当我们的项目在不知不觉中做大了之后，各种问题就出来了，真jb头疼，比如性能，业务系统的并行计算的一致性协调问题，比如分布式架构的事务问题，

我们需要多台机器共同commit事务，经典的案例当然是银行转账，支付宝转账这种，如果是一台机器的话，这个还是很方便的，windows中自带了一个事务协

调器mstsc，但是呢，你那种很大很牛逼的项目不可能全是windows服务器，对吧，有些人为了解决这个问题，会采用2pc，3pc这种算法，或者是paxos的思

想进行分布式下的一致性处理，当然在这个世界上，真的不需要你自己去开发这种协调性，因为现在已经有了专门解决这种问题的解决方案，比如zookeeper。

一：zookeeper集群搭建

　　有些人应该明白，zookeeper正是google的chubby的开源实现，使用zookeeper之前，我们先来搭建一个集群。

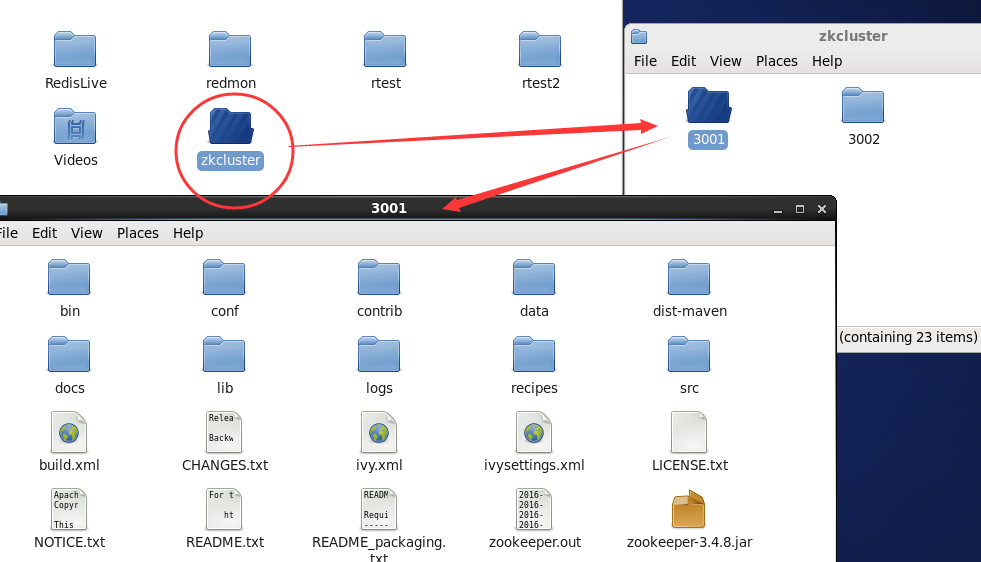
1. 下载

   从官网上，我们可以看到，zookeeper的最新版本是3.4.8，下载地址是：http://apache.fayea.com/zookeeper/zookeeper-3.4.8/，可以下载一下：



2. 文件夹配置

接下来我们解压一下，根目录为zkcluster，下面使用clientport（3000，3001，3002）这样的端口作为文件夹名称，里面就是zookeeper解压包，如下面这样：



3. 配置zoo.cfg

   现在我们有三个文件夹，也就是3个zookeeper程序，在3001/conf/下面有一个zoo\_sample.cfg文件，现在我们改成zoo.cfg，并且修改如下：

[复制代码](javascript:void(0);)

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/root/zkcluster/3001/data

dataLogDir=/root/zkcluster/3001/logs

# the port at which the clients will connect

clientPort=3001

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=6

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=192.168.161.134:2888:3888

server.2=192.168.161.134:2889:3889

server.3=192.168.161.134:2890:3890

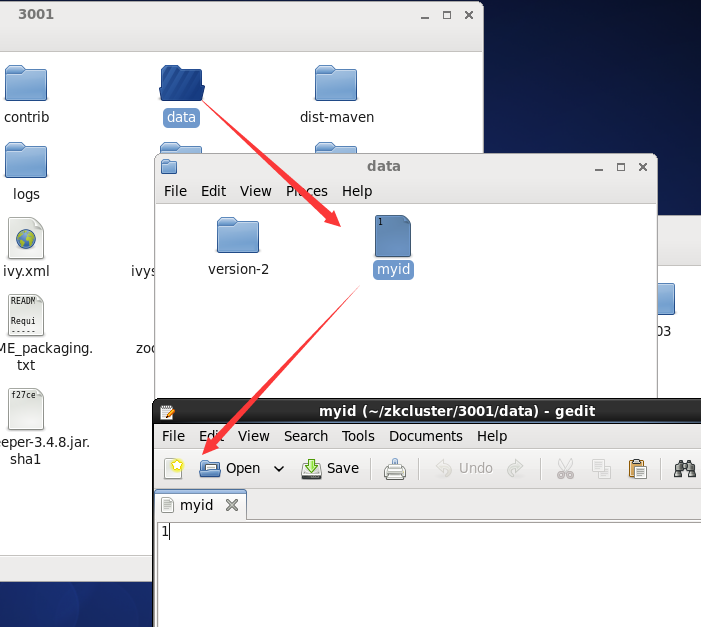
[复制代码](javascript:void(0);)

这里我们要注意的是，红色的部分分别就是：指定zookeeper的data和log文件夹，指定clientport访问的端口和servers的列表。

4. 生成pid文件

    我们在servers列表中，可以看到有server.1 ,server.2, server.3 三个字符串，生成pid文件的内容就取决如此，比如server.1的地址，

我们的pid文件里面就是1，不过要知道的是，pid文件要在data目录下，比如下面这样:



ok，同样的道理，3002和3003的文件夹同3001就可以了，比如他们的zoo.cfg如下：

--------  3002 --------------

[复制代码](javascript:void(0);)

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/root/zkcluster/3002/data

dataLogDir=/root/zkcluster/3002/logs

# the port at which the clients will connect

clientPort=3002

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=6

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=192.168.161.134:2888:3888

server.2=192.168.161.134:2889:3889

server.3=192.168.161.134:2890:3890

[复制代码](javascript:void(0);)

--------  3003 --------------

[复制代码](javascript:void(0);)

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/root/zkcluster/3003/data

dataLogDir=/root/zkcluster/3003/logs

# the port at which the clients will connect

clientPort=3003

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=6

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

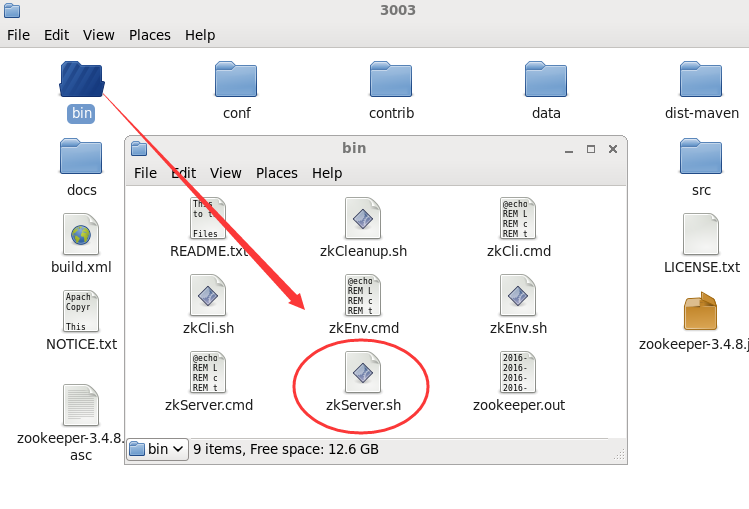
server.1=192.168.161.134:2888:3888

server.2=192.168.161.134:2889:3889

server.3=192.168.161.134:2890:3890

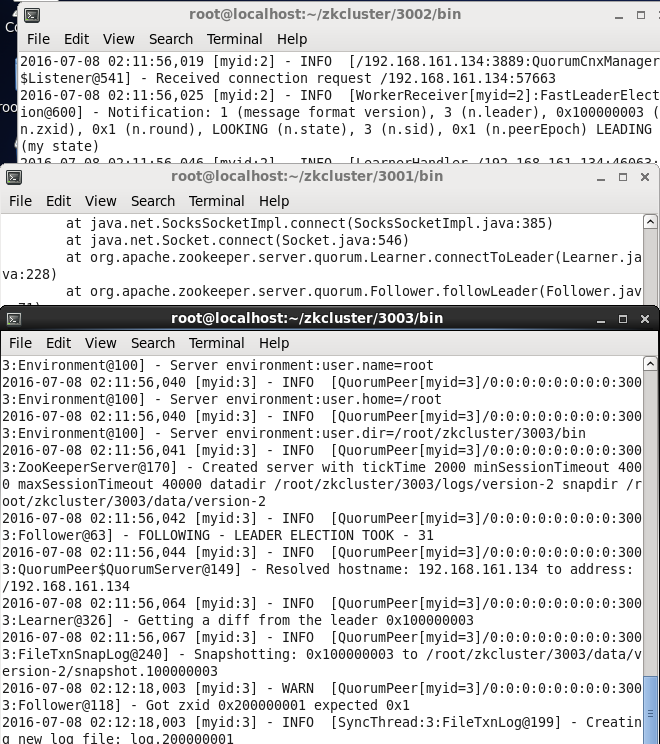
[复制代码](javascript:void(0);)

5. 启动各自服务器

    到现在为止，我们各个zookeeper程序的配置都结束了，接下来我们到各自目录的bin目录下，通过zkServer.sh来进行启动，比如下面这样：

ok，接下来我们来开始启动，通过如下命令即可：

./zkServer.sh start-foreground



现在我们都启动了，接下来我们可以用命令看下哪个server是leader，哪些是follower。。。

[复制代码](javascript:void(0);)

[root@localhost bin]# ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /root/zkcluster/3001/bin/../conf/zoo.cfg

Mode: follower

[root@localhost bin]#

[root@localhost bin]# ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /root/zkcluster/3002/bin/../conf/zoo.cfg

Mode: leader

[root@localhost bin]#

[root@localhost bin]# ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /root/zkcluster/3003/bin/../conf/zoo.cfg

Mode: follower

[root@localhost bin]#

[复制代码](javascript:void(0);)

到目前为止，我们的服务端操作都ok啦，，，是不是好吊。。。

测试代码

**package** com.enn.zkCluster;

**import** java.io.IOException;

**import** org.apache.zookeeper.CreateMode;

**import** org.apache.zookeeper.KeeperException;

**import** org.apache.zookeeper.WatchedEvent;

**import** org.apache.zookeeper.Watcher;

**import** org.apache.zookeeper.ZooKeeper;

**import** org.apache.zookeeper.ZooDefs.Ids;

/\*\*

\* zkCluster 测试

\*

\* **@author** houjianpo

\*

\*/

**public** **class** ZKDemo {

// 会话超时时间， 设置为与系统默认时间一致

**private** **static** **final** **int** ***SESSION\_TIMEOUT*** = 30 \* 1000;

// 创建zookeeper实例

**private** ZooKeeper zk;

// 创建Watcher实例

**private** Watcher wh = **new** Watcher() {

/\*\*

\* Watched事件

\*/

**public** **void** **process**(WatchedEvent event) {

System.***out***.println("WatchedEvent >>> " + event.toString());

}

};

// 初始化zookeeper实例

@SuppressWarnings("static-access")

**private** **void** **createZKInstance**() **throws** Exception {

// 连接到zk服务，多个服务可以用,分割写

zk = **new** ZooKeeper("10.4.82.161:3000,10.4.82.161:3001,10.4.82.161:3002", **this**.***SESSION\_TIMEOUT***, **this**.wh);

}

// zookeeper 操作

**private** **void** **ZKOperations**() **throws** IOException, InterruptedException, KeeperException {

System.***out***.println("\n1. 创建Zookeeper 节点(znode : zoo2, 数据: myData2, 权限: OPEN\_ACL\_UNSAFE, 节点类型: Persistent.)");

zk.create("/zoo2", "myData2".getBytes(), Ids.***OPEN\_ACL\_UNSAFE***, CreateMode.***PERSISTENT***);

System.***out***.println("\n2. 查看是否创建成功:");

System.***out***.println(**new** String(zk.getData("/zoo2", **this**.wh, **null**)));// 添加Watcher,设置下次修改监控

// 前面一行我们添加了对/zoo2节点的监视，所以这里对/zoo2进行修改的时候，会触发Watch时间

System.***out***.println("\n3. 修改/zoo2节点数据");

zk.setData("/zoo2", "helloword".getBytes(), -1);

// 这里再次进行修改，则不会触发Watch事件，这就是我们验证ZK的一个特性“一次触发”，也就是说设置一次监控，只会对下次操作起一次作用。

System.***out***.println("\n3-1. 再次修改/zoo2节点数据");

zk.setData("/zoo2", "helloword-ABCD".getBytes(), -1);

// 查看

System.***out***.println("\n4. 查看是否修改成功:");

System.***out***.println(**new** String(zk.getData("/zoo2", **false**, **null**)));// 本次不再设置Watch监控

// 删除节点

System.***out***.println("\n5. 删除节点");

zk.delete("/zoo2", -1);

System.***out***.println("\n6. 查看节点是否被删除:");

System.***out***.println("节点状态: [" + zk.exists("/zoo2", **false**) + " ]" );

}

// 关闭zookeeper连接

@SuppressWarnings("unused")

**private** **void** **ZKClose**() **throws** InterruptedException {

**if**(zk.getState() == ZooKeeper.States.***CONNECTED***){

zk.close();

}

}

**public** **static** **void** **main**(String[] args) **throws** Exception {

ZKDemo **zkDemo** = **new** ZKDemo();

zkDemo.createZKInstance();

zkDemo.ZKOperations();

zkDemo.ZKClose();

}

}