文档

1. kafka官方文档：https://kafka.apache.org/documentation/
2. kafka docker ： https://kafka.apache.org/documentation/#docker
3. kafka docker 配置：
   https://github.com/apache/kafka/blob/trunk/docker/examples/README.md
4. kafka go lib：https://github.com/IBM/sarama/tree/main

kafka

1. docker镜像

```
docker pull apache/kafka:3.7.0
```

2. 按默认配置运行kafka

```
docker run -p 9092:9092 apache/kafka:3.7.0
```

3. 使用文件输入提供配置
    1. 要求用户提供一个包含Kafka属性文件的本地文件夹路径，并使用Docker卷将其挂载到Docker容器中
    2. 将包含kafka属性文件的文件夹装载到docker容器中的/mnt/shared/config
    3. 例如：docker run --volume path/to/property/folder:/mnt/shared/config -p 9092:9092 apache/kafka:latest
4. 使用环境变量
    1. 通过env变量定义的Kafka属性将覆盖在文件输入和默认配置中定义的该属性的值
    2. 如果属性仅通过环境变量提供，则默认配置将被用户提供的属性所取代
    3. 构建服务器属性配置的环境键变量名时，可以按照以下步骤进行操作：
        1. 将"."替换为"_"
        2. 将"_"替换为"__"（双下划线）
        3. 将"-"替换为"___"（三个下划线）
        4. 在结果前面加上 "KAFKA"
        5. 举例说明：
            1. 对于 abc.def，使用 KAFKA_ABC_DEF
            2. 对于 abc-def，使用 KAFKA_ABC___DEF
            3. 对于 abc_def，使用 KAFKA_ABC__DEF
    4. 在为log4j属性文件提供配置时，应考虑以下几点：
    5. 通过环境变量提供的log4j属性将会追加到默认的属性文件中（与kafka捆绑在一起的log4j属性文件）
    6. 可以通过设置 KAFKA_LOG4J_ROOT_LOGLEVEL 来设置 log4j.properties 和 tools-log4j.properties 中 log4j.rootLogger 的值

7. 可以通过将它们以逗号分隔的形式设置在 KAFKA_LOG4J_LOGGERS 环境变量中来添加 log4j 日志记录器到 log4j.properties 文件中
8. 示例：
    1. 假设向Docker容器提供了 KAFKA_LOG4J_LOGGERS='property1=value1,property2=value2' 环境变量。
    2. log4j.logger.property1=value1 和 log4j.logger.property2=value2 将被添加到Docker容器内部的 log4j.properties 文件中。
9. 常用于Kafka的环境变量可以通过环境变量进行提供，例如 CLUSTER_ID。
10. 可以使用命令 docker run --env CONFIG_NAME=CONFIG_VALUE -p 9092:9092 apache/kafka:latest 来为Docker容器提供环境变量。
11. 请注意，建议使用docker-compose文件来使用环境变量提供配置。

运行kafka集群（docker-compose）

```
git clone  https://github.com/apache/kafka.git
cd kafka
 docker compose -f docker/examples/jvm/cluster/combined/plaintext/docker-
compose.yml up -d
```

运行kafka集群（无身份认证）

创建配置

```
####################服务器基本设置############################
# 当前服务器的角色，设置该选项后进入kraft模式（即无需zookeeper）
process.roles=broker,controller

# 与此实例的角色关联的节点id
node.id=1

# controller角色，所有法人的连接字符串
controller.quorum.voters=1@kafka-1:9093,2@kafka-2:9093,3@kafka-3:9093

# 用于控制消费者组在发生重新平衡（rebalance）时的初始延迟时间
group.initial.rebalance.delay.ms=0
######################套接字服务器设置#################################
# 套接字服务器监听的地址，格式为：listener_name://host_name:port
listeners=PLAINTEXT://:19092,CONTROLLER://:9093,PLAINTEXT_HOST://:9092

# 用于brokers之间通信的侦听器的名称
inter.broker.listener.name=PLAINTEXT

# 公开的监听地址，若没有设置则默认为 listeners指定的值
advertised.listeners=PLAINTEXT://kafka-
1:19092,PLAINTEXT_HOST://192.168.239.161:29092

# 控制器使用的侦听器名称的逗号分隔列表。
# 如果“listener.security.protocol.map”中未设置显式映射，则默认使用PLAINTEXT协议
# 如果在KRaft模式下运行，这是必需的
controller.listener.names=CONTROLLER
```

```
# 将侦听器名称映射到安全协议，默认情况下它们是相同的
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT
_HOST:PLAINTEXT

# 服务器用于从网络接收请求并向网络发送响应的线程数
num.network.threads=3

# 服务器用于处理请求的线程数，其中可能包括磁盘I/O
num.io.threads=8

# 套接字服务器使用的发送缓冲区（SO_SNDBUF）
socket.send.buffer.bytes=102400

# 套接字服务器使用的接收缓冲区（SO_RCVBUF）
socket.receive.buffer.bytes=102400

# 套接字服务器将接受的请求的最大大小（防止OOM）
socket.request.max.bytes=104857600

############################## 日志基本设置 ##############################

# 存储日志文件的目录的逗号分隔列表
log.dirs=/tmp/kraft-combined-logs

# 每个主题的默认日志分区数。更多的分区允许更大的并行性以供使用，但这也会导致代理之间有更多的文件
num.partitions=1

# 启动时用于日志恢复和关闭时用于刷新的每个数据目录的线程数。
# 对于数据目录位于RAID阵列中的安装，建议增加此值
num.recovery.threads.per.data.dir=1

############################## 内部主题设置  ##############################
# 用于指定存储消费者偏移量（offset）的内部主题的副本因子。每个消费者组在Kafka中都会有一个特殊的内部主题，用于存储和管理消费者组中每个分区的消费偏移量
offsets.topic.replication.factor=1
# 用于指定存储事务状态日志（transaction state log）的内部主题的副本因子。事务状态日志是用来维护和管理Kafka事务性消息的状态信息
transaction.state.log.replication.factor=1
# 用于设置事务状态日志（transaction state log）内部主题中每个分区的最小同步副本数
transaction.state.log.min.isr=1

############################## 日志刷新策略 ##############################

# 消息会立即写入文件系统，但默认情况下，我们只使用fsync（）延迟同步操作系统缓存。以下配置控制将数据刷新到磁盘。这里有一些重要的权衡：
# 1.持久性：如果不使用复制，未清理的数据可能会丢失。
# 2.延迟：当刷新发生时，非常大的刷新间隔可能会导致延迟峰值，因为将有大量数据需要刷新。
# 3.吞吐量：刷新通常是最昂贵的操作，小的刷新间隔可能会导致过多的寻道。
# 下面的设置允许配置刷新策略，以便在一段时间后或每N条消息（或两者兼有）刷新数据。这可以全局完成，并在每个主题的基础上覆盖

# 强制将数据刷新到磁盘之前要接受的消息数
#log.flush.interval.messages=10000

# 在我们强制刷新之前，消息可以在日志中停留的最长时间
```

```
#log.flush.interval.ms=1000

############################## 日志保留策略 ##############################

# 以下配置控制日志段的处理。可以将该策略设置为在一段时间后删除分段，或者在累积了给定大小之后删除分段。
# 只要满足这些条件中的任意一个，段就会被删除。删除总是从日志的末尾开始

# 指一个日志文件被删除的最小年龄条件
log.retention.hours=168

# 基于大小的日志保留策略。除非剩余的段低于log.retension.bytes，否则将从日志中删除段。
# 独立于log.retension.hours的功能
#log.retention.bytes=1073741824

# 日志段文件的最大大小。当达到此大小时，将创建一个新的日志段
log.segment.bytes=1073741824

# 检查日志段以查看是否可以根据保留策略删除它们的间隔
log.retention.check.interval.ms=300000
```

不同节点配置上的差异

```
node.id=1
advertised.listeners=PLAINTEXT://kafka-
1:19092,PLAINTEXT_HOST://192.168.239.161:29092

node.id=2
advertised.listeners=PLAINTEXT://kafka-
2:19092,PLAINTEXT_HOST://192.168.239.161:39092

node.id=3
advertised.listeners=PLAINTEXT://kafka-
3:19092,PLAINTEXT_HOST://192.168.239.161:49092
```

创建网络

```
docker network create kafka-net
```

获取集群ID

```
KAFKA_CLUSTER_ID=$(uuidgen)
```

启动kafka节点

```
docker run -d --name kafka-1 -p 29092:9092 --network kafka-net \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-v /home/nick/work/kafkaconf1/kafka1:/mnt/shared/config apache/kafka:3.7.0

docker run -d --name kafka-2 -p 39092:9092 --network kafka-net \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-v /home/nick/work/kafkaconf1/kafka2:/mnt/shared/config apache/kafka:3.7.0

docker run -d --name kafka-3 -p 49092:9092 --network kafka-net \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-v /home/nick/work/kafkaconf1/kafka3:/mnt/shared/config apache/kafka:3.7.0
```

验证

```
# 查看每一个节点在启动时，是否加入到了同一个集群
docker logs -f kafka-1
```

运行kafka集群（SASL身份认证）

创建配置

1. server.properties

```
####################服务器基本设置############################
# 当前服务器的角色，设置该选项后进入kraft模式（即无需zookeeper）
process.roles=broker,controller

# 与此实例的角色关联的节点id
node.id=1

# controller角色，所有法人的连接字符串
controller.quorum.voters=1@kafka-1:9093,2@kafka-2:9093,3@kafka-3:9093

# 用于控制消费者组在发生重新平衡（rebalance）时的初始延迟时间
group.initial.rebalance.delay.ms=0

######################套接字服务器设置#################################
# sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required \
#   username="alice" \
#    password="abcdefg";

sasl.mechanism.inter.broker.protocol=PLAIN
sasl.enabled.mechanisms=PLAIN
security.inter.broker.protocol=PLAINTEXT

# 套接字服务器监听的地址，格式为：listener_name://host_name:port
listeners=PLAINTEXT://:19092,CONTROLLER://:9093,PLAINTEXT_HOST://:9092
```

```
# 用于brokers之间通信的侦听器的名称
# inter.broker.listener.name=PLAINTEXT

# 公开的监听地址，若没有设置则默认为 listeners指定的值
advertised.listeners=PLAINTEXT://kafka-
1:19092,PLAINTEXT_HOST://192.168.239.161:29092

# 控制器使用的侦听器名称的逗号分隔列表。
# 如果“listener.security.protocol.map”中未设置显式映射，则默认使用PLAINTEXT协议
# 如果在KRaft模式下运行，这是必需的
controller.listener.names=CONTROLLER

# 将侦听器名称映射到安全协议，默认情况下它们是相同的
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:SASL_PLAINTEXT,PLAI
NTEXT_HOST:SASL_PLAINTEXT

# 服务器用于从网络接收请求并向网络发送响应的线程数
num.network.threads=3

# 服务器用于处理请求的线程数，其中可能包括磁盘I/O
num.io.threads=8

# 套接字服务器使用的发送缓冲区（SO_SNDBUF）
socket.send.buffer.bytes=102400

# 套接字服务器使用的接收缓冲区（SO_RCVBUF）
socket.receive.buffer.bytes=102400

# 套接字服务器将接受的请求的最大大小（防止OOM）
socket.request.max.bytes=104857600

############################## 日志基本设置 ##############################

# 存储日志文件的目录的逗号分隔列表
log.dirs=/tmp/kraft-combined-logs

# 每个主题的默认日志分区数。更多的分区允许更大的并行性以供使用，但这也会导致代理之间有更多的文件
num.partitions=1

# 启动时用于日志恢复和关闭时用于刷新的每个数据目录的线程数。
# 对于数据目录位于RAID阵列中的安装，建议增加此值
num.recovery.threads.per.data.dir=1

############################## 内部主题设置  ##############################
# 用于指定存储消费者偏移量（offset）的内部主题的副本因子。每个消费者组在Kafka中都会有一个特殊的
内部主题，用于存储和管理消费者组中每个分区的消费偏移量
offsets.topic.replication.factor=1
# 用于指定存储事务状态日志（transaction state log）的内部主题的副本因子。事务状态日志是用来维
护和管理Kafka事务性消息的状态信息
transaction.state.log.replication.factor=1
# 用于设置事务状态日志（transaction state log）内部主题中每个分区的最小同步副本数
transaction.state.log.min.isr=1

############################## 日志刷新策略 ##############################
```

```
# 消息会立即写入文件系统，但默认情况下，我们只使用fsync（）延迟同步操作系统缓存。以下配置控制将数据刷新到磁盘。这里有一些重要的权衡：
# 1.持久性：如果不使用复制，未清理的数据可能会丢失。
# 2.延迟：当刷新发生时，非常大的刷新间隔可能会导致延迟峰值，因为将有大量数据需要刷新。
# 3.吞吐量：刷新通常是最昂贵的操作，小的刷新间隔可能会导致过多的寻道。
# 下面的设置允许配置刷新策略，以便在一段时间后或每N条消息（或两者兼有）刷新数据。这可以全局完成，并在每个主题的基础上覆盖

# 强制将数据刷新到磁盘之前要接受的消息数
#log.flush.interval.messages=10000

# 在我们强制刷新之前，消息可以在日志中停留的最长时间
#log.flush.interval.ms=1000

############################# 日志保留策略 #############################

# 以下配置控制日志段的处理。可以将该策略设置为在一段时间后删除分段，或者在累积了给定大小之后删除分段。
# 只要满足这些条件中的任意一个，段就会被删除。删除总是从日志的末尾开始

# 指一个日志文件被删除的最小年龄条件
log.retention.hours=168

# 基于大小的日志保留策略。除非剩余的段低于log.retension.bytes，否则将从日志中删除段。
# 独立于log.retension.hours的功能
#log.retention.bytes=1073741824

# 日志段文件的最大大小。当达到此大小时，将创建一个新的日志段
log.segment.bytes=1073741824

# 检查日志段以查看是否可以根据保留策略删除它们的间隔
log.retention.check.interval.ms=300000
```

2. kafka_server_jaas.conf

```
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    serviceName="kafka"
    username="admin"
    password="123456"
    user_admin="123456"
    user_alice="abcdefg";
};
```

1. username与password 用于当前节点连接到其他节点
2. user_= 表示创建一个用户并指定密码
3. 格式，不能包含#注释，最后一行必须以分号结尾

不同节点配置上的差异

```
node.id=1
advertised.listeners=PLAINTEXT://kafka-
1:19092,PLAINTEXT_HOST://192.168.239.161:29092

node.id=2
advertised.listeners=PLAINTEXT://kafka-
2:19092,PLAINTEXT_HOST://192.168.239.161:39092

node.id=3
advertised.listeners=PLAINTEXT://kafka-
3:19092,PLAINTEXT_HOST://192.168.239.161:49092
```

创建网络

```
docker network create kafka-net
```

获取集群ID

```
KAFKA_CLUSTER_ID=$(uuidgen)
```

启动kafka节点

```
docker run -d --name kafka-1 -p 29092:9092 --network kafka-net  \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-e KAFKA_OPTS="-
Djava.security.auth.login.config=/mnt/shared/config/kafka_server_jaas.conf" \
-v /home/nick/work/kafkaconf/kafka1:/mnt/shared/config apache/kafka:3.7.0

docker run -d --name kafka-2 -p 39092:9092 --network kafka-net  \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-e KAFKA_OPTS="-
Djava.security.auth.login.config=/mnt/shared/config/kafka_server_jaas.conf" \
-v /home/nick/work/kafkaconf/kafka2:/mnt/shared/config apache/kafka:3.7.0


docker run -d --name kafka-3 -p 49092:9092 --network kafka-net  \
-e CLUSTER_ID=$KAFKA_CLUSTER_ID \
-e KAFKA_OPTS="-
Djava.security.auth.login.config=/mnt/shared/config/kafka_server_jaas.conf" \
-v /home/nick/work/kafkaconf/kafka3:/mnt/shared/config apache/kafka:3.7.0
```

验证

```
# 查看每一个节点在启动时，是否加入到了同一个集群
docker logs -f kafka-1
```