zookeeper概述

Zookeeper概述

官方网址：http://zookeeper.apache.org/

What is ZooKeeper?

ZooKeeper is a **centralized service** for maintaining **configuration information**, **naming**, providing **distributed synchronization**, and providing **group services**. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them ,which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

Zookeeper是一个分布式的协调服务框架，Zookeeper可以解决分布式环境常见的问题：集群管理、统一命名服务，信息配置管理，分布式锁等等。

Zookeeper要解决的问题

zookeeper旨在在分布式应用中，提供可靠的、可扩展的、分布式的、可配置的协调机制来管理整个集群的状态。

Zookeeper

动物园管理员

Zookeeper的重要性

1.Hadoop
2.Hbase
3.Storm
4.Kafka

Zookeeper应用场景（整理）

2018年8月21日 16:48

zookeeper单机模式安装配置

安装步骤：

0.关闭虚拟机的防火墙 ,执行：service iptables stop

1.准备虚拟机，安装并配置jdk，1.6以上

配置示例：

JAVA_HOME=/home/software/jdk1.8

CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

PATH=$JAVA_HOME/bin:$PATH

export JAVA_HOME PATH CLASSPATH

2.上传zookeeper的安装包 3.4.7版本

3.解压安装 tar -xvf ……

4.进入zookeeper安装目录下的conf目录，有一个zoosample.cfg的文件

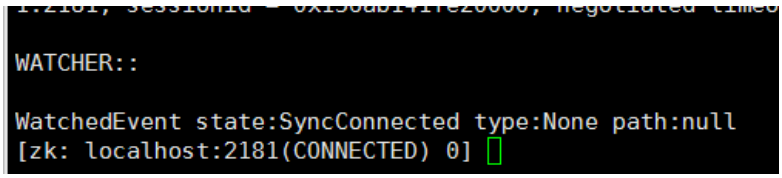
复制一份，并重命名为zoo.cfg文件，这个名字固定写死，因为zookeeper启动会检查这个文件，根据这个配置文件里的信息来启动服务

5.进入bin目录

执行：./zkServer.sh start 启动zookeeper

6.进入zookeeper客户端，操作zookeeper

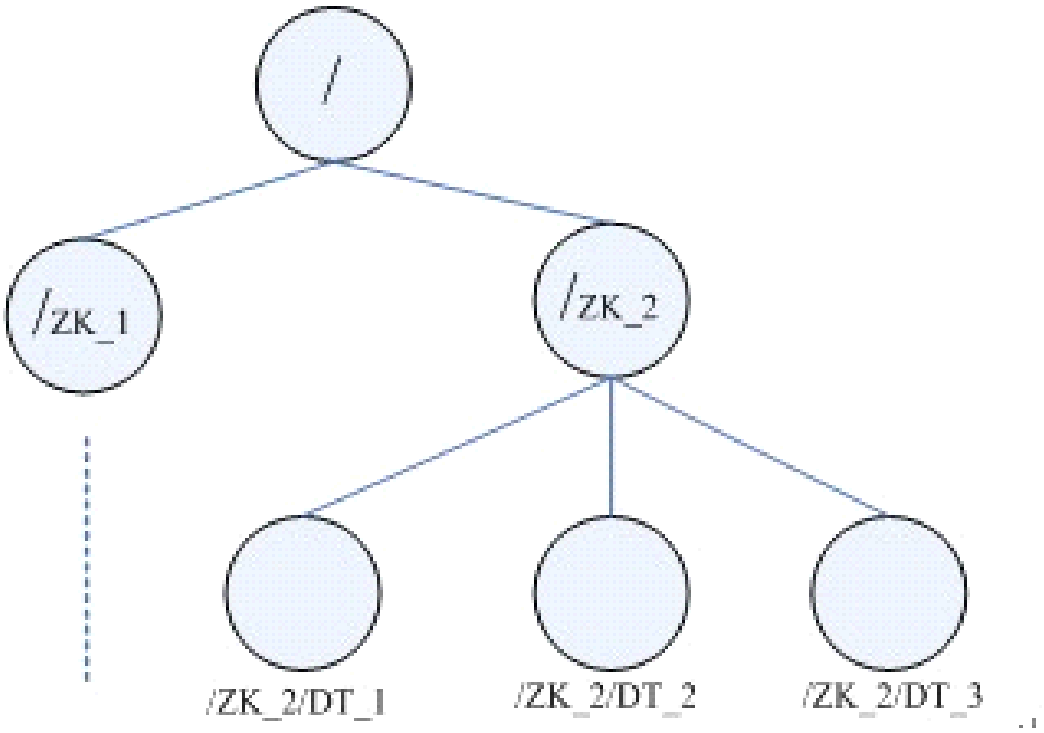
执行：./zkCli.sh



zk基础概念

2018年1月17日 10:51

Zookeeper结构



知识点:

1.zk有一个根节点 /

2.每一个节点都可以有自己的子节点

3.每一个节点称为znode节点

4.多个znode节点最终所形成的称为znode树

5.每一个znode节点都可以存储数据

6.zk为了提供快速的数据访问，会将数据维系在内存中

7.zk为了防止数据丢失，也会在磁盘上进行落地存储，存储的路径可以通过配置文件来指定

8.zk的所有操作指令，比如 查看，创建，更新，删除都是基于路径来操作的。

比如 ：ls /zookeeper

9.zk中的路径是具有全局唯一性的，所有可以基于这个特性来实现命名服务

Zookeeper服务端指令

指令	说明
sh zkServer.sh start	启动zk服务的
sh zkServer.sh stop	停止zk服务
sh zkServer.sh restart	重启zk服务
sh zkServer.sh status	查看zk服务角色，有： Standalone Leader Follower Observer
sh zkCli.sh	进入zk客户端

Zookeeper客户端指令

指令	说明	示例
ls	查看	ls / 查看根路径 ls /park01 查看park01路径
create	创建	create /park02 "" create /park02/node01 创建park02的子节点

get	获取指定节点信息	cZxid = 0x2 #创建此节点的事务id
		ctime = Wed Jan 17 10:55:25 PST 2018 #创建此节点的时间戳
		mZxid = 0x2 #修改此节点的事务id
		mtime = Wed Jan 17 10:55:25 PST 2018 #修改此节点的时间戳
		pZxid = 0x4
		cversion = 1
		dataVersion = 0

		aclVersion = 0
		ephemeralOwner = 0x0 #如果此节点不是临时节点，则为0
		dataLength = 9 #数据长度
		numChildren = 1 #子节点数量

set	更新节点数据	set /park01 hellozk
delete	删除子节点为空的节点	delete /park01
rmr	递归删除指定节点	rmr /park02
quit （或ctrl+c）	退出客户端	
create -e	创建**临时**节点，当创建此节点的客户端下线时，节点被删除。	
create -s	创建**顺序**节点，每次创建节点时，会跟上一个**递增**的顺序号	
create -e -s	创建**临时顺序**节点	

zookeeperAPI操作

2016年8月20日 21:19

```
/**
 * 知识点1:Zookeeper的连接是一个非阻塞连接方法
 * @author ysq
 *
 */
public class TestDemo {

	@Test
	public void testConnect() throws Exception{
		//①connectString:指定zk服务器的ip地址:端口号
		//如果是云服务器，写对应的外网ip地址,此外，要开方对应的端口
		//关闭防火墙执行令：service iptables stop
		//②sessionTimeout:客户端连接服务的超时时间
		//③监听器,可以指定事件的发生

		final CountDownLatch cdl=new CountDownLatch(1);
		ZooKeeper zk=
		new ZooKeeper("192.168.234.231:2181",30000,new Watcher(){

			/*
			 * KeeperState.SyncConnected表示连接成功事件
			 */
			@Override
			public void process(WatchedEvent event) {
				if(event.getState().equals(KeeperState.SyncConnected)){
					System.out.println("连接成功");
					cdl.countDown();
				}

			}

		});
		cdl.await();

	}
	@Test
	public void create_set_get_delete() throws Exception{
		final CountDownLatch cdl=new CountDownLatch(1);
		ZooKeeper zk=new ZooKeeper("192.168.234.231:2181",30000,new Watcher(){

			@Override
			public void process(WatchedEvent event) {
				if(event.getState().equals(KeeperState.SyncConnected)){
					System.out.println("连接成功");
					cdl.countDown();
				}

			}
```

```
        });
        cdl.await();
        //--创建节点：①节点路径 ②节点数据，字节数组 ③节点的权限 ④节点类型
        zk.create("/park06","hello1708".getBytes(),
                        Ids.OPEN_ACL_UNSAFE, CreateMode.EPHEMERAL);

        //--更新节点：①节点路径 ②更新的数据 ③节点的数据版本号
        //每次更新节点数据，dataversion都会递增1，此外，如果版本号不一致，会报错
        //如果想做到无论如何都更新，版本号写成-1
        zk.setData("/park01","hello123".getBytes(),-1);

        //--获取节点数据：①节点路径 ②监听器 ③节点状态
        byte[] data=zk.getData("/park01",null,null);
        System.out.println(new String(data));

        //--删除节点：①节点路径 ②版本号，-1表示无视版本号删除
        zk.delete("/park01",-1);

        while(true);
}
/*
 * getChildren获取的是子节点的名字，不是路径，
 * 所以在很多场景下，需要自己拼成一个完整路径
 */
@Test
public void getChild() throws Exception{
        final CountDownLatch cdl=new CountDownLatch(1);
        ZooKeeper zk=new ZooKeeper("192.168.234.231:2181",30000,new Watcher(){


                @Override
                public void process(WatchedEvent event) {
                        if(event.getState().equals(KeeperState.SyncConnected)){
                                System.out.println("连接成功");
                                cdl.countDown();
                        }

                }

        });

        cdl.await();
        List<String> paths=zk.getChildren("/park01",null);
        for(String path:paths){
                System.out.println(path);
        }
}
@Test
public void watchDataChange() throws Exception{
        final CountDownLatch cdl=new CountDownLatch(1);
        ZooKeeper zk=new
        ZooKeeper("192.168.234.11:2181,192.168.234.210:2181,192.168.234.211:2181",30000,
```

```
        new Watcher(){

            @Override
            public void process(WatchedEvent event) {
                if(event.getState().equals(KeeperState.SyncConnected)){
                    System.out.println("连接成功");
                    cdl.countDown();
                }

            }

        });
        cdl.await();

        //--监听指定节点数据发生变化的事件,默认的监听机制是一次性监听
        for(;;){
            final CountDownLatch lock=new CountDownLatch(1);
            zk.getData("/park01",new Watcher(){

                @Override
                public void process(WatchedEvent event) {
                    if(event.getType().equals(EventType.NodeDataChanged)){
                        System.out.println("有数据发生变化");
                        lock.countDown();
                    }

                }

            },null);
            lock.await();
        }



}

@Test
public void watchChildChange() throws Exception{
        final CountDownLatch cdl=new CountDownLatch(1);
        ZooKeeper zk=new
        ZooKeeper("192.168.234.11:2181,192.168.234.210:2181,192.168.234.211:2181",30000,
        new Watcher(){

            @Override
            public void process(WatchedEvent event) {
                if(event.getState().equals(KeeperState.SyncConnected)){
                    System.out.println("连接成功");
                    cdl.countDown();
                }

            }

        });
        cdl.await();
```

```
        //--监听指定节点子节点发生变化的事件
        zk.getChildren("/park01",new Watcher(){

            @Override
            public void process(WatchedEvent event) {
                if(event.getType().equals(EventType.NodeChildrenChanged)){
                    System.out.println("有子节点发生变化");
                }

            }

        });
        while(true);
    }
}
```

zookeeper集群安装配置

安装步骤：

提示：要关闭虚拟机的防火墙，执行：service iptables stop

1.准备虚拟机，安装并配置jdk，1.6以上

2.上传zookeeper的安装包 3.4.7版本

3.解压安装 tar -xvf ……

4.配置zookeeper。

5.配置集群模式

①切换到zookeeper安装目录的conf目录，其中有一个zoo_sample.cfg的配置文件，这个一个配置模板文件，我们需要复制这个文件，并重命名为 zoo.cfg。zoo.cfg才是真正的配置文件

```
configuration.xsl  log4j.properties  zoo_sample.cfg
[root@localhost conf]#  mv zoo_sample.cfg zoo.cfg
```

②配置zoo.cfg=》vim zoo.cfg 更改如下几个参数配置：

dataDir。这个参数是存放zookeeper集群环境配置信息的。这个参数默然是配置在/tmp/zookeeper下的。但是注意，tmp是一个临时文件夹，这个是linux自带的一个目录，是linux本身用于存放临时文件用的目录。但是这个目录极有可能被清空,所以，重要的文件一定不要存在这个目录下。

所以改成：/home/work/zkdata

注意：这个路径是自定义的，所以目录需要手动创建

```
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/home/work/zkdata
```

clientport。客户端连接服务器的端口，默认是2181，一般不用修改

在配置文件里，需要在加上如下的配置：

server.1=192.168.234.10:2888:3888

server.2=192.168.234.11:2888:3888

server.3=192.168.234.12:2888:3888

①server是关键字，写死

②后面的数字是选举id，在zk集群的选举过程中会用到。

补充：此数字不固定，但是需要注意选举id不能重复，相互之间要能比较大小

然后保存退出

③192.168.234.10:2888:3888

说明：2888原子广播端口，3888选举端口

zookeeper有几个节点，就配置几个server,

```
1 zebra_1
# 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/home/word/zkdata
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
server.1=192.168.234.10:2888:3888
server.2=192.168.234.11:2888:3888
server.3=192.168.234.12:2888:3888
-- INSERT --
```

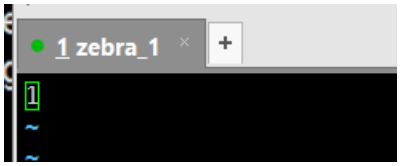
③配置文件配置好，需要在dataDir目录下创建一个文件

即在：/home/work/zkdata 目录下，创建 myid

vim myid

给当前的节点编号。zookeeper节点在启动时，就会到这个目录下去找myid文件，得知自己的

编号



保存退出

6.配置集群环境的其他节点

scp -r 目录 远程ip地址：存放的路径

scp -r /home/software/zookeeper 192.168.234.151: /home/

①更改节点的ip

②更改myid的id号

③关闭防火墙 ，执行：service iptables stop;

7.启动zookeeper

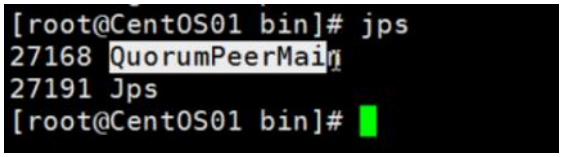
进入到zookeeper安装目录的bin目录

执行：./zkServer.sh start

```
[root@CentOS01 bin]# ./zkServer.sh
ZooKeeper JMX enabled by default
Using config: /root/work/zookeeper-3.4.7/bin/../conf/zoo.cfg
Usage: ./zkServer.sh {start|start-foreground|stop|restart|status|upgrade|pri
nt-cmd}
[root@CentOS01 bin]#
```

然后可以输入jps命令，查看有哪些java进程，

执行：jps



8.其他两台节点启动zookeeper服务

9.执行 ：./zkServer.sh status 查看当前zookeeper节点状态

📖 配置说明：

tickTime: zookeeper中使用的基本时间单位, 毫秒值.

dataDir: 数据目录. 可以是任意目录.

dataLogDir: log目录, 同样可以是任意目录. 如果没有设置该参数, 将使用和dataDir相同的设置.

clientPort: 监听client连接的端口号

initLimit: zookeeper集群中的包含多台server, 其中一台为leader, 集群中其余的server为follower. initLimit参数配置初始化连接时, follower和leader之间的最长心跳时间. 此时该参数设置为5, 说明时间限制为5倍tickTime, 即5*2000=10000ms=10s.

syncLimit: 该参数配置leader和follower之间发送消息, 请求和应答的最大时间长度. 此时该参数设置为2, 说明时间限制为2倍tickTime, 即4000ms.

Zookeeper选举机制（整理）

2018年8月21日 16:48