参考：<http://www.cnblogs.com/linuxbug/p/5023677.html>

Zookeeper的ACL机制和Quota机制网上资料较少，这里做一个总结，以供大家参考。

**1 Zookeeper ACL**

ZooKeeper的权限管理亦即ACL控制功能通过Server、Client两端协调完成：

Server端：

一个ZooKeeper的节点（znode）存储两部分内容：数据和状态，状态中包含ACL信息。创建一个znode会产生一个ACL列表，列表中每个ACL包括：

  验证模式(scheme)

  具体内容(Id)（当scheme=“digest”时，Id为用户名密码，例如“root：J0sTy9BCUKubtK1y8pkbL7qoxSw=”）

  权限(perms)

**1.1 scheme**

ZooKeeper提供了如下几种验证模式（scheme）：

  digest：Client端由用户名和密码验证，譬如user:password，digest的密码生成方式是Sha1摘要的base64形式

  auth：不使用任何id，代表任何已确认用户。

  ip：Client端由IP地址验证，譬如172.2.0.0/24

  world：固定用户为anyone，为所有Client端开放权限

  super：在这种scheme情况下，对应的id拥有超级权限，可以做任何事情(cdrwa）

注意的是，exists操作和getAcl操作并不受ACL许可控制，因此任何客户端可以查询节点的状态和节点的ACL。

节点的权限（perms）主要有以下几种：

  Create 允许对子节点Create操作

  Read 允许对本节点GetChildren和GetData操作

  Write 允许对本节点SetData操作

  Delete 允许对子节点Delete操作

  Admin 允许对本节点setAcl操作

Znode ACL权限用一个int型数字perms表示，perms的5个二进制位分别表示setacl、delete、create、write、read。比如0x1f=adcwr，0x1=----r，0x15=a-c-r。

**1.1.1 world scheme固定id为anyone，表示对所有Client端开放权限：**

[zk: localhost:2181(CONNECTED) 13] create /123 "123"

Created /123

[zk: localhost:2181(CONNECTED) 14] getAcl /123

'world,'anyone

: cdrwa

**1.1.2 ip scheme设置可以访问的ip地址（比如127.0.0.1）或ip地址段（比如192.168.1.0/16）**

10.194.157.58这台机器上创建/test并设置ip访问权限

[zk: 10.194.157.58:2181(CONNECTED) 0] create /test "123"

Created /test

[zk: 10.194.157.58:2181(CONNECTED) 1] setAcl /test ip:10.194.157.58:crwda

cZxid = 0x740021e467

ctime = Wed Dec 02 18:09:09 CST 2015

mZxid = 0x740021e467

mtime = Wed Dec 02 18:09:09 CST 2015

pZxid = 0x740021e467

cversion = 0

dataVersion = 0

aclVersion = 1

ephemeralOwner = 0x0

dataLength = 5

numChildren = 0

[zk: 10.194.157.58:2181(CONNECTED) 2] ls /test

[]

可以看到，本机是可以访问的。

在10.205.148.152上登陆

[zk: 10.194.157.58:2181(CONNECTED) 1] ls /test

Authentication is not valid : /test

可以看到，连接的ip不在授权中，提示访问错误。

**1.1.3 digest scheme的id表示为username:BASE64(SHA1(password))**

[root@rocket zookeeper-server1]# cd /usr/local/zookeeper-server1/

[root@rocket zookeeper-server1]# pwd

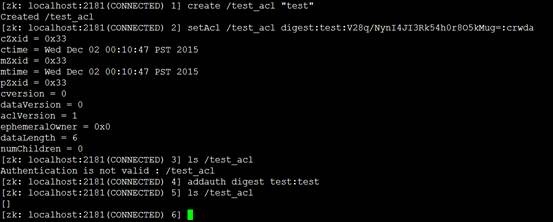
/usr/local/zookeeper-server1

# 生成密文

[root@rocket zookeeper-server1]# java -cp ./zookeeper-3.4.6.jar:./lib/log4j-1.2.16.jar:./lib/slf4j-log4j12-1.6.1.jar:./lib/slf4j-api-1.6.1.jar org.apache.zookeeper.server.auth.DigestAuthenticationProvider test:test

test:test->test:V28q/NynI4JI3Rk54h0r8O5kMug=

创建acl



通过认证后，可以访问数据：

[zk: localhost:2181(CONNECTED) 0]

[zk: localhost:2181(CONNECTED) 0] ls /test\_acl

Authentication is not valid : /test\_acl

[zk: localhost:2181(CONNECTED) 1] getAcl /test\_acl

'digest,'test:V28q/NynI4JI3Rk54h0r8O5kMug=

: cdrwa

[zk: localhost:2181(CONNECTED) 2] addauth digest test:test

[zk: localhost:2181(CONNECTED) 3] ls /test\_acl

[]

[zk: localhost:2181(CONNECTED) 4] get /test\_acl

"test"

cZxid = 0x33

ctime = Wed Dec 02 00:10:47 PST 2015

mZxid = 0x33

mtime = Wed Dec 02 00:10:47 PST 2015

pZxid = 0x33

cversion = 0

dataVersion = 0

aclVersion = 1

ephemeralOwner = 0x0

dataLength = 6

numChildren = 0

**1.2 SuperDigest超级管理员**

当设置了znode权限，但是密码忘记了怎么办？还好Zookeeper提供了超级管理员机制。

一次Client对znode进行操作的验证ACL的方式为：

a) 遍历znode的所有ACL：

i. 对于每一个ACL，首先操作类型与权限（perms）匹配

ii. 只有匹配权限成功才进行session的auth信息与ACL的用户名、密码匹配

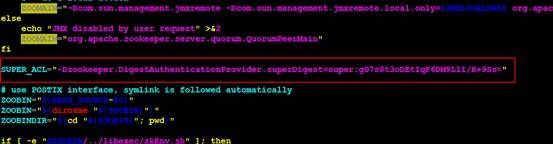
b) 如果两次匹配都成功，则允许操作；否则，返回权限不够error（rc=-102）

备注：如果znode ACL List中任何一个ACL都没有setAcl权限，那么就算superDigest也修改不了它的权限；再假如这个znode还不开放delete权限，那么它的所有子节点都将不会被删除。唯一的办法是通过手动删除snapshot和log的方法，将ZK回滚到一个以前的状态，然后重启，当然这会影响到该znode以外其它节点的正常应用。

superDigest设置的步骤

修改zkServer.sh，加入super权限设置

-Dzookeeper.DigestAuthenticationProvider.superDigest=super:gG7s8t3oDEtIqF6DM9LlI/R+9Ss=





重新启动Zookeeper

# ./zkServer.sh restart

这时候

不使用test:test进行认证，而是使用super:super进行认证：

[zk: localhost:2181(CONNECTED) 0] ls /test\_acl

Authentication is not valid : /test\_acl

[zk: localhost:2181(CONNECTED) 1] addauth digest super:super

[zk: localhost:2181(CONNECTED) 2] ls /test\_acl

[]

[zk: localhost:2181(CONNECTED) 3] get /test\_acl

"test"

cZxid = 0x33

ctime = Wed Dec 02 00:10:47 PST 2015

mZxid = 0x33

mtime = Wed Dec 02 00:10:47 PST 2015

pZxid = 0x33

cversion = 0

dataVersion = 0

aclVersion = 1

ephemeralOwner = 0x0

dataLength = 6

numChildren = 0

**1.3 ACL机制的缺陷**

然而，ACL毕竟仅仅是访问控制，并非完善的权限管理，通过这种方式做多集群隔离，还有很多局限性：

ACL并无递归机制，任何一个znode创建后，都需要单独设置ACL，无法继承父节点的ACL设置。

除了ip这种scheme，digest和auth的使用对用户都不是透明的，这也给使用带来了很大的成本，很多依赖zookeeper的开源框架也没有加入对ACL的支持，例如hbase，storm。

**2 Zookeeper quota**

ZooKeeper quota机制支持节点个数（znode）和空间大小（字节数）。

[zk: localhost:2181(CONNECTED) 2] create /test\_quota "12345"

Created /test\_quota

[zk: localhost:2181(CONNECTED) 3] listquota /test\_quota

absolute path is /zookeeper/quota/test\_quota/zookeeper\_limits

quota for /test\_quota does not exist.

# 这里看到quota还没有设置

[zk: localhost:2181(CONNECTED) 4] setquota -n 5 /test\_quota

Comment: the parts are option -n val 5 path /test\_quota

# -n表示设置znode count限制，这里表示/test\_quota这个path下的znode count个数限制为5（包括/test\_quota本身）

# -b 表示设置znode数据的字节大小限制，这里不做演示了，有兴趣的同学下去自己实验

[zk: localhost:2181(CONNECTED) 5] listquota /test\_quota

absolute path is /zookeeper/quota/test\_quota/zookeeper\_limits

Output quota for /test\_quota count=5,bytes=-1   # 限制znode count为5

Output stat for /test\_quota count=1,bytes=7     # 目前znode count为1

[zk: localhost:2181(CONNECTED) 3] create /test\_quota/0 "0"

Created /test\_quota/0

[zk: localhost:2181(CONNECTED) 6] create /test\_quota/1 "1"

Created /test\_quota/1

[zk: localhost:2181(CONNECTED) 7] create /test\_quota/2 "2"

Created /test\_quota/2

[zk: localhost:2181(CONNECTED) 8] create /test\_quota/3 "3"

Created /test\_quota/3

[zk: localhost:2181(CONNECTED) 9] create /test\_quota/4 "4"

Created /test\_quota/4

# 上面新建了多个znode

看zookeeper的日志，发现有Quota exceeded的日志，这里要说明一下zookeeper的Quota机制是比较温和的，即使超限了，只是在日志中报告一下，并不会限制Client的行为，Client可以继续操作znode。

在实际项目中，Client可以查看/zookeeper/quota目录下的数据来确定是否超出quota限制，由此来做一些告警。

[zk: localhost:2181(CONNECTED) 4] get /zookeeper/quota/test\_quota/zookeeper\_limits

count=5,bytes=-1

[zk: localhost:2181(CONNECTED) 5] get /zookeeper/quota/test\_quota/zookeeper\_stats

count=7,bytes=25