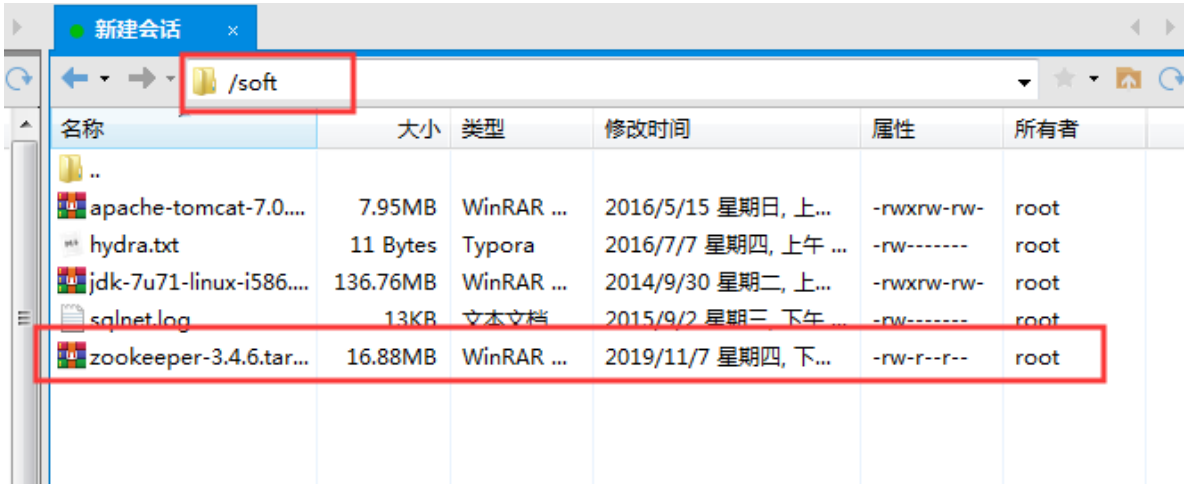
1.zookeeper 集群搭建

- 将zookeeper的tar包上传到linux 的soft下



- 解压zookeeper-3.4.6.tar.gz到 /usr/local

```
[root@localhost test]# tar zxf zookeeper-3.4.6.tar.gz -C /usr/local
```

- 进入解压后的zookeeper目录

```
[root@localhost local]# cd /usr/local/zookeeper-3.4.6/
```

- 创建data目录（在zookeeper目录下）

```
[root@localhost zookeeper-3.4.6]# mkdir data
```

- 在zookeeper目录下进入`conf目录`将`zoo_sample.cfg`文件改名为 `zoo.cfg`

```
[root@localhost conf]# mv zoo_sample.cfg zoo.cfg
```

- 在/usr/lcoal/ 目录下建立文件夹 zookeeper-cluster

```
[root@localhost local]# mkdir zookeeper-cluster/
```

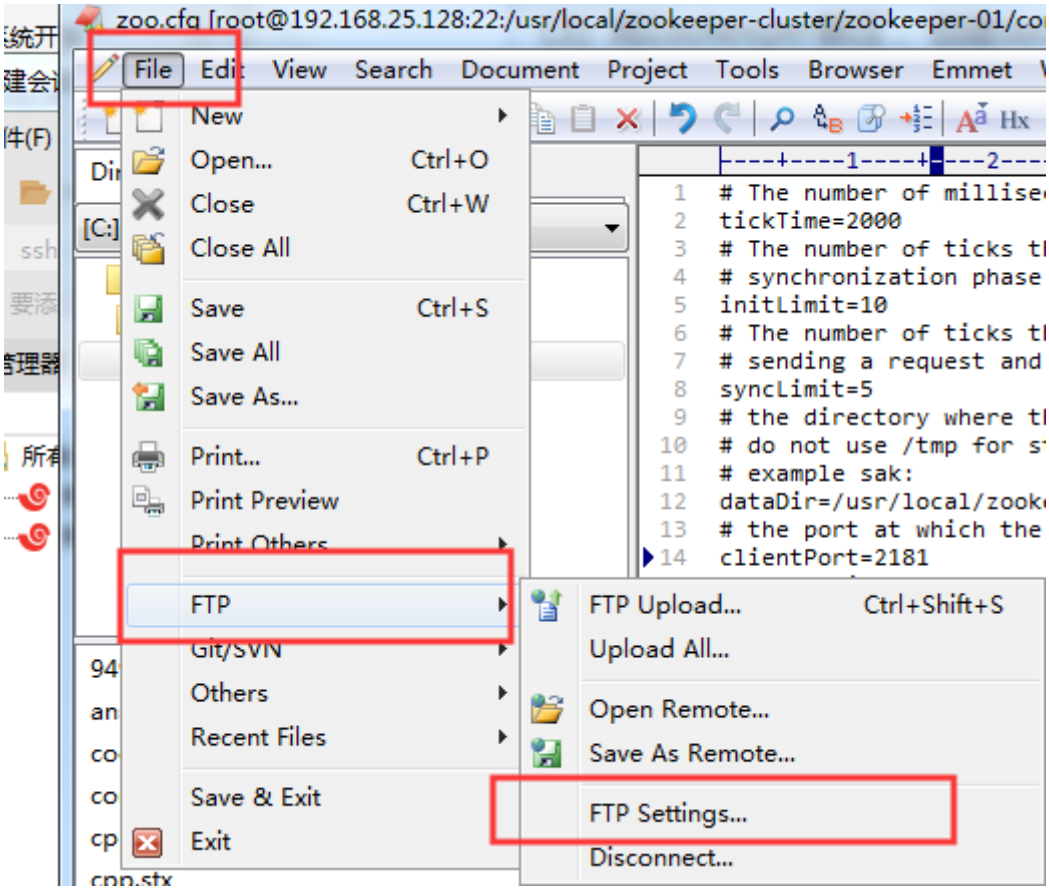
- 将解压后的zookeeper复制到建立的zookeeper-cluster中并命名zookeeper-01 以此类推 02 03

```
[root@localhost local]# cp -r zookeeper-3.4.6 /usr/local/zookeeper-cluster/zookeeper-02
[root@localhost local]# cp -r zookeeper-3.4.6 /usr/local/zookeeper-cluster/zookeeper-03
```

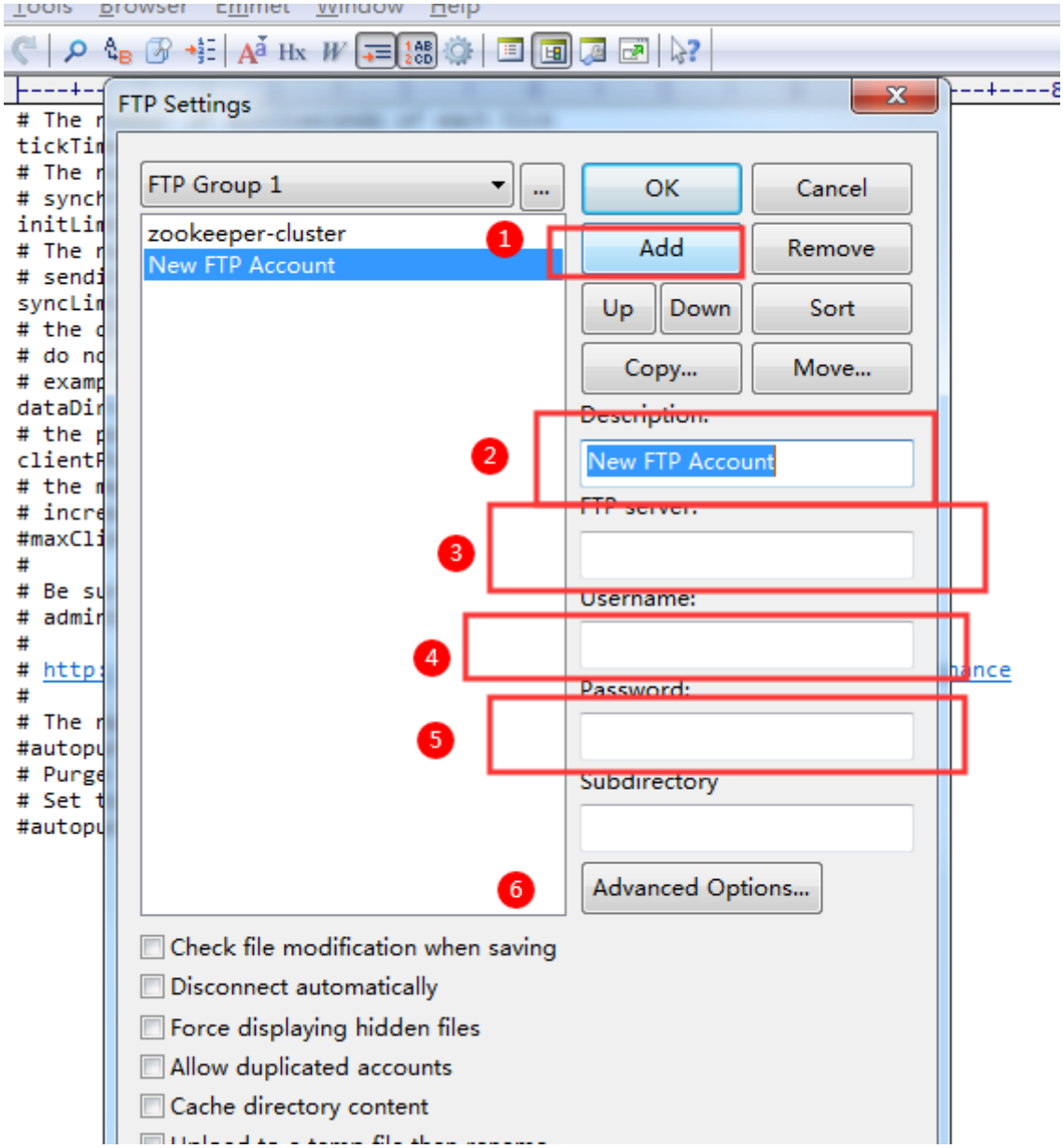
- 配置每一个zookeeper的dataDir 在每个zookeeper中的zoo.cfg中 并修改端口 分别为 2181 2182 2183

```
[root@localhost zookeeper-01]# cd data/
[root@localhost data]# pwd
/usr/local/zookeeper-cluster/zookeeper-01/data
```

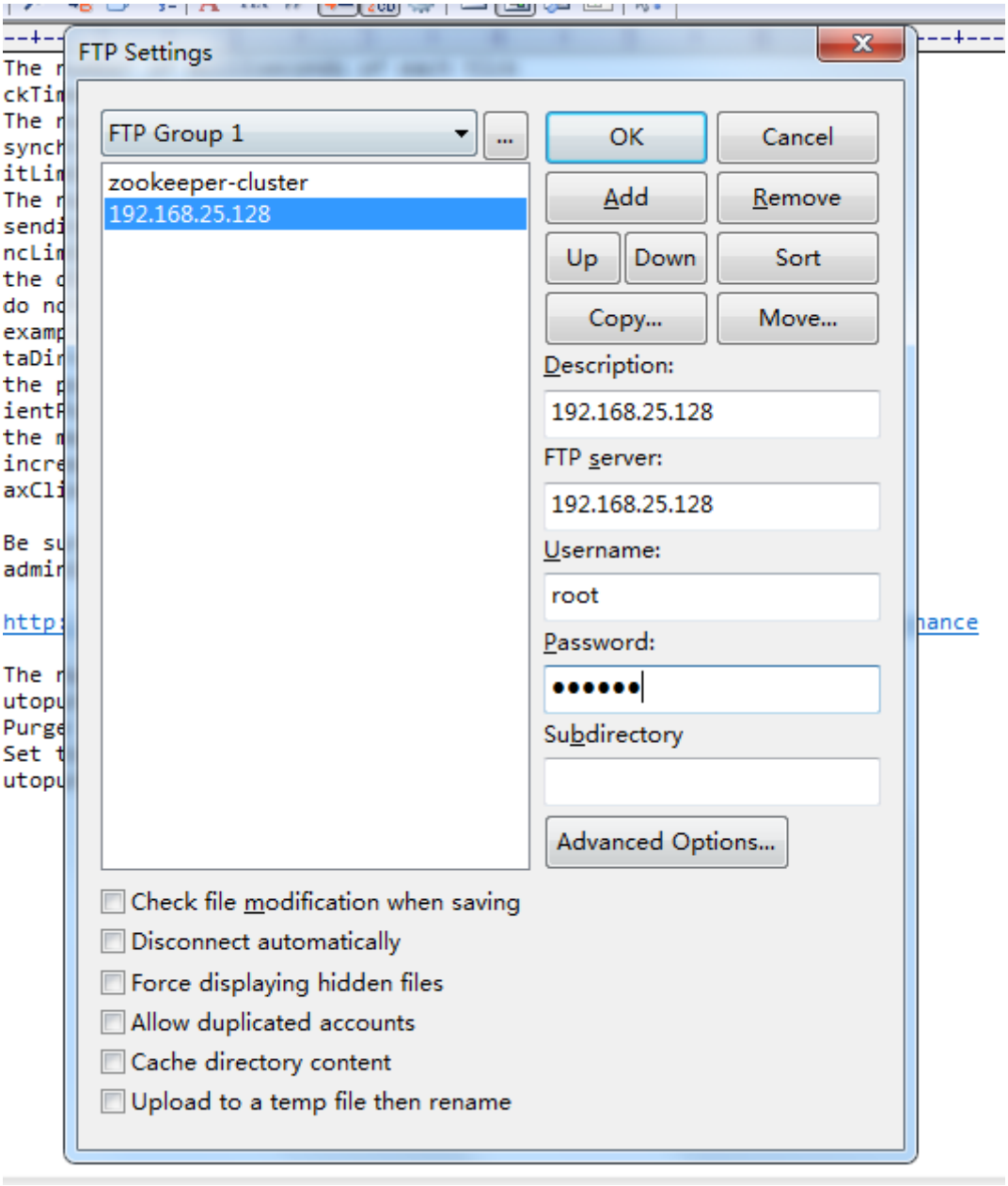
- 用editPlus连接192.168.25.128 具体过程如下


zoo.cfg [root@192.168.25.128:22:/usr/local/zookeeper-cluster/zookeeper-01/co
File
Edit
View
Search
Document
Project
Tools
Browser
Emmet
New
Open... Ctrl+O
Close Ctrl+W
Close All
Save Ctrl+S
Save All
Save As...
Print... Ctrl+P
Print Preview
Print Others
FTP
Git/SVN
Others
Recent Files
Save & Exit
Exit
FTP Upload... Ctrl+Shift+S
Upload All...
Open Remote...
Save As Remote...
FTP Settings...
Disconnect...
# The number of millise
tickTime=2000
# The number of ticks t
# synchronization phase
initLimit=10
# The number of ticks t
# sending a request and
syncLimit=5
# the directory where t
# do not use /tmp for s
# example sak:
dataDir=/usr/local/zook
# the port at which the
clientPort=2181

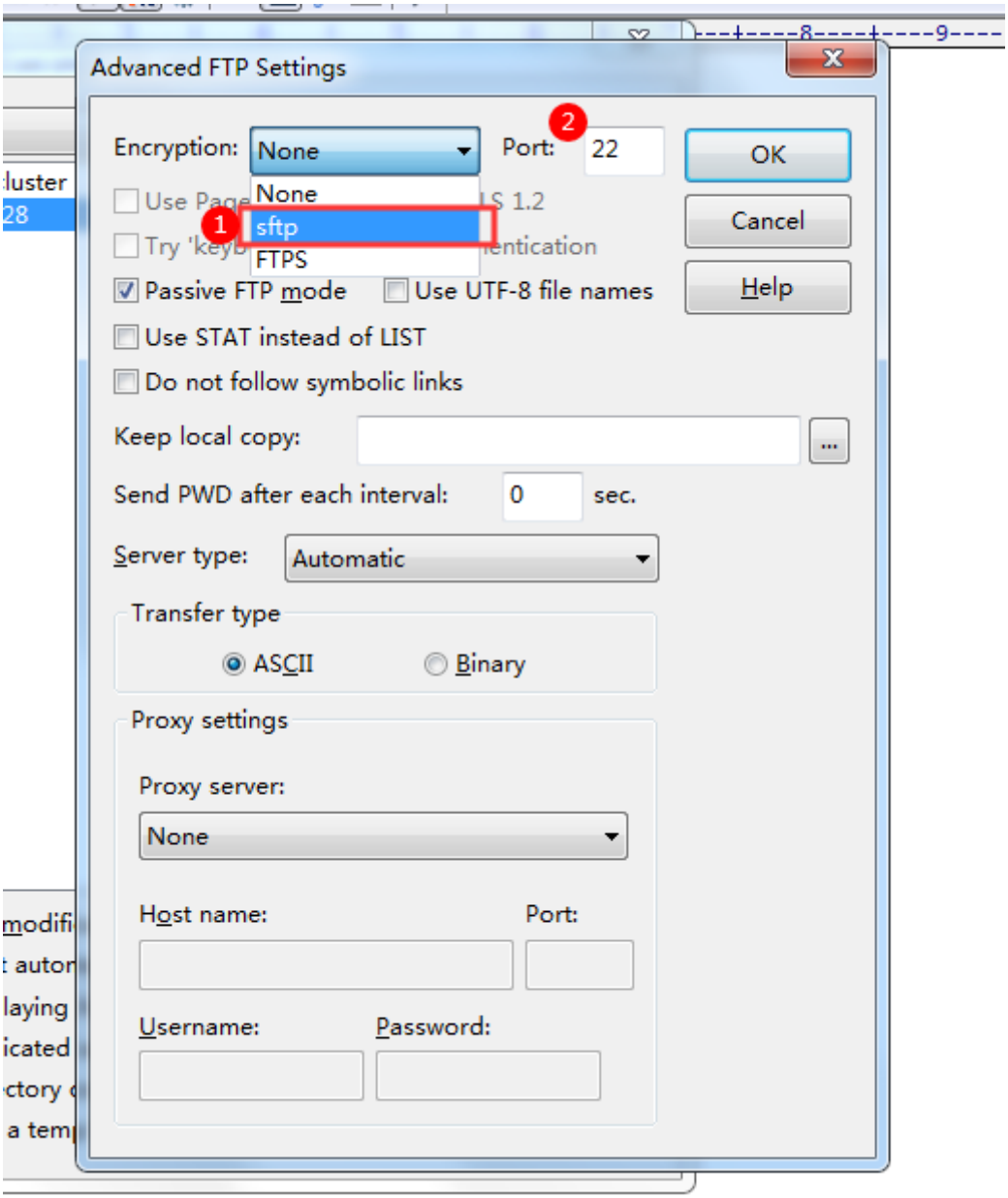
打开之后


FTP Settings
FTP Group 1
...
OK
Cancel
zookeeper-cluster
New FTP Account
Add
Remove
Up
Down
Sort
Copy...
Move...
Description:
New FTP Account
FTP server:
Username:
Password:
Subdirectory
Advanced Options...
Check file modification when saving
Disconnect automatically
Force displaying hidden files
Allow duplicated accounts
Cache directory content
1
2
3
4
5
6

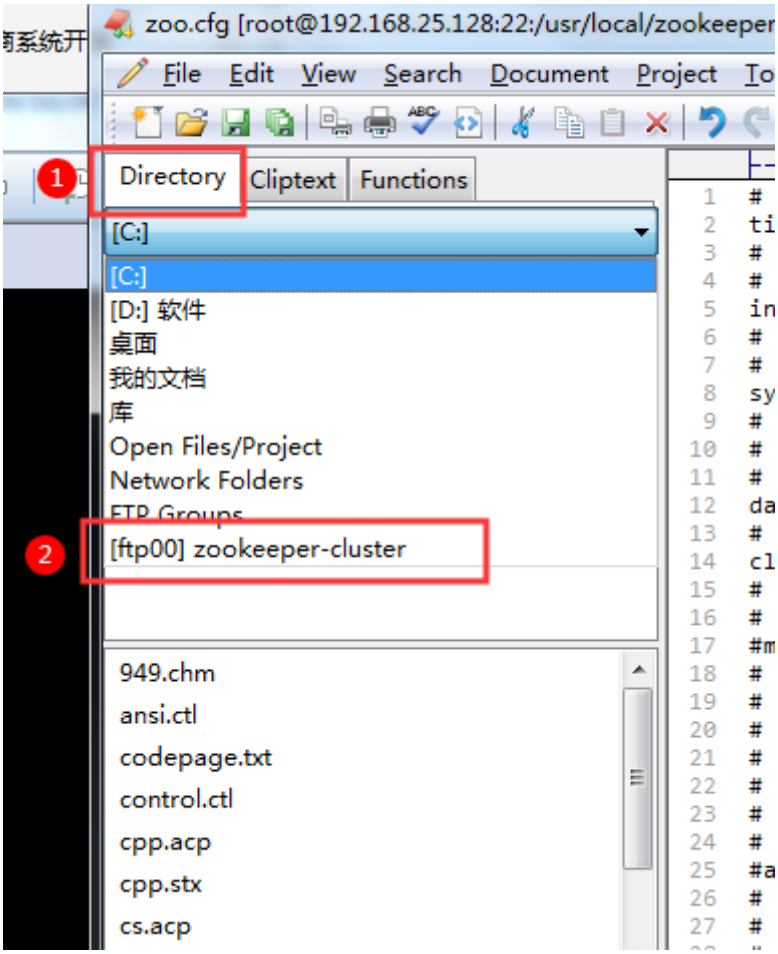
按顺序填写完成之后



点击Advanced Options


Advanced FTP Settings
Encryption:
None
Port:
22
OK
Cancel
Help
None
sftp
FTPS
Passive FTP mode
Use UTF-8 file names
Use STAT instead of LIST
Do not follow symbolic links
Keep local copy:
Send PWD after each interval:
0
sec.
Server type:
Automatic
Transfer type
ASCII
Binary
Proxy settings
Proxy server:
None
Host name:
Port:
Username:
Password:

完成后点击ok



`确定就行了`

这样就可以在windows里面改linux的文件了

- 完成后配置集群

  1. 在每个zookeeper的data目录下创建一个myid文件 文件的内容分别是1 2 3 这个文件的目的就是记录每个服务器的id

     ```
     [root@localhost zookeeper-03]# cd /usr/local/zookeeper-cluster/zookeeper-01
     [root@localhost zookeeper-01]# vi myid
     [root@localhost zookeeper-01]# cd /usr/local/zookeeper-cluster/zookeeper-02
     [root@localhost zookeeper-02]# vi myid
     [root@localhost zookeeper-02]# cd /usr/local/zookeeper-cluster/zookeeper-03
     [root@localhost zookeeper-03]# vi myid
     ```

     `vi 命令` 表示 `该文件存在就编辑 如果该文件不存在就创建并编辑`

  2. 在每个zookeeper的zoo.cfg配置客户端访问端口（clientPort）和集群服务器IP列表如下

     ```
     server.1=192.168.25.128:2881:3881
     server.2=192.168.25.128:2882:3882
     server.3=192.168.25.128:2883:3883
     ```

     ```
     解释：server.服务器ID=服务器IP地址：服务器之间通信端口：服务器之间投票选举端口
     ```

`注意这里的端口与前面的2181 2182 2183 端口无关！！！`

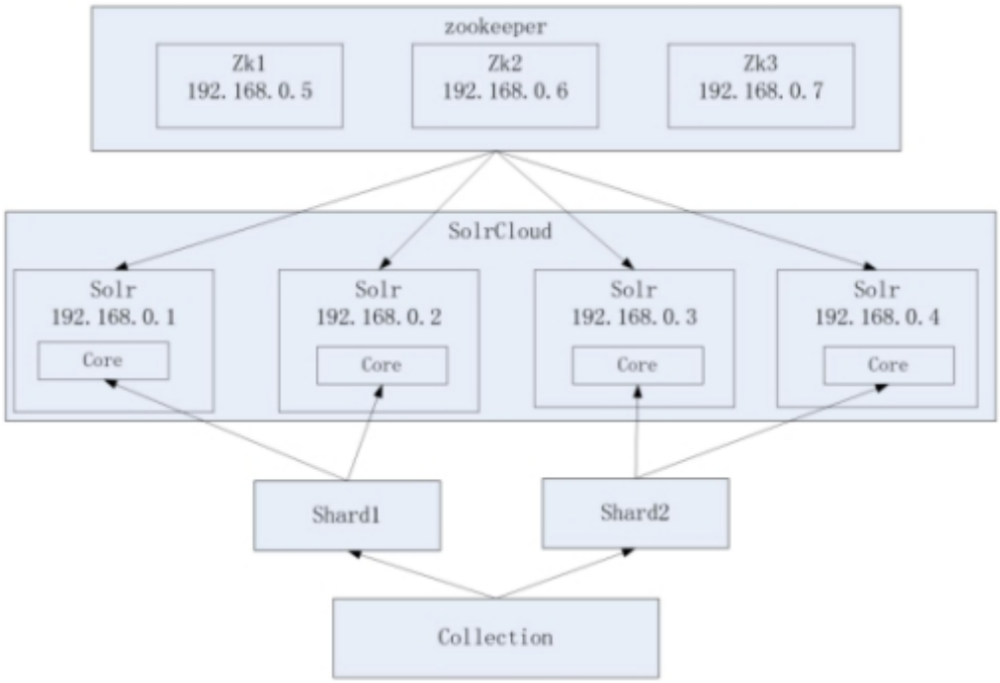
- 启动集群实际

  进入每个集群实例的bin目录下启动：

```
[root@localhost bin]# /usr/local/zookeeper-cluster/zookeeper-02/bin/zkServer.sh start
JMX enabled by default
Using config: /usr/local/zookeeper-cluster/zookeeper-02/bin/../conf/zoo.cfg
Starting zookeeper ... ^[[ASTARTED
[root@localhost bin]# /usr/local/zookeeper-cluster/zookeeper-02/bin/zkServer.sh start
JMX enabled by default
Using config: /usr/local/zookeeper-cluster/zookeeper-02/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@localhost bin]# /usr/local/zookeeper-cluster/zookeeper-03/bin/zkServer.sh start
JMX enabled by default
Using config: /usr/local/zookeeper-cluster/zookeeper-03/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

2.solrCloud 搭建

1. 搭建的要求



zookeeper作为句群的管理工具

1. 集群管理：容错、负载均衡。
2. 配置文件的集中管理
3. 集群的入口

需要实现 zookeeper 高可用，需要搭建zookeeper集群。建议是奇数节点。需要三个 zookeeper 服务

器。

搭建 `solr 集群需要 7 台服务器`（搭建伪分布式，建议虚拟机的内存 1G 以上）：

`需要三个 zookeeper 节点`

`需要四个 tomcat 节点`

2. 准备工作
   - 环境准备

     linux-cent-os

     linux-jdk.tar

     linux-Apache-tomcat-7.0.47.tar

linux-zookeeper-3.4.6.tar

linux-solr-4.10.3.tar

- 步骤

  1. 搭建zookeeper集群（见上文）
  2. 将部署了solr 的tomcat上传到linux
  3. 在linux中创建文件夹 /usr/local/solr-cloud 创建4个tomcat实例

     ```
     [root@localhost ~]# mkdir /usr/local/solr-cloud
     [root@localhost ~]# cp -r tomcat-solr /usr/local/solr-cloud/tomcat-1
     [root@localhost ~]# cp -r tomcat-solr /usr/local/solr-cloud/tomcat-2
     [root@localhost ~]# cp -r tomcat-solr /usr/local/solr-cloud/tomcat-3
     [root@localhost ~]# cp -r tomcat-solr /usr/local/solr-cloud/tomcat-4
     ```

  4. 将本地的solrhome上传到linux
  5. 在linux中创建文件夹 /usr/local/solrhomes ,将solrhome复制4份

     ```
     [root@localhost ~]# mkdir /usr/local/solrhomes
     [root@localhost ~]# cp -r solrhome /usr/local/solrhomes/solrhome-1
     [root@localhost ~]# cp -r solrhome /usr/local/solrhomes/solrhome-2
     [root@localhost ~]# cp -r solrhome /usr/local/solrhomes/solrhome-3
     [root@localhost ~]# cp -r solrhome /usr/local/solrhomes/solrhome-4
     ```

  6. 修改每个solr的 web.xml 文件，关联solrhome(以第一个solr服务器的修改为例)

     ```
     <env-entry>
         <env-entry-name>solr/home</env-entry-name>
         <env-entry-value>/usr/local/solrhomes/solrhome-1</env-entry-value>
         <env-entry-type>java.lang.String</env-entry-type>
     </env-entry>
     ```

  7. 修改每个tomcat的原运行端口8085 8080 8009 ，分别为

     ```
     8185 8180 8109
     8285 8280 8209
     8385 8380 8309
     8485 8480 8409
     ```

     举一个例子

     ```
     -->
     <!-- Note:  A "Server" is not itself a "Container", so you may not
          define subcomponents such as "Valves" at this level.
          Documentation at /docs/config/server.html
     -->
     <Server port="8485" shutdown="SHUTDOWN">
       <!-- Security listener. Documentation at /docs/config/listeners.html
       <Listener className="org.apache.catalina.security.SecurityListener"
       -->
     ```

```
66          Java AJP  Connector: /docs/config/ajp.html
67          APR (HTTP/AJP) Connector: /docs/apr.html
68          Define a non-SSL HTTP/1.1 Connector on port 8080
69      -->
70      <Connector port="8480" protocol="HTTP/1.1"
71                 connectionTimeout="20000"
72                 redirectPort="8443" />
73      <!-- A "Connector" using the shared thread pool-->
74      <!--
75      <Connector executor="tomcatThreadPool"
76                 port="8080" protocol="HTTP/1.1"
77                 connectionTimeout="20000"
78                 redirectPort="8443" />
79      -->
80      <!-- Define a SSL HTTP/1.1 Connector on port 8443
81           This connector uses the JSSE configuration, when using APR, the
82           connector should be using the OpenSSL style configuration
83           described in the APR documentation -->
84      <!--
85      <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
86                 maxThreads="150" scheme="https" secure="true"
87                 clientAuth="false" sslProtocol="TLS" />
88      -->
89
90      <!-- Define an AJP 1.3 Connector on port 8009 -->
91      <Connector port="8409" protocol="AJP/1.3" redirectPort="8443" />
92
93
```

文件目录在

```
/usr/local/solr-cloud/tomcat-01/conf/server.xml
/usr/local/solr-cloud/tomcat-02/conf/server.xml
/usr/local/solr-cloud/tomcat-03/conf/server.xml
/usr/local/solr-cloud/tomcat-04/conf/server.xml
```

- 配置集群

1. 修改每个 tomcat实例 bin 目录下的 catalina.sh 文件，把此配置添加到 catalina.sh中(第237行) ：

```
JAVA_OPTS="-
DzkHost=192.168.25.128:2181,192.168.25.128:2182,192.168.25.128:2183"
```

JAVA_OPTS ,顾名思义,是用来设置JVM相关运行参数的变量.此配置用于在tomcat启动时找到

zookeeper集群。

2. 配置 solrCloud 相关的配置。每个 solrhome 下都有一个solr.xml，把其中的 ip 及端口号配置好（是对应的tomcat的IP和端口）。

```
路径：solrhomes/solrhome-01/solr.xml
其他类推。。。
```

```
<solrcloud>
    <str name="host">192.168.25.128</str>
    <int name="hostPort">8180</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}
</bool> </solrcloud>
```

```
路径：solrhomes/solrhome-02/solr.xml
其他类推。。。
```

```
<solrcloud>
    <str name="host">192.168.25.140</str>
    <int name="hostPort">8280</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}
</bool> </solrcloud>
```

```
路径：solrhomes/solrhome-03/solr.xml
其他类推。。。
```

```
<solrcloud>
    <str name="host">192.168.25.128</str>
    <int name="hostPort">8380</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}
</bool> </solrcloud>
```

```
路径：solrhomes/solrhome-04/solr.xml
其他类推。。。
```

```
<solrcloud>
    <str name="host">192.168.25.128</str>
    <int name="hostPort">8480</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}
</bool> </solrcloud>
```

3. 让 zookeeper 统一管理配置文件。需要把 solrhome下collection1/conf 目录上传到zookeeper。上传任意 solrhome 中的配置文件即可。

   我们需要使用solr给我们提供的工具上传配置文件：

   `solr-4.10.3/example/scripts/cloud-scripts/zkcli.sh`

   将solr-4.10.3压缩包上传到linux，解压，然后进入solr-4.10.3/example/scripts/cloud-scripts目录，执行下列命令

```
//中间可能存在权限问题,给对应的文件权限就行了:

chmod -R 777 文件名
```

```
./zkcli.sh -zkhost
192.168.25.128:2181,192.168.25.128:2182,192.168.25.128:2183 -cmd
upconfig -confdir /usr/local/solrhomes/solrhome-01/collection1/conf -
confname myconf
```

   `参数解释`

   -zkhost ： 指定zookeeper地址列表

   -cmd ： 指定命令。upconfifig 为上传配置的命令

-confdir ： 配置文件所在目录

-confname ： 配置名称

- 启动集群

  1. 启动每个tomcat实例。`要保证 zookeeper 集群是启动状态`

```
回到根目录:
cd /

然后执行命令启动tomcat:
./usr/local/solr-cloud/tomcat-01/bin/startup.sh
./usr/local/solr-cloud/tomcat-02/bin/startup.sh
./usr/local/solr-cloud/tomcat-03/bin/startup.sh
./usr/local/solr-cloud/tomcat-04/bin/startup.sh
```

3. srpingDataSolr连接SolrCloud

   在SolrJ中提供一个叫做CloudSolrServer的类，它是SolrServer的子类，用于连接solrCloud

   它的构造参数就是zookeeper的地址列表，另外它要求要指定defaultCollection属性（默认的

   collection名称）

   我们现在修改springDataSolrDemo工程的配置文件 ，把原来的solr-server注销，替换为

   CloudSolrServer指定构造参数为地址列表，设置默认 collection名称

   1. `修改pinyougou-search-service工程的solr配置文件`

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:solr="http://www.springframework.org/schema/data/solr"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/data/solr
       http://www.springframework.org/schema/data/solr/spring-solr.xsd">
    <!--&lt;!&ndash;单击版的solr&ndash;&gt;-->
    <!--<solr:solr-server id="solrServer"
url="http://localhost:8080/solr/collection2"/>-->

    <!--集群版的solr-->
    <bean id="solrServer"
class="org.apache.solr.client.solrj.impl.CloudSolrServer">
        <!--构造注入 solr集群的地址列表-->
        <constructor-arg
value="192.168.25.128:2181,192.168.25.128:2182,192.168.25.128:2183"/>
        <!--默认的collection-->
        <property name="defaultCollection" value="collection1"/>
    </bean>
    <bean id="solrTemplate"
class="org.springframework.data.solr.core.SolrTemplate">
        <constructor-arg name="solrServer" ref="solrServer"/>
    </bean>
</beans>
```

4. 分片设置

- 创建新的collection进行分片处理。

  在浏览器输入以下地址，可以按照我们的要求 创建新的Collection

  ```
  http://192.168.25.140:8180/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2
  ```

  参数：

  name:将被创建的集合的名字

  numShards:集合创建时需要创建逻辑碎片的个数

  replicationFactor:分片的副本数。

  看到这个提示表示成功


```
▼<response>
  ▼<lst name="responseHeader">
     <int name="status">0</int>
     <int name="QTime">5912</int>
   </lst>
  ▼<lst name="success">
    ▼<lst>
      ▼<lst name="responseHeader">
         <int name="status">0</int>
         <int name="QTime">4468</int>
       </lst>
       <str name="core">collection2_shard1_replica1</str>
     </lst>
```


- 删除不用的collection

  ```
  http://192.168.25.140:8480/solr/admin/collections?action=DELETE&name=collection1
  ```


```
▼<response>
  ▼<lst name="responseHeader">
     <int name="status">0</int>
     <int name="QTime">699</int>
   </lst>
  ▼<lst name="success">
    ▼<lst name="192.168.25.135:8480_solr">
      ▼<lst name="responseHeader">
         <int name="status">0</int>
         <int name="QTime">126</int>
       </lst>
     </lst>
```


3.Redis集群搭建

3.1 redis 集群简介

3.1.1 什么是redis集群

为何要搭建Redis集群。Redis是在内存中保存数据的，而我们的电脑一般内存都不大，这也就意味着Redis不适

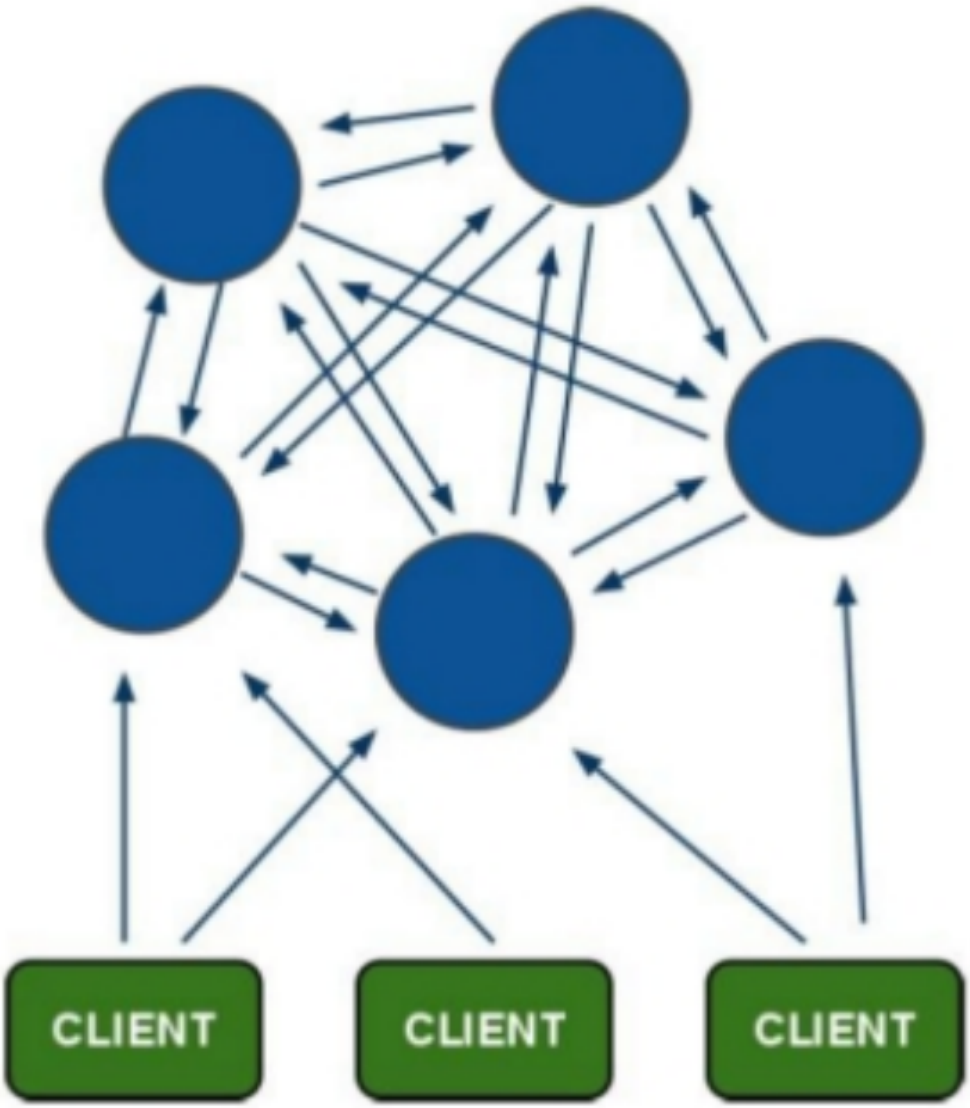
合存储大数据，适合存储大数据的是Hadoop生态系统的Hbase或者是MogoDB。Redis更适合处理高并发，一台设

备的存储能力是很有限的，但是多台设备协同合作，就可以让内存增大很多倍这就需要用到集群。

Redis集群搭建的方式有多种，例如使用客户端分片、Twemproxy、Codis等，但从redis 3.0之后版本支持

redis-cluster集群，它是Redis官方提出的解决方案，Redis-Cluster采用无中心结构，每个节点保存数据和

整个集群状态,每个节点都和其他所有节点连接。其redis-cluster架构图如下：



客户端与 redis 节点直连,不需要中间 proxy 层客户端不需要连接集群所有节点连接集群中任何一个可用节点

即可。

所有的 redis 节点彼此互联(PING-PONG 机制),内部使用二进制协议优化传输速度和带宽

3.1.2 分布式存储机制-槽

1. redis-cluster 把所有的物理节点映射到[0-16383]slot 上,cluster 负责维护node<->slot<->value
2. Redis 集群中内置了 16384 个哈希槽，当需要在 Redis 集群中放置一个 key-value 时，redis 先对key

   使用 crc16 算法算出一个结果，然后把结果对 16384 求余数，这样每个key 都会对应一个编号在 0-

   16383 之间的哈希槽，redis 会根据节点数量大致均等的将哈希槽映射到不同的节点。

   例如三个节点：槽分布的值如下：

   SERVER1: 0-5460

   SERVER2: 5461-10922

   SERVER3: 10923-16383

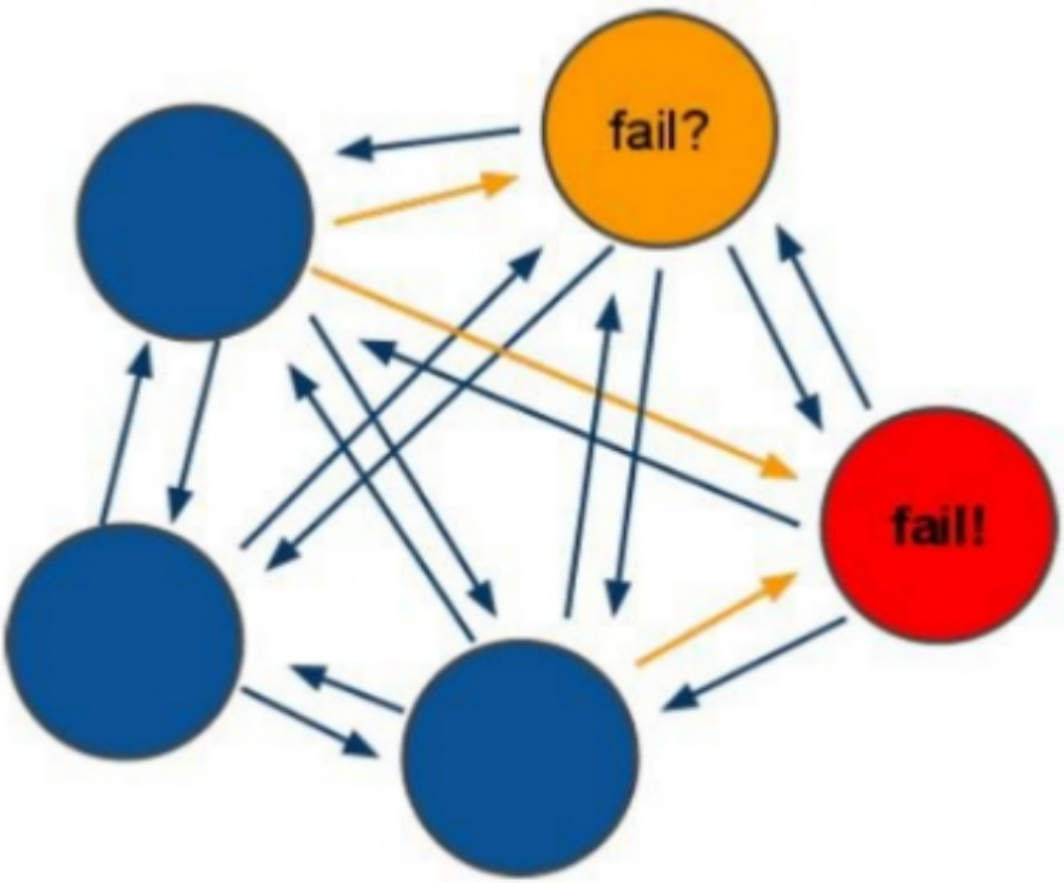
3.1.3 容错机制-投票

1. 选举过程是集群中所有master参与,如果半数以上master节点与故障节点通信超过(cluster-node-

   timeout),认为该节点故障，自动触发故障转移操作故障节点对应的从节点自动升级为主节点
2. 什么时候整个集群不可用(cluster_state:fail)?

   如果集群任意master挂掉,且当前master没有slave.集群进入fail状态,也可以理解成集群的slot映射

   [0-16383]不完成时进入fail状态



3.2搭建redis集群

3.2.1 搭建要求

需要 6 台 redis 服务器。搭建伪集群。

需要 6 个 redis 实例。

需要运行在不同的端口 7001-7006

3.2.2 准备工作

1. 安装gcc（如果安装过了跳过此步）

   Redis 是 c 语言开发的。安装 redis 需要 c 语言的编译环境。如果有 gcc 需要在线安装。

   ```
   yum install gcc-c++
   ```

2. 使用yum命令安装ruby（我们需要ruby脚本来实现集群的搭建）

   ```
   yum install ruby
   yum install rubygems
   ```

   `Ruby`

一种简单快捷的面向对象（面向对象程序设计）脚本语言，在20世纪90年代由日本人松本行弘(Yukihiro

Matsumoto)开发，遵守GPL协议和Ruby License。它的灵感与特性来自于 Perl、Smalltalk、Eiffffel、

Ada以及 Lisp 语言。由 Ruby 语言本身还发展出了JRuby（Java平台）、IronRuby（.NET平台）等其他平

台的 Ruby 语言替代品。Ruby的作者于1993年2月24日开始编写Ruby，直至1995年12月才正式公开发布于

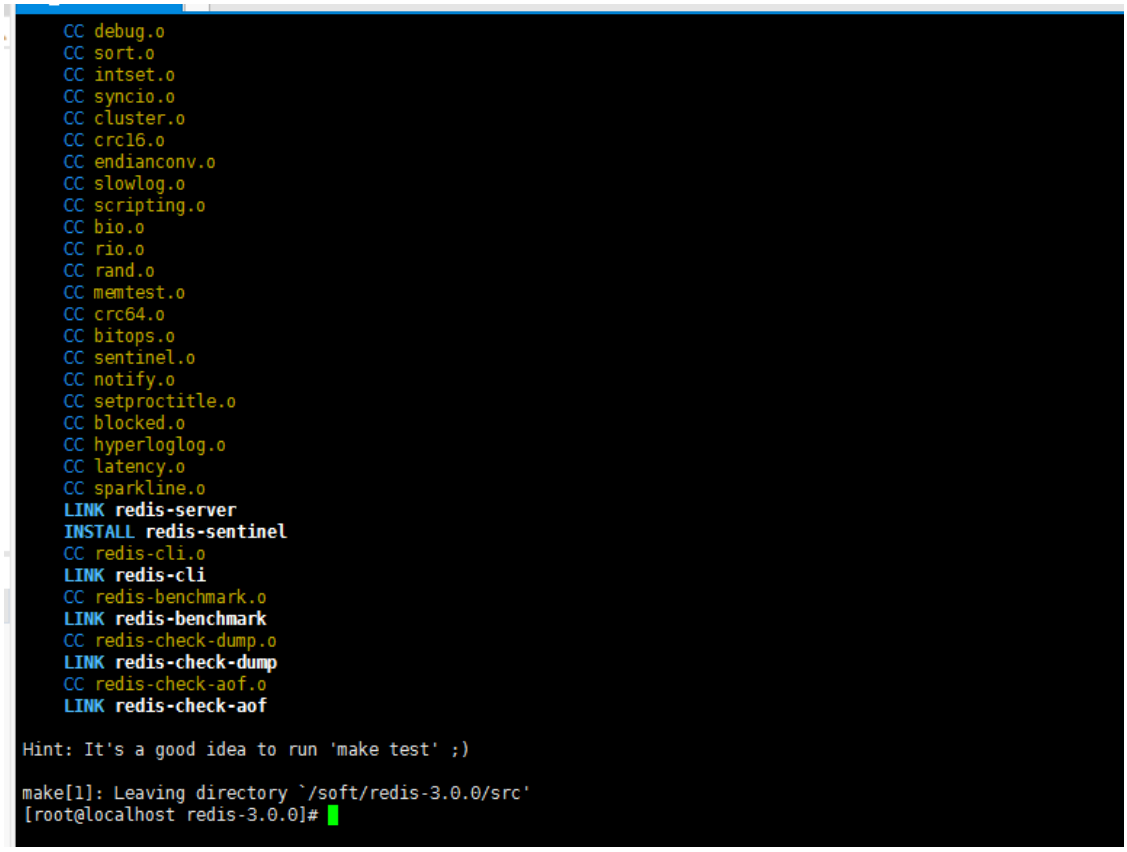
fj（新闻组）。因为Perl发音与6月诞生石pearl（珍珠）相同，因此Ruby以7月诞生石ruby（红宝石）命名

RubyGems简称gems，是一个用于对 Ruby组件进行打包的 Ruby 打包系统

3. 将redis源码包上传到 linux 系统 ，解压redis源码包
4. 编译redis源码 ，进入redis源码文件夹

```
make
```

看到以下输出结果表示编译成功



5. 创建redis 集群的目录/usr/local/redis-cluster,安装6个redis实例：

```
mkdir /usr/local/redis-cluster
```

（在刚刚make的目录下创建到上面指定的目录中）以第一个实例为例安装实例：

```
make install PREFIX=/usr/local/redis-cluster/redis-01
```

```
[root@localhost redis-3.0.0]# make install PREFIX=/usr/local/redis-cluster/redis-01
cd src && make install
make[1]: Entering directory `/soft/redis-3.0.0/src'

Hint: It's a good idea to run 'make test' ;)

    INSTALL install
    INSTALL install
    INSTALL install
    INSTALL install
    INSTALL install
make[1]: Leaving directory `/soft/redis-3.0.0/src'
[root@localhost redis-3.0.0]#
```

出现上面的提示表示安装实例成功！

> ！注意：这六个实例都是make install 得到的不是copy的！！！！

6. 复制配置文件 将redis-3.0.0/redis.conf 复制到每个实例的bin中

```
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-01/bin/
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-02/bin/
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-03/bin/
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-04/bin/
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-05/bin/
[root@localhost redis-3.0.0]# cp -r redis.conf /usr/local/redis-cluster/redis-06/bin/
```

3.2.3 配置集群

1. 修改每个redis节点的配置文件redis.conf

- 修改其中的端口从 7001 到 7006

```
# include /path/to/local.conf
# include /path/to/other.conf

################################## GENERAL  #####################################

# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize no

# When running daemonized, Redis writes a pid file in /var/run/redis.pid by
# default. You can specify a custom pid file location here.
pidfile /var/run/redis.pid

# Accept connections on the specified port, default is 6379.
# If port 0 is specified Redis will not listen on a TCP socket.
port 7001

# TCP listen() backlog.
#
# In high requests-per-second environments you need an high backlog in order
# to avoid slow clients connections issues. Note that the Linux kernel
# will silently truncate it to the value of /proc/sys/net/core/somaxconn so
# make sure to raise both the value of somaxconn and tcp_max_syn_backlog
# in order to get the desired effect.
tcp-backlog 511

# By default Redis listens for connections from all the network interfaces
```

- 将cluster-enabled-yes 前的注释去掉

```
617  # Set it to 0 or a negative value for unlimited execution without warnings.
618  lua-time-limit 5000
619
620  ################################ REDIS CLUSTER  ################################
621  #
622  # ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
623  # WARNING EXPERIMENTAL: Redis Cluster is considered to be stable code, however
624  # in order to mark it as "mature" we need to wait for a non trivial percentage
625  # of users to deploy it in production.
626  # ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
627  #
628  # Normal Redis instances can't be part of a Redis Cluster; only nodes that are
629  # started as cluster nodes can. In order to start a Redis instance as a
630  # cluster node enable the cluster support uncommenting the following:
631  #
632  cluster-enabled yes
633
634  # Every cluster node has a cluster configuration file. This file is not
635  # intended to be edited by hand. It is created and updated by Redis nodes.
636  # Every Redis Cluster node requires a different cluster configuration file.
637  # Make sure that instances running in the same system do not have
638  # overlapping cluster configuration file names.
```

2. 带配置启动每个redis实例

```
//以第一个实例为例启动
//先进到bin
cd /usr/local/redis-cluster/redis-1/bin/
//再带配置启动
./redis-server redis.conf
        //完了之后复制会话进入下个窗口否则redis会退出（如果不想退出也可以后台redis）
```

```
的箭头按钮。
1 新建会话   2 新建会话   3 新建会话   4 新建会话
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Tue Dec 10 01:55:11 2019 from 192.168.25.1
[root@localhost ~]# cd /usr/local/redis-cluster/redis-04/bin/
[root@localhost bin]# ./redis-server redis.conf
23909:M 10 Dec 01:56:15.162 * Increased maximum number of open files to 10032 (it was originally set to 102
4).
23909:M 10 Dec 01:56:15.163 # Warning: 32 bit instance detected but no memory limit set. Setting 3 GB maxme
mory limit with 'noeviction' policy now.
23909:M 10 Dec 01:56:15.165 * No cluster configuration found, I'm a1f90d90aa0b59db8011e0679cfb7f7f338ed1c5
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 3.0.0 (00000000/0) 32 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7004
      |    `-._   `._    /     _.-'    |     PID: 23909
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

23909:M 10 Dec 01:56:15.176 # Server started, Redis version 3.0.0
23909:M 10 Dec 01:56:15.176 # WARNING overcommit_memory is set to 0! Background save may fail under low mem
ory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run
the command 'sysctl vm.overcommit_memory=1' for this to take effect.
23909:M 10 Dec 01:56:15.176 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/
net/core/somaxconn is set to the lower value of 128.
23909:M 10 Dec 01:56:15.178 * The server is now ready to accept connections on port 7004
```

```
1 新建会话    2 新建会话
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Tue Dec 10 01:56:19 2019 from 192.168.25.1
[root@localhost ~]# cd /usr/local/redis-cluster/redis-06/bin/
[root@localhost bin]# ./redis-server redis.conf
23952:M 10 Dec 01:57:17.645 * Increased maximum number of open files to 10032 (it was originally set to 102
4).
23952:M 10 Dec 01:57:17.647 # Warning: 32 bit instance detected but no memory limit set. Setting 3 GB maxme
mory limit with 'noeviction' policy now.
23952:M 10 Dec 01:57:17.649 * No cluster configuration found, I'm 27817058dd3b7718d122196fb9241d8eb336c9f5

                Redis 3.0.0 (00000000/0) 32 bit

                Running in cluster mode
                Port: 7006
                PID: 23952

                      http://redis.io

23952:M 10 Dec 01:57:17.670 # Server started, Redis version 3.0.0
23952:M 10 Dec 01:57:17.670 # WARNING overcommit_memory is set to 0! Background save may fail under low mem
ory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run
the command 'sysctl vm.overcommit_memory=1' for this to take effect.
23952:M 10 Dec 01:57:17.670 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/
net/core/somaxconn is set to the lower value of 128.
23952:M 10 Dec 01:57:17.672 * The server is now ready to accept connections on port 7006
```

3.上传redis-3.0.0.gem ,安装ruby用于搭建redis集群脚本

```
//在redis.3.0.0.gem 所在的目录下安装 ,先给与权限参照上面的权限授予
gem install redis-3.0.0.gem
```

```
[root@localhost soft]# chmod -R 777 redis-3.0.0.gem
[root@localhost soft]# ll
total 166904
drwxr-xr-x   9 root root      4096 Dec  9 22:25 apache-tomcat-7.0.40
-rwxrw-rw-.  1 root root   8340063 May 14  2016 apache-tomcat-7.0.52.tar.gz
-rw-------.  1 root root        11 Jul  6  2016 hydra.txt
-rwxrw-rw-.  1 root root 143398235 Sep 29  2014 jdk-7u71-linux-i586.tar.gz
drwxrwxr-x   6 root root      4096 Apr  1  2015 redis-3.0.0
-rwxrwxrwx   1 root root     57856 Dec 10 01:20 redis-3.0.0.gem
-rw-r--r--   1 root root   1358081 Dec 10 01:20 redis-3.0.0.tar.gz
drwxrwxrwx   8 root root      4096 Dec  9 23:31 solr-4.10.3
drwxr-xr-x   5 root root      4096 Dec  9 22:39 solrHome
-rw-------.  1 root root     13057 Sep  1  2015 sqlnet.log
drwxr-xr-x  10 1000 1000      4096 Feb 20  2014 zookeeper-3.4.6
-rw-r--r--   1 root root  17699306 Nov  6 23:10 zookeeper-3.4.6.tar.gz
[root@localhost soft]# gem install redis-3.0.0.gem
Successfully installed redis-3.0.0
1 gem installed
Installing ri documentation for redis-3.0.0...
Installing RDoc documentation for redis-3.0.0...
```

4.使用ruby脚本搭建集群

进入redis源码目录中的src目录 执行命令:

```
./redis-trib.rb create --replicas 1 192.168.25.128:7001 192.168.25.128:7002
192.168.25.128:7003 192.168.25.128:7004 192.168.25.128:7005 192.168.25.128:7006
```

```
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join....
>>> Performing Cluster Check (using node 192.168.25.128:7001)
M: bfcb53306634248d32c739e49d28a5499576e4f9 192.168.25.128:7001
   slots:0-5460 (5461 slots) master
M: 044f7b478870e9c14b557f39d4789a8af0954246 192.168.25.128:7002
   slots:5461-10922 (5462 slots) master
M: a0f9416fc84f4f54496cadc20f0d3e0dda2f93e8 192.168.25.128:7003
   slots:10923-16383 (5461 slots) master
M: a1f90d90aa0b59db8011e0679cfb7f7f338ed1c5 192.168.25.128:7004
   slots: (0 slots) master
   replicates bfcb53306634248d32c739e49d28a5499576e4f9
M: fbb82fc63962958fe3f50fe101a866b172232455 192.168.25.128:7005
   slots: (0 slots) master
   replicates 044f7b478870e9c14b557f39d4789a8af0954246
M: 27817058dd3b7718d122196fb9241d8eb336c9f5 192.168.25.128:7006
   slots: (0 slots) master
   replicates a0f9416fc84f4f54496cadc20f0d3e0dda2f93e8
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
[root@localhost src]#
```

成功则出现以上提示！

3.3 连接redis-cluster

3.3.1 windows客户端工具连接

redis-client连接集群：

```
redis-cli -h 主机ip -p 端口（集群中任意端口） -c
```

-c：代表连接的是 redis 集群

测试值的存取：

（1）从本地连接到集群redis 使用7001端口加-c 参数

（2）存入name值为abc ， 系统提示此值被存入到了7002端口所在的redis （槽是xxxx）

（3）提取name的值，可以提取

（4）退出（quit）

（5）再次以7001端口进入 ，不带-c

（6）查询name值，无法获取，因为值在7002端口的redis上

（7）我们以7002端口进入，获取name值发现是可以获取的,而以其它端口进入均不能获取

3.3.2 SpringDataRedis连接Redis集群

修改pinyougou-common工程 添加spring配置文件：

`applicationContext-redis-cluster.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
    <!--0.读取配置文件-->
    <context:property-placeholder ignore-unresolvable="true"
                                  location="classpath*:properties/*.properties"/>

    <!--配置redis的集群配置-->
    <bean id="redisClusterConfiguration"
class="org.springframework.data.redis.connection.RedisClusterConfiguration">
        <property name="clusterNodes">
            <set>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host1}"/>
                    <constructor-arg name="port" value="${redis.port1}"/>
                </bean>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host2}"/>
                    <constructor-arg name="port" value="${redis.port2}"/>
                </bean>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host3}"/>
                    <constructor-arg name="port" value="${redis.port3}"/>
                </bean>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host4}"/>
                    <constructor-arg name="port" value="${redis.port4}"/>
                </bean>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host5}"/>
                    <constructor-arg name="port" value="${redis.port5}"/>
                </bean>
                <bean
class="org.springframework.data.redis.connection.RedisClusterNode">
                    <constructor-arg name="host" value="${redis.host6}"/>
                    <constructor-arg name="port" value="${redis.port6}"/>
                </bean>
            </set>
        </property>
    </bean>

    <!--配置redis的连接池-->
    <bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">
        <property name="maxIdle" value="${redis.maxIdle}" />
        <property name="maxWaitMillis" value="${redis.maxWait}" />
        <property name="testOnBorrow" value="${redis.testOnBorrow}" />
    </bean>
    <!--2.配置connectionFactory工厂对象 -->
```

```
    <bean id="connectionFactory"
class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">
        <!--2.1)引入集群的配置-->
        <constructor-arg name="clusterConfig" ref="redisClusterConfiguration"/>
        <!--引入连接池-->
        <constructor-arg ref="poolConfig"/>
    </bean>
    <!--配置redisTemplate对象 -->
    <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate">
        <property name="connectionFactory" ref="connectionFactory"/>
    </bean>
</beans>
```