

加餐 | 搭建开发环境、阅读源码方法、经典学习资料大揭秘

2019-08-31 胡夕



你好，我是胡夕。

截止到现在，专栏已经更新了**38**讲，你掌握得怎么样了呢？如果暂时掌握得不是很好，也没有关系，慢慢来，有问题记得在留言区留言，我们一起讨论。

今天，我们来聊点儿不一样的。我总结了**3**个讨论热度很高的话题，现在一一来为你“揭秘”。

1. 如何搭建**Kafka**开发环境？很多人对于编译和调试**Kafka**饶有兴致，却苦于无从下手。今天我就给你完整地演示一遍搭建**Kafka**开发环境的过程。
2. 如何阅读**Kafka**源码？我曾经在专栏[第1讲](#)提到过我自己阅读**Kafka**源码的经历，后来我收到很多留言，问我是如何阅读的，今天，我就跟你分享一些阅读**Kafka**源代码的比较好的法则或者技巧。
3. **Kafka**的学习资料。幸运的是，我在这方面还是有过一些总结的，今天我会毫无保留地把资料全部分享给你。

Kafka开发环境搭建

现在，我先来回答第1个问题：如何搭建**Kafka**开发环境。我以**IDEA**为例进行说明，**Eclipse**应该也是类似的。

第1步：安装Java和Gradle

要搭建**Kafka**开发环境，你必须安装好**Java**和**Gradle**，同时在**IDEA**中安装**Scala**插件。你最好

把Java和Gradle环境加入到环境变量中。

第2步：下载Kafka的源码

完成第1步之后，下载Kafka的源码，命令如下：

```
$ cd Projects  
$ git clone https://github.com/apache/kafka.git
```

这个命令下载的是Kafka的trunk分支代码，也就是当前包含所有已提交Patch的最新代码，甚至比Kafka官网上能够下载到的最新版本还要超前很多。值得注意的是，如果你想向Kafka社区贡献代码，通常要以trunk代码为主体进行开发。

第3步：下载Gradle的Wrapper程序套件

代码下载完成之后，会自动创建一个名为kafka的子目录，此时需要进入到该目录下，执行下面的这条命令，主要目的是下载Gradle的Wrapper程序套件。

```
$ gradle  
Starting a Gradle Daemon (subsequent builds will be faster)  
  
> Configure project :  
Building project 'core' with Scala version 2.12.9  
Building project 'streams-scala' with Scala version 2.12.9  
  
Deprecated Gradle features were used in this build, making it incompatible with Gradle 6.0.  
Use '--warning-mode all' to show the individual deprecation warnings.  
See https://docs.gradle.org/5.3/userguide/command_line_interface.html#sec:command_line_warning
```

第4步：将Kafka源码编译打包成Jar文件

现在，你可以运行下列命令，将Kafka源码编译打包成Jar文件：

```
./gradlew clean releaseTarGz
```

通常你需要等待一段时间，经过一系列操作之后，比如Gradle拉取依赖Jar包、编译Kafka源码、打包等，你可以在core的build/distributions下面找到生成的tgz包：kafka_2.12-2.4.0-SNAPSHOT。解压之后，这就是一个可以正常启动运行的Kafka环境了。

第5步：把Kafka源码工程导入到IDEA中

这也是搭建开发环境的最后一步。你可以先执行下面的命令去创建IDEA项目所需要的项目文件：

```
$ ./gradlew idea #如果你用的是Eclipse，执行./gradlew eclipse即可
```

接着，你需要打开IDEA，选择“打开工程”，然后再选择kafka目录即可。

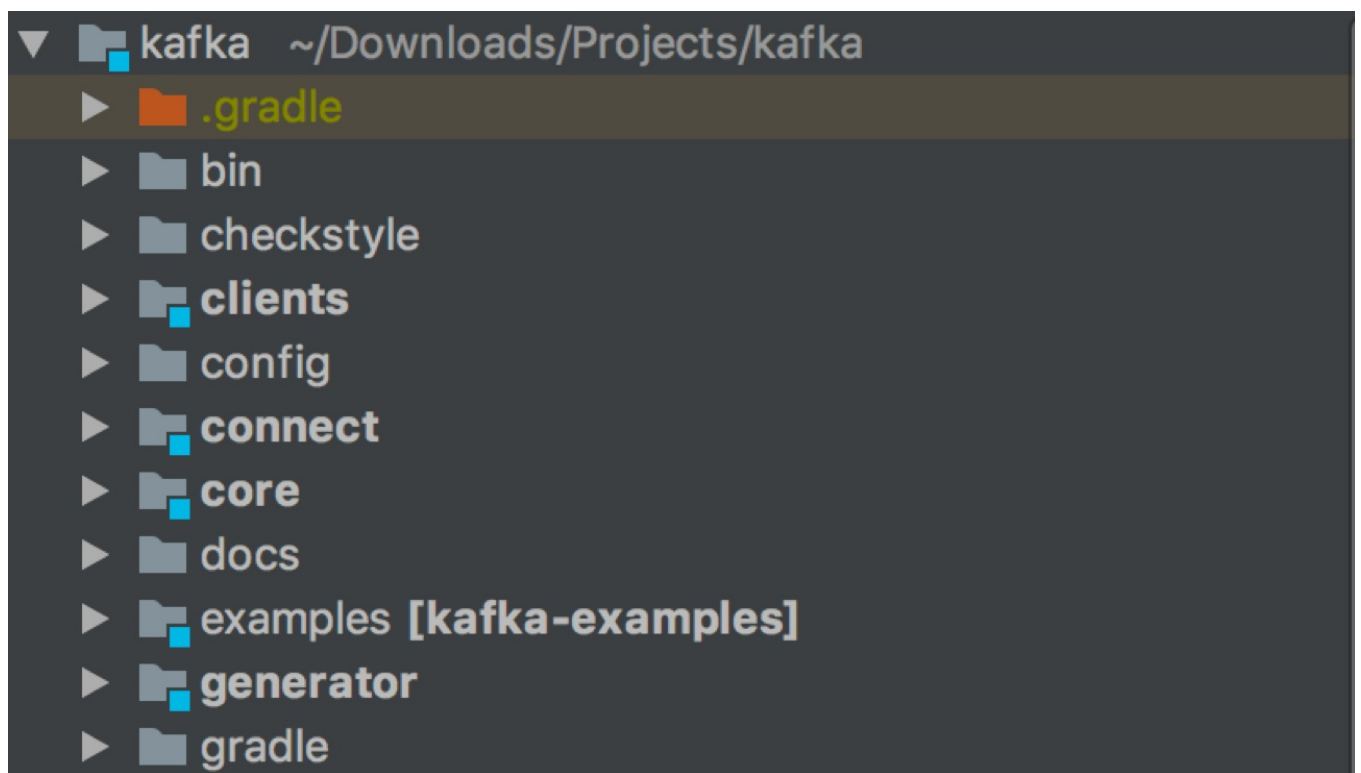
至此，我们就在IDEA中搭建了Kafka源码环境。你可以打开Kafka.scala文件，右键选择“运行”，这时，你应该可以看到启动Kafka Broker的命令行用法说明，如下图所示：

































```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
USAGE: java [options] KafkaServer server.properties [--override property=value]*
Option                                Description
-----                                -
--override <String>                  Optional property that should override values set in
                                     server.properties file
--version                             Print version information and exit.
```

总体来说，Kafka工程自从由使用sbt改为使用Gradle管理之后，整个项目的编译和构建变得简单多了，只需要3、4条命令就能在本机环境中搭建测试开发环境了。

Kafka源码阅读方法

搭建好了开发环境，下一步自然就是阅读Kafka源码并尝试自行修改源码了。下图是IDEA上Kafka工程的完整目录列表。



- ▶  **jmh-benchmarks**
- ▶  **log4j-appender**
- ▶  **streams**
- ▶  tests
- ▶  **tools**
- ▶  vagrant
-  .gitignore
-  .travis.yml
-  build.gradle
-  CONTRIBUTING.md
-  doap_Kafka.rdf
-  gradle.properties
-  gradlew
-  gradlew.bat
-  HEADER
-  jenkins.sh
-  kafka.iml
-  kafka.ipr
-  kafka.iws
-  kafka.main.iml
-  kafka.test.iml
-  kafka-merge-pr.py
-  LICENSE
-  NOTICE
-  PULL_REQUEST_TEMPLATE.md
-  README.md
-  release.py
-  release_notes.py
-  settings.gradle
-  TROGDOR.md
-  Vagrantfile
-  wrapper.gradle

在这张图中，有几个子目录需要你重点关注一下。

- **core**: Broker端工程，保存Broker代码。
- **clients**: Client端工程，保存所有Client代码以及所有代码都会用到的一些公共代码。
- **streams**: Streams端工程，保存Kafka Streams代码。
- **connect**: Connect端工程，保存Kafka Connect框架代码以及File Connector代码。

我之前说过，Kafka源码有50万行之多，没有重点地进行通读，效率会特别低。最初我就是盲读源码的，深感效果极差，所以，我觉得非常有必要为你推荐几条最佳实践。

我建议你先从**core**包读起，也就是先从Broker端的代码着手。你可以按照下面的顺序进行阅读。

1. **log包**。log包中定义了Broker底层消息和索引保存机制以及物理格式，非常值得一读。特别是Log、LogSegment和LogManager这几个类，几乎定义了Kafka底层的消息存储机制，一定要重点关注。
2. **controller包**。controller包实现的是Kafka Controller的所有功能，特别是里面的KafkaController.scala文件，它封装了Controller的所有事件处理逻辑。如果你想弄明白Controller的工作原理，最好多读几遍这个将近2000行的大文件。
3. **coordinator包下的group包代码**。当前，coordinator包有两个子package: group和transaction。前者封装的是Consumer Group所用的Coordinator；后者封装的是支持Kafka事务的Transaction Coordinator。我个人觉得你最好把group包下的代码通读一遍，了解下Broker端是如何管理Consumer Group的。这里比较重要的是GroupMetadataManager和GroupCoordinator类，它们定义了Consumer Group的元数据信息以及管理这些元数据的状态机机制。
4. **network包代码以及server包下的部分代码**。如果你还有余力的话，可以再读一下这些代码。前者的SocketServer实现了Broker接收外部请求的完整网络流程。我们在专栏第24讲说过，Kafka用的是Reactor模式。如果你想搞清楚Reactor模式是怎么在Kafka“落地”的，就把这个类搞明白吧。

从总体流程上看，Broker端顶部的入口类是KafkaApis.scala。这个类是处理所有入站请求的总入口，下图展示了部分请求的处理方法：


```
def handle(request: RequestChannel.Request): Unit = {
  try {
    trace(s"Handling request:${request.requestDesc(true)} from connection ${request.context.connectionId};" +
      s"securityProtocol:${request.context.securityProtocol},principal:${request.context.principal}")
    request.header.apiKey match {
      case ApiKeys.PRODUCE => handleProduceRequest(request)
      case ApiKeys.FETCH => handleFetchRequest(request)
      case ApiKeys.LIST_OFFSETS => handleListOffsetRequest(request)
      case ApiKeys.METADATA => handleTopicMetadataRequest(request)
      case ApiKeys.LEADER_AND_ISR => handleLeaderAndIsrRequest(request)
      case ApiKeys.STOP_REPLICA => handleStopReplicaRequest(request)
      case ApiKeys.UPDATE_METADATA => handleUpdateMetadataRequest(request)
      case ApiKeys.CONTROLLED_SHUTDOWN => handleControlledShutdownRequest(request)
      case ApiKeys.OFFSET_COMMIT => handleOffsetCommitRequest(request)
      case ApiKeys.OFFSET_FETCH => handleOffsetFetchRequest(request)
      case ApiKeys.FIND_COORDINATOR => handleFindCoordinatorRequest(request)
      case ApiKeys.JOIN_GROUP => handleJoinGroupRequest(request)
      case ApiKeys.HEARTBEAT => handleHeartbeatRequest(request)
      case ApiKeys.LEAVE_GROUP => handleLeaveGroupRequest(request)
      case ApiKeys.SYNC_GROUP => handleSyncGroupRequest(request)
      case ApiKeys.DESCRIBE_GROUPS => handleDescribeGroupRequest(request)
      case ApiKeys.LIST_GROUPS => handleListGroupsRequest(request)
      case ApiKeys.SASL_HANDSHAKE => handleSaslHandshakeRequest(request)
      case ApiKeys.API_VERSIONS => handleApiVersionsRequest(request)
      case ApiKeys.CREATE_TOPICS => handleCreateTopicsRequest(request)
      case ApiKeys.DELETE_TOPICS => handleDeleteTopicsRequest(request)
      case ApiKeys.DELETE_RECORDS => handleDeleteRecordsRequest(request)
      case ApiKeys.INIT_PRODUCER_ID => handleInitProducerIdRequest(request)
      case ApiKeys.OFFSET_FOR_LEADER_EPOCH => handleOffsetForLeaderEpochRequest(request)
      case ApiKeys.ADD_PARTITIONS_TO_TXN => handleAddPartitionToTxnRequest(request)
      case ApiKeys.ADD_OFFSETS_TO_TXN => handleAddOffsetsToTxnRequest(request)
      case ApiKeys.END_TXN => handleEndTxnRequest(request)
      case ApiKeys.WRITE_TXN_MARKERS => handleWriteTxnMarkersRequest(request)
      case ApiKeys.TXN_OFFSET_COMMIT => handleTxnOffsetCommitRequest(request)
      case ApiKeys.DESCRIBE_ACLS => handleDescribeAcls(request)
      case ApiKeys.CREATE_ACLS => handleCreateAcls(request)
      case ApiKeys.DELETE_ACLS => handleDeleteAcls(request)
      case ApiKeys.ALTER_CONFIGS => handleAlterConfigsRequest(request)
      case ApiKeys.DESCRIBE_CONFIGS => handleDescribeConfigsRequest(request)
      case ApiKeys.ALTER_REPLICA_LOG_DIRS => handleAlterReplicaLogDirsRequest(request)
      case ApiKeys.DESCRIBE_LOG_DIRS => handleDescribeLogDirsRequest(request)
    }
  }
}
```

你可以进到不同的方法里面去看实际的请求处理逻辑。比如`handleProduceRequest`方法是处理Producer生产消息请求的，而`handleFetchRequest`方法则是处理消息读取请求的。

我们刚刚说的都是core代码包下的重要类文件。在客户端clients包下，我推荐你重点阅读4个部分的内容。

1. **org.apache.kafka.common.record**包。这个包下面是各种Kafka消息实体类，比如用于在内存中传输的MemoryRecords类以及用于在磁盘上保存的FileRecords类。
2. **org.apache.kafka.common.network**包。这个包不用全看，你重点关注下Selector、KafkaChannel就好了，尤其是前者，它们是实现Client和Broker之间网络传输的重要机制。如果你完全搞懂了这个包下的Java代码，Kafka的很多网络异常问题也就迎刃而解了。
3. **org.apache.kafka.clients.producer**包。顾名思义，它是Producer的代码实现包，里面的Java类很多，你可以重点看看KafkaProducer、Sender和RecordAccumulator这几个类。
4. **org.apache.kafka.clients.consumer**包。它是Consumer的代码实现包。同样地，我推荐你重点阅读KafkaConsumer、AbstractCoordinator和Fetcher这几个Java文件。

另外，在阅读源码的时候，不管是Broker端还是Client端，你最好结合Java调试一起来做。通过Debug模式下打断点的方式，一步一步地深入了解Kafka中各个类的状态以及在内存中的保存信息，这种阅读方式会让你事半功倍。

Kafka推荐学习资料

如果你暂时对搭建开发环境或阅读源码没有兴趣，但又想快速深入地学习Kafka的话，直接学习现成的资料也不失为一个妙法。接下来，我就向你推荐一些很有价值的Kafka学习资料。

第1个不得不提的当然就是[Kafka官网](#)。很多人会忽视官网，但其实官网才是最重要的学习资料。你只需要通读几遍官网，并切实掌握里面的内容，就已经能够较好地掌握Kafka了。

第2个是Kafka的[JIRA列表](#)。当你碰到Kafka抛出的异常的时候，不妨使用异常的关键字去JIRA中搜索一下，看看是否是已知的Bug。很多时候，我们碰到的问题早就已经被别人发现并提交到社区了。此时，JIRA列表就是你排查问题的好帮手。

第3个是[Kafka KIP列表](#)。KIP的全称是Kafka Improvement Proposals，即Kafka新功能提议。你可以看到Kafka的新功能建议及其讨论。如果你了解Kafka未来的发展路线，KIP是不能不看的。当然，如果你想到了一些Kafka暂时没有的新功能，也可以在KIP中提交自己的提议申请，等待社区的评审。

第4个是Kafka内部团队维护的[设计文档](#)。在这里，你几乎可以找到所有的Kafka设计文档。其中关于Controller和新版本Consumer的文章都很有深度，我建议你一定要重点读一读。

第5个是著名的[StackOverflow论坛](#)。当今，StackOverflow论坛对程序员意味着什么，想必我不说你也知道。这里面的Kafka问题很有深度。事实上，从仅仅是StackOverflow上的一个问题，到最后演变成了Kafka的Bug修复或新功能实现的情况屡见不鲜。

第6个是Confluent公司维护的[技术博客](#)。这是Kafka商业化公司Confluent团队自己维护的技术博客，里面的技术文章皆出自Kafka Committer之手，质量上乘，我从中受益匪浅。比如讲述Kafka精确一次处理语义和事务的文章，含金量极高，你一定要去看一下。

第7个是我自己的[博客](#)。我会定期在博客上更新Kafka方面的原创文章。有的是我对Kafka技术的一些理解，有的是Kafka的最新动态。虽然不是国内质量最好的，但应该是坚持时间最长的。毕竟，我这个博客就只有Kafka的内容，而且已经写了好几年了。

最后，我给推荐你3本学习Kafka的书。

第1本是我的[《Apache Kafka实战》](#)，我在里面总结了我这几年使用和学习Kafka的各种实战心得。这本书成书于2018年，虽然是以Kafka 1.0为模板撰写的，而Kafka目前已经出到了2.3版本，但其消息引擎方面的功能并没有什么重大变化，因此绝大部分内容依然是有效的。

第2本是[《Kafka技术内幕》](#)。我个人非常喜欢这个作者的书写风格，而且这本书内容翔实，原理分析得很透彻，配图更是精彩。

第3本是2019年新出的一本名为[《深入理解Kafka》](#)的书。这本书的作者是一位精通RabbitMQ和Kafka的著名技术人，对消息中间件有着自己独特的见解。

这些资料各有侧重，你可以根据自己的实际需求，选择相应的资料进行学习。

小结

好了，我们来小结一下。在今天的文章里，我跟你分享了很多经验，比如如何搭建**Kafka**开发环境、如何阅读**Kafka**源码等，希望这些经验可以帮你有效地节省时间，避免走一些弯路。另外，我把我收集到的相关学习资料全部列了出来，分享给你，也希望这些资料能够帮你更好地学习**Kafka**。

讲到这里，我想再强调一下，学习是个持续的过程。经验和外部帮助固然重要，但最关键的，还是自己要付出努力，持之以恒。

还是那句话：**Stay focused and work hard!**

加餐 | 3个问题大揭秘

- 搭建Kafka开发环境的5个步骤：安装Java和Gradle；下载Kafka的源码；下载Gradle的Wrapper程序套件；将Kafka源码编译打包成Jar文件；把Kafka源码工程导入到IDEA中。
- 源码阅读方法：建议你先从core包读起，也就是先从Broker端的代码着手，依次按照log包、controller包、coordinator包下的group包代码、network包代码以及server包下的部分代码的顺序阅读。
- 经典学习资料：Kafka官网、JIRA列表、KIP列表、设计文档、StackOverflow论坛、Confluent公司维护的技术博客和《Apache Kafka实战》《Kafka技术内幕》《深入理解Kafka》3本书。



开放讨论

最后，我们来讨论这样一个问题，你觉得学习Kafka或者任何一种技术，最重要的是什么？

欢迎写下你的思考和答案，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

Kafka 核心技术与实战

全面提升你的 Kafka 实战能力

胡夕

人人贷计算平台部总监

Apache Kafka Contributor



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言



每天晒白牙

4

这篇加餐很及时，正好在读kafka的源码，感谢老师

2019-08-31



姜戈

1

收藏了，刚好消息中间件学习，阅读Kafka源码，太及时了

2019-08-31



james

0

请问线上kafka数据可以迁移到线下kafka吗 也就是不同kafka直接数据迁移 复制文件的方式可否呢

2019-09-26

作者回复

嗯，就我个人而言，这属于生僻的用法，但不是不能用，只是把它当做最后的选项吧。尽量还是按照Kafka推荐的方式来做迁移

2019-09-27



DesertSnow

0

输入gradle后，等了很久，然后报错了，有遇到的吗？

2019-09-24

作者回复

可能是被墙了吧。。。。

2019-09-25



dengy

0

老师，最近发现由kafka topic+偏移量+分区组成的ID有重复，es使用这些重复ID的时候，会只保留最新的一条。请问如何使用kafka参数组成一个唯一的ID

2019-09-20

作者回复

有重复说明是否存在重复消费的问题，本身就值得好好查一下。如果一定要唯一ID，引入UUID就可以了

2019-09-23



sonald

0

跑./gradlew clean releaseTarGz其实会自动下载gradle吧。之前安装gradle似乎没有必要？

2019-09-15

作者回复

推荐还是使用Gradle的wrapper

2019-09-16



godtrue

0

课后思考及问题

你觉得学习 Kafka 或者任何一种技术，最重要的是什么？

学习是个持续的过程，经验和外部帮助固然重要，但最关键的，还是自己要付出努力，持之以恒。

还是那句话：Stay focused and work hard!

不怕不占先，就怕缠的粘。基础好脑子聪明这些条件有最好，否则除了Stay focused and work hard别无他法。

2019-09-15



墙角儿的花

0

老师 对于im服务器集群，客户端的socket均布在各个服务器，目标socket不在同一个服务器上时，服务器间需要转发消息，这个场景需要低延迟无需持久化，服务器间用redis的发布订阅，因其走内存较快，即使断电还可以走库。im服务器和入库服务间用其他mq解耦，因为这个环节需要持久化，所以选rocketmq或kafka，但kafka会延迟批量发布消息 所以选rocketmq，这两个环节的mq选型可行吗。

2019-09-11



曾轼麟

0

我觉得最重要的是两个，坚持和热情，老师还有一本书也挺好的《apache kafka源码剖析》

2019-09-11



每天晒白牙

0

周六日写的Kafka服务端之网络层的源码分析

<https://mp.weixin.qq.com/s/-VzDU0V8J2guNXwhiBEEyg>

2019-09-10

作者回复

赞~

2019-09-11



double

0

老师，partition与replication是怎么分配到broker上的

2019-09-10

作者回复

分区是个虚拟概念，分区下的副本才是broker实际分配的对象。默认情况下，你大致可以认为创建topic时副本是按照round-robin策略分配在不同broker上的。

2019-09-10



Allen Lei

0

我觉得最重要的是要知道为什么这门技术会存在，解决了什么问题，最重要的是思想

2019-09-02



陈华应

0

1.学习实现原理，经典实现的技术细节，编程思想，架构设计

2.坚持住，形成体系

3.实践，实践，实践

2019-09-01



开水

0

太及时了，昨天刚留言，今天就分享了。赞一个👍

2019-08-31



许童童

0

老师真是太牛了，向老师学习。

2019-08-31



锦

0

Scala语言需要学习到什么程度才能读懂源码呢？

2019-08-31

作者回复

不需要了解太高深，就当是个better java就行。Kafka里面也没有用到Scala很高大上的语法特性

2019-09-01



北冥Master

0

消息队列是分布式架构里面非常重要的一环，而kafka又是消息队列里面最重要的实现之一。

2019-08-31