
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
Zookeeper

目录 Contents

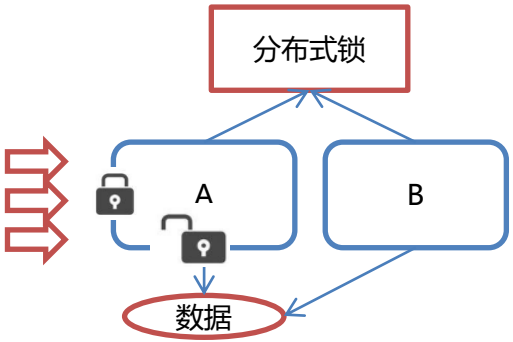
初识 Zookeeper

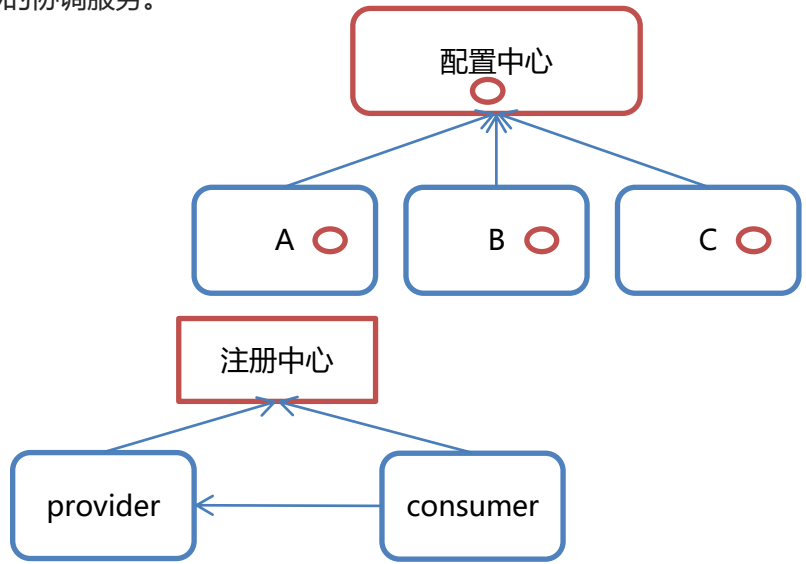
- Zookeeper 概念

初识 Zookeeper

Zookeeper 概念

- Zookeeper 是 Apache Hadoop 项目下的一个子项目，是一个树形目录服务。
- Zookeeper 翻译过来就是 动物园管理员，他是用来管 Hadoop（大象）、Hive(蜜蜂)、Pig(小 猪)的管理员。简称zk
- Zookeeper 是一个分布式的、开源的分布式应用程序的协调服务。
- Zookeeper 提供的主要功能包括：
  - 配置管理
  - 分布式锁
  - 集群管理





目录 Contents

目录 Contents

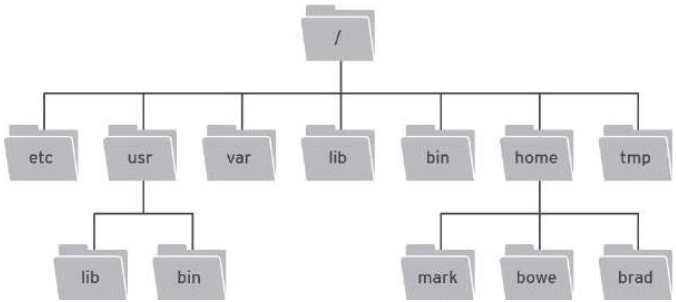
ZooKeeper 命令操作

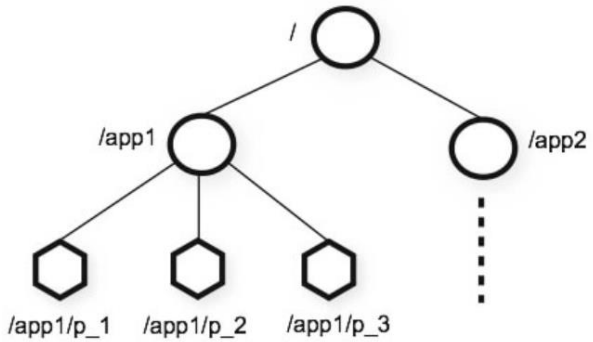
- Zookeeper 数据模型
- Zookeeper 服务端常用命令
- Zookeeper 客户端常用命令

ZooKeeper 命令操作

Zookeeper 数据模型

- ZooKeeper 是一个树形目录服务,其数据模型和Unix的文件系统目录树很类似，拥有一个层次化结构。
- 这里面的每一个节点都被称为： ZNode，每个节点上都会保存自己的数据和节点信息。
- 节点可以拥有子节点，同时也允许少量（1MB）数据存储在该节点之下。
- 节点可以分为四大类:
    - PERSISTENT 持久化节点
    - EPHEMERAL 临时节点： -e
    - PERSISTENT_SEQUENTIAL 持久化顺序节点： -s
    - EPHEMERAL_SEQUENTIAL 临时顺序节点 ： -es





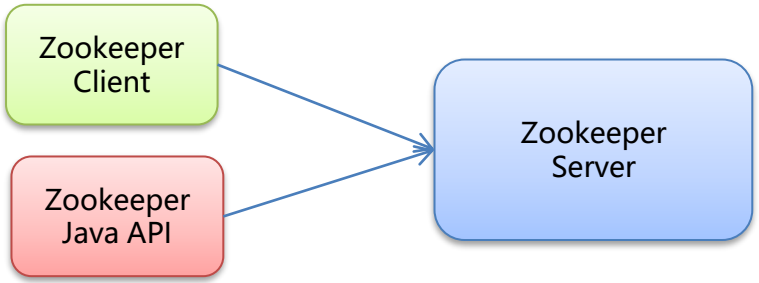
ZooKeeper 命令操作

- Zookeeper 数据模型
- Zookeeper 服务端常用命令
- Zookeeper 客户端常用命令

ZooKeeper 命令操作

Zookeeper 服务端常用命令

- 启动 ZooKeeper 服务: ./zkServer.sh start
- 查看 ZooKeeper 服务状态: ./zkServer.sh status
- 停止 ZooKeeper 服务: ./zkServer.sh stop
- 重启 ZooKeeper 服务: ./zkServer.sh restart



ZooKeeper 命令操作

- Zookeeper 数据模型
- Zookeeper 服务端常用命令
- Zookeeper 客户端常用命令

ZooKeeper 命令操作

Zookeeper 客户端常用命令

- 连接ZooKeeper服务端

```
./zkCli.sh –server ip:port
```

- 断开连接

```
quit
```

- 查看命令帮助

```
help
```

- 显示指定目录下节点

```
ls 目录
```

- 创建节点

```
create /节点path value
```

- 获取节点值

```
get /节点path
```

- 设置节点值

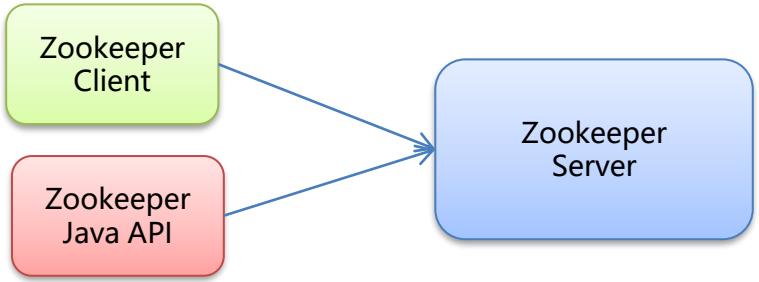
```
set /节点path value
```

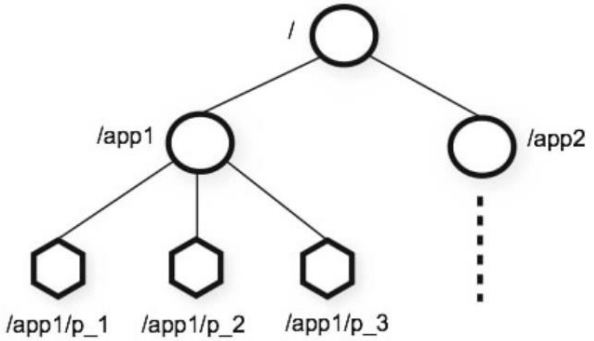
- 删除单个节点

```
delete /节点path
```

- 删除带有子节点的节点

```
deleteall /节点path
```





ZooKeeper 命令操作

Zookeeper 客户端常用命令

- 创建临时节点

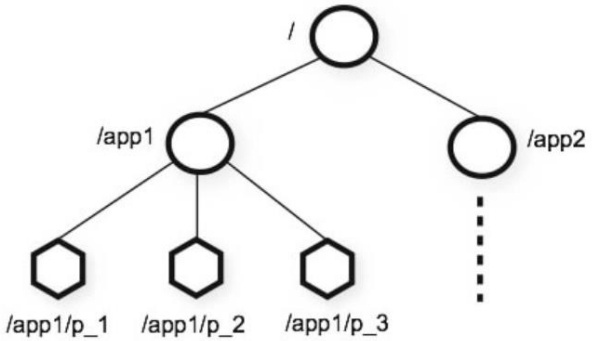
```
create -e /节点path value
```

- 创建顺序节点

```
create -s /节点path value
```

- 查询节点详细信息

```
ls -s /节点path
```



- czxid：节点被创建的事务ID
- ctime: 创建时间
- mzxid: 最后一次被更新的事务ID
- mtime: 修改时间
- pzxid：子节点列表最后一次被更新的事务ID
- cversion：子节点的版本号
- dataversion：数据版本号
- aclversion：权限版本号
- ephemeralOwner：用于临时节点，代表临时节点的事务ID，如果为持久节点则为0
- dataLength：节点存储的数据的长度
- numChildren：当前节点的子节点个数

目录 Contents

ZooKeeper JavaAPI 操作

- Curator 介绍
- Curator API 常用操作
- 分布式锁
- 模拟12306售票案例

ZooKeeper JavaAPI 操作

Curator 介绍

- Curator 是 Apache ZooKeeper 的Java客户端库。
- 常见的ZooKeeper Java API：
  - 原生Java API
  - ZkClient
  - Curator
- Curator 项目的目标是简化 ZooKeeper 客户端的使用。
- Curator 最初是 Netfix 研发的,后来捐献了 Apache 基金会,目前是 Apache 的顶级项目。
- 官网：http://curator.apache.org/

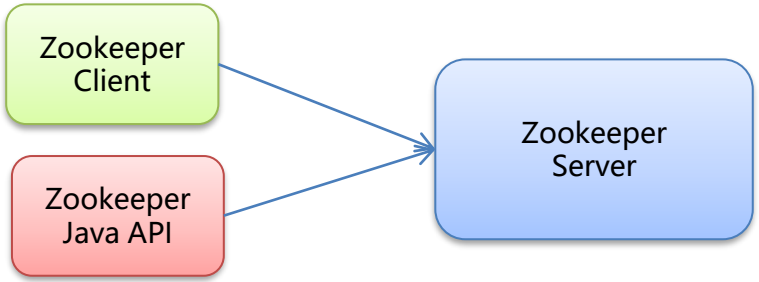
ZooKeeper JavaAPI 操作

- Curator 介绍
- Curator API 常用操作
- 分布式锁
- 模拟12306售票案例

ZooKeeper JavaAPI 操作

Curator API 常用操作

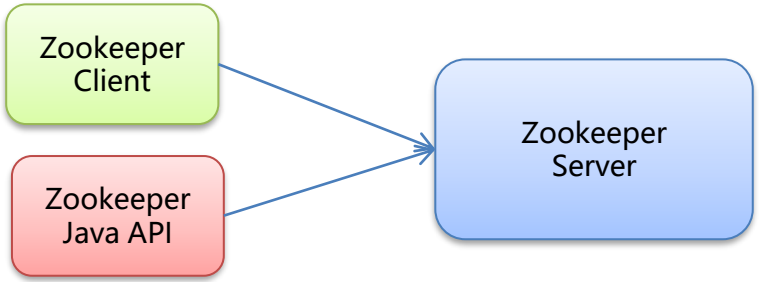
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

Curator API 常用操作

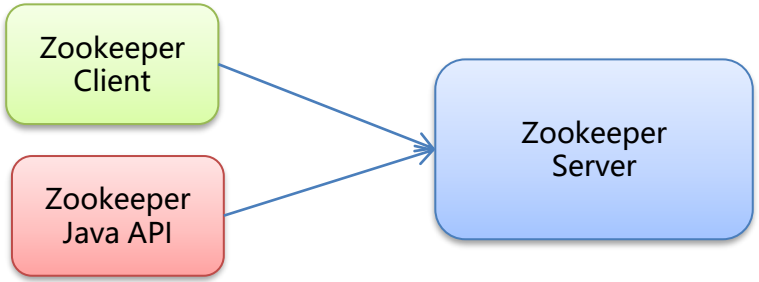
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

Curator API 常用操作

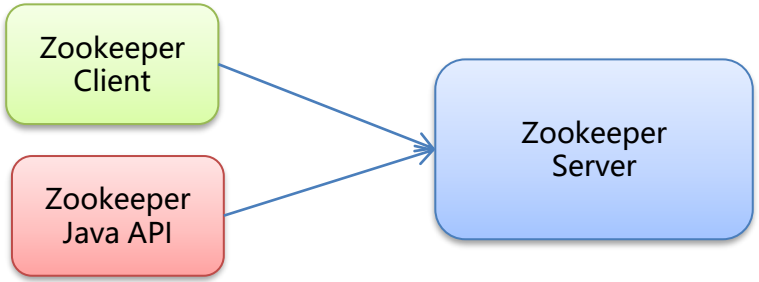
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

Curator API 常用操作

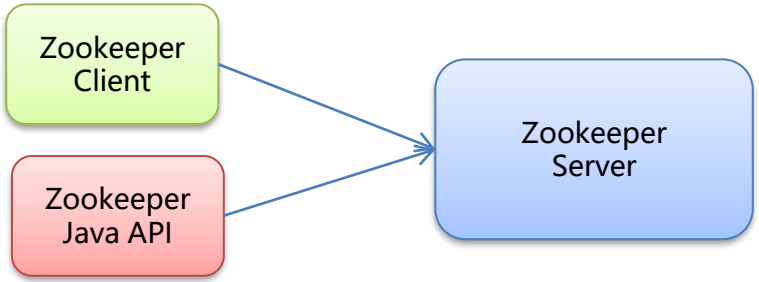
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

Curator API 常用操作

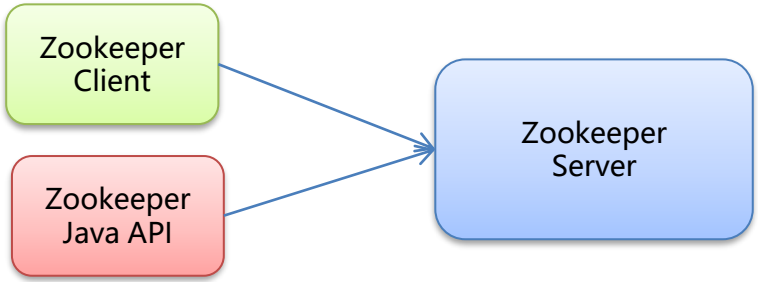
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

Curator API 常用操作

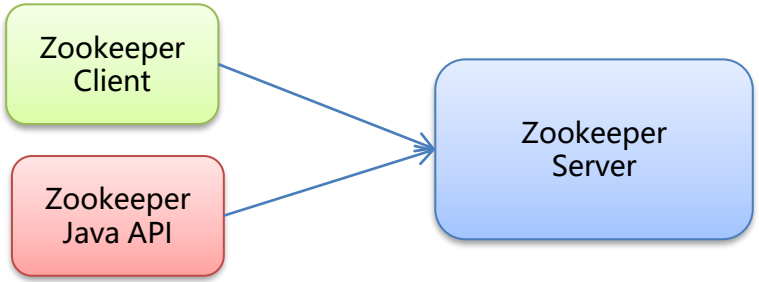
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

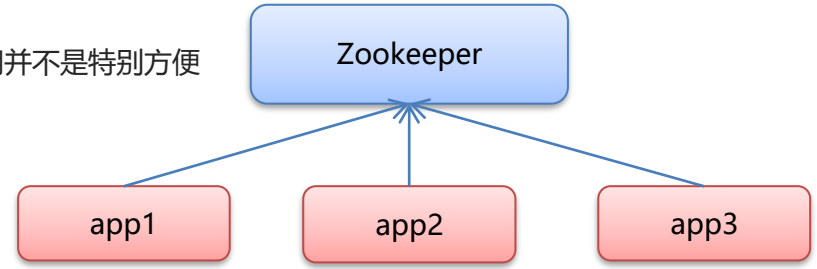
Curator API 常用操作

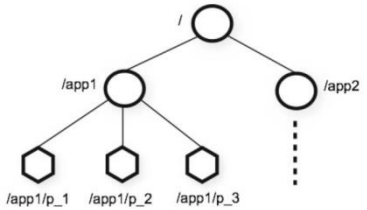
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



Curator API 常用操作 - Watch事件监听

- ZooKeeper 允许用户在指定节点上注册一些Watcher，并且在一些特定事件触发的时候，ZooKeeper 服务端会将事件通知到感兴趣的客户端上去，该机制是 ZooKeeper 实现分布式协调服务的重要特性。
- ZooKeeper 中引入了Watcher机制来实现了发布/订阅功能能，能够让多个订阅者同时监听某一个对象，当一个对象自身状态变化时，会通知所有订阅者。
- ZooKeeper 原生支持通过注册Watcher来进行事件监听，但是其使用并不是特别方便需要开发人员自己反复注册Watcher，比较繁琐。
- Curator引入了 Cache 来实现对 ZooKeeper 服务端事件的监听。
- ZooKeeper提供了三种Watcher：
  - NodeCache : 只是监听某一个特定的节点
  - PathChildrenCache : 监控一个ZNode的子节点.
  - TreeCache : 可以监控整个树上的所有节点，类似于PathChildrenCache和NodeCache的组合

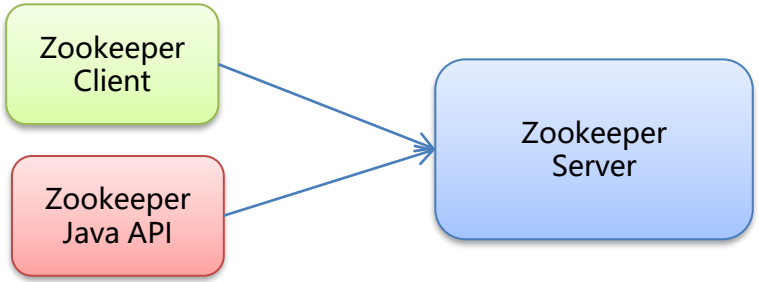




ZooKeeper JavaAPI 操作

Curator API 常用操作

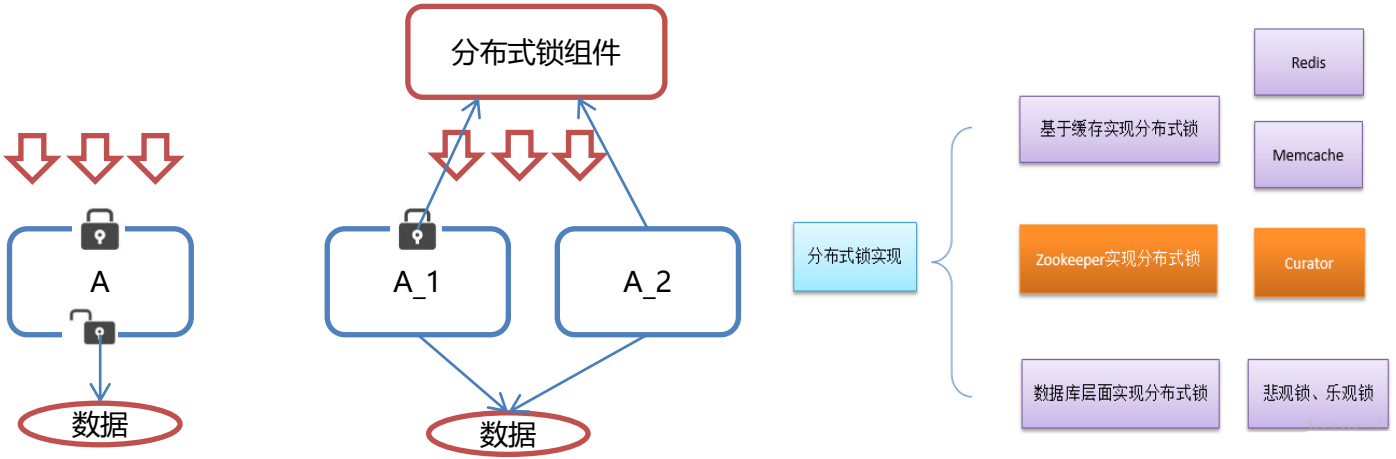
- 建立连接
- 添加节点
- 删除节点
- 修改节点
- 查询节点
- Watch事件监听
- 分布式锁实现



ZooKeeper JavaAPI 操作

分布式锁

- 在我们进行单机应用开发，涉及并发同步的时候，我们往往采用synchronized或者Lock的方式来解决多线程间的代码同步问题，这时多线程的运行都是在同一个JVM之下，没有任何问题。
- 但当我们的应用是分布式集群工作的情况下，属于多JVM下的工作环境，跨JVM之间已经无法通过多线程的锁解决同步问题。
- 那么就需要一种更加高级的锁机制，来处理种跨机器的进程之间的数据同步问题——这就是分布式锁。

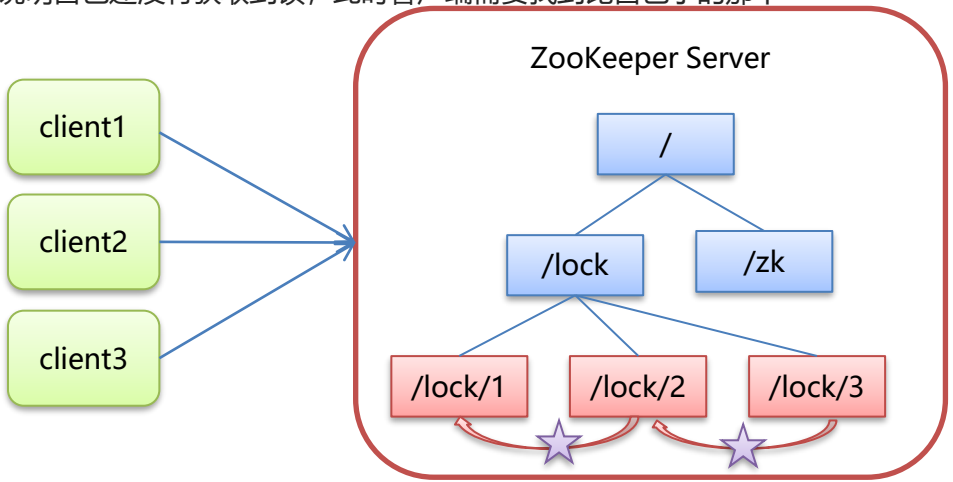


ZooKeeper JavaAPI 操作

ZooKeeper分布式锁原理

- 核心思想：当客户端要获取锁，则创建节点，使用完锁，则删除该节点。

1. 客户端获取锁时，在lock节点下创建临时顺序节点。
2. 然后获取lock下面的所有子节点，客户端获取到所有的子节点之后，如果发现自己创建的子节点序号最小，那么就认为该客户端获取到了锁。使用完锁后，将该节点删除。
3. 如果发现自己创建的节点并非lock所有子节点中最小的，说明自己还没有获取到锁，此时客户端需要找到比自己小的那个节点，同时对其注册事件监听器，监听删除事件。
4. 如果发现比自己小的那个节点被删除，则客户端的Watcher会收到相应通知，此时再次判断自己创建的节点是否是lock子节点中序号最小的，如果是则获取到了锁，如果不是则重复以上步骤继续获取到比自己小的一个节点并注册监听。



ZooKeeper JavaAPI 操作

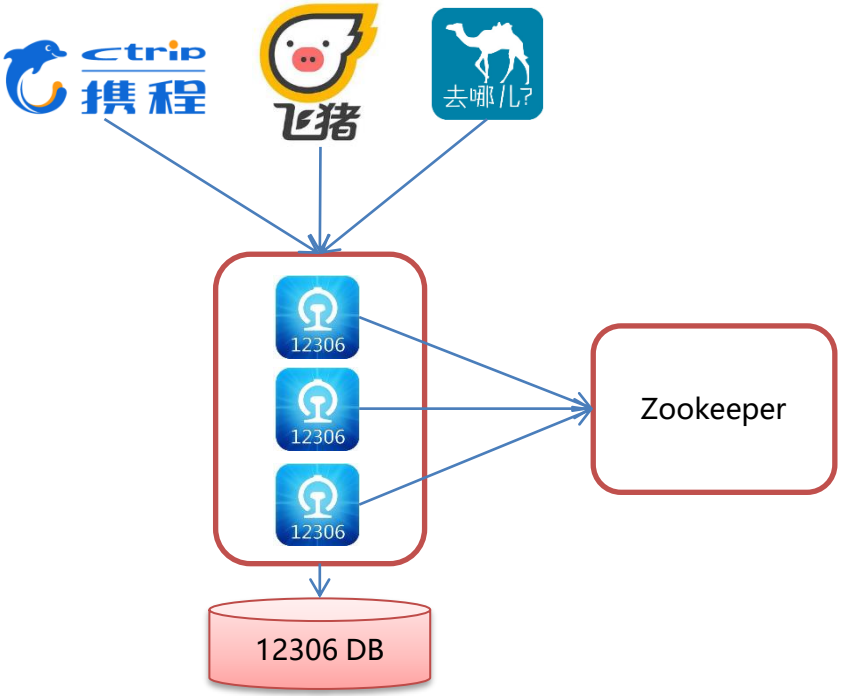
- Curator 介绍
- Curator API 常用操作
- 分布式锁
- 模拟12306售票案例

Curator实现分布式锁API

- 在Curator中有五种锁方案：
  - InterProcessSemaphoreMutex：分布式排它锁（非可重入锁）
  - InterProcessMutex：分布式可重入排它锁
  - InterProcessReadWriteLock：分布式读写锁
  - InterProcessMultiLock：将多个锁作为单个实体管理的容器
  - InterProcessSemaphoreV2：共享信号量

ZooKeeper JavaAPI 操作

分布式锁案例 – 模拟12306售票



目录 Contents

ZooKeeper 集群搭建

- ZooKeeper 集群介绍
- ZooKeeper 集群搭建

ZooKeeper 集群搭建

Zookeeper 集群介绍

Leader选举：

- Serverid：服务器ID

  比如有三台服务器，编号分别是1,2,3。

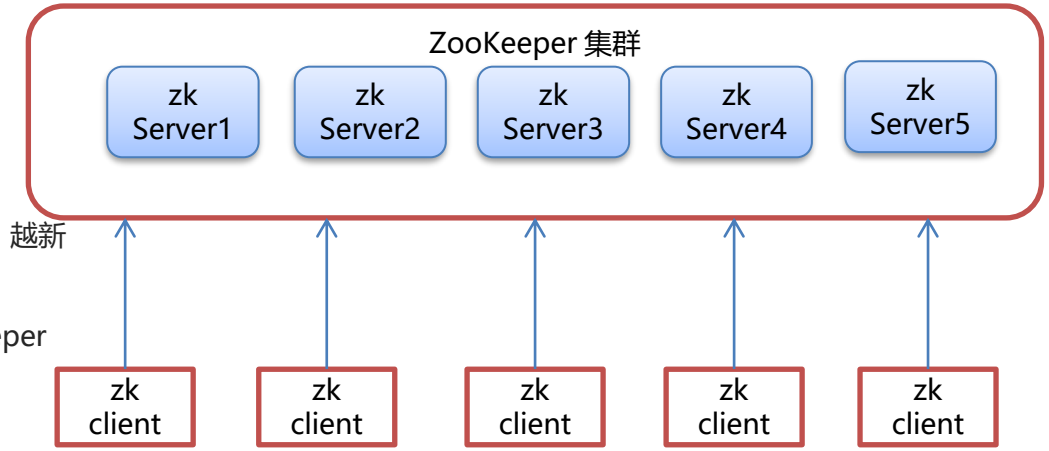
  编号越大在选择算法中的权重越大。

- Zxid：数据ID

  服务器中存放的最大数据ID.值越大说明数据 越新，在选举算法中数据越新权重越大。

- 在Leader选举的过程中，如果某台ZooKeeper获得了超过半数的选票，

  则此ZooKeeper就可以成为Leader了。



ZooKeeper 集群搭建

- ZooKeeper 集群介绍
- ZooKeeper 集群搭建

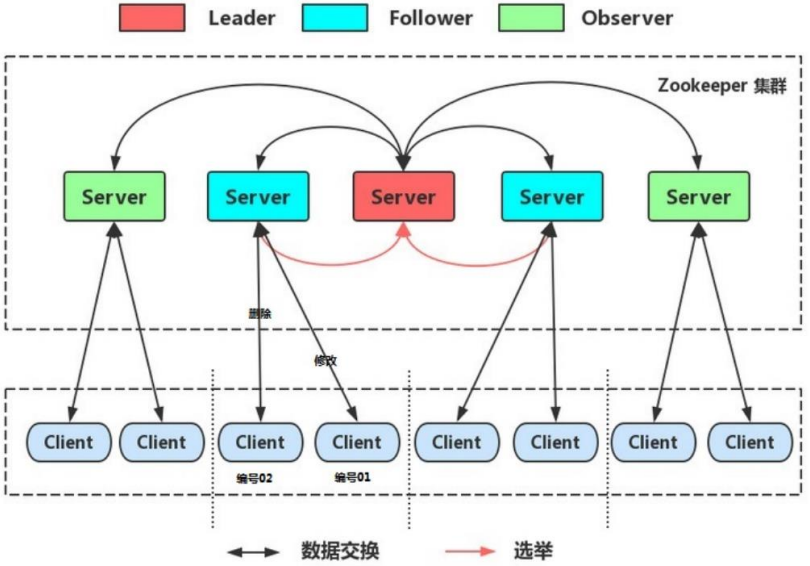
目录 Contents

ZooKeeper 集群搭建

Zookeeper 集群角色

在ZooKeeper集群服中务中有三个角色：

- Leader 领导者：
  1. 处理事务请求
  2. 集群内部各服务器的调度者
- Follower 跟随者：
  1. 处理客户端非事务请求，转发事务请求给Leader服务器
  2. 参与Leader选举投票
- Observer 观察者：
  1. 处理客户端非事务请求，转发事务请求给Leader服务器



传智播客旗下高端IT教育品牌