大数据导论和Apache Zookeeper

1.大数据基础介绍

1.1数据与数据分析

- 数据 : 就是对客观事物的逻辑归纳,是用于标识客观事物的未经加工的原始素材.
- 数据分析:所谓的数据就是通过工具或者方法把隐藏在数据背后的规律和价值提取处理的过程
- 数据分析在商业中的作用 : 面向分析,分析的结果支持决策
  - 数据分析的结果给企业的决策提供支撑,支持决策.
  - 数据仓库的出现也是集成的数据分析平台,分析的结果支撑决策

1.2 数据分析的作用

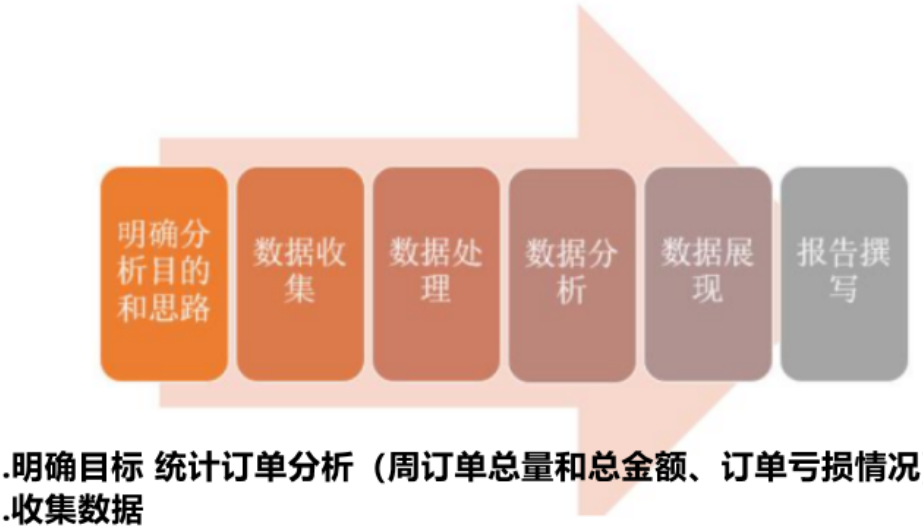
- 我们更加关注商业领域,也就是企业为什么需要数据分析
  - 1.客观原因--对应历史数据
    - T+1数据(T: 天 月 年 小时)
  - 2.现状分析--对应当下数据
    - 实时分析:流处理实时分析,当前生成的数据
  - 3.预测分析--结合数据预测未来

  - 离线分析(批处理分析 batch processing)
    - 机器学习: 根据历史经验、根据机器学习算法,进行预测分析 聚类、分类、关联规则等等
    - 分析已有的数据 历史数据,面向过去分析
  - 实时分析:数据流 streaming
    - 面相当下,尽量缩短时间间隔
  - 机器学习 （Machine Learning，ML）
    - 基于历史数据和当下产生的实时数据预测未来发生的事情。
    - 侧重于数学算法的运用。 分类 聚类 关联 预测。

数据分析六部曲

1.明确目标 统计订单分析（周订单总量和总金额、订单亏损情况）
2.收集数据
爬虫爬取数据
数据埋点，前端点击事件埋点，埋点日志，点击流日志
业务数据，业务数据库中
静态数据
企业年鉴、图书
3.数据的 ETL （抽取、转换、加载）
抽取：拉取数据
转换： 去重、去噪、过滤、脱敏、解析、数据补全等
加载： 数据落地 load
4.数据的分析
统计、分析
挖掘
5.可视化的图表，数据可视化展示
6.撰写报告

- 数据分析六部曲 -- 数据分析基本步骤



1.明确目标 统计订单分析（周订单总量和总金额、订单亏损情况）
2.收集数据
爬虫爬取数据
数据埋点，前端点击事件埋点，埋点日志，点击流日志
业务数据，业务数据库中
静态数据
企业年鉴、图书
3.数据的 ETL （抽取、转换、加载）
抽取：拉取数据
转换： 去重、去噪、过滤、脱敏、解析、数据补全等
加载： 数据落地 load
4.数据的分析
统计、分析
挖掘
5.可视化的图表，数据可视化展示
6.撰写报告

- 1.明确分析目的和思路 --分析主题 框架
  - 明确目标 统计订单分析(周订单总量和总金额、订单亏损情况)
- 2.数据收集
  - 爬虫爬取数据
  - 数据埋点,前端点击事件埋点,埋点日志,点击流日志
  - 业务数据,业务数中来
  - 静态数据
  - 业务年鉴、图书
- 3.数据处理 -- 数据结构化
  - 数据格式分为三类
    - 结构化数据
      - 所谓的结构化数据指的是具有schema约束信息的数据.通俗理解已与程序处理解读的数据.
        - schema代表:字段和字段类型(Hive Spark Flink 都会使用到)
      - MySQL数据库的数据就是结构化的数据

        - 字段固定(字段的名称和字段的类型)
    - 半结构化数据
        - 数据类型有结构,结构不固定(内容不固定),例子:HTML、json、xml数据
    - 非结构化数据
        - 唯美的文章(文本)、视频、图片、音频等
- 数据的ETL(抽取、转换、加载)
    - 抽取:拉取数据
    - 转换:去重、去噪(错误的数据)、过滤、脱敏(敏感信息)、解析、数据补全等
    - 加载:数据落地load
- 4.数据分析
    - 统计、分析
    - 挖掘
- 5.数据展现
    - 可视化的图表,数据可视化展示
- 6.报告撰写

1.3 大数据时代

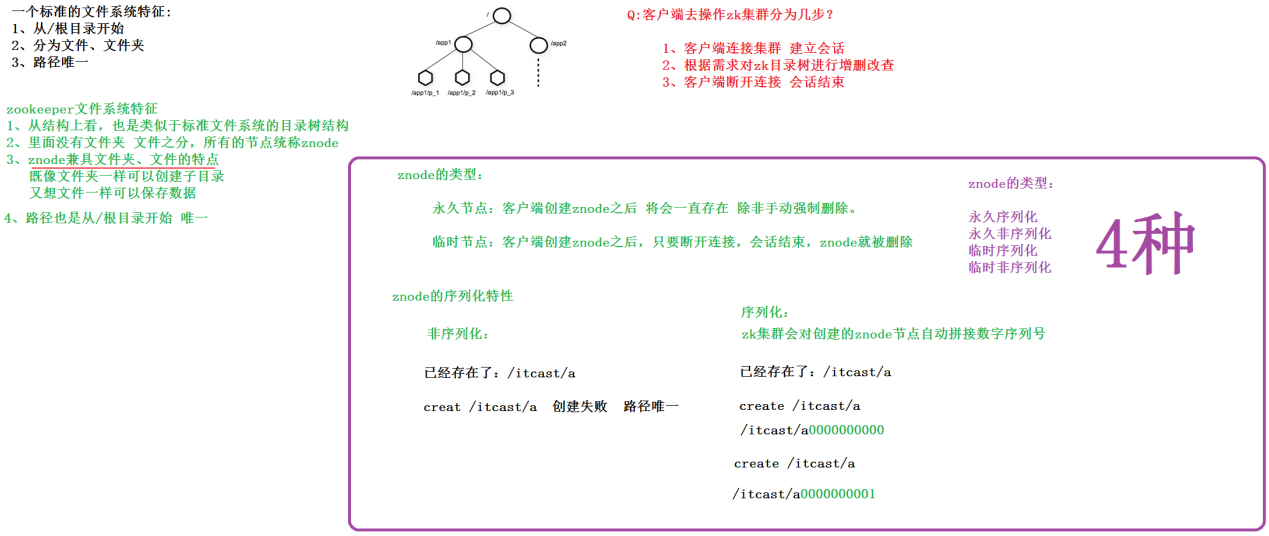
- 定义:信息爆炸时代产生的海量数据,并命名为与之相关的技术发展与创新
- 大数据5V特点
    - 大 : 数据量大,采集、存储、计算都量大
    - 多 : 多样化,数据种类多 (结构化、半结构化、非结构化)
    - 值 : 价值密度低
    - 信 : 数据质量可信 真实有效可信
    - 快 : 生成数据快、处理快 (时效性高)
- 数据大爆炸和面临的挑战
    - 海量数据如何存储
    - 海量数据如何高效计算
- 海量数据存储和计算解决方案
    - 解决方案 : 分布式存储、分布式计算
    - Hadoop生态系统
        - HDFS分布式文件系统用于截取大的文件称数据块最终存储到各个节点上,元数据(存储位置、存储节点、存储大小、创建时间等)保存起来
        - MapReduce分布式计算 Map:分 Reduce :合

1.4分布式技术

- 分布式: 一个软件或者服务多个组件分别运行在不同的机器上,通过通信配合共同对外提供服务
    - 是指在多台不同的服务器中部署不同的服务模块,通过远程调用协同工作,对外提供服务

- 在网络内不同的计算机通过网络通信连接一起,组成了完整功能,每台节点属于在分布式系统内

- 集群: 计算后的数据是同步的
  - 是指在多台不同的服务器中部署相同应用或服务模块,构成一个集群,通过负载均衡设备对外提供服务
  - 在计算机网络内,每台节点的状态和数据保持一致.
- 共同点:多台机器 不是单机的
- 不同点:
  - 集群:每台机器上的服务、状态是一样的
  - 分布式:每台机群上的服务、组件是不一样的
- 分布式演化



2.Apache Zookeeper

2.1基础概念

2.1.1是一个**分布式的协调服务软件**(distributed **coordination**)

- Zookeeper是一个分布式的小的文件系统
  - 分布式是多台机器
  - 小的:存储的数量小1M以内(存储的是控制数据)
  - 文件系统 :
    - 1.文件或文件夹 ZNode
    - 2.Linux系统以根目录开始 / 有绝对路径和相对路径的概念 , Zookeeper必须从 / 开始 ,树形结构,没有相对路径
- 分布式:多台机器的环境
- 协调服务:在分布式环境下,如何控制大家有序的去做某件事
  - 顺序 : 谁先做 A先做 B后做 先来先服务
  - 一致 : Zookeeper状态保持一致
  - 共同 : 共同操作,共同做一件事

- 共享 : 数据是共享的

2.1.2 zookeeper的本质:分布式的小文件存储系统

- 存储系统:存储数据、存储文件 目录树结构
- 小文件:上面存储的数据有大小限制
- 分布式: 可以部署在多台机器上运行,对比单机来理解
- 问题: zk这个存储系统和我们常见的存储系统不一样.基于这些不一样产生了很多应用

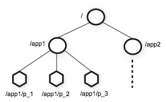
2.1.3 Zookeeper是一个标准的 -**主从结构**- 集群

- zk是标准的主从结构,只不过为了扩大集群的读写能力,同时又不增加选举复杂度,又提供了观察者角色 Zookeeper架构FastLeaderElection 快速选举机制 万一主服务器崩溃了会自动选举另一台服务器作为主服务器 观察者角色不参与
- 主角色 leader : 事务性请求的唯一调度和处理者,leader全局统一调度处理,事务ID
- 从角色 follower : 处理非事务性操作 , 转发事务性操作给leader(主服务器) 参与zk内部选举机制
- 观察者 Observe : 处理非事务性操作 转发事务性操作给给leader 不参与zk内部选举机制 . 通俗化:是一群被剥夺政治终身的follower
- 主从各司其职,共同配合,对外提供服务

2.2 Zookeeper特性

- 1.全局数据一致 : 集群中每个服务器保存一份相同的数据副本,client 无论连接到哪个服务器,展示的数据都 是一致的,这是最重要的特征
  - 事务:通俗理解 多个操作组成一个事务,要么一起成功,要么一起失败,不会存在中间的状态.如果中间失败了要进行回滚操作
  - 事务一般是:增 删 改 非事务 : 查 读
  - 主从一致性:Master 主节点、 Follower从节点,主节点负责管理集群,事务操作(增删改),从节点负责实务操作的转化非事务操作(读)
  - 非事务操作,多个分布式服务器操作都可以自行操作 读 查 事务操作 : 从服务器把事务都发送给主服务器进行操作,自身不进行操作 ,然后进行数据的拉取(同步数据-和主服务器进行相同的操作)

一个标准的文件系统特征:
1、从/根目录开始
2、分为文件、文件夹
3、路径唯一



Q:客户端去操作zk集群分为几步?

1、客户端连接集群 建立会话
2、根据需求对zk目录树进行增删改查
3、客户端断开连接 会话结束

zookeeper文件系统特征
1、从结构上看, 也是类似于标准文件系统的目录树结构
2、里面没有文件夹 文件之分, 所有的节点统称znode
3、znode兼具文件夹、文件的特点
既像文件夹一样可以创建子目录
又想文件一样可以保存数据
4、路径也是从/根目录开始 唯一

znode的类型:

永久节点:客户端创建znode之后 将会一直存在 除非手动强制删除。

临时节点:客户端创建znode之后,只要断开连接,会话结束,znode就被删除

znode的类型:

永久序列化
永久非序列化
临时序列化
临时非序列化

4种

znode的序列化特性

非序列化:

已经存在了:/itcast/a

creat /itcast/a 创建失败 路径唯一

序列化:
zk集群会对创建的znode节点自动拼接数字序列号

已经存在了:/itcast/a

create /itcast/a
/itcast/a0000000000

create /itcast/a
/itcast/a0000000001

- 2.可靠性 : 如果消息被其中一台服务器接受，那么将被所有的服务器接受。
- 3.顺序性 : 1. 包括全局有序和偏序两种：全局有序是指如果在一台服务器上消息a在消息b前发布，则在所有Server上消息a都将在消息b前被发布；偏序是指如果一个消息b在消息a后被同一个发送者发布，a必将排在b前面。
- 4.数据更新原子性 : 一次数据更新要么成功（半数以上节点成功），要么失败，不存在中间状态；
- 5.实时性 : Zookeeper保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息。

2.3 Zookeeper集群的搭建

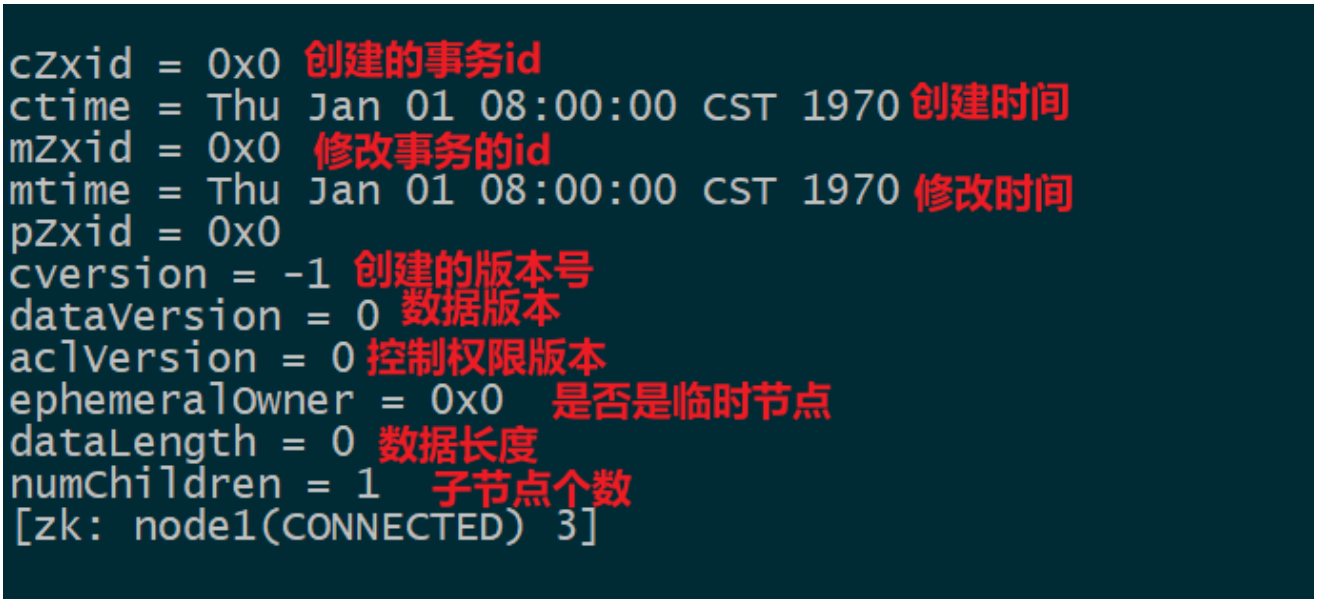
- zk集群在搭建部署的时候,通常选择 2n+1 奇数台.底层paxos算法支持(过半成功)

```
先上传文件
然后修改主机名hosts映射
防火墙关闭
时间同步
ssh免密登陆
jdk环境 解压文件
配置jdk环境变量
vim /etc/profile
输入以下内容
#set java environment
export JAVA_HOME=/export/server/jdk1.8.0_241
export PATH=:$PATH:$JAVA_HOME/bin

使环境变量生效
source /etc/profile

确认jdk生效
Java -version
```

```
zk具体安装部署
上传解压重命名,为了更新找不到路径
cd /export/server

tar zxvf zookeeper-3.4.6.tar.gz
mv zookeeper-3.4.6/ zookeeper

修改配置文件
#zk默认加载的配置文件是zoo.cfg 因此需要针对模板进行修改。保证名字正确。
```

```
cd zookeeper/conf
mv zoo_sample.cfg zoo.cfg

vi zoo.cfg

#修改
dataDir=/export/data/zkdata
#文件最后添加 2888心跳端口 3888选举端口
server.1=node1:2888:3888
server.2=node2:2888:3888
server.3=node3:2888:3888

myid
#在每台机器的dataDir指定的目录下创建一个文件 名字叫做myid
#myid里面的数字就是该台机器上server编号。server.N  N的数字就是编号
[root@node1 conf]# mkdir -p /export/data/zkdata
[root@node1 conf]# echo 1 >/export/data/zkdata/myid

把安装包同步到其他节点上

shell
cd /export/server
scp -r zookeeper/ node2:$PWD
scp -r zookeeper/ node3:$PWD

创建其他机器上myid和datadir目录
[root@node2 ~]# mkdir -p /export/data/zkdata
[root@node2 ~]# echo 2 > /export/data/zkdata/myid

[root@node3 ~]# mkdir -p /export/data/zkdata
[root@node3 ~]# echo 3 > /export/data/zkdata/myid

每台机器上单独启动服务
#在哪个目录执行启动命令 默认启动日志就生成当前路径下 叫做zookeeper.out

/export/server/zookeeper/bin/zkServer.sh  start|stop|status

#3台机器启动完毕之后 可以使用status查看角色是否正常。
#还可以使用jps命令查看zk进程是否启动。
[root@node3 ~]# jps
```

```
2034 Jps
1980 QuorumPeerMain  #看我，我就是zk的java进程


```



- 一键关闭脚本

```
[root@node1 ~]# vim stopZk.sh

#!/bin/bash
hosts=(node1 node2 node3)
for host in ${hosts[*]}
do
 ssh $host "/export/server/zookeeper/bin/zkServer.sh stop"
done

- 注意：关闭java进程时候 根据进程号 直接杀死即可就可以关闭。启动java进程的时候 需要JDK。
- shell程序ssh登录的时候不会自动加载/etc/profile 需要shell程序中自己加载。
```

- 一键启动脚本

```
[root@node1 ~]# vim startZk.sh

#!/bin/bash
hosts=(node1 node2 node3)
for host in ${hosts[*]}
do
 ssh $host "source /etc/profile;/export/server/zookeeper/bin/zkServer.sh start"
```

```
done
```

- 基本操作
  - 创建
    - 查看

```
[zk: node2(CONNECTED) 28] ls /itcast   #查看指定路径下有哪些节点
[aaa0000000000, bbbb0000000002, aaa0000000001]
[zk: node2(CONNECTED) 29] get /

zookeeper   itcast
[zk: node2(CONNECTED) 29] get /itcast  #获取znode的数据和stat属性信息
1111
cZxid = 0x200000003   #创建事务ID
ctime = Fri May 21 16:20:37 CST 2021 #创建的时间
mZxid = 0x200000003   #上次修改时事务ID
mtime = Fri May 21 16:20:37 CST 2021  #上次修改的时间
pZxid = 0x200000009
cversion = 3
dataVersion = 0  #数据版本号  只要有变化 就自动+1
aclVersion = 0
ephemeralOwner = 0x0   #如果为0 表示永久节点 如果是sessionID数字 表示临时节点
dataLength = 4   #数据长度
numChildren = 3  #子节点个数

更新节点
set path data
```

    - 删除节点

```
  [zk: node2(CONNECTED) 43] ls /itcast
  [aaa0000000000, bbbb0000000002, aaa0000000001]
  [zk: node2(CONNECTED) 44] delete /itcast/bbbb0000000002
  [zk: node2(CONNECTED) 45] delete /itcast
Node not empty: /itcast
  [zk: node2(CONNECTED) 46] rmr /itcast  #递归删除
```

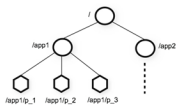
- quota==限制== 软性限制
  - 限制某个节点下面可以创建几个子节点 数据大小。
    - 超过限制，zk不会强制禁止操作 而是在日志中给出warn警告提示。

```
[zk: node2(CONNECTED) 47] create /itheima 111
```

```
Created /itheima
[zk: node2(CONNECTED) 49] listquota /itheima    #查看限制
absolute path is /zookeeper/quota/itheima/zookeeper_limits
quota for /itheima does not exist.

2021-05-21 16:54:42,697 [myid:3] - WARN  [CommitProcessor:3:DataTree@388] - Quota exceeded: /itheima count=6 limit=3
```

Zookeeper数据模型

一个标准的文件系统特征：
1、从/根目录开始
2、分为文件、文件夹
3、路径唯一



Q:客户端去操作zk集群分为几步？

1、客户端连接集群 建立会话
2、根据需求对zk目录树进行增删改查
3、客户端断开连接 会话结束

zookeeper文件系统特征
1、从结构上看，也是类似于标准文件系统的目录树结构
2、里面没有文件夹 文件之分，所有的节点统称znode
3、znode兼具文件夹、文件的特点
既像文件夹一样可以创建子目录
又想文件一样可以保存数据
4、路径也是从/根目录开始 唯一

znode的类型：

永久节点：客户端创建znode之后 将会一直存在 除非手动强制删除。

临时节点：客户端创建znode之后，只要断开连接，会话结束，znode就被删除

znode的类型：

永久序列化
永久非序列化
临时序列化
临时非序列化

4种

znode的序列化特性

非序列化：

已经存在了：/itcast/a

creat /itcast/a 创建失败 路径唯一

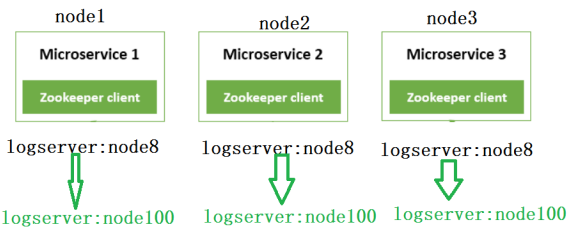
序列化：
zk集群会对创建的znode节点自动拼接数字序列号

已经存在了：/itcast/a

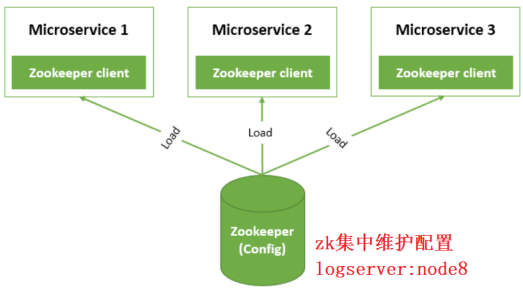
create /itcast/a
/itcast/a0000000000

create /itcast/a
/itcast/a0000000001

Zookeeper选举机制



需求：实现配置集中管理维护 自动更新同步的效果。



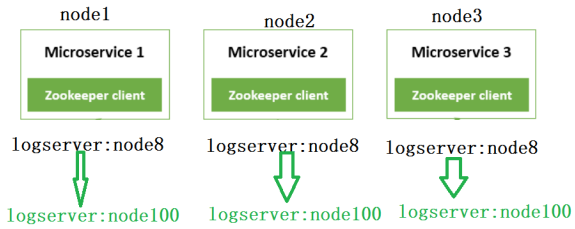
zk集中维护配置
logserver:node8

1、各个服务启动的时候去zk集群上读取配置文件信息
并且注册一个监听 监听znode的节点数据是否发生变化

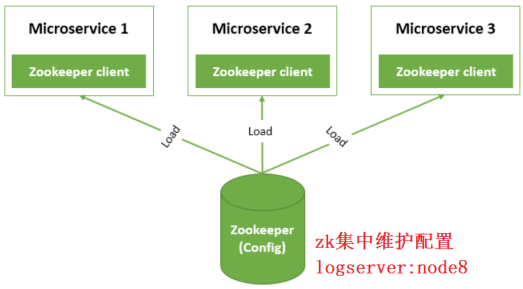
2、运维更新zk的节点数据（更新配置）--->触发监听--->通知给设置监听的客户端（各个服务）--->重新读取配置信息并且再次注册监听

核心使用监听机制

Zookeeper监听机制watch



需求：实现配置集中管理维护 自动更新同步的效果。



1、各个服务启动的时候去zk集群上读取配置文件信息
并且注册一个监听 监听znode的节点数据是否发生变化

2、运维更新zk的节点数据（更新配置）--->触发监听--->通知给设置监听的客户端（各个服务）--->重新读取配置信息并且再次注册监听

核心使用监听机制

- 监听机制
  - 监听实现需要几步?
  - 1、设置监听 2、执行监听 3、事件发生，触发监听 通知给设置监听的 回调callback
  - zk中的监听是什么?
    - 谁监听谁?
      - 客户端监听zk服务
    - 监听什么事?
      - 监听zk上目录树znode的变化情况。 znode增加了 删除了 增加子节点了 不见了
  - zk中监听实现步骤

```
#1、设置监听 然后zk服务执行监听
ls path [watch]
        没有watch 没有监听 就是查看目录下子节点个数
        有watch  有监听  设置监听子节点是否有变化
get path [watch]
        监听节点数据是否变化

e.g: get /itheima  watch
#2、触发监听
set /itheima 2222  #修改了被监听的节点数据 触发监听

#3、回调通知客户端
WATCHER::
```

```
14
15 WatchedEvent state:SyncConnected type:NodeDataChanged path:/itheima
```

- zk的监听特性
  - ==先注册 再触发==
  - ==一次性的监听==
  - ==异步通知==
  - ==通知是使用event事件来封装的==
    - path 、type

```
1 state:SyncConnected type:NodeDataChanged path:/itheima
2
3 type：发生了什么
4 path:哪里发生的
```

- zk中监听类型
  - 连接状态事件监听 系统自动触发 用户如果不关心可以忽略不计
  - 上述所讲的是用户自定义监听 主要监听zk目录树的变化 这类监听必须先注册 再监听。

- ==总结：zk的很多功能都是基于这个特殊文件系统而来的。==
  - ==特殊1：znode有临时的特性。==
  - ==特殊2：znode有序列化的特性。顺序==
  - ==特殊3：zk有监听机制 可以满足客户端去监听zk的变化。==
  - ==特殊4：在非序列化节点下，路径是唯一的。不能重名。==

zk典型应用

- zk核心特性:
  - 非序列节点唯一性
  - 序列的顺序性
  - 临时节点的自动删除
  - 坚挺机制
- 应用
  - 数据发布与订阅
  - 提供集群选举
  - 软负载均衡
- 分布式锁服务
  - 排他锁
    - 排他锁（Exclusive Locks），又被称为写锁或独占锁，如果事务T1对数据对象O1加上排他锁，那么整个加锁期间，只允许事务T1对O1进行读取和更新操作，其他任何事务都不能进行读或写。
  - 共享锁

- 共享锁（Shared Locks），又称读锁。如果事务T1对数据对象O1加上了共享锁，那么当前事务只能对O1进行读取操作，其他事务也只能对这个数据对象加共享锁，直到该数据对象上的所有共享锁都释放。