# Инструкция по установке Q.StreameR.

Exported on Jul 08, 2019

**1 Шаг №1: Скачайте дистрибутив. ....4**

**2 Шаг №2: Запустите Q.StreameR. ....5**

**3 Шаг №3: Создание топика. ....6**

**4 Шаг №4: Отправка сообщений. ....7**

**5 Шаг №5: Запуск обработчика сообщений. ....8**

**6 Шаг №6: Настройка кластера Q.StreameR. ....9**

**7 Параметры конфигурации для промышленного использования ....12**

7.1 Общие конфигурационные параметры ....12

7.2 Параметры репликации ....12

7.3 Файловые дескрипторы и mmap ....13

7.4 Мультикластерная конфигурация ....13

В руководстве предполагается, что вы разворачиваете решение с нуля и у вас нет данных Q.StreameR или ZooKeeper. Поскольку консольные сценарии Q.StreameR различаются для платформ Unix и Windows, на платформах Windows используйте **bin\windows** вместо **bin/** и измените расширение сценария с **.sh** на **.bat**.


- Шаг №1: Скачайте дистрибутив.
- Шаг №2: Запустите Q.StreameR.
- Шаг №3: Создание топика.
- Шаг №4: Отправка сообщений.
- Шаг №5: Запуск обработчика сообщений.
- Шаг №6: Настройка кластера Q.StreameR.
- Параметры конфигурации для промышленного использования
    - Общие конфигурационные параметры
    - Параметры репликации
    - Файловые дескрипторы и mmap
    - Мультикластерная конфигурация

## 1 Шаг №1: Скачайте дистрибутив.

Скачайте архив дистрибутива и разархивируйте его.

```
> tar -xzf qstreamer-2.02.01-19062601.tgz
> cd qstreamer-2.02.01-19062601
```

## 2 Шаг №2: Запустите Q.StreameR.

Q.StreameR использует ZooKeeper, поэтому вам нужно сначала запустить сервер ZooKeeper, если у вас его еще нет.

Вы можете использовать вспомогательный скрипт, поставляемый вместе с Q.StreameR, чтобы запустить экземпляр ZooKeeper с одним узлом:

```
> bin/zookeeper-server-start.sh config/zookeeper.properties
```

Теперь запускаем сервер Q.StreameR:

```
> bin/qstreamer-server-start.sh config/server.properties
```

## 3 Шаг №3: Создание топика.

Теперь создадим топик с именем "test" с одним разделом и одной репликой:

```
> bin/qstreamer-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
```

Мы можем увидеть этот топик, если выполним команду вывода списка топиков:

```
> bin/qstreamer-topics.sh --list --bootstrap-server localhost:9092

test
```

## 4 Шаг №4: Отправка сообщений.

Q.StreameR поставляется с клиентом командной строки, который будет принимать входные данные из файла или из стандартного ввода и отправлять их в виде сообщений в кластер Q.StreameR. По умолчанию каждая строка будет отправлена как отдельное сообщение.

Запустите qstreamer-console-producer и напечатайте несколько сообщений в консоли для отправки:

```
> bin/qstreamer-console-producer.sh --broker-list localhost:9092 --topic test
This is a message
This is another message
```

## 5 Шаг №5: Запуск обработчика сообщений.

У Q.StreameR также есть утилита командной строки обработчика сообщений, которая выводит полученные сообщения в стандартный вывод.

```
> bin/qstreamer-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning
This is a message
This is another message
```

Если вы запустите команды qstreamer-console-producer и qstreamer-console-consumer в разных терминалах, то вы сможете набирать сообщения в терминале producer и видеть, как они появляются в терминале consumer.

Все утилиты командной строки имеют дополнительные параметры, выполнение команд без параметров отобразит документацию с подробными пояснениями.

## 6 Шаг №6: Настройка кластера Q.StreameR.

До сих пор мы работали только с одним брокером сообщений. Для Q.StreameR один брокер сообщений - это кластер с одним узлом и ничего особенно не изменится, если мы запустим еще несколько экзмепляров брокеров сообщений.

Давайте расширим наш кластер до трех узлов (все еще на нашей локальной машине).

Сначала мы создадим файл конфигурации для каждого из брокеров (в Windows используйте команду copy):

```
> cp config/server.properties config/server-1.properties
> cp config/server.properties config/server-2.properties
```

Теперь отредактируйте эти новые файлы и установите следующие свойства:

```
config/server-1.properties:
    broker.id=1
    listeners=PLAINTEXT://:9093
    log.dirs=/tmp/qstreamer-logs-1

config/server-2.properties:
    broker.id=2
    listeners=PLAINTEXT://:9094
    log.dirs=/tmp/qstreamer-logs-2ties
```

Свойство broker.id - это уникальное и постоянное имя каждого узла в кластере. Мы должны переопределить каталог портов и журналов только потому, что все они выполняются на одном компьютере, и мы хотим, чтобы брокеры не пытались зарегистрироваться на одном и том же порту или перезаписывать данные друг друга.

Zookeeper у нас уже запущен, поэтому нам нужно просто запустить два новых узла Q.StreameR:

```
> bin/qstreamer-server-start.sh config/server-1.properties &
...
> bin/qstreamer-server-start.sh config/server-2.properties &
...
```

Теперь создадим новый топик с коэффициентом репликации три:

```
> bin/qstreamer-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 3 --partitions
1 --topic my-replicated-topic
```

Хорошо, но теперь, когда у нас есть кластер, как мы можем узнать, какой брокер что делает? Чтобы увидеть это, запустите команду «describe topics»:

```
> bin/qstreamer-topics.sh --describe --bootstrap-server localhost:9092 --topic my-replicated-topic
Topic:my-replicated-topic   PartitionCount:1    ReplicationFactor:3 Configs:
    Topic: my-replicated-topic  Partition: 0    Leader: 1   Replicas: 1,2,0 Isr: 1,2,0
```

В первой строке дается сводная информация обо всех разделах, каждая дополнительная строка содержит информацию об одном разделе. Поскольку у нас только один раздел для топика, вывелась всего одна строка.

«Leader» - это узел, отвечающий за все операции чтения и записи для данного раздела. Каждый узел будет лидером для случайно выбранной части разделов.
«Replicas» - это список узлов, которые реплицируют журнал для этого раздела.

«isr» - это набор «синхронизированных» реплик. Это подмножество списка реплик, которое в настоящее время живо и захвачено лидером.

Мы можем выполнить ту же команду для исходного топика "test":

```
> bin/qstreamer-topics.sh --describe --bootstrap-server localhost:9092 --topic test
Topic:test  PartitionCount:1    ReplicationFactor:1 Configs:
   Topic: test Partition: 0    Leader: 0   Replicas: 0 Isr: 0
```

Поэтому нет ничего удивительного в том, что исходный топик не имеет реплик и находится на сервере 0, единственном сервере в нашем кластере, когда мы его создали.

Давайте опубликуем несколько сообщений в наш новый топик:

```
> bin/qstreamer-console-producer.sh --broker-list localhost:9092 --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

Теперь давайте обработаем эти сообщения:

```
> bin/qstreamer-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

Теперь давайте проверим отказоустойчивость. Брокер 1 действовал как лидер, поэтому давайте остановим его:

```
> ps aux | grep server-1.properties
7564 ttys002    0:15.91
/System/Library/Frameworks/JavaVM.framework/Versions/1.8/Home/bin/java...
> kill -9 7564
```

На Windows используйте команду:

```
> wmic process where "caption = 'java.exe' and commandline like '%server-1.properties%'" get processid
ProcessId
6016
> taskkill /pid 6016 /f
```

Лидером стал один из подписчиков, и узел 1 больше не находится в наборе синхронных реплик:

```
> bin/qstreamer-topics.sh --describe --bootstrap-server localhost:9092 --topic my-replicated-topic
Topic:my-replicated-topic   PartitionCount:1    ReplicationFactor:3 Configs:
   Topic: my-replicated-topic  Partition: 0    Leader: 2   Replicas: 1,2,0 Isr: 2,0
```

Но сообщения все еще доступны для потребителей, хотя лидер, который первоначально записывал записи, не работает:

```
> bin/qstreamer-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

## 7 Параметры конфигурации для промышленного использования

Настройки по умолчанию Q.StreameR должны работать в большинстве случаев. Но есть некоторые конфигурационные параметры, которые должны быть изменены при промышленном использовании.

### 7.1 Общие конфигурационные параметры

- zookeeper.connect
  Список хостов ZooKeeper, на которых зарегистрирован брокер. Рекомендуется настроить это для всех хостов в вашем кластере ZooKeeper
- broker.id
  Целочисленный идентификатор, который идентифицирует брокера. Брокеры в одном и том же кластере Q.StreameR не должны иметь одинаковый идентификатор.
- log.dirs
  Каталоги, в которых находятся данные журнала Q.StreameR.
- listeners
  Разделенный запятыми список URI (включая протокол), который будет прослушивать брокер. Укажите имя хоста как 0.0.0.0 для привязки ко всем интерфейсам или оставьте его пустым для привязки к интерфейсу по умолчанию. Примером является PLAINTEXT: // myhost: 9092.
- advertised.listeners
  Слушатели для публикации в ZooKeeper для использования клиентами. В средах IaaS это может отличаться от интерфейса, с которым связывается брокер. Если это не установлено, будет использоваться значение для слушателей.
- num.partitions
  Количество разделов журнала по умолчанию для автоматически создаваемых тем. Вы должны увеличить это, так как лучше разделить тему. Чрезмерное разделение темы приводит к лучшей балансировке данных и помогает потребительскому параллелизму. Для ключевых данных следует избегать изменения количества разделов в теме.

### 7.2 Параметры репликации

- default.replication.factor
  Коэффициент репликации по умолчанию, который применяется к автоматически созданным темам. Вы должны установить это по крайней мере 2.
    - Тип: int
    - По умолчанию: 1
    - Важность: средняя
- min.insync.replicas
  Минимальное количество реплик в ISR. Вы должны установить значение required.acks = -1 (или всеми).
    - Тип: int
    - По умолчанию: 1
    - Важность: средняя
- unclean.leader.election.enable
  Указывает, следует ли включать реплики, не входящие в набор ISR, в качестве лидера в качестве последнего средства, даже если это может привести к потере данных.
    - Тип: int

- По умолчанию: 1
- Важность: средняя

### 7.3 Файловые дескрипторы и mmap

Кафка использует очень большое количество файлов и большое количество сокетов для связи с клиентами. Все это требует относительно большого количества доступных файловых дескрипторов.

Многие современные дистрибутивы Linux поставляются только с 1024 дескрипторами файлов, разрешенными для каждого процесса. Это слишком низко для Кафки.

Вы должны увеличить количество файловых дескрипторов как минимум до 100 000. Этот процесс может быть сложным и сильно зависит от вашей конкретной ОС и дистрибутива. Обратитесь к документации для вашей ОС, чтобы определить, как лучше всего изменить количество разрешенных дескрипторов файлов.

### 7.4 Мультикластерная конфигурация

В производственной среде требуется несколько брокеров. Во время запуска брокеры регистрируются в ZooKeeper, чтобы стать участником кластера.

Перейдите к конфигурационном файлу Q.StreameR (/etc/qstreamer/config/server.properties) и настройте следующее:

- Подключитесь к одному и тому ZooKeeper, установив для zookeeper.connect во всех узлах одинаковое значение. Замените все экземпляры localhost на имя хоста или полное доменное имя вашего узла, где развернут ZooKeeper. Например, если ваше имя хоста - zookeeper:

```
zookeeper.connect=zookeeper:2181
```

Настройте идентификаторы брокера для каждого узла в кластере, используя один из этих методов:

- Динамически генерируйте идентификаторы брокера: добавьте broker.id.generation.enable = true и закомментируйте broker.id. Например:

```
############################## Server Basics ###############################

# The ID of the broker. This must be set to a unique integer for each broker.
#broker.id=0
broker.id.generation.enable=true
```

- Вручную установите идентификаторы брокера: установите уникальное значение для broker.id на каждом узле.

Сконфигурируйте, как другие брокеры и клиенты общаются с брокером, используя listeners и, по желанию, advertised.listeners.

- **listeners**: разделенный запятыми список URI и имен слушателей для прослушивания.
- **advertised.listeners:** Разделенный запятыми список URI и имен слушателей для использования другими брокерами и клиентами. Параметр advertised.listeners гарантирует, что посредник объявляет адрес, доступный как с локальных, так и с внешних хостов.