# 第二单元 Zookeeper入门

# 【授课重点】

1. Zookeeper简介
2. Zookeeper安装配置

# 【考核要求】

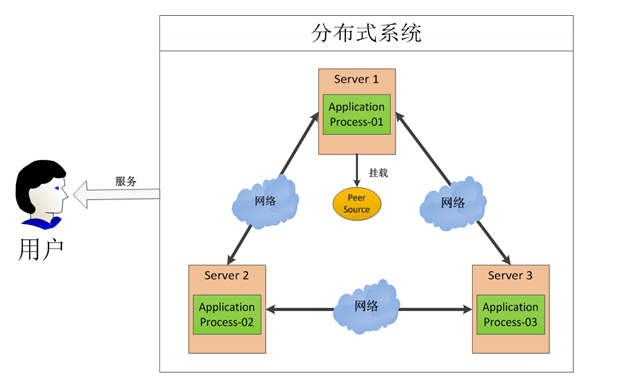
1. 了解Zookeeper
2. 掌握Zookeeper安装配置

# 【教学内容】

## 2.1 课程导入

1. 集群当中 N 多的配置信息如何做到全局一致并且单点修改迅速响应到整个集群？ --- 配置管理
2. 集群中的 namonode 和 resourcemanager 的单点故障怎么解决？ --- 集群的主节点的单点故障

### 2.1.1 分布式协调技术

在给大家介绍ZooKeeper之前先来给大家介绍一种技术——分布式协调技术。那么什么是分布式协调技术？那么我来告诉大家，其实分布式协调技术 主要用来解决分布式环境当中多个进程之间的同步控制，让他们有序的去访问某种临界资源，防止造成"脏数据"的后果。这时，有人可能会说这个简单，写一个调 度算法就轻松解决了。说这句话的人，可能对分布式系统不是很了解，所以才会出现这种误解。如果这些进程全部是跑在一台机上的话，相对来说确实就好办了，问 题就在于他是在一个分布式的环境下，这时问题又来了，那什么是分布式呢？这个一两句话我也说不清楚，但我给大家画了一张图希望能帮助大家理解这方面的内 容，如果觉得不对尽可拍砖，来咱们看一下这张图，如图1.1所示。 

### 2.2.2 分布式锁的实现

#### 面临的问题

在看了图1.1所示的分布式环境之后，有人可能会感觉这不是很难。无非是将原来在同一台机器上对进程调度的原语，通过网络实现在分布式环境中。是的，表面上是可以这么说。但是问题就在网络这，在分布式系统中，所有在同一台机器上的假设都不存在：因为网络是不可靠的。

比如，在同一台机器上，你对一个服务的调用如果成功，那就是成功，如果调用失败，比如抛出异常那就是调用失败。但是在分布式环境中，由于网络的不可靠，你对一个服务的调用失败了并不表示一定是失败的，可能是执行成功了，但是响应返回的时候失败了。还有，A和B都去调用C服务，在时间上 A还先调用一些，B后调用，那么最后的结果是不是一定A的请求就先于B到达呢？ 这些在同一台机器上的种种假设，我们都要重新思考，我们还要思考这些问题给我们的设计和编码带来了哪些影响。还有，在分布式环境中为了提升可靠性，我们往往会部署多套服务，但是如何在多套服务中达到一致性，这在同一台机器上多个进程之间的同步相对来说比较容易办到，但在分布式环境中确实一个大难题。

所以分布式协调远比在同一台机器上对多个进程的调度要难得多，而且如果为每一个分布式应用都开发一个独立的协调程序。一方面，协调程序的反复编写浪费，且难以形成通用、伸缩性好的协调器。另一方面，协调程序开销比较大，会影响系统原有的性能。所以，急需一种高可靠、高可用的通用协调机制来用以协调分布式应用。

#### 分布式锁的实现者

目前，在分布式协调技术方面做得比较好的就是Google的Chubby还有Apache的ZooKeeper他们都是分布式锁的实现者。有人会问既然有了Chubby为什么还要弄一个ZooKeeper，难道Chubby做得不够好吗？不是这样的，主要是Chbby是非开源的，Google自家用。后来雅虎模仿Chubby开发出了ZooKeeper，也实现了类似的分布式锁的功能，并且将ZooKeeper作为一种开源的程序捐献给了Apache，那么这样就可以使用ZooKeeper所提供锁服务。而且在分布式领域久经考验，它的可靠性，可用性都是经过理论和实践的验证的。所以我们在构建一些分布式系统的时候，就可以以这类系统为起点来构建我们的系统，这将节省不少成本，而且bug也 将更少。



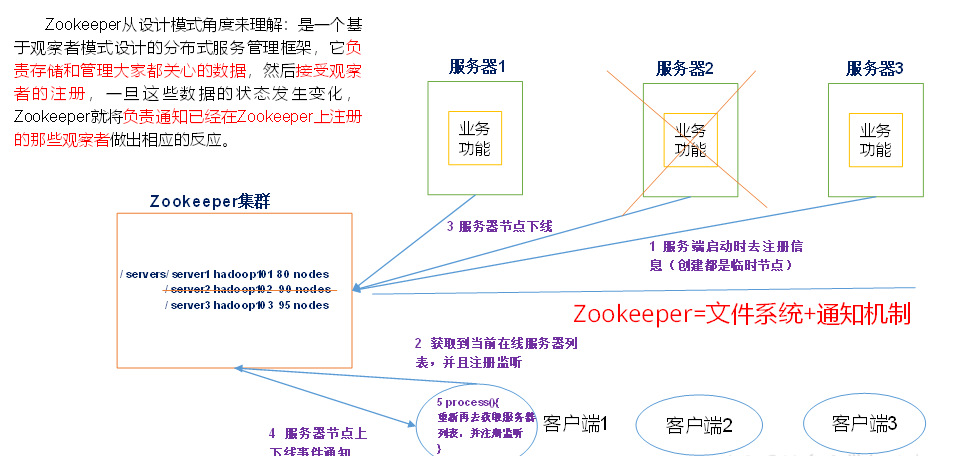
## 2.2 zookeepr简介

Zookeeper是一个开源的**分布式**的，为分布式应用提供**协调**服务的Apache项目。

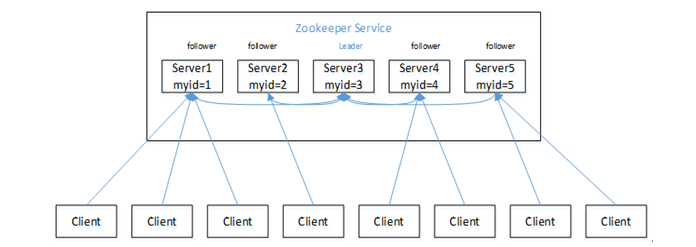
提供的功能包括：

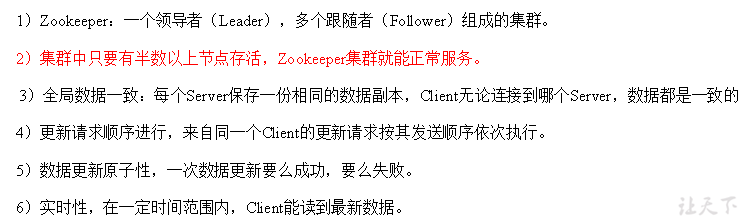
1. 配置维护
2. 域名服务
3. 分布式同步
4. 组服务。

### zookeeper工作机制

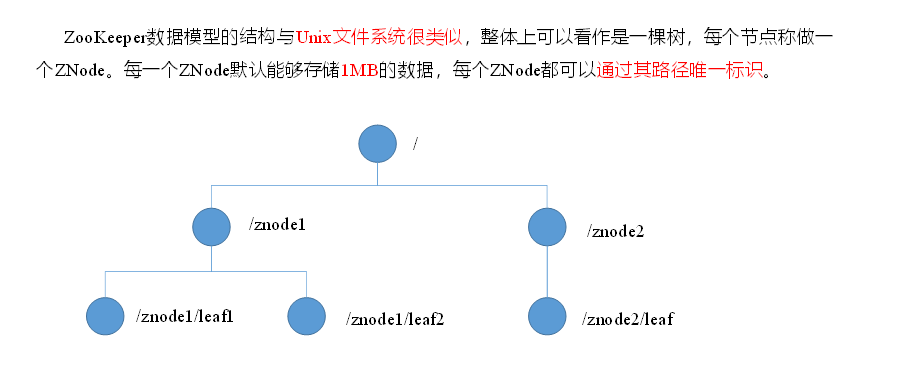


### 2.2.1 特点





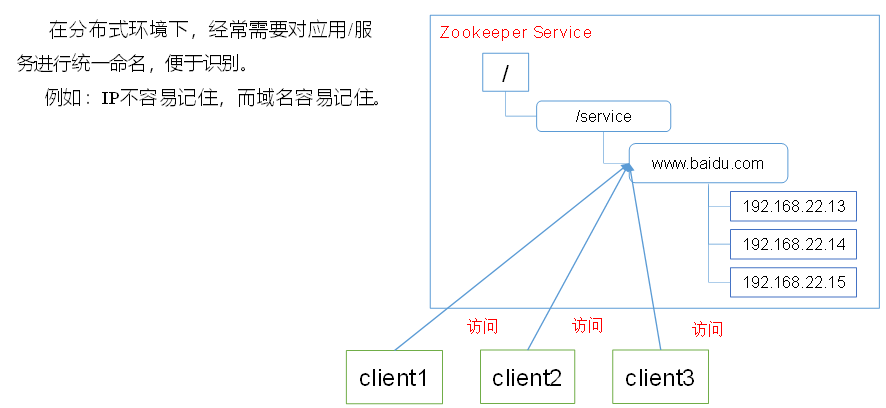
### 2.2.2 **数据结构**



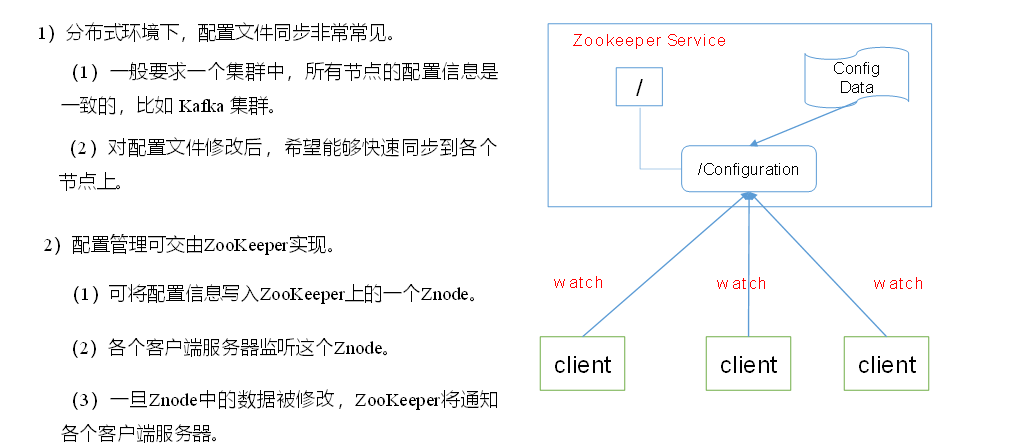
### 2.2.4 **应用场景**

提供的服务包括：统一命名服务、统一配置管理、统一集群管理、服务器节点动态上下线、软负载均衡等。

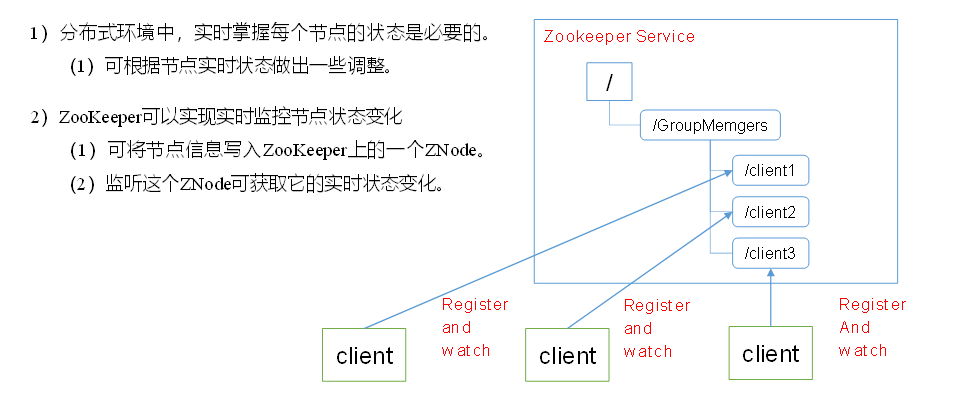
#### 统一命名服务



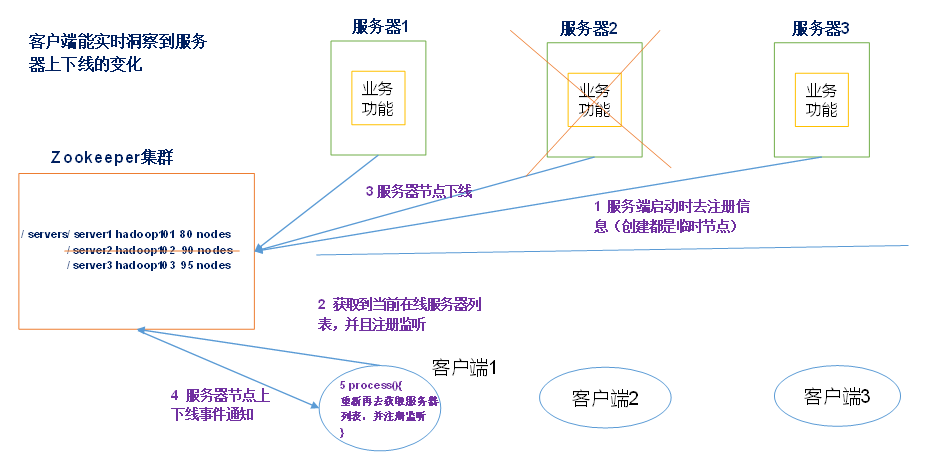
#### 统一配置管理



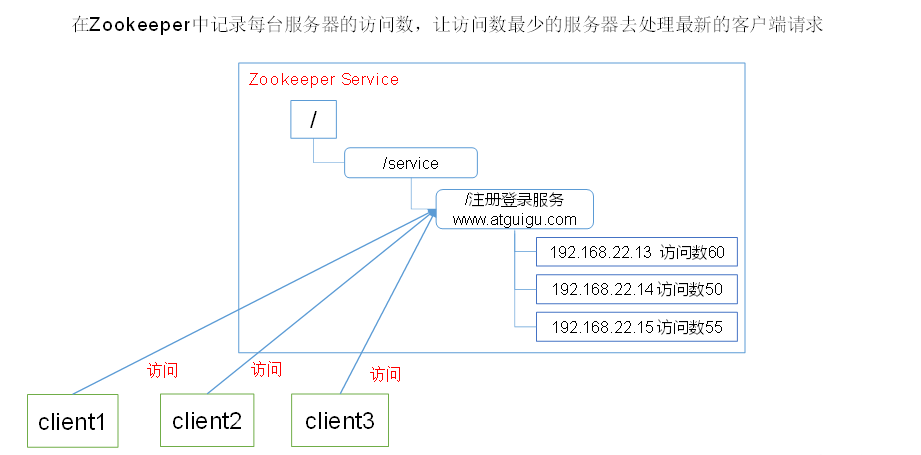
#### 统一集群管理



#### 服务器动态上下线



#### 负载均衡



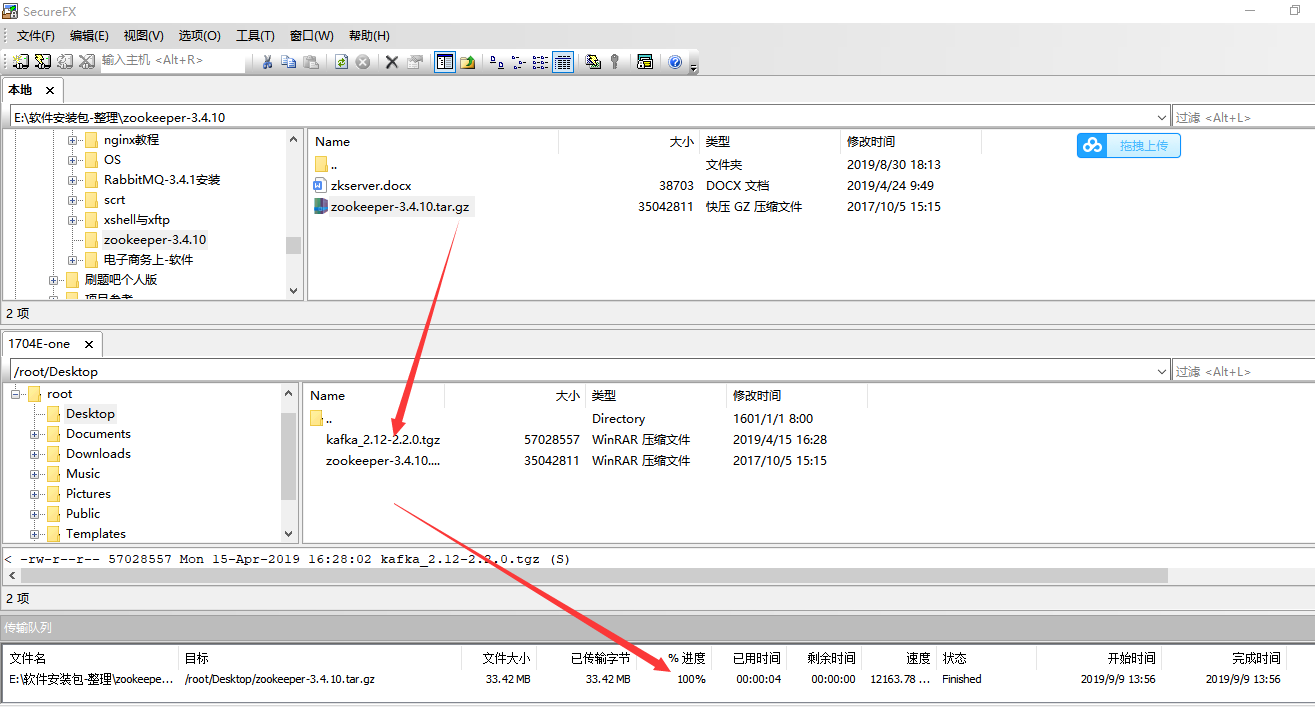
## 2.3 zookeepr安装配置

### 2.3.1 学生独立安装

第一步：安装 jdk(参考小二课程)

第二步：把 zookeeper安装包上传到 linux 系统

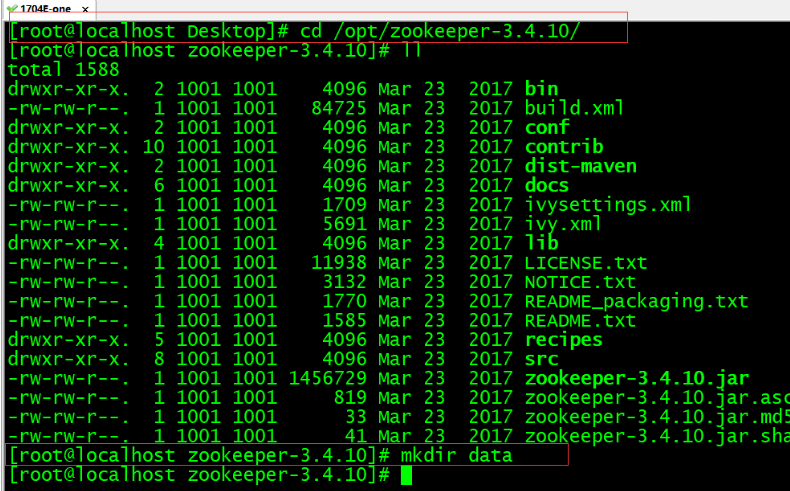
首先在官网 https://zookeeper.apache.org/ 下载zookeeper安装的包,然后上传到指定目录



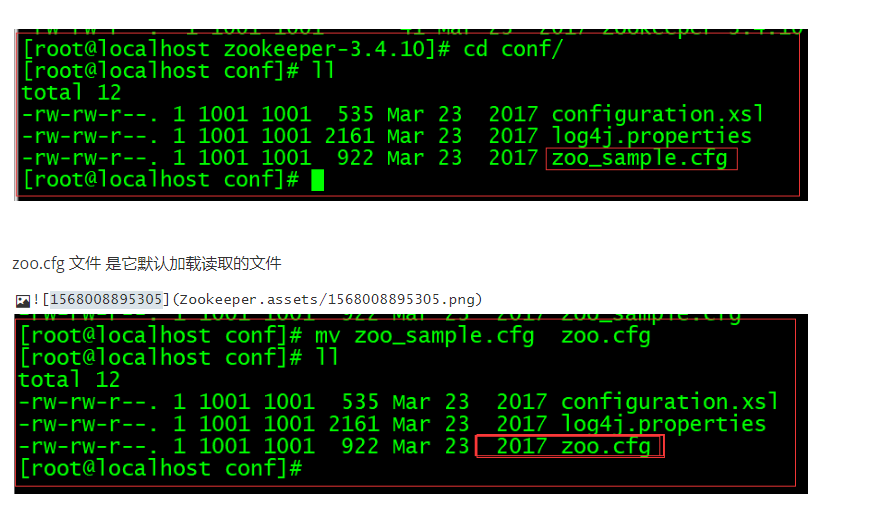
第三步：解压缩压缩包 tar -zxvf 解压到指定目录



第四步：进入 zookeeper的安装目录，创建 data 文件夹，用于存储zookeeper的数据

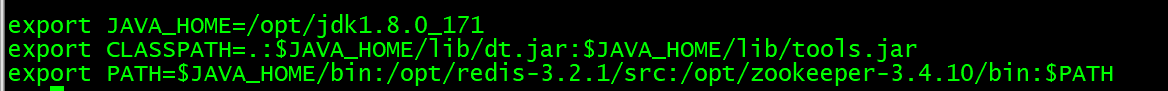


第五步：进入conf目录 ，把 zoo\_sample.cfg 改名为 zoo.cfg，默认识别zoo.cfg配置文件



第六步：打开zoo.cfg , 修改 data 属性：dataDir=/opt/zookeeper-3.4.10/data,指定数据存储的路径地址

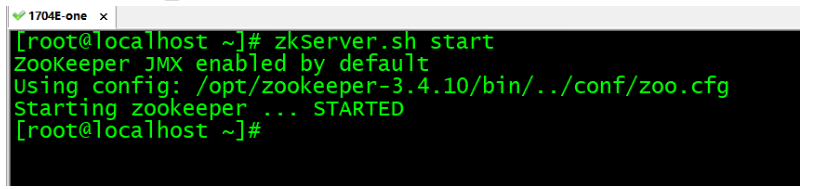
第七步：配置环境遍历 vim /etc/profile



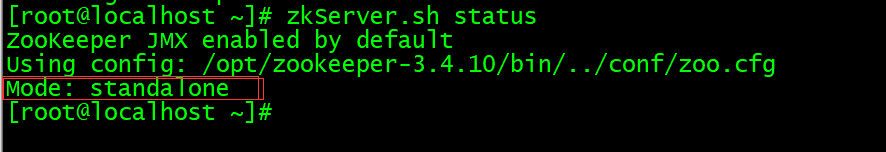
生效文件



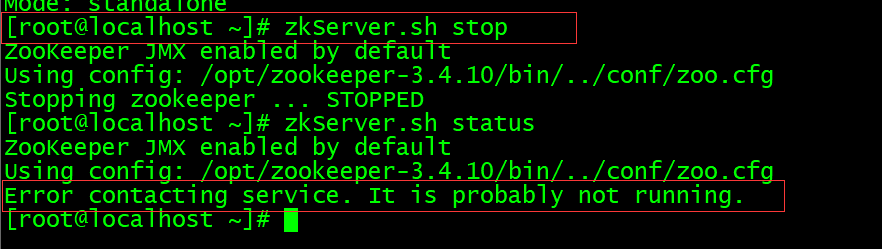
第八步:启动zookeeper



第九步:查看状态： zkServer.sh status

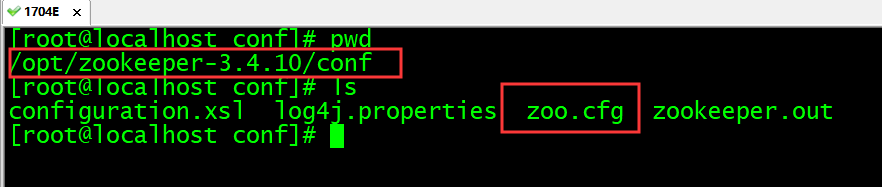


第十步:关闭zookeeper 输入命令zkServer.sh stop

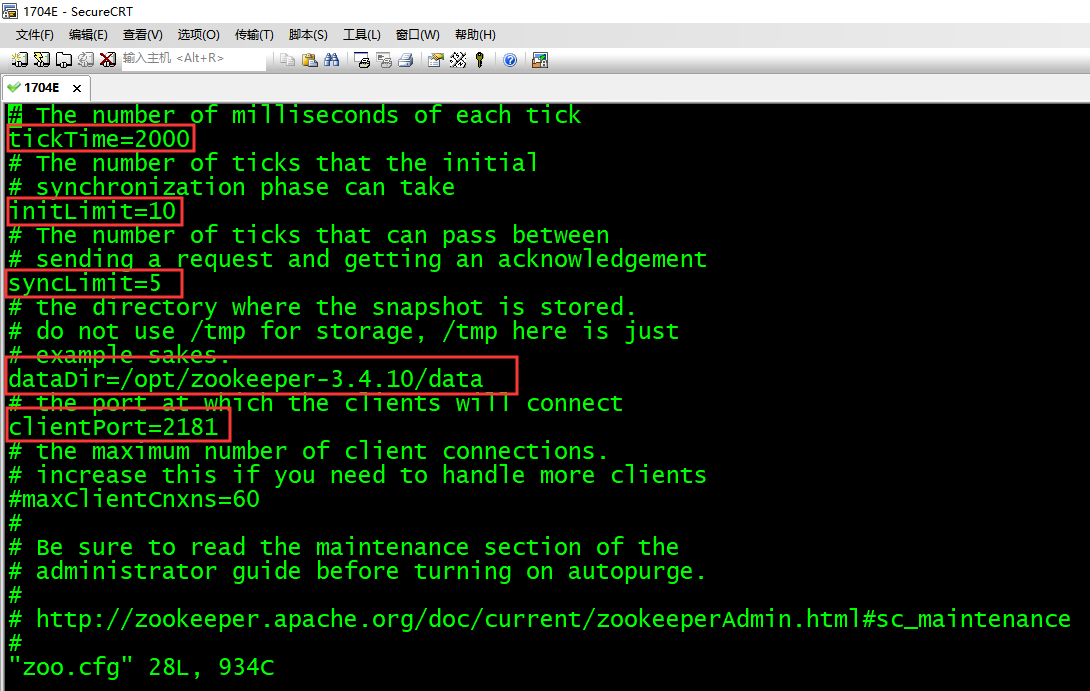


### 2.3.2 配置文件详解

切换到指定的目录



查看zoo.cfg配置文件



配置文件说明：

1. 基本配置

* tickTime :心跳基本时间单位，毫秒级，ZK基本上所有的时间都是这个时间的整数倍。  
    
  initLimit :tickTime的个数，表示在leader选举结束后，followers与leader同步需要的时间，如果followers比较多或者说leader的数据灰常多时，同步时间相应可能会增加，那么这个值也需要相应增加。当然，这个值也是follower和observer在开始同步leader的数据时的最大等待时间(setSoTimeout)  
    
  syncLimit :tickTime的个数，这时间容易和上面的时间混淆，它也表示follower和observer与leader交互时的最大等待时间，只不过是在与leader同步完毕之后，进入正常请求转发或ping等消息交互时的超时时间。  
    
  dataDir:内存数据库快照存放地址，如果没有指定事务日志存放地址(dataLogDir)，默认也是存放在这个路径下，建议两个地址分开存放到不同的设备上。  
    
  clientPort:配置ZK监听客户端连接的端口

1. 集群配置

* server.serverid=host:tickpot:electionport  
  server：固定写法  
  serverid：每个服务器的指定ID（必须处于1-255之间，必须每一台机器不能重复）  
  host：主机名  
  tickpot：心跳通信端口  
  electionport：选举端口

1. 高级配置

* dataLogDir：将事务日志存储在该路径下，比较重要，这个日志存储的设备效率会影响ZK的写吞吐量。  
    
  globalOutstandingLimit：(Java system property: zookeeper.globalOutstandingLimit)默认值是1000，限定了所有连接到服务器上但是还没有返回响应的请求个数(所有客户端请求的总数，不是连接总数)，这个参数是针对单台服务器而言，设定太大可能会导致内存溢出。  
    
  preAllocSize：(Java system property: zookeeper.preAllocSize)默认值64M，以KB为单位,预先分配额定空间用于后续transactionlog 写入，每当剩余空间小于4K时，就会又分配64M，如此循环。如果SNAP做得比较频繁(snapCount比较小的时候)，那么请减少这个值。  
    
  snapCount：(Java system property: zookeeper.snapCount)默认值100,000，当transaction每达到snapCount/2+rand.nextInt(snapCount/2)时，就做一次SNAPSHOT,默认情况下是50,000~100,000条transactionlog就会做一次，之所以用随机数是为了避免所有服务器可能在同一时间做snapshot.  
    
  traceFile (Java system property: requestTraceFile)  
    
  maxClientCnxns：默认值是10，一个客户端能够连接到同一个服务器上的最大连接数，根据IP来区分。如果设置为0，表示没有任何限制。设置该值一方面是为了防止DoS攻击。  
    
  clientPortAddress ：与clientPort匹配，表示某个IP地址，如果服务器有多个网络接口(多个IP地址),如果没有设置这个属性，则clientPort会绑定到所有IP地址上，否则只绑定到该设置的IP地址上。  
    
  minSessionTimeout：最小的session time时间，默认值是2个tick time,客户端设置的session time 如果小于这个值，则会被强制协调为这个最小值。  
    
  maxSessionTimeout：最大的session time 时间，默认值是20个tick time. ,客户端设置的session time 如果大于这个值，则会被强制协调为这个最大值。

## 2.4 课程总结

1. 了解zookeeper
2. 掌握zookeepr的安装与配置

## 2.5 作业

1. 完成zookeeper单机安装
2. 完成zookeeper集群搭建