Zookeeper 介绍：

“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.”

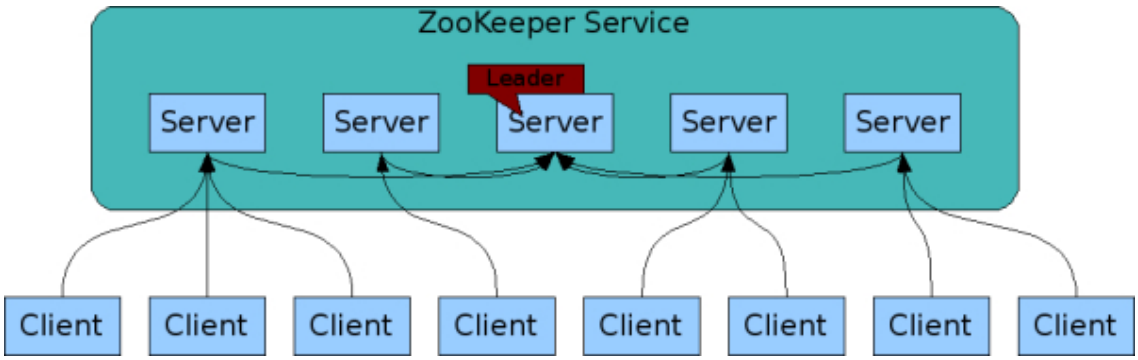
Zookeeper 高可用、高性能且一致的开源协调服务，它提供了一项基本服务：统一命名服务、、布式协调、存储数据、监听与通知等功能

官网： http://zookeeper.apache.org/

源码： https://github.com/apache/zookeeper

快速开始：

集群搭建：



Zoo.cfg 参数

参数	意义
tickTime	2000
syncLimit	Leader 和 follower 之间的通讯时长 最长不能超过 initLimt*ticktime
initLimt	接受客户端链接 zk 初始化的最长等待心跳时长 initLimt*ticktime
dataDir	数据目录
dataLogDir	日志文件
clientPort	客户端链接服务端端口号
Server.A=B:C:D	A:第几号服务器 B 服务 IP C 代表 Leader 和 follower 通讯端口 D 备用选 leader 端口

角色：

Leader：

Leader 作为整个 ZooKeeper 集群的主节点，负责响应所有对 ZooKeeper 状态变更的请求。它会将每个状态更新请求进行排序和编号，以便保证整个集群内部消息处理的 FIFO，写操作都走 leader

Follower：

Follower 的逻辑就比较简单了。除了响应本服务器上的读请求外，follower 还要处理 leader 的提议，并在 leader 提交该提议时在本地也进行提交。 另外需要注意的是，leader 和 follower 构成 ZooKeeper 集群的法定人数，也就是说，只有他们才参与新 leader 的选举、响应 leader 的提议。

Observer：

如果 ZooKeeper 集群的读取负载很高，或者客户端多到跨机房，可以设置一些 observer 服务器，以提高读取的吞吐量。Observer 和 Follower 比较相似，只有一些小区别：首先 observer 不属于法定人数，即不参加选举也不响应提议；其次是 observer 不需要将事务持久化到磁盘，一旦 observer 被重启，需要从 leader 重新同步整个名字空间。

Zookeeper 特性：

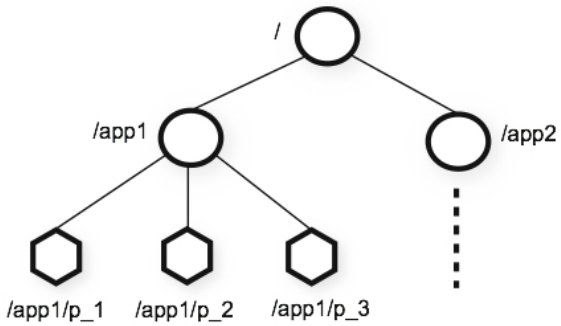
Zookeeper 是一个由多个 server 组成的集群
一个 leader，多个 follower
每个 server 保存一份数据副本

全局数据一致
分布式读 follower，写由 leader 实施
更新请求转发，由 leader 实施
更新请求顺序进行，来自同一个 client 的更新请求按其发送顺序依次执行
数据更新原子性，一次数据更新要么成功，要么失败
全局唯一数据视图，client 无论连接到哪个 server，数据视图都是一致的
实时性，在一定事件范围内，client 能读到最新数据

Session 会话：

客户端会话，客户端和服务端建立一个 TCP 长连接
org.apache.zookeeper.ClientCnxn

Znode 数据模型：



Znode 结构：

Stat：状态信息、版本、权限相关

状态属性	说明
czxid	节点创建时的 zxid
mzxid	节点最新一次更新发生时的 zxid
ctime	节点创建时的时间戳.
mtime	节点最新一次更新发生时的时间戳.
dataVersion	节点数据的更新次数.
cversion	其子节点的更新次数

aclVersion	节点 ACL(授权信息)的更新次数.
ephemeralOwner	如果该节点为 ephemeral 节点, ephemeralOwner 值表示与该节点绑定的 session id. 如果该节点不是 ephemeral 节点, ephemeralOwner 值为 0. 至于什么是 ephemeral 节点
dataLength	节点数据的字节数.
numChildren	子节点个数.

节点类型：

2 大类、四种类型 持久、临时、持久有序、临时有序
PERSISTENT 持久类型，如果不手动删除 是一直存在的
PERSISTENT_SEQUENTIAL
EPHEMERAL 临时 客户端 session 失效就会随着删除节点 没有子节点
EPHEMERAL_SEQUENTIAL 有序 自增

顺序号：

创建 znode 时设置顺序标识，znode 名称后会附加一个值
顺序号是一个单调递增的计数器，由父节点维护
在分布式系统中，顺序号可以被用于为所有的事件进行全局排序，这样客户端可以通过顺序号推断事件的顺序

ACL(Access Control List)

org.apache.zookeeper.ZooDefs
内置的 ACL schemes：
world：默认方式，相当于全世界都能访问
auth：代表已经认证通过的用户(cli 中可以通过 addauth digest user:pwd 来添加当前上下文中的授权用户)
digest：即用户名:密码这种方式认证，这也是业务系统中最常用的
ip：使用 Ip 地址认证

ACL 支持权限：

CREATE：能创建子节点

READ：能获取节点数据和列出其子节点

WRITE：能设置节点数据

DELETE：能删除子节点

ADMIN：能设置权限

Watcher:监听

KeeperState	EventType	触发条件	说明	操作
SyncConnected（3）	None（-1）	客户端与服务端成功建立连接	此时客户端和服务器处于连接状态	
	NodeCreated（1）	Watcher 监听的对应数据节点被创建		Create
	NodeDeleted（2）	Watcher 监听的对应数据节点被删除		Delete/znode
	NodeDataChanged（3）	Watcher 监听的对应数据节点的数据内容发生变更		setDate/znode
	NodeChildChanged（4）	Wather 监听的对应数据节点的子节点列表发生变更		Create/child
Disconnected（0）	None（-1）	客户端与 ZooKeeper 服务器断开连接	此时客户端和服务器处于断开连接状态	
Expired（-112）	None（-1）	会话超时	此时客户端会话失效，通常同时也会受到 SessionExpiredException 异常	
AuthFailed（4）	None（-1）	通常有两种情况，1：使用错误的 schema 进行权限检查 2：SASL 权限检查失败	通常同时也会收到 AuthFailedException 异常	

ZAB 协议： Zookeeper 的核心是原子广播，这个机制保证了各个 server 之间的同步。实现这个机制的协议叫做 Zab 协议。Zab 协议有两种模式，它们分别是恢复模式和广播模式。当服务启动或者在领导者崩溃后，Zab 就进入了恢复模式，当领导者被选举出来，且大多数 server 的完成了和 leader 的状态同步以后，恢复模式就结束了。状态同步保证了 leader 和 server 具有相同的系统状态。一旦 leader 已经和多数的 follower 进行了状态同步后，他就可以开始广播消息了，即进入广播状态。
这时候当一个 server 加入 zookeeper 服务中，它会在恢复模式下启动，发现 leader，并和 leader 进行状态同步。待到同步结束，它也参与消息广播。

Zookeeper 服务一直维持在 Broadcast 状态，直到 leader 崩溃了或者 leader 失去了大部分的 followers 支持。
广播模式需要保证 proposal 被按顺序处理，因此 zk 采用了递增的事务 id 号(zxid)来保证。所有的提议(proposal)都在被提出的时候加上了 zxid。实现中 zxid 是一个 64 为的数字，它高 32 位是 epoch 用来标识 leader 关系是否改变，每次一个 leader 被选出来，它都会有一个新的 epoch。低 32 位是个递增计数。
当 leader 崩溃或者 leader 失去大多数的 follower，这时候 zk 进入恢复模式，恢复模式需要重新选举出一个新的 leader，让所有的 server 都恢复到一个正确的状态。

Leader 选举：

LOOKING, FOLLOWING, LEADING, OBSERVING

总结：

这节课我们讲了集群搭建、和常用命令使用。
Zookeeper 是一个有上下级关系（**Leader、follower、Observer** ）的集群。
客户端链接 zookeeper 集群是通过 Seesion 链接(TCP 长链接)。
客户端链接以后可以对节点(存储在 zookeeper 上 znode)增删改查。
Znode 有四种类型：临时、临时有序、持久、持久有序
对（znode）节点做增删改查时我们可以监控其动作（Watcher 机制）还可以对节点设置权限访问。