

Kafka Monitoring and Operations



- Some of the most important metrics are:
- Under Replicated Partitions: Number of partitions are have problems with the ISR (in-sync replicas). May indicate a high load on the system
- Request Handlers: utilization of threads for IO, network, etc... overall utilization of an Apache Kafka broker.
- Request timing: how long it takes to reply to requests. Lower is better, as latency will be improved.

Link to the...

grafana цветовой маркировкой сама покажет отклонения от метрик

Important Metrics to monitor



- A few metrics are super important to have:
 - Number of active controller: should always be 1
 - Number of Under Replicated Partitions: should always be 0
 - Number of Offline Partitions: should always be 0
- There are a ton of metrics you can find online:
 - <https://kafka.apache.org/documentation/#monitoring>
 - <https://docs.confluent.io/current/kafka/monitoring.html>
- It's better to have more metrics monitored than less to easily troubleshoot issues when they arise

производительность как продюсера, так и потребителя зависит от пропускной способности

- В плане генераторов нас интересует в основном скорость отправки сообщений брокеру.

Разумеется, чем выше пропускная способность — тем лучше.

- В плане потребителей нас также интересует производительность, а именно: насколько быстро мы можем читать сообщения от брокера.

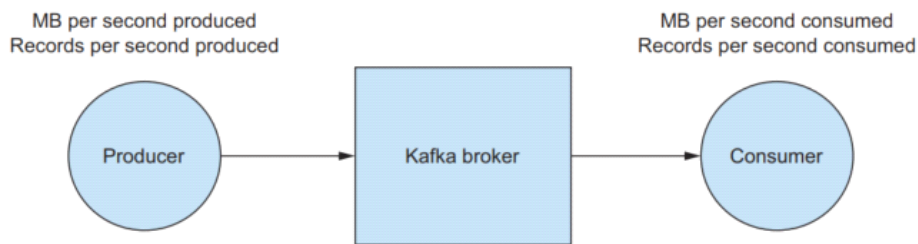


Figure 7.1 Kafka producer and consumer performance concerns, writing to and reading from a broker

consumer lag - другой показатель производительности потребителя: отставание потребителя

- Но существует и другой показатель производительности потребителя: отставание потребителя
- Как видите, нас интересует, какой объем сообщений и насколько быстро могут наши генераторы отправлять брокеру, а одновременно и то, насколько быстро наши потребители могут прочесть сообщения с него.
- Разница между скоростью записи генераторами сообщений на брокер и скоростью чтения потребителями сообщений с него называется отставанием потребителя (consumer lag).
-

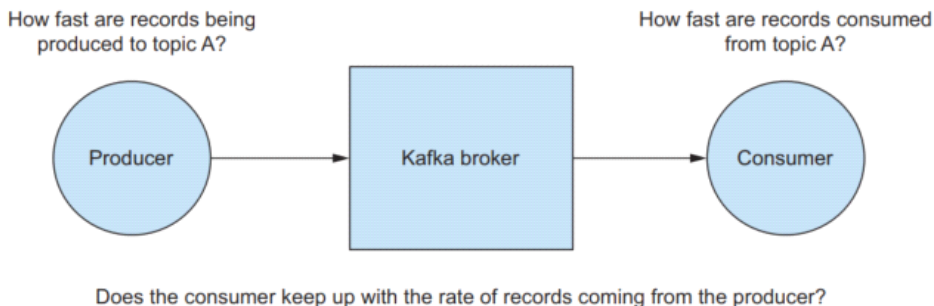
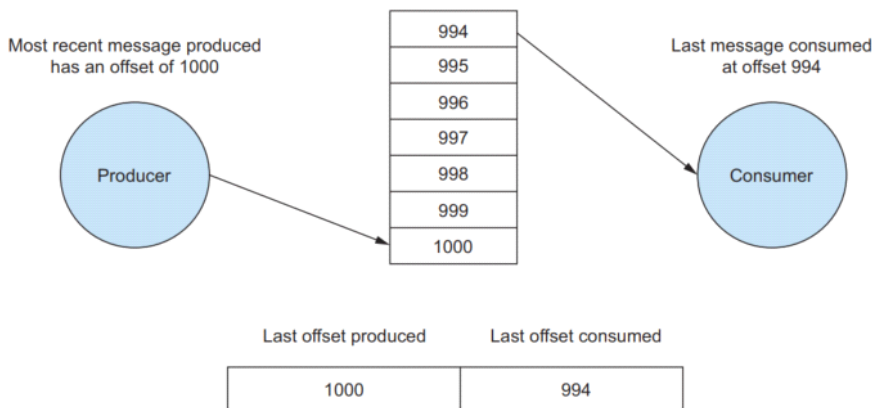


Figure 7.2 Kafka producer and consumer performance revisited

- что отставание потребителя представляет собой разницу между последним зафиксированным потребителем смещением и последним смещением записанного на брокер сообщения.
- Некоторое отставание потребителя неизбежно, но в идеале потребитель наверстывает упущенное или по крайней мере отставание не растет.



The difference between the most recent offset produced and the last offset consumed (from the same topic) is known as consumer lag.

In this case, the consumer lags behind the producer by six records.

Figure 7.3 Consumer lag is the difference in offsets committed by the consumer and offsets written by the producer

когда отставание растет со временем — это признак того, что потребителю требуется больше ресурсов

Небольшое либо постоянное отставание вполне нормально, но когда отставание растет со временем — это признак того, что потребителю требуется больше ресурсов

как вычислить consumer lag

Во-первых, воспользуемся командой `list` для вывода списка всех активных групп потребителей. Результаты ее выполнения приведены на рис. 7.4.

```
<kafka-install-dir>/bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--list
```

```
oddball:bin bbejeck$ ./kafka-consumer-groups.sh --list --bootstrap-server localhost:9092
Note: This will only show information about consumers that use the Java consumer API (non-ZooKeeper-based consumers).
console-consumer-59026
```

Рис. 7.4. Вывод перечня доступных групп потребителей из командной строки

Получив эту информацию, можно выбрать название группы потребителей и выполнить следующую команду:

```
<kafka-install-dir>/bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--group <GROUP-NAME> \
--describe
```

Результаты (состояние работы потребителя) показаны на рис. 7.5.

Количество прочитанных сообщений = 3 Количество отправленных в топик сообщений = 10

```
oddball:bin bbejeck$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group console-consumer-59026 --describe
Note: This will only show information about consumers that use the Java consumer API (non-ZooKeeper-based consumers).
Consumer group 'console-consumer-59026' has no active members.
TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID     HOST                CLIENT-ID
consumer_lag_demo  0           3                10               7               console-consumer-59026  localhost:9092      -
```

10 (отправлено сообщений) – 3 (прочитано сообщений) = 7 (отставание на такое число записей)

Рис. 7.5. Состояние группы потребителей

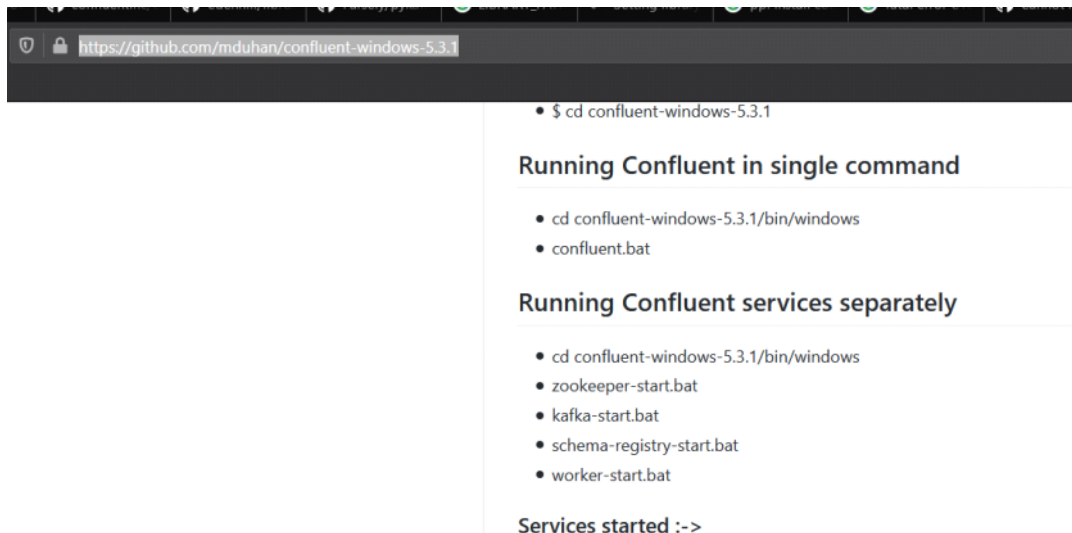
Эти результаты показывают, что отставание потребителя невелико. Наличие отставания потребителя не всегда означает наличие проблем — потребители читают сообщения пакетами и не получают следующий пакет до окончания обработки текущего. Обработка записей занимает определенное время, так что небольшое отставание неудивительно.

GUI TOOLS 1

3 ноября 2020 г. 17:26

работающий виндовый клиент

<https://github.com/mduhan/confluent-windows-5.3.1>



kafka-broker-api-versions --command-config kafka.properties --bootstrap-server kafka:31000

```
C:\ProgramFilesMy\confluent-windows-5.0.1\bin\windows>kafka-broker-api-versions --command-config kafka.properties --bootstrap-server kafka:31000
kafka:31002 (id: 0 rack: 0) -> (
  Produce(0): 0 to 8 [usable: 6],
  Fetch(1): 0 to 11 [usable: 8],
  ListOffsets(2): 0 to 5 [usable: 3],
  Metadata(3): 0 to 9 [usable: 6],
  LeaderAndIsr(4): 0 to 4 [usable: 1],
  StopReplica(5): 0 to 3 [usable: 0],
  UpdateMetadata(6): 0 to 6 [usable: 4],
  ControlledShutdown(7): 0 to 3 [usable: 1],
  OffsetCommit(8): 0 to 8 [usable: 4],
  OffsetFetch(9): 0 to 7 [usable: 4],
  FindCoordinator(10): 0 to 3 [usable: 2],
  JoinGroup(11): 0 to 7 [usable: 3],
  Heartbeat(12): 0 to 4 [usable: 2],
  LeaveGroup(13): 0 to 4 [usable: 2],
  SyncGroup(14): 0 to 5 [usable: 2],
  DescribeGroups(15): 0 to 5 [usable: 2],
  ListGroups(16): 0 to 4 [usable: 2],
  SaslHandshake(17): 0 to 1 [usable: 1],
  ApiVersions(18): 0 to 3 [usable: 2],
```

работающий GUI клиент

<https://www.kafkamagic.com/>

админ GUI клиент

<https://www.kafkatool.com/>

<https://docs.conduktor.io/>

<https://dev.to/dariusx/recommend-a-simple-kafka-ui-tool-5gob>

<https://github.com/obsidiandynamics/kafdrop>

<https://www.getkadeck.com/#/?tab=desktop>

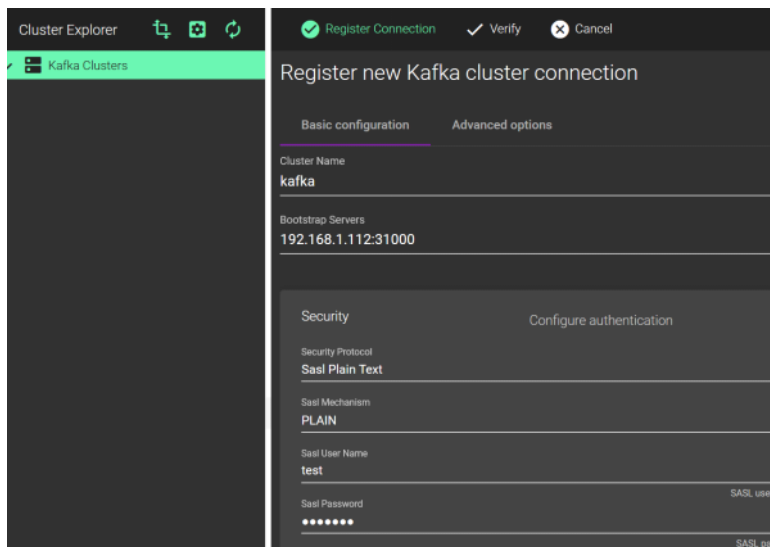
<https://github.com/yahoo/CMAK>

<https://blog.usejournal.com/overview-of-ui-monitoring-tools-for-apache-kafka-clusters-9ca516c165bd>
<https://rmoff.net/2018/08/02/kafka-listeners-explained/>

Windows

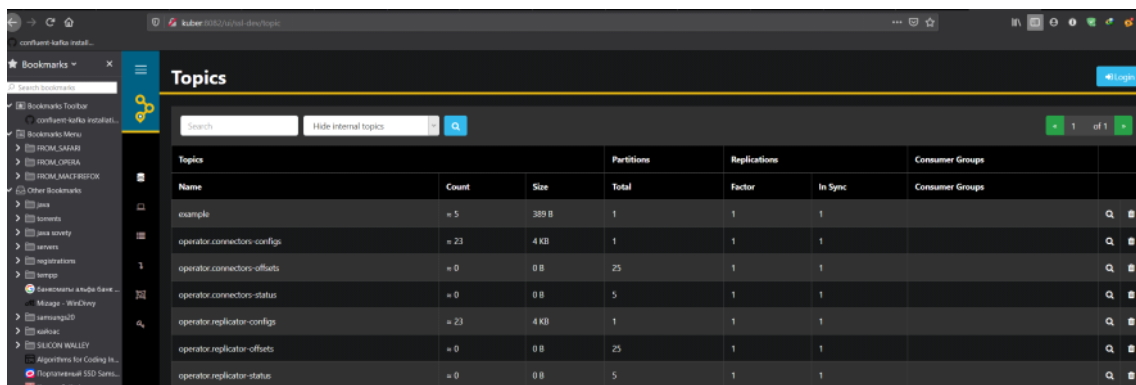
Extract *zip* file into a new folder. Run *KafkaMagic.exe* app.

In your browser navigate to *http://localhost:5000*
(you can change the port number - see below)



kafka GUI akhq

```
docker run -d -p 8082:8080 -v C:\application.yaml:/app/application.yaml --add-host kafka:192.168.1.112 --name kafka2 tchiotludo/akhq
```



добавить на диск С файл конфига

<https://www.edureka.co/community/63666/is-there-any-web-ui-for-kafka>
<https://github.com/tchiotludo/akhq/releases>

```
docker run -d -p 8082:8080 -v C:\application.yaml:/app/application.yaml tchiotludo/akhq
```

akhq:

connections:

ssl-dev:

properties:

bootstrap.servers: "192.168.1.112:31000"

security.protocol: SASL_PLAINTEXT

sasl.mechanism: PLAIN

sasl.jaas.config : org.apache.kafka.common.security.plain.PlainLoginModule required username="test"

password="test123";

kafkacat

<https://docs.confluent.io/current/app-development/kafkacat-usage.html>

<https://tsuyoshiushio.medium.com/configuring-kafka-on-kubernetes-makes-available-from-an-external-client-with-helm-96e9308ee9f4>

<https://rmoff.net/2018/08/02/kafka-listeners-explained/>

kafkacat Utility

kafkacat is a command line utility that you can use to test and debug Apache Kafka® deployments. You can use kafkacat to produce, consume, and list topic and partition information for Kafka. Described as “netcat for Kafka” it is a swiss-army knife of tools for inspecting and creating data in Kafka.

It is similar to Kafka Console Producer (`kafka-console-producer`) and Kafka Console Consumer (`kafka-console-consumer`), but even more powerful.

Important

kafkacat is an open-source utility, available at <https://github.com/edenhill/kafkacat>. It is not supported by Confluent and is not included in Confluent Platform.

плагин для ruCharm

<https://plugins.jetbrains.com/plugin/11946-kafkalytic>

https://plugins.jetbrains.com/plugin/11946-kafkalytic

JET BRAINS Marketplace

Resources Sign In

Tools integration

Kafkalytic

★★★★★

Daniil Ermakov

Install to IDE

Compatible with all IntelliJ-based IDEs

Overview Versions Reviews

The screenshot shows the Kafkalytic plugin interface. On the left, there's a tree view of Kafka topics and consumers. On the right, a 'New cluster' dialog box is open, prompting the user to enter Kafka bootstrap servers as host:port,host2:port. The dialog also includes fields for Cluster name (optional), Truststore path, Truststore password, Keystore path, Keystore password, and Request timeout, ms. There are buttons for 'Load properties from file', 'Test connection', 'OK', and 'Cancel'.

New cluster

Enter Kafka bootstrap servers as host:port,host2:port

192.168.1.112:31000

Cluster name (optional)

Truststore path local path

Truststore password

Keystore path local path

Keystore password

Request timeout, ms 5000

Load properties from file

security.protocol SASL_PLAINTEXT

sasl.jaas.config org.apache.kafka.common.security.plain.PlainLoginModule required username="test" password="test123";

sasl.mechanism PLAIN

bootstrap.servers 192.168.1.112:31000

Test connection

OK Cancel

landoop/kafka-topics-ui

<https://hub.docker.com/r/landoop/kafka-topics-ui/>

lensesio/kafka-topics-ui

<https://github.com/lensesio/kafka-topics-ui>

<https://github.com/lensesio/kafka-topics-ui/releases>

lenses kafka topic ui

<https://lenses.io/blog/2017/05/kafka-topics-ui-v2/>

<https://lenses.io/blog/2016/08/kafka-topics-ui/>

Admin and Monitoring Learning



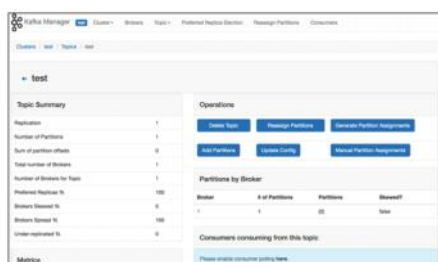
- **Administration tools:**
 - ZooNavigator UI: view Zookeeper nodes if we need to
 - Yahoo Kafka Manager: easily view brokers, topics, easily create topics
 - LinkedIn Kafka Monitor: UI to visualize a continuously running producer / consumer and measure end-to-end latency
- **Monitoring Tools:**
 - Prometheus: acquire + store metrics
 - Grafana: create dashboards and visualize metrics against Prometheus
- **Disclaimer:** *The setup is meant to show capabilities and may not match your deployment model. If you need to do things differently in production, learn from the process and adapt it for your use case*

kafka manager

<https://hub.docker.com/r/kafkamanager/kafka-manager>
<https://github.com/yahoo/kafka-manager>

Hands On: Cluster Management with Kafka Manager

- Setup security groups to access our web tools instance
- Run Kafka Manager using Docker Compose
- Demo of Kafka Manager



```
#!/bin/bash
```

```
# make sure you open port 9000 on the security group
```

```
# make sure you can access the zookeeper endpoints
```

```
nc -vz zookeeper1 2181
```

```
nc -vz zookeeper2 2181
```

```
nc -vz zookeeper3 2181
```

make sure you can access the kafka endpoints

nc -vz kafka1 9092

nc -vz kafka2 9092

nc -vz kafka3 9092

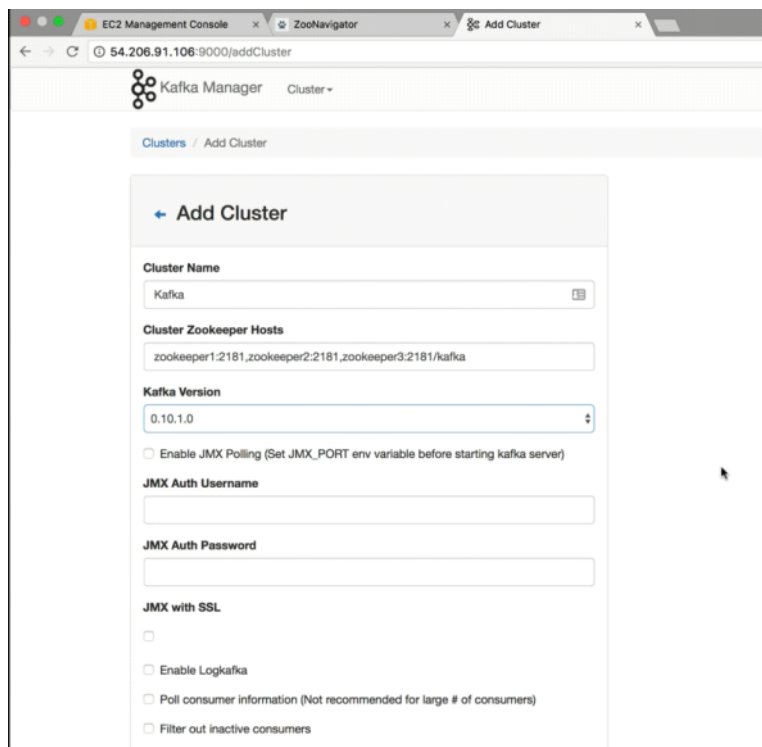
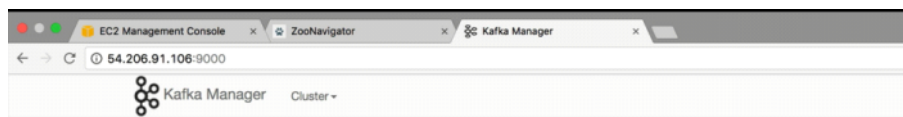
copy the kafka-manager-docker-compose.yml file

nano kafka-manager-docker-compose.yml

launch it

docker-compose -f kafka-manager-docker-compose.yml up -d

настроим GUI



100

logkafkaUpdatePeriodSeconds

30

partitionOffsetCacheTimeoutSecs

5

brokerViewThreadPoolSize

2

Must be greater or equal to 2

brokerViewThreadPoolQueueSize

1000

offsetCacheThreadPoolSize

2

Must be greater or equal to 2

offsetCacheThreadPoolQueueSize

1000

kafkaAdminClientThreadPoolSize

2

Must be greater or equal to 2

kafkaAdminClientThreadPoolQueueSize

1000

Save Cancel

References

EC2 Management Console x ZooNavigator x Broker List x

54.206.91.106:9000/clusters/Kafka/brokers

Kafka Manager Kafka Cluster Brokers Topic Preferred Replica Election Reassign Partitions Consumers

Clusters / Kafka / Brokers

Brokers						Combined Metrics
Id	Host	Port	JMX Port	Bytes In	Bytes Out	
1	kafka1	9092	-1	==	==	Please enable JMX polling here .
2	kafka2	9092	-1	==	==	
3	kafija3	9092	-1	==	==	

[docker compose](#)

version: '2'

services:

<https://github.com/yahoo/kafka-manager>

kafka-manager:

image: **qnib/plain-kafka-manager**

network_mode: host

environment:

ZOOKEEPER_HOSTS: "zookeeper1:2181,zookeeper2:2181,zookeeper3:2181"

APPLICATION_SECRET: change_me_please

restart: always

[landoop kafka topics UI](#)

10

- 10

10

10

10



10

version: '2'

services:

<https://github.com/confluentinc/schema-registry>

confluent-schema-registry:

image: confluentinc/cp-schema-registry:3.2.1

network_mode: host

environment:

SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL: **zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka**

SCHEMA_REGISTRY_LISTENERS: <http://0.0.0.0:8081>

please replace this setting by the IP of your web tools server

SCHEMA_REGISTRY_HOST_NAME: "54.206.91.106" **!! установить public IP этого сервера, на который я буду ставить эти тулы**

restart: always

<https://github.com/confluentinc/kafka-rest>

confluent-rest-proxy:

image: confluentinc/cp-kafka-rest:3.2.1

network_mode: host

environment:

KAFKA_REST_BOOTSTRAP_SERVERS: **kafka1:9092,kafka2:9092,kafka3:9092**

KAFKA_REST_ZOOKEEPER_CONNECT: **zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka**

KAFKA_REST_LISTENERS: <http://0.0.0.0:8082/>

KAFKA_REST_SCHEMA_REGISTRY_URL: <http://localhost:8081/>

please replace this setting by the IP of your web tools server

KAFKA_REST_HOST_NAME: "54.206.91.106" **!! установить public IP этого сервера, на который я буду ставить эти тулы**

depends_on:

- confluent-schema-registry

restart: always

<https://github.com/Landoop/kafka-topics-ui>

kafka-topics-ui:

image: **landoop/kafka-topics-ui:0.9.2**

network_mode: host

environment:

KAFKA_REST_PROXY_URL: <http://localhost:8082>

PROXY: "TRUE"

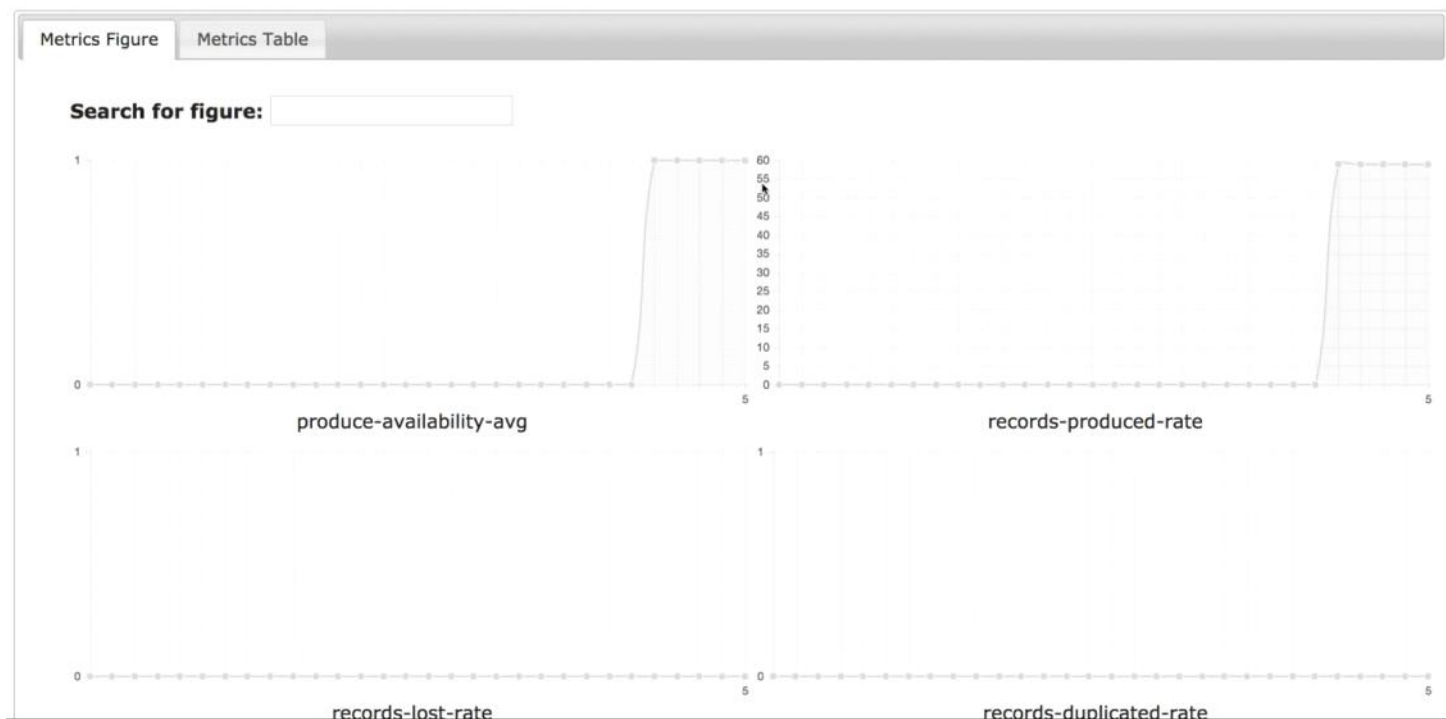
depends_on:

- confluent-rest-proxy

restart: always

linkedin kafka monitor

Kafka Monitor GUI



Search or jump to...

Pull requests Issues Marketplace Explore

linkedin / kafka-monitor

Watch 118 Unstar 936 Fork 230

Code Issues Pull requests Projects Wiki Insights

Monitor the availability of Kafka clusters with generated messages.

67 commits 6 branches 1 release 14 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

Commit	Message	Latest commit	Time ago
lindong28 Upgrade to kafka 1.0.1 (#127)	Update kafka-monitor-start.bat to use the correct class path (#67)	f37f8e4	29 days ago
bin	Add Kafka Monitor code		a year ago
checkstyle	Export to prometheus supported format using jmx exporter with this co...		2 years ago
config	Docker build hardening (#79)		2 months ago
docker	Update figures in the wiki to replace Test with App		a year ago
docs/images	Upgrade Gradle to 3.2.1 (#39)		a year ago
gradle/wrapper	Upgrade to kafka 1.0.1 (#127)		29 days ago
src	Allow producer and consumer props to be passed to services as a map I...		2 years ago
webapp			

```
[ec2-user@ip-172-31-31-225 kafka-monitor]$ ll
total 60
drwxrwxr-x 3 ec2-user ec2-user 134 Aug 8 09:11 bin
drwxrwxr-x 7 ec2-user ec2-user 90 Aug 8 09:12 build
-rw-rw-r-- 1 ec2-user ec2-user 1618 Aug 8 09:11 build.gradle
drwxrwxr-x 2 ec2-user ec2-user 28 Aug 8 09:11 checkstyle
drwxrwxr-x 2 ec2-user ec2-user 134 Aug 8 09:11 config
drwxrwxr-x 2 ec2-user ec2-user 94 Aug 8 09:11 docker
drwxrwxr-x 3 ec2-user ec2-user 20 Aug 8 09:11 docs
drwxrwxr-x 3 ec2-user ec2-user 21 Aug 8 09:11 gradle
-rwxrwxr-x 1 ec2-user ec2-user 4971 Aug 8 09:11 gradlew
-rw-rw-r-- 1 ec2-user ec2-user 2404 Aug 8 09:11 gradlew.bat
-rw-rw-r-- 1 ec2-user ec2-user 28832 Aug 8 09:11 LICENSE
-rw-rw-r-- 1 ec2-user ec2-user 1371 Aug 8 09:11 NOTICE
-rw-rw-r-- 1 ec2-user ec2-user 6277 Aug 8 09:11 README.md
-rw-rw-r-- 1 ec2-user ec2-user 0 Aug 8 09:11 settings.gradle
drwxrwxr-x 4 ec2-user ec2-user 30 Aug 8 09:11 src
drwxrwxr-x 3 ec2-user ec2-user 34 Aug 8 09:11 webapp
[ec2-user@ip-172-31-31-225 kafka-monitor]$ ll config/
total 20
-rw-rw-r-- 1 ec2-user ec2-user 5251 Aug 8 09:11 kafka-monitor.properties
-rw-rw-r-- 1 ec2-user ec2-user 1339 Aug 8 09:11 log4j.properties
-rw-rw-r-- 1 ec2-user ec2-user 3788 Aug 8 09:11 multi-cluster-monitor.properties
-rw-rw-r-- 1 ec2-user ec2-user 183 Aug 8 09:11 prometheus-exporter.yaml
[ec2-user@ip-172-31-31-225 kafka-monitor]$
```

```
# Install Kafka-Monitor
sudo yum install -y git
git clone https://github.com/linkedin/kafka-monitor.git
cd kafka-monitor
sudo yum install -y java-1.8.0-openjdk-devel
./gradlew jar
./bin/kafka-monitor-start.sh config/kafka-monitor.properties
# Setup as SystemD component
sudo nano /etc/systemd/system/kafka-monitor.service
```

kafka-monitor.properties

```
{
  "single-cluster-monitor": {
    "class.name": "com.linkedin.kmf.apps.SingleClusterMonitor",
    "topic": "kafka-monitor-topic",
    "zookeeper.connect": "172.31.9.21:2181",
    "bootstrap.servers": "172.31.32.31:9092,172.31.9.32:9092,172.31.16.33:9092",
    "produce.record.delay.ms": 100,
    "topic-management.topicCreationEnabled": true,
    "topic-management.replicationFactor": 3,
    "topic-management.partitionsToBrokersRatio": 2.0,
    "topic-management.rebalance.interval.ms": 600000,
    "topic-management.topicFactory.props": {
    },
    "topic-management.topic.props": {
      "retention.ms": "3600000"
    },
    "produce.producer.props": {
      "client.id": "kmf-client-id"
    },
    "consume.latency.sla.ms": "20000",
    "consume.consumer.props": {
    }
  },

  "jetty-service": {
    "class.name": "com.linkedin.kmf.services.JettyService",
    "jetty.port": 8000
  },

  "jolokia-service": {
    "class.name": "com.linkedin.kmf.services.JolokiaService"
  },

  "reporter-service": {
    "class.name": "com.linkedin.kmf.services.DefaultMetricsReporterService",
    "report.interval.sec": 1,
    "report.metrics.list": [
      "kmf:type=kafka-monitor:offline-runnable-count",
      "kmf.services:type=produce-service,name=:produce-availability-avg",
      "kmf.services:type=consume-service,name=:consume-availability-avg",
      "kmf.services:type=produce-service,name=:records-produced-total",
      "kmf.services:type=consume-service,name=:records-consumed-total",
      "kmf.services:type=consume-service,name=:records-lost-total",
      "kmf.services:type=consume-service,name=:records-duplicated-total",
      "kmf.services:type=consume-service,name=:records-delay-ms-avg",
      "kmf.services:type=produce-service,name=:records-produced-rate",
      "kmf.services:type=produce-service,name=:produce-error-rate",
      "kmf.services:type=consume-service,name=:consume-error-rate"
    ]
  }
}
```

kafka-monitor.service

```
# /etc/systemd/system/kafka-monitor.service
[Unit]
Description=Kafka Monitor
After=network.target
```

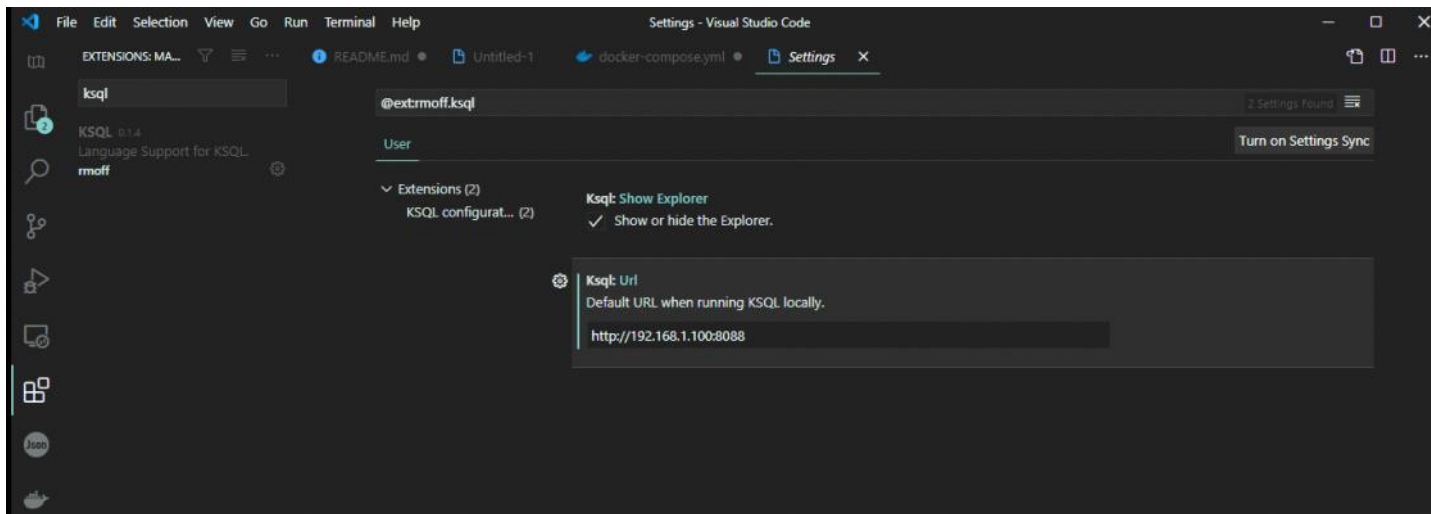
```
[Service]
User=ec2-user
Group=ec2-user
WorkingDirectory=/home/ec2-user/kafka-monitor
ExecStart=/home/ec2-user/kafka-monitor/bin/kafka-monitor-start.sh /home/ec2-user/kafka-monitor/config/kafka-monitor.properties
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```


GUI TOOLS 3

11 декабря 2020 г. 17:34

visual studio code + ksql extension



d

Overview of Monitoring Solutions



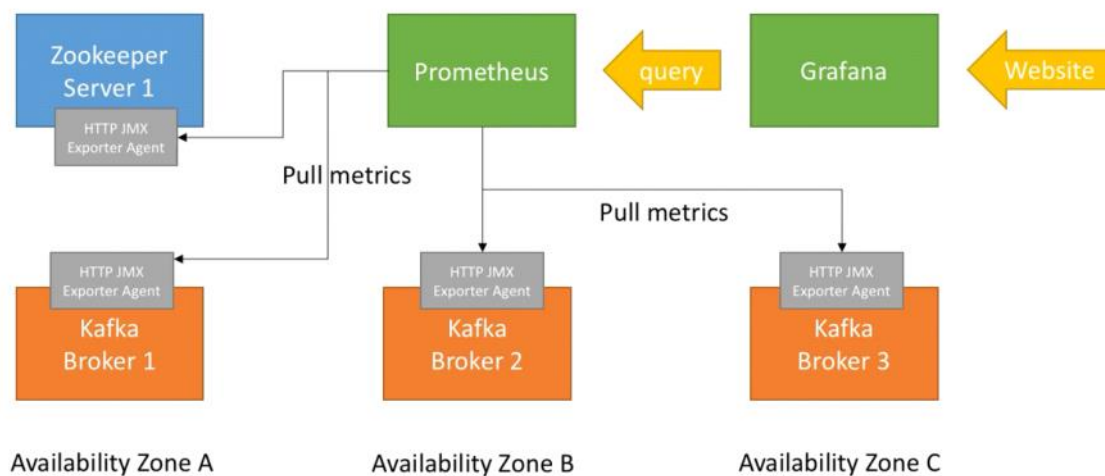
- Kafka is independent from your monitoring stack.
- By default, Kafka exposes its metrics using JMX (Java Management Extensions) and therefore most monitoring back-end should have a way to integrate with JMX directly or using an ETL process
- In this course, we go over the setup of some monitoring using Prometheus and Grafana, but the idea remains the same for whichever tool you are using.
- Self-monitoring available solutions on the market are:
 - **Prometheus + Grafana (the hands on solution for this course)**
 - ELK Stack (ElasticSearch + Kibana)
 - Confluent Control Centre (paid monitoring tool by Confluent)
 - LinkedIn Cruise Control (no UI, automated monitoring)
- As well as managed monitoring :
 - Datadog
 - NewRelic
 - Splunk
 - CloudWatch Monitoring (requires a lot of work to get it working)

Setup for admin & monitoring in this course



- Administration tools:
 - ZooNavigator UI: view Zookeeper nodes if we need to
 - Yahoo Kafka Manager: easily view brokers, topics, easily create topics
- Monitoring Tools:
 - LinkedIn Kafka Monitor: UI to visualize a continuously running producer / consumer and measure end-to-end latency
 - Prometheus: acquire + store metrics
 - Grafana: create dashboards and visualize metrics against Prometheus

Setting up Prometheus + Grafana Target Architecture



мониторинг работает через JMX exporter

Setting up Kafka on Prometheus

- Install JMX Exporter Agent on Kafka Broker
- Install Prometheus on Admin Machine as a Service
- View in Prometheus that the Kafka data is being pulled
- You'll have to setup the two other brokers and Zookeeper as an exercise!

GitHub, Inc. [US] | https://github.com/prometheus/jmx_exporter

Features Business Explore Marketplace Pricing Search Sign in or Sign up

prometheus / jmx_exporter Watch 60 Star 530 Fork 280

<> Code Issues 14 Pull requests 13 Projects 0 Insights

Join GitHub today
GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.
Sign up

A process for exposing JMX Beans via HTTP for Prometheus consumption

jmx prometheus mbean java-agent monitoring prometheus-exporter

211 commits 1 branch 12 releases 51 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

Commit	Message	Time
andynxng and brian-brazil	add username_and_password_to readme configuration example (#293)	Latest commit 365ae13 29 days ago
collector	Registering JMX Exporter build info (#279)	2 months ago
example_configs	wildfly: Whitelist objects for performance reason (#284)	2 months ago
jmx_prometheus_httpserver	Registering JMX Exporter build info (#279)	2 months ago
jmx_prometheus_javaagent	Fix typo in comment (#289)	2 months ago
.gitignore	Use simple HTTP Server (#162)	a year ago

```
GNU nano 2.3.1 File: /etc/systemd/system/kafka.service Modified
[Unit]
Description=Kafka
After=network.target

[Service]
User=ec2-user
Group=ec2-user
Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"
Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-$
ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka.properties
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

f

Getting Started

APIs

Kafka Streams

Kafka Connect

Configuration

Design

Implementation

Operations

Security

PERFORMANCE

POWERED BY

PROJECT INFO

ECOSYSTEM

CLIENTS

EVENTS

CONTACT US

APACHE

Download

@apachekafka

6.6 Monitoring

Kafka uses Yammer Metrics for metrics reporting in the server and Scala clients. The Java clients use Kafka Metrics, a built-in metrics registry that minimizes transitive dependencies pulled into client applications. Both expose metrics via JMX and can be configured to report stats using pluggable stats reporters to hook up to your monitoring system.

All Kafka rate metrics have a corresponding cumulative count metric with suffix `-total`. For example, `records-consumed-rate` has a corresponding metric named `records-consumed-total`.

The easiest way to see the available metrics is to fire up `jconsole` and point it at a running kafka client or server; this will allow browsing all metrics with JMX.

We do graphing and alerting on the following metrics:

DESCRIPTION	MBean NAME	NORMAL VALUE
Message in rate	kafka.server:type=Broker-TopicMetrics,name=MessagesInPerSec	
Byte in rate from clients	kafka.server:type=Broker-TopicMetrics,name=BytesInPerSec	
Byte in rate from other brokers	kafka.server:type=Broker-TopicMetrics,name=ReplicationBytesInPerSec	
Request rate	kafka.network:type=RequestMetrics,name=RequestsPerSec,request={Produce FetchConsumer FetchFollower}	
Error rate	kafka.network:type=RequestMetrics,name=ErrorsPerSec,request={[-.\w +],error={[-.\w +}	Number of errors in responses counted per-request-type, per-error-code. If a response contains multiple errors, all are counted. <code>error=NONE</code> indicates suc...

confluent

Product Cloud Developers Blog Docs Download

» Monitoring Kafka

• Server Metrics

• Producer Metrics

• New Consumer Metrics

• Old Consumer Metrics

• Confluent Metrics Reporter

Confluent Cloud

Confluent Control Center

Multi-Datacenter Replication

Schema Registry Operations

Kafka REST Proxy Operations

Docker Operations

Security

Version 5.0.0 » Operations » Kafka Operations »

Monitoring Kafka

Apache Kafka brokers and clients report many internal metrics. JMX is the default reporter, though you can add any pluggable reporter.

You can deploy [Confluent Control Center](#) for out-of-the-box Kafka cluster monitoring so you don't have to build your own monitoring system. Control Center makes it easy to manage the entire Confluent Platform. Control Center is a web-based application that allows you to manage your cluster, to monitor Kafka [system health](#) in predefined dashboards and to alert on triggers. Additionally, Control Center reports end-to-end [stream monitoring](#) to assure that every message is delivered from producer to consumer, measures how long messages take to be delivered, and determines the source of any issues in your cluster.

Refer to [Optimizing Your Apache Kafka Deployment](#) whitepaper for a practical guide to optimizing your Apache Kafka deployment for various services goals including throughput, latency, durability and availability, and useful metrics to monitor for performance and cluster health.

* consumer lag

2 февраля 2021 г. 13:19

производительность как продюсера, так и потребителя зависит от пропускной способности

- В плане генераторов нас интересует в основном скорость отправки сообщений брокеру. Разумеется, чем выше пропускная способность — тем лучше.
- В плане потребителей нас также интересует производительность, а именно: насколько быстро мы можем читать сообщения от брокера.

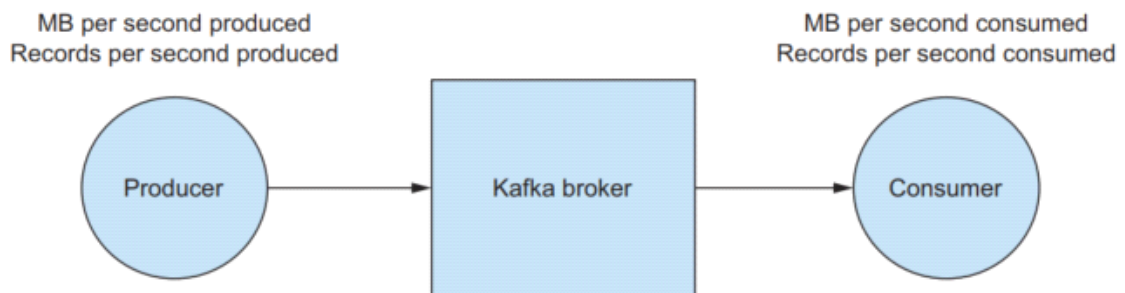


Figure 7.1 Kafka producer and consumer performance concerns, writing to and reading from a broker

consumer lag - другой показатель производительности потребителя: отставание потребителя

- Но существует и другой показатель производительности потребителя: отставание потребителя
- Как видите, нас интересует, какой объем сообщений и насколько быстро могут наши генераторы отправлять брокеру, а одновременно и то, насколько быстро наши потребители могут прочитать сообщения с него.
- Разница между скоростью записи генераторами сообщений на брокер и скоростью чтения потребителями сообщений с него называется отставанием потребителя (consumer lag).
-

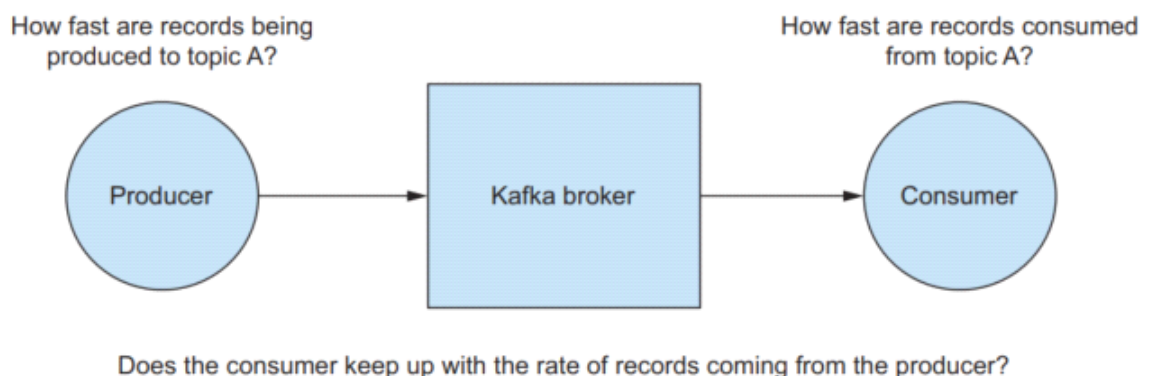


Figure 7.2 Kafka producer and consumer performance revisited

- что отставание потребителя представляет собой разницу между последним зафиксированным

потребителем смещением и последним смещением записанного на брокер сообщения.

- Некоторое отставание потребителя неизбежно, но в идеале потребитель наверстывает упущенное или в крайней мере отставание не растет.

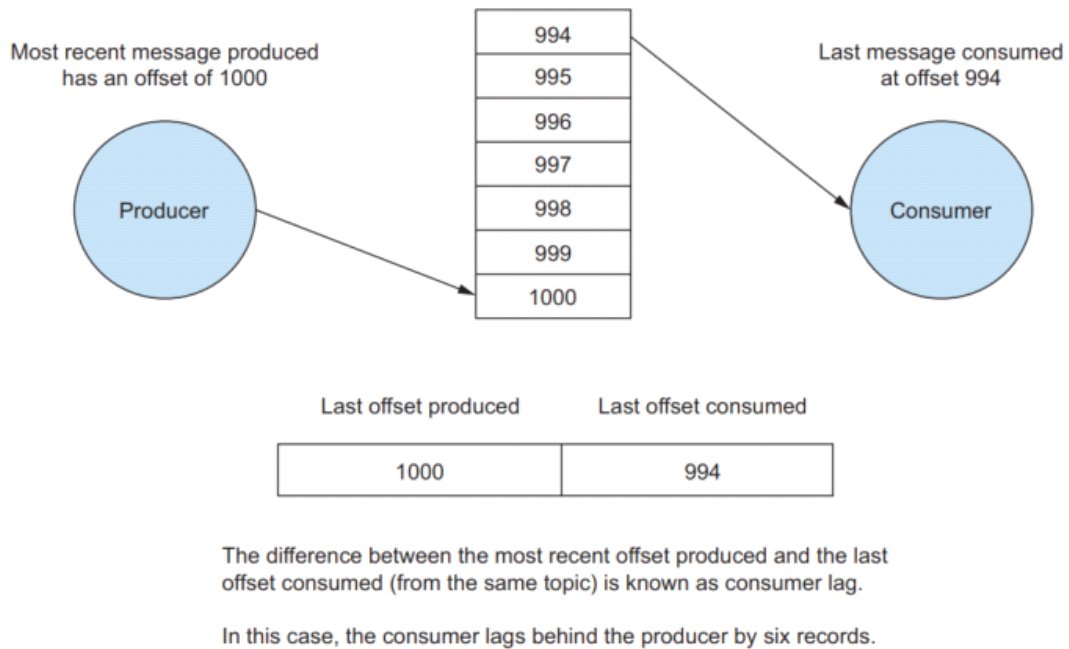


Figure 7.3 Consumer lag is the difference in offsets committed by the consumer and offsets written by the producer

когда отставание растет со временем — это признак того, что потребителю требуется больше ресурсов

- Небольшое либо постоянное отставание вполне нормально, но когда отставание растет со временем — это признак того, что потребителю требуется больше ресурсов

как вычислить consumer lag

Во-первых, воспользуемся командой `list` для вывода списка всех активных групп потребителей. Результаты ее выполнения приведены на рис. 7.4.

```
<kafka-install-dir>/bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--list
```

```
oddball:bin bbejeck$ ./kafka-consumer-groups.sh --list --bootstrap-server localhost:9092
Note: This will only show information about consumers that use the Java consumer API (non-ZooKeeper-based consumers).
console-consumer-59026
```

Рис. 7.4. Вывод перечня доступных групп потребителей из командной строки

Получив эту информацию, можно выбрать название группы потребителей и выполнить следующую команду:

```
<kafka-install-dir>/bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--group <GROUP-NAME> \
--describe
```

Результаты (состояние работы потребителя) показаны на рис. 7.5.

Количество прочитанных сообщений = 3 Количество отправленных в топик сообщений = 10

```
oddball:bin bbejeck$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group console-consumer-59026 --describe
Note: This will only show information about consumers that use the Java consumer API (non-ZooKeeper-based consumers).
Consumer group 'console-consumer-59026' has no active members.
TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG           CONSUMER-ID     HOST                CLIENT-ID
consumer_log_demo  0           3               10              7             -               -                  -
```


TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSUMER-ID	HOST	CLIENT-ID
consumer_log_demo	0	3	10	7	-	-	-

$10 \text{ (отправлено сообщений)} - 3 \text{ (прочитано сообщений)} = 7 \text{ (отставание на такое число записей)}$

Рис. 7.5. Состояние группы потребителей

Эти результаты показывают, что отставание потребителя невелико. Наличие отставания потребителя не всегда означает наличие проблем — потребители читают сообщения пакетами и не получают следующий пакет до окончания обработки текущего. Обработка записей занимает определенное время, так что небольшое отставание неудивительно.

* interceptors

2 февраля 2021 г. 13:20

consumer interceptors - мониторинга (перехвата) информации о поведении клиентов (генераторов и потребителей).

- раче Kafka, KIP-42: Add Producer and Consumer Interceptors («KIP-42: Добавление перехватчиков для генераторов и потребителей»), <http://mng.bz/g8oX>.
- Перехватчики принимают в качестве параметра возвращаемый с брокера внутри метода `Consumer.poll()` объект `consumerRecords` и получают возможность выполнить над ним любые операции, включая фильтрацию и модификацию, до того как `KafkaConsumer` вернет эти записи из метода `poll`.

```
ConsumerRecords<String, String> poll(long timeout) {  
    ConsumerRecords<String, String> consumerRecords =  
    ➡ ...consuming records  
    return interceptors.onConsume(consumerRecords);  
}
```

The diagram illustrates the process of fetching records and running them through the interceptor chain. An arrow points from the text "Fetches new records from the broker" to the line `...consuming records`. Another arrow points from the text "Runs records through the interceptor chain and returns the results" to the line `return interceptors.onConsume(consumerRecords);`.

- `ConsumerInterceptor` указываются через ключ конфигурации `ConsumerConfig.INTERCEPTOR_CLASSES_CONFIG` и принимают на входе объект `Collection`, состоящий из одного или нескольких классов реализации `ConsumerInterceptor`. При этом несколько перехватчиков соединяются цепочкой и выполняются в указанном в настройках порядке. `ConsumerInterceptor` принимает в качестве параметра и возвращает экземпляр `ConsumerRecords`. В случае нескольких перехватчиков возвращаемый из одного перехватчика объект `ConsumerRecords` служит входным параметром для следующего перехватчика в цепочке. Таким образом, все произведенные одним перехватчиком изменения передаются далее по цепочке.

PRODUCER INTERCEPTOR

- ProducerInterceptor ведет себя аналогично ConsumerInterceptor, у него также есть две точки доступа: ProducerInterceptor.onSend() и ProducerInterceptor.onAcknowledgement().
- С помощью метода onSend() перехватчик может выполнить любое нужное действие, включая изменение объекта ProducerRecord. Каждый из перехватчиков генераторов в цепочке получает объект, возвращаемый предыдущим перехватчиком.

Listing 7.2 Logging producer interceptor

```
public class ZMartProducerInterceptor implements
➤ ProducerInterceptor<Object, Object> {
    // some details left out for clarity
    private static final Logger LOG =
➤ LoggerFactory.getLogger(ZMartProducerInterceptor.class);

    @Override
    public ProducerRecord<Object, Object> onSend(ProducerRecord<Object,
➤ Object> record) {
        LOG.info("ProducerRecord being sent out {} ", record);
        return record;
    }

    @Override
    public void onAcknowledgement(RecordMetadata metadata, Exception exception) {
        if (exception != null) {
            LOG.warn("Exception encountered producing record {}",
➤ exception);
        } else {
            LOG.info("record has been acknowledged {} ", metadata);
        }
    }
}
```

Logs right before the message is sent to the broker

Logs broker acknowledgement or whether error occurred (broker-side) during the produce phase

When configuring interceptors in a Kafka Streams application, you need to prefix the consumer and producer interceptors' property names

- with
props.put(StreamsConfig.consumerPrefix(ConsumerConfig.INTERCEPTOR_CLASSES_CONFIG) and StreamsConfig.producerPrefix(ProducerConfig.INTERCEPTOR_CLASSES_CONFIG), respectively

Отмечу, что, поскольку перехватчики обрабатывают каждую запись в приложении Kafka Streams, объем выводимой журналирующими перехватчиками информации весьма значителен.

- Результаты работы перехватчиков выводятся в файлы consumer_interceptor.log и producer_interceptor.log, располагающиеся в подкаталоге logs корневого каталога установки исходного кода.

* Application metrics

2 февраля 2021 г. 13:21

категории метрик

- Thread metrics
 - Average time for commits, poll, process operations
 - Tasks created per second, tasked closed per second
- Task metrics
 - Average number of commits per second
 - Average commit time
- Processor node metrics
 - Average and max processing time
 - Average number of process operations per second
 - Forward rate
- State store metrics
 - Average execution time for put, get, and flush operations
 - Average number put, get, and flush operations per second

полный список метрик

- <https://docs.confluent.io/platform/current/streams/monitoring.html>

нужно знать точно, в каком месте возникает затор

- Когда речь заходит об оценке производительности приложения, вы можете получить представление о том, сколько времени нужно для обработки одной записи, да и оценка длительности сквозной задержки — неплохой показатель общей производительности. Но для повышения производительности нужно знать точно, в каком месте возникает затор

В Kafka Streams есть уже готовый механизм сбора метрик производительности

Чаще всего достаточно только указать несколько параметров конфигурации.

Поскольку сбор метрик сам по себе снижает производительность, существует два его уровня: INFO и DEBUG.

- Уровни метрик подобны уровням журналирования. При поиске причин проблемы или при наблюдении за поведением приложения необходимо больше информации, так что можно воспользоваться уровнем DEBUG. В остальное время вся информация не требуется и достаточно уровня INFO.
- Обычно в промышленной эксплуатации уровень DEBUG не используется, поскольку падение производительности оказалось бы слишком большим.
- Уровень сбора метрик по умолчанию — INFO.

Table 7.1 Metrics availability by levels

Metrics category	DEBUG	INFO
Thread	x	x
Task	x	
Processor node	x	
State store	x	
Record cache	x	

Listing 7.3 Updating the configs for DEBUG metrics

```
private static Properties getProperties() {
    Properties props = new Properties();
    props.put(StreamsConfig.CLIENT_ID_CONFIG,
    ➡ "metrics-client-id");
    ➡ props.put(ConsumerConfig.GROUP_ID_CONFIG,
    ➡ "metrics-group-id");
    ➡ props.put(StreamsConfig.APPLICATION_ID_CONFIG,
    ➡ "metrics-app-id");
    ➡ props.put(StreamsConfig.METRICS_RECORDING_LEVEL_CONFIG,
    ➡ "DEBUG");
    ➡ props.put(StreamsConfig.BootstrapServersConfig,
    ➡ "localhost:9092");
    return props;
}
```

Client ID

Group ID

Application ID

Sets the metrics recording level to DEBUG

Sets the connection for the brokers

Как получить доступ к собранным метрикам

для работы JMX требуется работающее в данный момент приложение, так что просматривать метрики мы будем во время работы приложения.

- Можно также получить доступ к метрикам программным способом. Пример этого вы найдете в файле `src/main/java/bbejeck/chapter_7/StockPerformanceStreamsAndProcessorMetricsApplication.java`
- JMX — стандартный способ наблюдения за поведением программ, запущенных на виртуальной машине Java
- достаточно просто подключиться с помощью утилит Java VisualVM (<http://mng.bz/euif>), JConsole (<http://mng.bz/Ea71>) или Java Mission Control (<http://mng.bz/Or5B>).

Here's the
process-rate value.

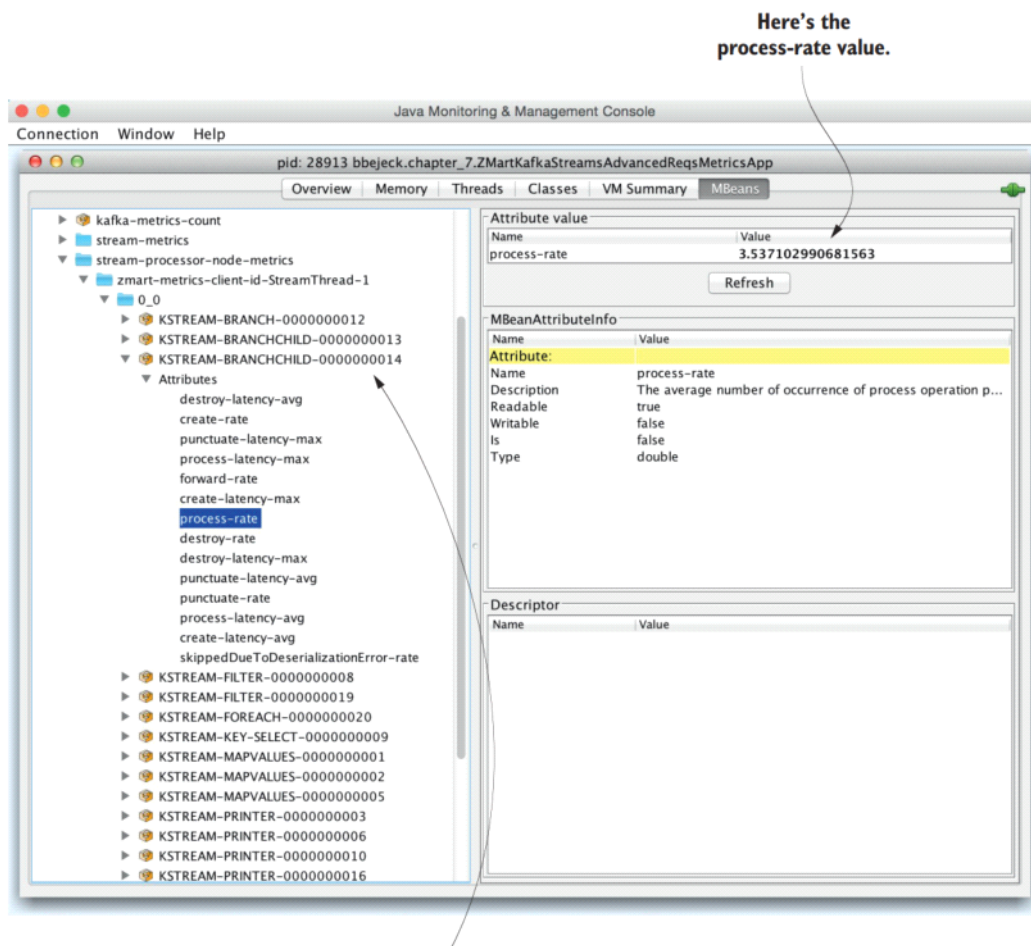


Figure 7.11 JConsole metrics for ZMart

* debug kafka streams

2 февраля 2021 г. 13:41

Метод `Topology.describe()` дает возможность просмотреть общую структуру приложения

- Он выводит информацию о структуре программы, включая все внутренние топики, созданные для целей повторного секционирования
- Как вы можете видеть, метод `Topology.describe()` выводит краткий, аккуратный обзор структуры приложения

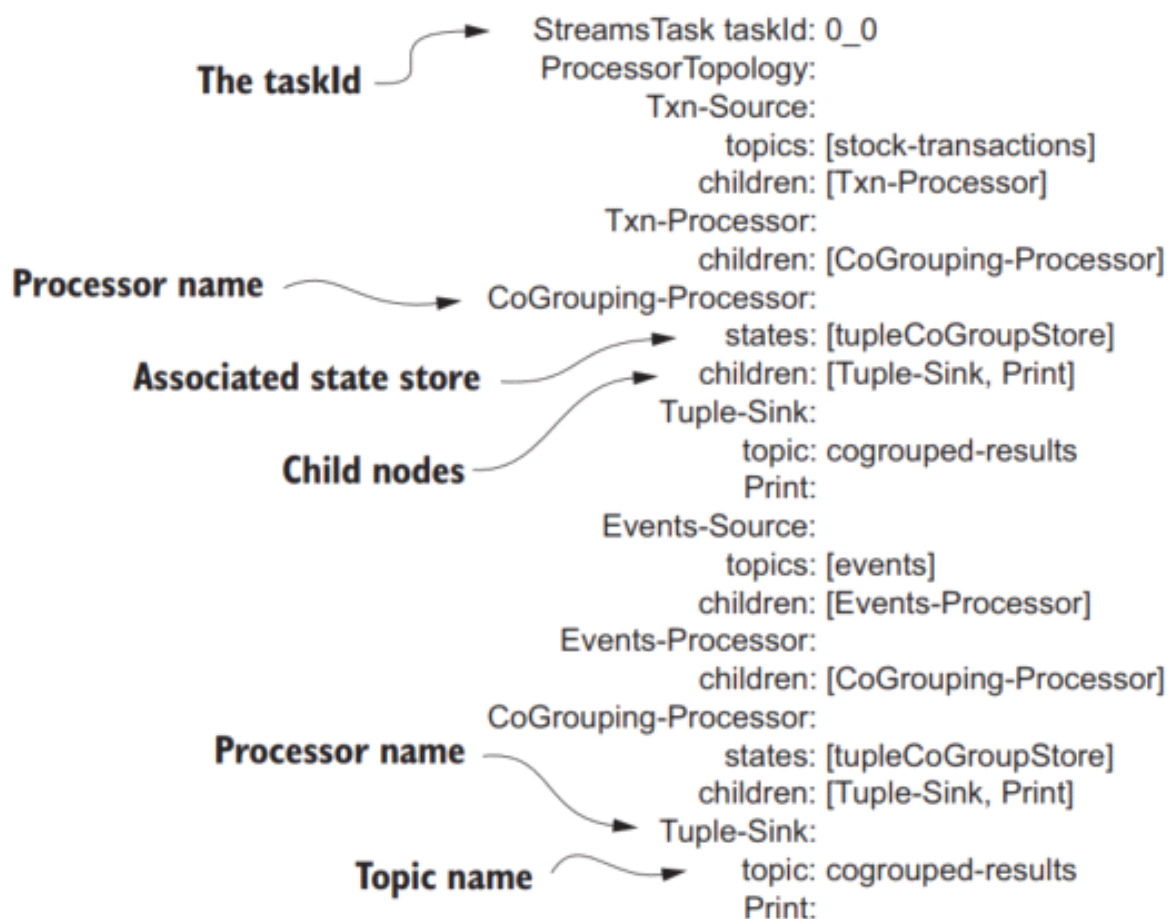


Figure 7.12 Displaying node names, associated child nodes, and other info

При работе с API DSL Kafka Streams класс `Topology` непосредственно не применяется, но к нему можно легко обратиться при необходимости. Если вам нужно вывести в консоль физическую топологию приложения, воспользуйтесь методом `StreamsBuilder.build()`, возвращающим объект `Topology`, а затем вызовите метод `Topology.describe()`, как мы вам только что показывали.

Для доступа к информации объектов `StreamThread` воспользуйтесь методом

Getting notification on various states of the application

- Запущенное приложение Kafka Streams не начинает сразу же обрабатывать данные — сначала нужно произвести определенную координацию действий. Потребитель должен извлечь информацию о метаданных и подписках, приложение должно запустить экземпляры StreamThread и распределить TopicPartition по StreamTask.
- Этот процесс распределения или перераспределения задач (рабочей нагрузки) называется перебалансировкой (rebalancing). Перебалансировка означает возможность автоматического вертикального масштабирования Kafka Streams в обе стороны. Это ключевое его преимущество — возможность добавить новые экземпляры приложения во время работы существующего приложения в расчете на то, что процесс перебалансировки перераспределит нагрузку.
- Например, допустим, что ваше приложение Kafka Streams содержит два узлаисточника, по две секции в каждом топике — итого четыре требующих распределения объекта TopicPartition. Сначала вы запускаете приложение с одним потоком выполнения. Kafka Streams определяет, сколько нужно создать задач, на основе максимального размера секции среди всех входных топиков. В данном случае у каждого топика две секции, так что максимум равен 2, следовательно, будет создано две задачи. Затем в процессе перебалансировки каждой из двух задач назначается по два объекта TopicPartition.
- После того как приложение поработало какое-то время, вы решаете, что хотите обрабатывать записи быстрее. Для этого вам достаточно запустить еще один экземпляр приложения с тем же идентификатором приложения, после чего процесс перебалансировки распределит нагрузку с учетом нового потока выполнения приложения. В результате две задачи окажутся распределенными по двум потокам выполнения.

Приложение Kafka Streams в каждый заданный момент времени может находиться в одном из шести состояний

- переходе из состояния выполнения в состояние перебалансировки. Это наиболее частый и сильнее всего влияющий на производительность тип перехода, поскольку в фазе перебалансировки не производится никакой обработки
- хотя возможность автоматической перебалансировки нагрузки — одна из сильных сторон Kafka Streams, **число перебалансировок лучше минимизировать**. Ведь во время перебалансировки данные не обрабатываются, а хотелось бы, чтобы наше приложение обрабатывало данные как можно больше времени.

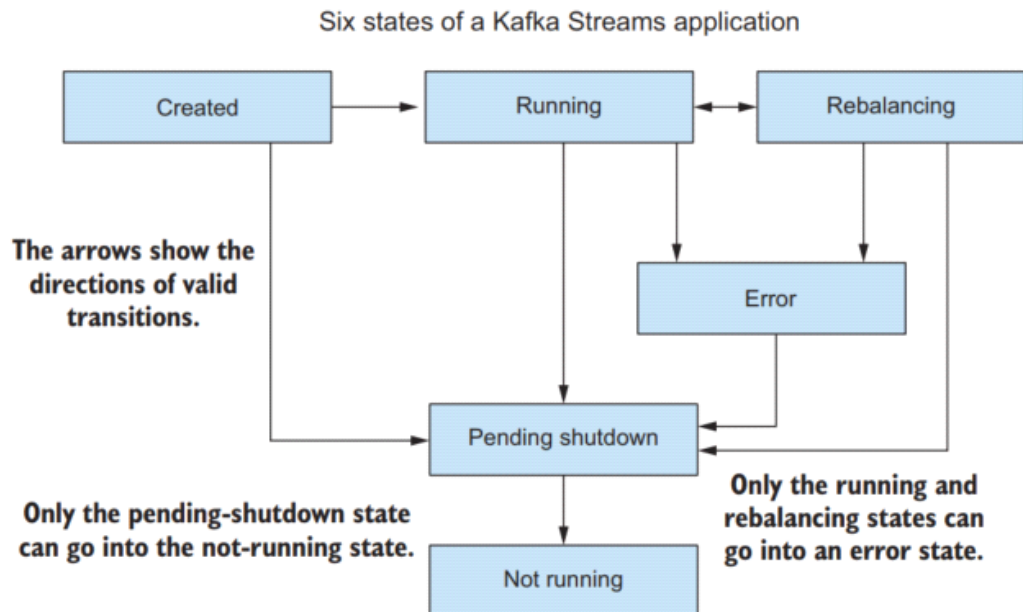


Figure 7.13 Possible states of a Kafka Streams application

Чтобы уловить эти смены состояния, мы воспользуемся методом `KafkaStreams.setStateListener`, принимающим на входе экземпляр интерфейса `StateListener`.

Listing 7.4 Adding a state listener

```

KafkaStreams.StateListener stateListener = (newState, oldState) -> {
    if (newState == KafkaStreams.State.RUNNING &&
    ➤ oldState == KafkaStreams.State.REBALANCING) {
        LOG.info("Application has gone from REBALANCING to RUNNING ");
        LOG.info("Topology Layout {}",
    ➤ streamsBuilder.build().describe());
    }
};
  
```

Prints out the structure of the topology

Checks that you're transitioning from REBALANCING to RUNNING

TIP Listing 7.4, running `ZMartKafkaStreamsAdvancedReqsMetricsApp.java`, involves viewing JMX metrics and the state-transition notification, so I've turned off printing the streaming results to the console. You're writing solely to Kafka topics. When you run the app, you should see the listener output in the console.

Let's take this example a little further and show how to signal when the application is going into a rebalancing state. You can update your code to handle this additional state transition as follows (found in `src/main/java/bbejeck/chapter_7/ZMartKafkaStreamsAdvancedReqsMetricsApp.java`).

Listing 7.5 Updating the state listener when REBALANCING

```
KafkaStreams.StateListener stateListener = (newState, oldState) -> {
    if (newState == KafkaStreams.State.RUNNING &&
    ➡ oldState == KafkaStreams.State.REBALANCING) {
        LOG.info("Application has gone from REBALANCING to RUNNING ");
        LOG.info("Topology Layout {}", streamsBuilder.build().describe());
    }

    if (newState == KafkaStreams.State.REBALANCING) {
        LOG.info("Application is entering REBALANCING phase");
    }
};
```

← Adds an action when entering the rebalancing phase