

1-1 kafka开发环境搭建

1 kafka开发环境

1.1 安装Java环境

1.1.1 下载linux下的安装包

1.1.2 解压安装包jdk-8u202-linux-x64.tar.gz

1.1.3 将解压后的文件移到/usr/lib目录下

1.1.4 配置java环境变量

1.1.5 执行命令使修改立即生效

1.1.6 测试安装是否成功

1.2 Kafka的安装部署

1.2.1 下载kafka

1.2.2 安装kafka

1.2.3 配置和启动zookeeper

1.2.4 启动和停止kafka

1.3 kafka的基本操作

1.3.1 创建topic

1.3.2 查看topic

1.3.3 查看topic属性

13.4 消费消息

1.3.5 发送消息

1.4 kafka-topics.sh 使用方式

1.4.1 查看帮助--help

1.4.2 副本数量不能大于broker的数量

1.4.3 创建主题--create

1.4.4 查看broker上所有的主题 --list

1.4.5 查看指定主题 topic 的详细信息 --describe

1.4.6 修改主题信息 --alter （增加主题分区数量）

1.4.7 删除主题 topic --delete

腾讯课堂零声教育：<https://ke.qq.com/course/420945?tuin=137bb271>

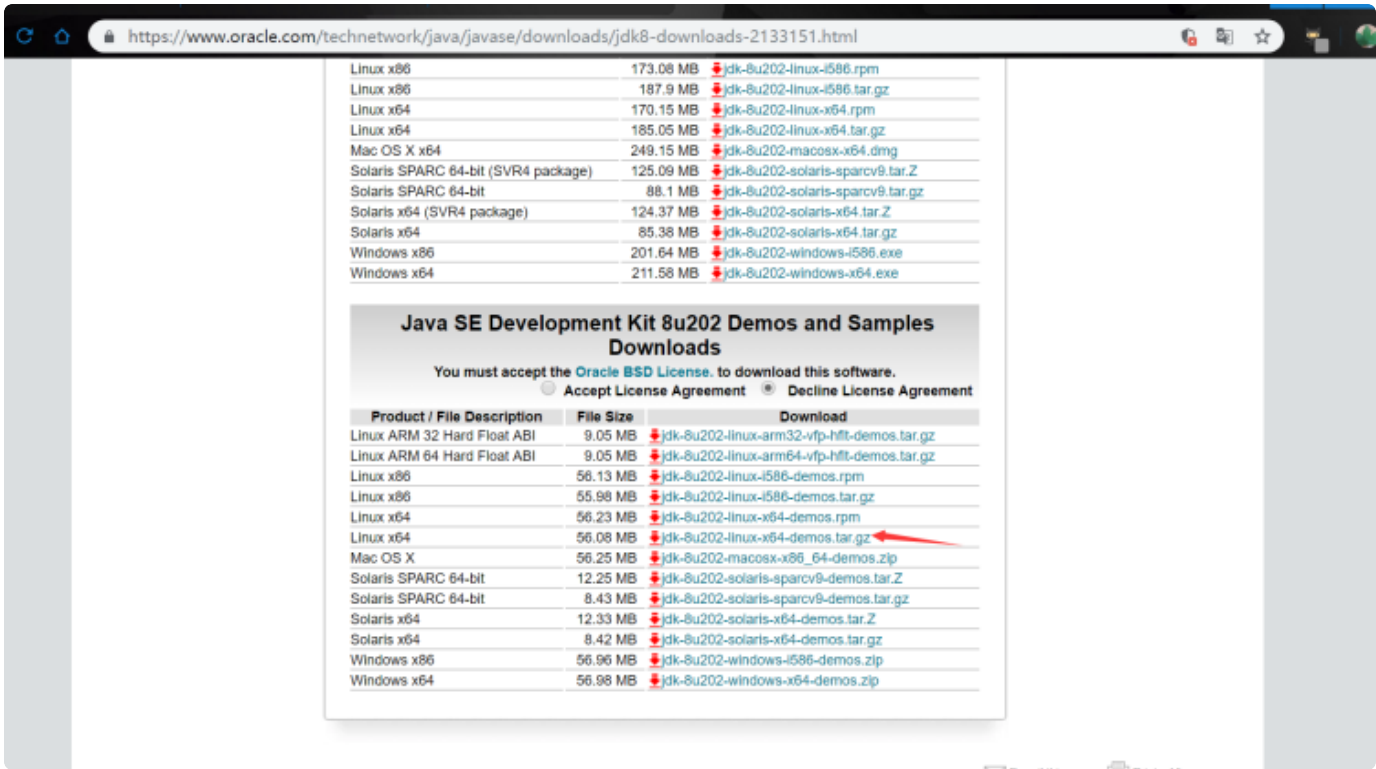
网页版本：<https://www.yuque.com/docs/share/fa589923-4368-4bcd-97ad-5e38d7207dee?#>
《1-1 kafka开发环境搭建》

1 kafka开发环境

1.1 安装Java环境

1.1.1 下载linux下的安装包

登陆网址<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



下载完成后，Linux默认下载位置在当前目录下的Download或下载文件夹下，通过命令cd ~/Downloads或cd ~/下载即可查看到对应的文件。

1.1.2 解压安装包jdk-8u202-linux-x64.tar.gz

Shell | 复制代码

```
1 tar -zxvf jdk-8u291-linux-x64.tar.gz
```

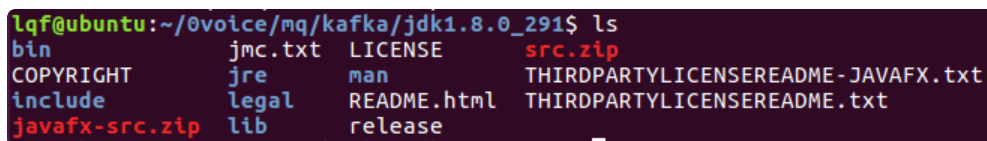
解压后的文件夹为**jdk1.8.0_291**

进入文件夹和查看文件

Shell | 复制代码

```
1 cd jdk1.8.0_291
2 ls
```

可以看到bin目录



```
lqf@ubuntu:~/0voice/mq/kafka/jdk1.8.0_291$ ls
bin          jmc.txt     LICENSE     src.zip
COPYRIGHT    jre         man          THIRDPARTYLICENSEREADME-JAVAFX.txt
include      legal       README.html THIRDPARTYLICENSEREADME.txt
javafx-src.zip lib         release
```

1.1.3 将解压后的文件移到/usr/lib目录下

在/usr/bin目录下新建jdk目录

Shell | 复制代码

```
1 sudo mkdir /usr/lib/jdk
```

将解压的jdk文件移动到新建的/usr/lib/jdk目录下

Shell | 复制代码

```
1 sudo mv jdk1.8.0_291 /usr/lib/jdk/
2
```

执行命令后可到 **usr/lib/jdk** 目录下查看是否移动成功。

1.1.4 配置java环境变量

这里是将环境变量配置在etc/profile，即为所有用户配置JDK环境。
使用命令打开/etc/profile文件

```
1  sudo vim /etc/profile
```

在末尾添加以下几行文字：

```
1  #set java env
2  export JAVA_HOME=/usr/lib/jdk/jdk1.8.0_291
3  export JRE_HOME=${JAVA_HOME}/jre
4  export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
5  export PATH=${JAVA_HOME}/bin:$PATH
```

```
#set java env
export JAVA_HOME=/usr/lib/jdk/jdk1.8.0_291
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
```

1.1.5 执行命令使修改立即生效

```
1  source /etc/profile
```

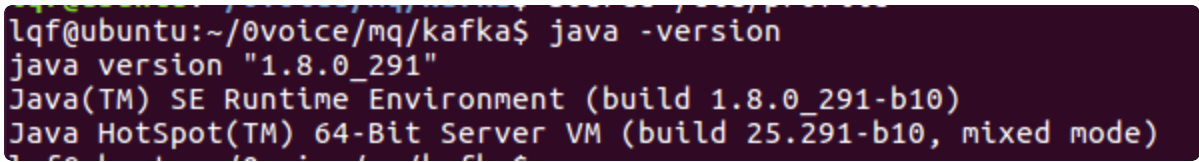
1.1.6 测试安装是否成功

在终端输入，出现版本号说明安装成功。

▼ Shell 复制代码

```
1 java -version
```

结果如图：



1.2 Kafka的安装部署

1.2.1 下载kafka

▼ Shell 复制代码

```
1 wget https://archive.apache.org/dist/kafka/2.0.0/kafka_2.11-2.0.0.tgz
```

1.2.2 安装kafka

我们下载的kafka是已经编译好的程序，只需要解压即可得到执行程序。

▼ Shell 复制代码

```
1 tar -zxvf kafka_2.11-2.0.0.tgz
```

进入kafka目录，以及查看对应的文件和目录

▼ Shell 复制代码

```
1 cd kafka_2.11-2.0.0
2 ls
```

`bin` `config` `libs` `LICENSE` `NOTICE` `site-docs`

bin: 为执行程序

config: 为配置文件

libs: 为库文件

1.2.3 配置和启动zookeeper

下载的kafka程序里自带了zookeeper，kafka自带的Zookeeper程序脚本与配置文件名与原生Zookeeper稍有不同。

kafka自带的Zookeeper程序使用`bin/zookeeper-server-start.sh`，以及`bin/zookeeper-server-stop.sh`来启动和停止Zookeeper。

kafka依赖于zookeeper来做master选举一起其他数据的维护。

- 启动zookeeper: `zookeeper-server-start.sh`
- 停止zookeeper: `zookeeper-server-stop.sh`

在config目录下，存在一些配置文件

```
1  zookeeper.properties
2  server.properties
```

Shell

复制代码

所以我们可以通过下面的脚本来启动zk服务，当然，也可以自己独立搭建zk的集群来实现。这里我们直接使用kafka自带的zookeeper。

启动zookeeper

```
1  前台运行: sh zookeeper-server-start.sh ../config/zookeeper.properties
2  后台运行: sh zookeeper-server-start.sh -daemon
    ../config/zookeeper.properties
```

Shell

复制代码

默认端口为: 2181，可以通过命令`lsyf -i:2181`查看zookeeper是否启动成功。

1.2.4 启动和停止kafka

- 修改server.properties（在config目录），增加zookeeper的配置，这里只是本地的配置，如果是另一台机器运行zookeeper，要配置对应的ip地址。

▼ Shell 复制代码

```
1  zookeeper.connect=localhost:2181
```

- 启动kafka

▼ Shell 复制代码

```
1  sh kafka-server-start.sh -daemon ../config/server.properties
```

默认启动端口9092

- 停止kafka

▼ Shell 复制代码

```
1  sh kafka-server-stop.sh -daemon ../config/server.properties
```

1.3 kafka的基本操作

1.3.1 创建topic

▼ Shell 复制代码

```
1  sh kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test
```

Replication-factor 表示该topic需要在不同的broker中保存几份，这里设置成1，表示在两个broker中保存两份

Partitions分区数

1.3.2 查看topic

Shell | [复制代码](#)

```
1 sh kafka-topics.sh --list --zookeeper localhost:2181
```

1.3.3 查看topic属性

Shell | [复制代码](#)

```
1 sh kafka-topics.sh --describe --zookeeper localhost:2181 --topic test
```

13.4 消费消息

Shell | [复制代码](#)

```
1 sh kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic  
test  
2 --from-beginning
```

1.3.5 发送消息

Shell | [复制代码](#)

```
1 sh kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic test
```

1.4 kafka-topics.sh 使用方式

围绕创建、修改、删除以及查看等功能。

1.4.1 查看帮助--help

/bin目录下的每一个脚本工具，都有着众多的参数选项，不可能所有命令都记得住，这些脚本都可以使用 --help 参数来打印列出其所需的参数信息。


```

1  $ sh kafka-topics.sh --help
2  Command must include exactly one action: --list, --describe, --create, --
   alter or --delete
3  Option                                Description
4  -----                                -
5  --alter                                Alter the number of partitions,
6                                         replica assignment, and/or
7                                         configuration for the topic.
8  --config <String: name=value>         A topic configuration override
   for the
9                                         topic being created or
10 altered.The                            following is a list of valid
11                                         configurations:
12                                         cleanup.policy
13                                         compression.type
14                                         delete.retention.ms
15                                         file.delete.delay.ms
16                                         flush.messages
17                                         flush.ms
18                                         follower.replication.throttled.
19                                         replicas
20                                         ....省略
21

```

下面我们挑选其中使用最为频繁且重要的参数进行说明，以及其中一些坑进行标明。

1.4.2 副本数量不能大于broker的数量

kafka 创建主题的时候其副本数量不能大于broker的数量，否则创建主题 topic 失败.

```
sh kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 1  
--topic test1
```

详细报错信息见下图，其中红色框注明了其副本数量已经超过了可使用的 kafka broker 数量。

Error while executing topic command : **Replication factor: 2 larger than available brokers: 1.**[2021-07-12 17:16:32,476] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: Replication factor: 2 larger than available brokers: 1.
(kafka.admin.TopicCommand\$)

1.4.3 创建主题--create

创建主题时候，有3个参数是必填的，分别是 --partitions（分区数量）、--topic（主题名）、--replication-factor（复制系数）， 同时还需使用 --create 参数表明本次操作是想要创建一个主题操作。

```
1 sh kafka-topics.sh --create --zookeeper localhost:2181 --replication-  
factor 1 --partitions 1 --topic test1
```

返回：

Created topic "test1".

此时主题 test1 就已经创建了。另外在创建主题的时候，还可以附加以下两个选项：--if-not-exists 和 --if-exists . 第一个参数表明仅当该主题不存在时候，创建； 第二个参数表明当修改或删除这个主题时候， 仅在该主题存在的时候去执行操作。

1.4.4 查看broker上所有的主题 --list

```
1 sh kafka-topics.sh --list --zookeeper localhost:2181
```

返回: test1

其中test1便为我们创建的主题。

1.4.5 查看指定主题 topic 的详细信息 --describe

该参数会将该主题的所有信息一一列出打印出来, 比如分区数量、副本系数、领导者等待。

```
1 sh kafka-topics.sh --describe --zookeeper localhost:2181 --topic test1
```

返回:

Topic:test1 PartitionCount:1 ReplicationFactor:1 Configs:

Topic: test1 Partition: 0 Leader: 0 Replicas: 0 Isr: 0

1.4.6 修改主题信息 --alter (增加主题分区数量)

```
1 sh kafka-topics.sh --zookeeper localhost:2181 --topic test1 --alter --  
  partitions 2  
2  
3 WARNING: If partitions are increased for a topic that has a key, the  
  partition logic or ordering of the messages will be affected  
4 Adding partitions succeeded!  
5
```

可以看到已经成功的将主题的分区数量从1修改为了2。

如果去修改一个不存在的topic信息会怎么样？比如修改主题 test2，当前这主题是不存在的。

```
▼ Shell | 复制代码

1 sh kafka-topics.sh --zookeeper localhost:2181 --topic test2 --alter --
  partitions 2
2
3 Error while executing topic command : Topic test2 does not exist on ZK
  path localhost:2181
4 [2021-07-12 17:28:59,253] ERROR java.lang.IllegalArgumentException: Topic
  test2 does not exist on ZK path localhost:2181
5   at kafka.admin.TopicCommand$.alterTopic(TopicCommand.scala:123)
6   at kafka.admin.TopicCommand$.main(TopicCommand.scala:65)
7   at kafka.admin.TopicCommand.main(TopicCommand.scala)
8   (kafka.admin.TopicCommand$)
9
```

注意：不要使用 `--alter` 去尝试减少分区数量，如果非要减少分区数量，只能删除整个主题 topic，然后重新创建

1.4.7 删除主题 topic `--delete`

```
▼ Shell | 复制代码

1 sh kafka-topics.sh --zookeeper localhost:2181 --delete --topic test1
2
3 Topic test1 is marked for deletion.
4 Note: This will have no impact if delete.topic.enable is not set to true.
```

日志信息提示，主题 test1 已经被标记删除状态，但是若 `delete.topic.enable` 没有设置为 `true`，则不会有任何作用。

启动生产者：sh kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic test1

启动消费者：sh kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic test1 --from-beginning

发现此时还是可以发送消息和接收消息。

如果要支持能够删除主题的操作，则需要在 /bin 的同级目录 /config目录下的文件server.properties 中，修改配置delete.topic.enable=true（如果置为false，则kafka broker 是不允许删除主题的）。

```
# topic setting
delete.topic.enable=true
```

需要server.properties中设置delete.topic.enable=true否则只是标记删除或者直接重启。

重启kafka

停止：sh kafka-server-stop.sh -daemon ../config/server.properties

启动：sh kafka-server-start.sh -daemon ../config/server.properties

再次删除

```
1 sh kafka-topics.sh --zookeeper localhost:2181 --delete --topic test1
```