失败自动切换 -->
    <property>
        <name>dfs.ha.automatic-failover.enabled</name>
        <value>true</value>
    </property>
    <!-- 启用 webhdfs -->
    <property>
        <name>dfs.webhdfs.enabled</name>
        <value>true</value>
    </property>
    <!-- 配置 sshfence 隔离机制超时时间 -->
    <property>
        <name>dfs.ha.fencing.ssh.connect-timeout</name>
        <value>30000</value>
    </property>
    <property>
        <name>ha.failover-controller.cli-check.rpc-timeout.ms</name>
        <value>60000</value>
    </property>
```

保存并退出

2.5.2.7. 步骤七：配置 mapred-site.xml 配置文件

```
[root@master hadoop]# cp mapred-site.xml.template mapred-site.xml
[root@master hadoop]# vi mapred-site.xml
```

进行如下配置：

```
<!-- 指定 mr 框架为 yarn 方式 -->
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <!-- 指定 mapreduce jobhistory 地址 -->
    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>master:10020</value>
    </property>
    <!-- 任务历史服务器的 web 地址 -->
    <property>
```

```
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>master:19888</value>
    </property>
```

保存并退出

2.5.2.8. 步骤八：配置 yarn-site.xml 配置文件

[root@master hadoop]# vi yarn-site.xml

进行如下配置：

```
<!-- Site specific YARN configuration properties -->
    <!-- 开启 RM 高可用 -->
    <property>
        <name>yarn.resourcemanager.ha.enabled</name>
        <value>true</value>
    </property>
    <!-- 指定 RM 的 cluster id -->
    <property>
        <name>yarn.resourcemanager.cluster-id</name>
        <value>yrc</value>
    </property>
    <!-- 指定 RM 的名字 -->
    <property>
        <name>yarn.resourcemanager.ha.rm-ids</name>
        <value>rm1,rm2</value>
    </property>
    <!-- 分别指定 RM 的地址 -->
    <property>
        <name>yarn.resourcemanager.hostname.rm1</name>
        <value>master</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname.rm2</name>
        <value>slave1</value>
    </property>
    <!-- 指定 zk 集群地址 -->
    <property>
        <name>yarn.resourcemanager.zk-address</name>
        <value>master:2181,slave1:2181,slave2:2181</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
```

```
    </property>
    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>86400</value>
    </property>
    <!-- 启用自动恢复 -->
    <property>
        <name>yarn.resourcemanager.recovery.enabled</name>
        <value>true</value>
    </property>
    <!-- 制定 resourcemanager 的状态信息存储在 zookeeper 集群上 -->
    <property>
        <name>yarn.resourcemanager.store.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>
    </property>
```

保存并退出

2.5.2.9. 步骤九：配置 slaves 配置文件

[root@master hadoop]# vi slaves

进行如下配置：

master

slave1

slave2

2.5.2.10. 步骤十：解压包到指定目录

namenode、datanode、journalnode 等存放数据的公共目录为/usr/local/src/hadoop/tmp；

在 master 上执行如下：

```
[root@master hadoop]#    mkdir -p /usr/local/src/hadoop/tmp/hdfs/nn
[root@master hadoop]#    mkdir -p /usr/local/src/hadoop/tmp/hdfs/dn
[root@master hadoop]#    mkdir -p /usr/local/src/hadoop/tmp/hdfs/jn
[root@master hadoop]#    mkdir -p /usr/local/src/hadoop/tmp/logs
```

2.5.2.11. 步骤十一：分发文件

```
[root@master hadoop]# scp -r /etc/profile root@slave1:/etc/
[root@master hadoop]# scp -r /etc/profile root@slave2:/etc/

[root@master hadoop]# scp -r /usr/local/src/hadoop root@slave1:/usr/local/src/
[root@master hadoop]# scp -r /usr/local/src/hadoop root@slave2:/usr/local/src/
```

2.5.2.12. 步骤十二：修改目录所有者和所有者组

上述安装完成的 Hadoop 软件只能让 root 用户使用，要让 hadoop 用户能够运行 Hadoop 软件，需要将目录/usr/local/src 的所有者改为 hadoop 用户。

```
[root@master ~]# chown -R hadoop:hadoop /usr/local/src/hadoop/
[root@slave1~]# chown -R hadoop:hadoop /usr/local/src/hadoop/
[root@slave2~]# chown -R hadoop:hadoop /usr/local/src/hadoop/
```

2.5.2.13. 步骤十三：生效环境变量

[root@master hadoop]# su hadoop
[hadoop@master hadoop]$ source /etc/profile
[root@slave1 hadoop]# su hadoop
[hadoop@slave1 hadoop]$ source /etc/profile
[root@slave2 hadoop]# su hadoop
[hadoop@slave2 hadoop]$ source /etc/profile