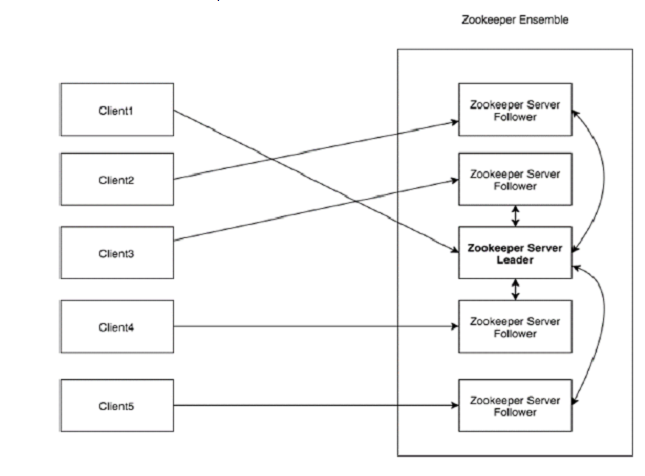
**Zookeeper 基础**

1. Architecture（架构）
2. Hierarchical namespace（层次命名空间）
3. Session（会话）
4. Watches（监视）

客户端-服务器架构



Client （客户端） 客户端，我们的分布式应用集群中的一个节点，从服务器访问信息。对于特定的时间间隔，每个客户端向服务器发送消息以使服务器知道客户端是活跃的。

类似地，当客户端连接时，服务器发送确认码。如果连接的服务器没有响应，客户端会自动将消息重定向到另一个服务器。

Server（服务器）服务器，我们的ZooKeeper总体中的一个节点，为客户端提供所有的服务。向客户端发送确认码以告知服务器是活跃的。

Ensemble Zookeeper服务器组件，形成ensemble所需要的最小节点数为3

Leader 服务器节点

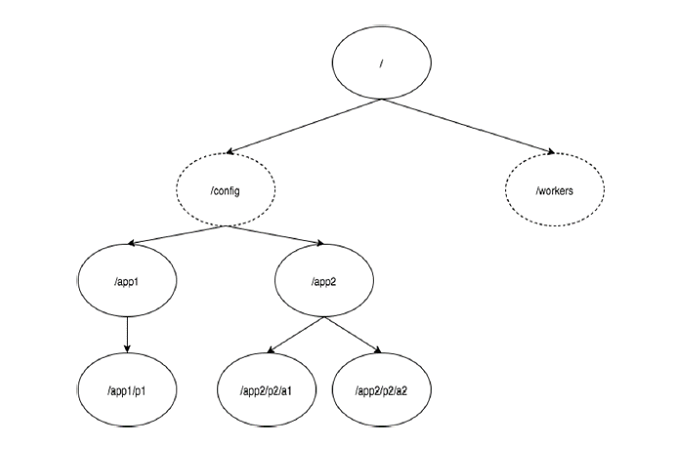
Follower

## 层次命名空间

ZooKeeper节点称为**znode**。每个znode由一个名称标识，并用路径(/)序列分隔。

首先一个由/分隔的znode，在根目录下，有两个逻辑命名空间，config和workers

在config下，每个znode节点最多存储1M数据，存储同步数据并描述znode的元数据



ZooKeeper数据模型中的每个znode都维护着一个**stat**结构。一个stat仅提供一个znode的**元数据**。它由版本号，操作控制列表(ACL)，时间戳和数据长度组成。

版本号：每个znode都有一个版本号，每当与znode相关联的数据发生变化时，其对应的版本号也会增加

操作控制列表ACL,ACL是访问znode的认证机制，管理所有znode读取和写入操作

时间戳：时间戳表示创建和修改znode所经过的时间，通常以毫秒为单位，zookeeper从事务ID(zxid)，标识znode的每个更改，zxid是唯一的，并且为每个事务保留一时间，以便可以轻松确定一个请求到另一个请求锁经过的时间

数据长度：存储在znode中的数据总量是数据长度，最多可以存储1m的数据

### Znode的类型

Znode被分为持久（persistent）节点，顺序（sequential）节点和临时（ephemeral）节点。

* **持久节点** - 即使在创建该特定znode的客户端断开连接后，持久节点仍然存在。默认情况下，除非另有说明，否则所有znode都是持久的。
* **临时节点**- 客户端活跃时，临时节点就是有效的。当客户端与ZooKeeper集合断开连接时，临时节点会自动删除。因此，只有临时节点不允许有子节点。如果临时节点被删除，则下一个合适的节点将填充其位置。临时节点在leader选举中起着重要作用。
* **顺序节点**- 顺序节点可以是持久的或临时的。当一个新的znode被创建为一个顺序节点时，ZooKeeper通过将10位的序列号附加到原始名称来设置znode的路径。例如，如果将具有路径**/myapp**的znode创建为顺序节点，则ZooKeeper会将路径更改为**/myapp0000000001**，并将下一个序列号设置为0000000002。如果两个顺序节点是同时创建的，那么ZooKeeper不会对每个znode使用相同的数字。顺序节点在锁定和同步中起重要作用。

## Sessions（会话）

会话对于ZooKeeper的操作非常重要。会话中的请求按FIFO顺序执行。一旦客户端连接到服务器，将建立会话并向客户端分配**会话ID**。

客户端以特定的时间间隔发送**心跳**以保持会话有效。如果ZooKeeper集合在超过服务器开启时指定的期间（会话超时）都没有从客户端接收到心跳，则它会判定客户端死机。

会话超时通常以毫秒为单位。当会话由于任何原因结束时，在该会话期间创建的临时节点也会被删除。

## Watches（监视）

监视是一种简单的机制，使客户端收到关于ZooKeeper集合中的更改的通知。客户端可以在读取特定znode时设置Watches。Watches会向注册的客户端发送任何znode（客户端注册表）更改的通知。

Znode更改是与znode相关的数据的修改或znode的子项中的更改。只触发一次watches。如果客户端想要再次通知，则必须通过另一个读取操作来完成。当连接会话过期时，客户端将与服务器断开连接，相关的watches也将被删除。

# Zookeeper 工作流

一旦ZooKeeper集合启动，它将等待客户端连接。客户端将连接到ZooKeeper集合中的一个节点。它可以是leader或follower节点。一旦客户端被连接，节点将向特定客户端分配会话ID并向该客户端发送确认。如果客户端没有收到确认，它将尝试连接ZooKeeper集合中的另一个节点。 一旦连接到节点，客户端将以有规律的间隔向节点发送心跳，以确保连接不会丢失。

**如果客户端想要读取特定的znode，**它将会向具有znode路径的节点发送**读取请求**，并且节点通过从其自己的数据库获取来返回所请求的znode。为此，在ZooKeeper集合中读取速度很快

**如果客户端想要将数据存储在ZooKeeper集合中**，则会将znode路径和数据发送到服务器。连接的服务器将该请求转发给leader，然后leader将向所有的follower重新发出写入请求。如果只有大部分节点成功响应，而写入请求成功，则成功返回代码将被发送到客户端。 否则，写入请求失败。绝大多数节点被称为**Quorum**。

## ZooKeeper集合中的节点

让我们分析在ZooKeeper集合中拥有不同数量的节点的效果

