

第 3 章

Kafka的基本操作

本章开始实战演练 Kafka 的基本操作，比如操作 Zookeeper 集群和 Kafka 集群、管理主题、修改分区和副本等。

3.1 本章教学视频说明

视频内容：Zookeeper 和 Kafka 在不同模式下的启用/停止操作、Kafka 主题操作等。视频时长：18 分钟。视频截图见图 3-1。

3.2 操作 Zookeeper 集群

Zookeeper 和 Kafka 密切相关。在启动 Kafka 集群之前，需要先启动 Zookeeper 集群。在管理和协调 Kafka 代理（Broker）时，Zookeeper 起着至关重要的作用。

3.2.1 Zookeeper 的作用及背景

Zookeeper 是一个分布式应用程序，它是 Hadoop、Kafka、HBase 等这些分布式系统中不可或缺的重要组件，它为分布式系统提供协调服务。

1. 管理代码中变量的配置

在实际项目开发中，实现一个应用程序通常会动态配置一些参数，比如线程数、数据库连接地址、定时调度时间间隔等。这些参数会用一个配置文件进行保存，在代码中可引用这些配置文件。如果业务系统非常复杂、配置文件非常多，多台服务器上的应用程序都依赖这些配置文件，则使用配置文件这种方式就略显吃力。

进阶一点的做法是使用数据库：将配置文件的内容存储到数据库，所有依赖这些配置的应用程序都可以通过访问数据库来获取配置信息。可这样会有一个问题：随着访问数据库的应用程序越来越多，对数据库的要求会越来越高。需要保证配置信息的高可用，不能因为数据库故障导致所有的应用程序瘫痪。

可以在此基础上进行改进——组建集群。集群虽然满足了配置信息的高可用，但是配置信息的一致性却难以保证。因此，需要一种拥有一致性协议的服务。

Zookeeper 的出现，很好地弥补这一空白。Zookeeper 使用 Zab 协议来提供一致性服务。现在很多分布式开源系统（比如 Hadoop、Kafka、HBase 等）都使用 Zookeeper 来维护和管理配置。



提示：

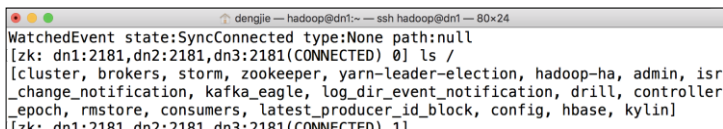
Zookeeper Atomic Broadcast 简称 Zab，该协议是实现 Zookeeper 一致性服务的核心。实现 Zab 协议的算法充分考虑了高吞吐量、低延迟、稳定、简单等特性。

2. 设置命名服务

命名服务是分布式系统中比较常见的一类应用场景。

在分布式系统中，通过使用命名服务，客户端应用程序可以根据指定名字来获取元数据信息，其中较为常见的是分布式服务框架中的服务地址列表。

在 Kafka 系统中，通过调用 Zookeeper 系统的应用接口来创建一个全局唯一的路径地址，在这些路径地址中存储了 Kafka 的主题名称、Kafka 主题分区与副本、Kafka 代理信息等内容，如图 3-2 所示。



```
WatchedEvent state:SyncConnected type:None path:null
[zk: dn1:2181,dn2:2181,dn3:2181(CONNECTED) 0] ls /
[cluster, brokers, storm, zookeeper, yarn-leader-election, hadoop-ha, admin, isr
_change_notification, kafka_eagle, log_dir_event_notification, drill, controller
_epoch, rmstore, consumers, latest_producer_id_block, config, hbase, kylin]
[zk: dn1:2181,dn2:2181,dn3:2181(CONNECTED) 1]
```

图 3-2 Zookeeper 系统中的命令空间

3. 提升系统的可用性和安全性

在分布式系统中，为了提高系统的可用性，集群中每一个节点应部署相同的进程服务。但如果一个客户端请求需要这些进程服务都参与，则这些进程服务之间的相互协调和编程实现就会变得困难。可如果只使用一个服务进程，则存在单点问题。



说明：

单点问题是分布式系统中应重点考虑的问题。在一个系统中，如果只有一个主进程来处理客户端请求，就会存在单点问题。即，当这个主进程出现异常变得不可用时，整个系统将不能正常运行。

所以一般情况下，在分布式系统中都会做高可用（High Availability, HA），即存在一个主进程和一个备用进程。

面对这种复杂的场景，通常有一种做法——使用分布式锁：

- 在某一时刻，只让一个进程服务处理请求，该服务进程处理完请求后会释放锁；
- 若在处理请求时出现异常，在释放锁的同时会将故障转移到其他可用的进程服务。

使用分布式锁的做法在分布式系统中比较常见，如 Leader 选举。Hadoop 的 NameNode Active、HBase Master、Kafka 所有分区（Partition）的 Leader 选举都是采用的这种机制。

4. 管理 Kafka 集群

分布式集群（比如 Kafka 集群）在运行过程中，可能会因为服务器硬件故障、软件故障、网络故障，导致集群中的某些代理节点时而处于离线状态，时而处于上线状态。

另外，在维护 Kafka 集群时，可能会添加新的代理节点，也可能淘汰老的代理节点。此时，Kafka 集群中的其他代理节点应感知到这样的变化，并能根据变化采取对应的决策。

在 Kafka 系统中，每一个代理节点会在 Zookeeper 系统中注册一个

监听器（Watcher），在会话（Session）期间不断地更新当前代理节点的状态信息。

3.2.2 实例 8：单机模式启动 Zookeeper 系统

Zookeeper 有两种启动模式：① 单机模式（Standalone），② 分布式模式（Distributed）。

实例描述

在一台 Linux 操作系统主机上安装一个 Zookeeper 系统，执行启动命令并观察结果。

单机模式配置比较简单，在 \$ZK_HOME/conf/zoo.cfg 文件中配置代码 3-1 所示的内容。

代码 3-1 单机模式配置

```
# 通信时间限制
syncLimit=5
# 元数据存储路径，推荐使用独立磁盘来存储
dataDir=/data/soft/new/zookeeper/data
# 客户端连接端口号
clientPort=2181
# 处理客户端最大连接数
maxClientCnxns=60
# 需要保留的文件数目
autopurge.snapRetainCount=3
# 日志清理频率，单位是小时。如果填写整数 0，则表示不开启自动清理功能
autopurge.purgeInterval=1
```

1. 配置全局变量

为了能够全局使用 Zookeeper 命令，可以在 Linux 操作系统中配置 Zookeeper 环境变量。

具体配置内容如下。

```
# 在 ~/.bash_profile 文件中进行配置
[hadoop@dn1 ~]$ vi ~/.bash_profile
```

```
# 添加如下内容
export ZK_HOME=/data/soft/new/zookeeper
export PATH=$PATH:$ZK_HOME/bin

# 保存并退出
```

然后，使用 `source` 命令使配置的 Zookeeper 环境变量立即生效。具体操作命令如下。

```
# 使用 source 命令
[hadoop@dn1 ~]$ source ~/.bash_profile
```

2. 单机模式启动 Zookeeper

在 Linux 操作系统中，可使用 Zookeeper 的 `zkServer.sh` 脚本来启动 Zookeeper 系统。

具体操作命令如下。

```
# 使用 start 参数启动
[hadoop@dn1 ~]$ zkServer.sh start
```

执行上述命令后，在控制台中会出现启动成功的日志信息，见下方代码。

```
JMX enabled by default
Using config: /data/soft/new/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

3. 验证单机模式的 Zookeeper 是否正常

可以通过 Linux 的 `jps` 命令来查看 Zookeeper 进程 `QuorumPeerMain` 是否存在，还可以通过 Zookeeper 的 `status` 命令或 `telnet` 命令来查看。具体操作命令如下。

代码 3-2 验证 Zookeeper 服务是否正常

```
# 使用 jps 命令查看
```

```
8210 QuorumPeerMain
```

```
# 使用 status 命令
[hadoop@dn1 ~]$ zkServer.sh status
JMX enabled by default
Using config: /data/soft/new/zookeeper/bin/../conf/zoo.cfg
Mode: standalone

# 使用 telnet 命令，访问 Zookeeper 的客户端端口 2181
[hadoop@dn1 ~]$ telnet dn1 2181
Trying 10.211.55.5...
Connected to dn1.

Escape character is '^]'.

^]
telnet>
```

3.2.3 实例 9：单机模式关闭 Zookeeper 系统

单机模式关闭 Zookeeper 有两种方法：

- 通过 `jps` 命令获取 Zookeeper 进程的 PID 值，然后使用 `kill` 命令关闭 Zookeeper 进程；
- 使用 `zkServer.sh` 脚本，通过 `stop` 参数来关闭 Zookeeper 进程。



提示：

Process Identification 简称 PID，表示每个进程的唯一编号。在进程启动时，由 Linux 操作系统随机分配，它并不代表专门的进程。

在进程运行期间 PID 的值不会改变，但是进程关闭后再启动时，会产生一个新的 PID 值，老的 PID 值会被系统回收。

实例描述：

(1) 执行 `kill` 命令关闭 Zookeeper 系统；(2) 使用 `stop` 参数关闭 Zookeeper 系统。

具体操作步骤如下。

1. 使用 kill 命令关闭 Zookeeper

在 Linux 操作系统中，先通过 `jps` 命令获取 PID 值，然后执行 `kill` 命令。具体操作如下。

```
# 使用 jps 命令
[hadoop@dn1 ~]$ jps
8210 QuorumPeerMain

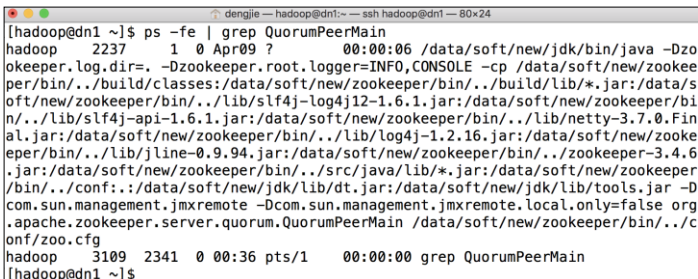
#使用 kill 命令
[hadoop@dn1 ~]$ kill -9 8210
```

在实际场景中，可能会出现这样一种情况：Zookeeper 的 `QuorumPeerMain` 进程是一直存在的，但是执行 Linux 操作系统的 `jps` 命令后，却无法找到对应的 PID 值。

这时可以通过使用 Linux 操作系统的 `grep` 命令来获取 PID 值。具体操作命令如下。

```
# 使用 grep
[hadoop@dn1 ~]$ ps -fe | grep QuorumPeerMain
```

执行 `grep` 命令后，在 Linux 控制台中会打印出 Zookeeper 的进程信息，如图 3-3 所示。



```
[hadoop@dn1 ~]$ ps -fe | grep QuorumPeerMain
hadoop 2237 1 0 Apr09 ? 00:00:06 /data/soft/new/jdk/bin/java -Dzookeeper.log.dir=. -Dzookeeper.root.logger=INFO,CONSOLE -cp /data/soft/new/zookeeper/bin/./build/classes:/data/soft/new/zookeeper/bin/./build/lib/*:/data/soft/new/zookeeper/bin/./lib/slf4j-log4j12-1.6.1.jar:/data/soft/new/zookeeper/bin/./lib/slf4j-api-1.6.1.jar:/data/soft/new/zookeeper/bin/./lib/netty-3.7.0.Final.jar:/data/soft/new/zookeeper/bin/./lib/log4j-1.2.16.jar:/data/soft/new/zookeeper/bin/./lib/jline-0.9.94.jar:/data/soft/new/zookeeper/bin/./zookeeper-3.4.6.jar:/data/soft/new/zookeeper/bin/./src/java/lib/*:/data/soft/new/zookeeper/bin/./conf:./data/soft/new/jdk/lib/dt.jar:/data/soft/new/jdk/lib/tools.jar -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.local.only=false org.apache.zookeeper.server.quorum.QuorumPeerMain /data/soft/new/zookeeper/bin/./conf/zoo.cfg
hadoop 3109 2341 0 00:36 pts/1 00:00:00 grep QuorumPeerMain
[hadoop@dn1 ~]$
```

图 3-3 使用 `grep` 命令获取 Zookeeper 进程信息

2. 使用 stop 参数关闭 Zookeeper

Zookeeper 提供了一种简便的方式来关闭进程：使用 `zkServer.sh` 脚本，通过输入 `stop` 参数来进行关闭。

具体操作命令如下。

```
# 使用 stop 参数
[hadoop@dn1 ~]$ zkServer.sh stop
JMX enabled by default
Using config: /data/soft/new/zookeeper/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

3.2.4 实例 10：分布式模式启动 Zookeeper 集群

与单机模式（Standalone）的 Zookeeper 相比较，分布式模式下的 Zookeeper 需要在\$ZK_HOME/conf/zoo.cfg 文件中配置一些额外的属性。具体内容见代码 3-3。

代码 3-3 分布式模式配置

```
# 服务器与客户端之间维持的心跳时间
tickTime=20000

# 集群中 Follower 服务器与 Leader 服务器之间最大的初始化连接数
initLimit=10

# 同步通信时间间隔
syncLimit=5

# 元数据存储路径，推荐使用独立的磁盘来存储
dataDir=/zookeeper/zkdata

# 事物日志存储的路径，推荐使用独立的磁盘来存储
dataLogDir=/zookeeper/logs

# 客户端连接服务器的端口号
clientPort=2181

# 配置集群节点信息，序号要保证唯一
server.1= dn1:2888: 3888
server.2= dn2:2888:3888
server.3= dn3:2888:3888

# 设置客户端最大连接数
maxClientCnxns=300
```



```
# 需要保留的文件数目
autopurge.snapRetainCount=3
# 日志清理频率，单位是小时。如果填写整数 0，则表示不开启自动清理功能
autopurge.purgeInterval=1
```

在配置 Zookeeper 集群时，需要在元数据存储路径中新建一个 myid 的文本文件，在该文本文件中填写一个正整数，并且数字要与配置文件中的内容一致。例如，在配置文件中配置的是 server.1= dn1:2888:3888 信息，则在主机 dn1 中的/zookeeper/zkdata/myid 文本文件中填写数字 1。其他节点以此类推。

Zookeeper 系统不包含分布式管理脚本，因此，需要到每一台 Zookeeper 节点启动 Zookeeper 进程。这样维护起来很不方便，因此需要自行编写一个分布式管理脚本。

实例描述

创建一个扩展名为 sh 的 Shell 源代码文件，将其作为分布式管理脚本。该模块的功能是管理 Zookeeper 集群的启动、查看状态、停止、重启。执行该脚本，并观察操作结果。

1. 编写分布式管理脚本

基于 Zookeeper 系统现有的管理脚本二次开发一个分布式管理脚本（zks-daemons.sh）。具体实现内容见代码 3-4。

代码 3-4 分布式管理脚本

```
#!/bin/bash

# 设置 Zookeeper 集群节点地址
hosts=(dn1 dn2 dn3)

# 获取输入 Zookeeper 命令参数
cmd=$1

# 执行分布式管理命令
function zookeeper()
{
```

```
for i in ${hosts[@]}
do
    ssh hadoop@$i "source ~/.bash_profile;zkServer.sh
$cmd;echo Zookeeper node is $i, run $cmd command. " &
    sleep 1
done
}

# 判断输入的 Zookeeper 命令参数是否有效
case "$1" in
    start)
        zookeeper
        ;;
    stop)
        zookeeper
        ;;
    status)
        zookeeper
        ;;
    start-foreground)
        zookeeper
        ;;
    upgrade)
        zookeeper
        ;;
    restart)
        zookeeper
        ;;
    print-cmd)
        zookeeper
        ;;
    *)
        echo "Usage:                                $0
{start|start-foreground|stop|restart|status|upgrade|print-cmd}"
        RETVAL=1
esac
```

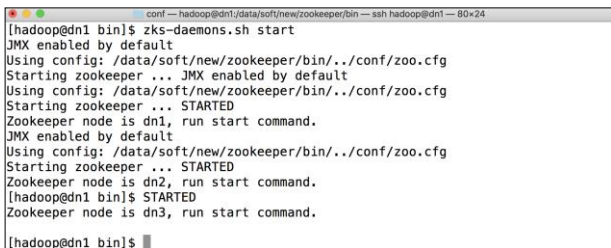
2. 分布式模式启动 Zookeeper 集群

执行编写好的分布式管理脚本来启动 Zookeeper 集群。具体操作命

令如下。

```
# 输入 start 命令来启动 Zookeeper 集群
[hadoop@dn1 bin]$ zks-daemons.sh start
```

执行上述启动命令后，Linux 控制台会打印出 Zookeeper 启动日志信息，如图 3-4 所示。



```
conf — hadoop@dn1:/data/soft/new/zookeeper/bin — ssh hadoop@dn1 — 80x24
[hadoop@dn1 bin]$ zks-daemons.sh start
JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Starting zookeeper ... JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Starting zookeeper ... STARTED
Zookeeper node is dn1, run start command.
JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Starting zookeeper ... STARTED
Zookeeper node is dn2, run start command.
[hadoop@dn1 bin]$ STARTED
Zookeeper node is dn3, run start command.
[hadoop@dn1 bin]$ █
```

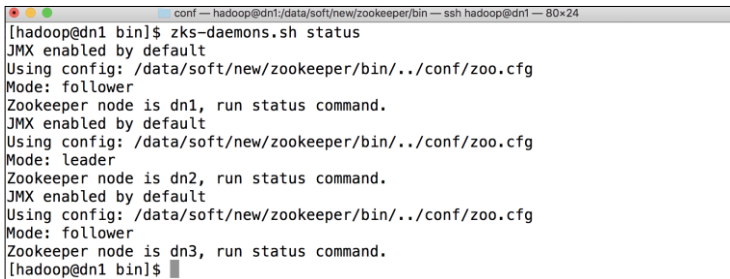
图 3-4 分布式模式启动 Zookeeper

3. 验证 Zookeeper 集群

可以执行 Zookeeper 的 status 命令来验证 Zookeeper 集群的运行状态。具体操作命令如下。

```
# 使用 Zookeeper 的 status 命令
[hadoop@dn1 bin]$ zks-daemons.sh status
```

执行完 status 命令后，Linux 控制台会打印出 Zookeeper 集群各个节点的角色信息，如图 3-5 所示。



```
conf — hadoop@dn1:/data/soft/new/zookeeper/bin — ssh hadoop@dn1 — 80x24
[hadoop@dn1 bin]$ zks-daemons.sh status
JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Mode: follower
Zookeeper node is dn1, run status command.
JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Mode: leader
Zookeeper node is dn2, run status command.
JMx enabled by default
Using config: /data/soft/new/zookeeper/bin/./conf/zoo.cfg
Mode: follower
Zookeeper node is dn3, run status command.
[hadoop@dn1 bin]$ █
```

图 3-5 分布式模式 Zookeeper 集群状态

3.1	本章教学视频说明	1
3.2	操作 Zookeeper 集群	1
3.2.1	Zookeeper 的作用及背景	1
3.2.2	实例 8：单机模式启动 Zookeeper 系统	4
3.2.3	实例 9：单机模式关闭 Zookeeper 系统	6
3.2.4	实例 10：分布式模式启动 Zookeeper 集群	8