Arenadata™ Streaming

Версия - v1.0.0

Руководство по работе с Arenadata Streaming

2018 Arenadata™

Оглавление

1 Введение и основные понятия **3**
- 1.1 Введение в Arenadata Streaming . . . 3
- 1.2 Концепция хранения . . . 4
- 1.3 Гарантии . . . 7
- 1.4 Рекомендации по использованию . . . 7

2 Установка Arenadata Streaming с помощью Ambari 2.7.0 **9**
- 2.1 Настройка удаленного репозитория YUM . . . 9

3 Подготовка к установке Arenadata Streaming MPack **12**

4 Архитектурные особенности **13**
- 4.1 Обзор . . . 13
- 4.2 Персистентность . . . 13
- 4.3 Производительность . . . 15
- 4.4 Поставщики данных . . . 16
- 4.5 Потребители данных . . . 16
- 4.6 Механизм доставки сообщений . . . 19
- 4.7 Репликация . . . 20
- 4.8 Сжатие журналов . . . 22

5 Настройки платформы **26**
- 5.1 Настройка брокера . . . 26
- 5.2 Настройка на уровне топика . . . 56
- 5.3 Конфигурирование Producer . . . 62
- 5.4 Конфигурирование Consumer . . . 70
- 5.5 Конфигурирование Connect . . . 79
- 5.6 Конфигурирование Streams . . . 79

6 Новые функциональности **86**

В документе приведено описание платформы Arenadata Streaming: концепции хранения, архитектурные особенности, APIs, настройки платформы.

Документ может быть полезен администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих внедрение и сопровождение кластеров Arenadata.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Important: В настоящий момент автоматическая установка Arenadata Streaming доступна только через Ambari версии не ниже 2.7.0

Глава 1

Введение и основные понятия

Arenadata Streaming (ADS) – платформа распределенных потоковых операций

1.1 Введение в Arenadata Streaming

Arenadata Streaming (ADS) – платформа распределенных потоковых операций. Что именно это значит?

Платформа потоковой передачи данных имеет три ключевые возможности:

- Публикация и подписка на потоковую передачу данных, похожую на очередь сообщений или корпоративную систему обмена сообщениями;
- Хранение потоков записей отказоустойчивым способом;
- Обработка потоков записей по мере их возникновения.

ADS, как правило, используется для двух обширных классов приложений:

- Создание канала для потоковой передачи данных в реальном времени с целью надежного обмена данными между системами и приложениями;
- Создание приложений для потоковой передачи данных в реальном времени с целью их преобразования и взаимодействия с другими потоками.

Чтобы понять, как **Arenadata Streaming** все это осуществляет, следует погрузиться и изучить возможности **ADS** снизу вверх.

Несколько концепций **Arenadata Streaming**:

- **ADS** запускается как кластер на одном или нескольких серверах, которые могут располагаться в нескольких центрах обработки данных;
- В кластере **ADS** потоки записей хранятся по категориям, называемым *topics* (топики);
- Каждая запись состоит из ключа, значения и временной метки.

ADS имеет четыре основных **API** (API, Рис.1.1.):

- Producer API (поставщик) позволяет приложению публиковать поток записей по одному или нескольким топикам платформы;
- Consumer API (потребитель) позволяет приложению подписываться на один или несколько топиков и обрабатывать принадлежащие им потоки записей;

- Streams API позволяет приложению выступать в качестве *stream processor* (потокового процессора), потребляя входной поток данных из одного или нескольких топиков и создавая выходной поток данных так же для одного или нескольких топиков, эффективно преобразуя входные потоки в выходные;
- Connector API позволяет создавать и запускать повторное использование поставщиков и потребителей, которые связывают топики с существующими приложениями или системами данных. Например, коннектор для реляционной базы данных может записывать каждое изменение в таблицу.

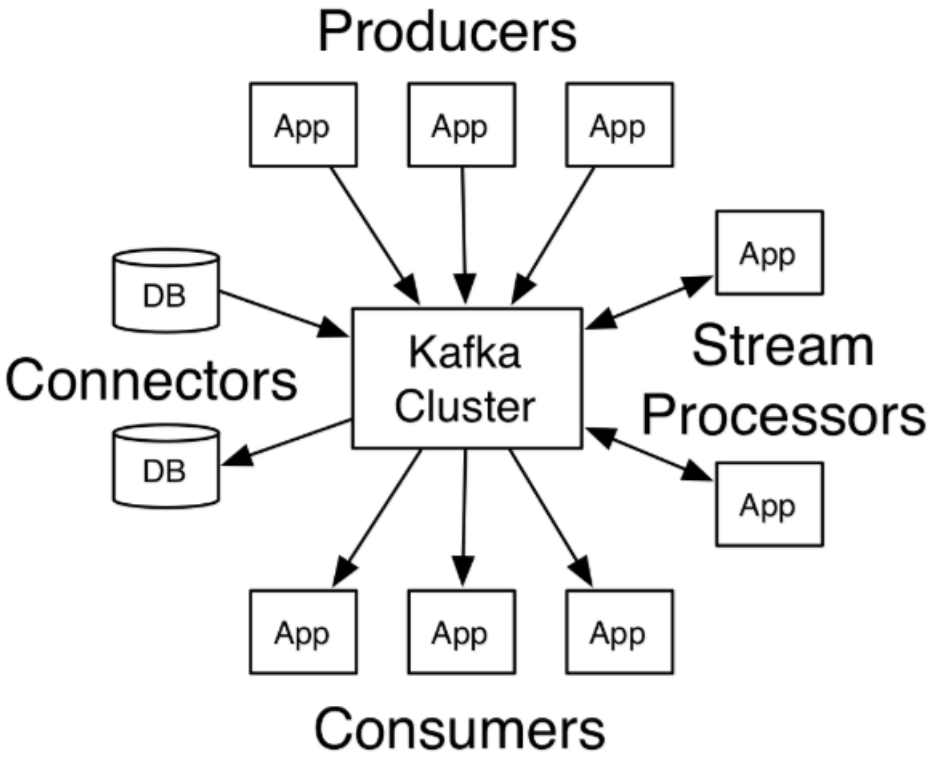


Рис.1.1.: API к платформе ADS

В **ADS** связь между клиентами и серверами осуществляется с помощью простого, высокопроизводительного протокола **TCP**. Протокол является версионным и поддерживает обратную совместимость с предыдущими версиями. **Arenadata** предоставляет Java-клиент для **ADS**, при этом клиенты доступны на многих языках программирования (ссылка на https://cwiki.apache.org/confluence/display/KAFKA/Clients).

1.2 Концепция хранения

Для начала в главе описывается основная абстракция, которую **ADS** обеспечивает для потока записей – топик.

Топики – это категории, по которым записи публикуются в платформе. В **ADS** топики могут иметь нескольких потребителей, которые подписываются на получение находящихся в них данных. Для каждого топика платформа поддерживает партиционированный журнал, схематично представленный на Рис.1.2.

Каждая партиция представляет собой упорядоченную неизменяемую последовательность записей, которая постоянно добавляется в структурированный журнал. Каждой записи в партиции присваивается порядковый номер *id*, называемый смещением (*offset*), который однозначно идентифицирует каждую запись.

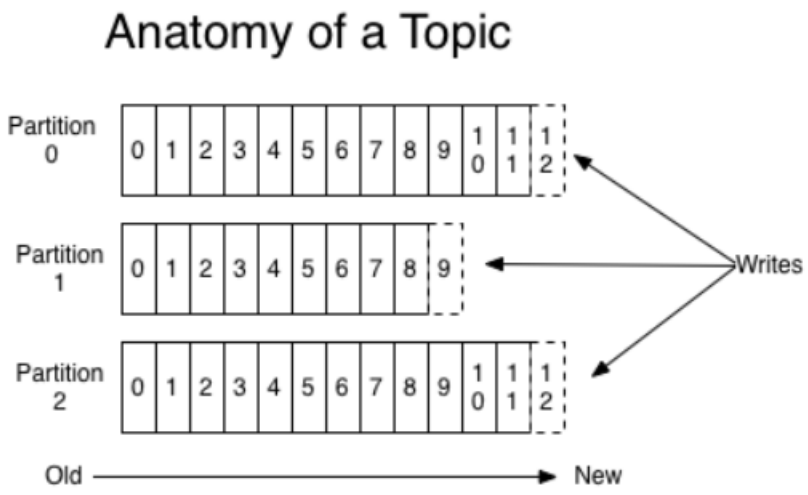


Рис.1.2.: Парицонированный журнал

Платформа **ADS** надежно сохраняет все опубликованные записи с настройкой периода их хранения. Например, если политика хранения установлена на два дня, то в течение двух дней после публикации запись доступна для потребления, после чего она удаляется с целью освобождения места. Производительность **ADS** фактически постоянна по отношению к размеру данных, поэтому хранение данных в течение длительного времени не является проблемой (Рис.1.3.).

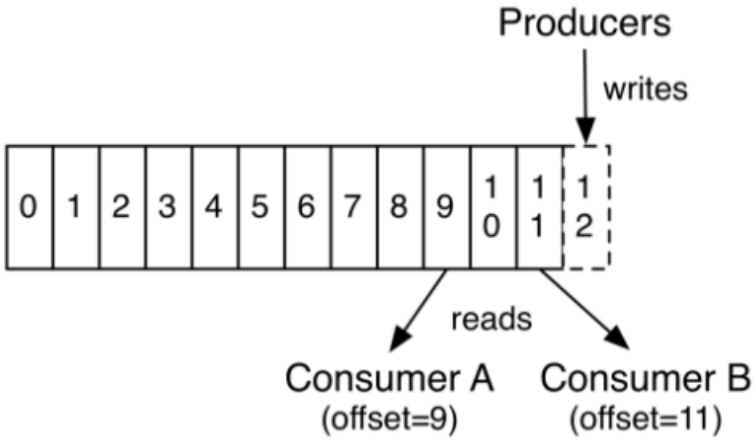


Рис.1.3.: Логика смещения

Фактически, метаданные, сохраненные для каждого потребителя, являются смещением или положением потребителя в журнале. Это смещение контролируется самим потребителем: обычно потребитель линейно продвигает свое смещение при считывании записи, но так как позиция контролируется потребителем, он может считывать записи в любом порядке. Например, потребитель может вернуться к более старому смещению для повторной обработки данных или перейти к самой последней записи и начать считывание с настоящего момента.

Такое сочетание функций означает, что потребители **ADS** могут приходить и уходить без особого влияния на кластер и на других потребителей. Например, можно использовать инструменты командной строки для считывания с конца содержимого любого топика без какого-либо влияния на то, что считывается другими потребителями.

Партиции в журнале служат нескольким целям. Во-первых, они позволяют журналу масштабироваться сверх размера, который помещается на одном сервере. Каждая отдельная партиция располагается на конкретном

сервере, но топик может иметь много партиций и располагаться на нескольких серверах для возможности обработки произвольного количества данных. Во-вторых, партиции действуют как единица параллелизма.

Партиции журнала распределяются по серверам кластера **ADS**, при этом каждый сервер обрабатывает данные и запросы чатсти партиций. Каждая партиция реплицируется на настраиваемое число серверов для обеспечения отказоустойчивости.

Каждая партиция имеет один сервер, который выступает в качетсве “лидера”. Лидер обрабатывает все запросы на чтение и запись для партиции, а остальные сервера пассивно реплицируют изменения лидера. Если лидер выходит из строя, один из брокеров автоматически становится новым лидером.

ADS MirrorMaker обеспечивает поддержку георепликации для кластеров. С помощью **MirrorMaker** сообщения реплицируются через несколько центров обработки данных или облачных сервисов. Это можно использовать в активных/пассивных сценариях резервного копирования и восстановления или в активных/активных сценариях для размещения данных ближе к пользователям или с целью поддержки требований к местоположению данных.

Поставщики публикуют данные по топикам по своему усмотрению и отвечают за выбор того, какую запись назначить для какой партиции. Это может быть сделано в циклическом режиме просто для балансировки нагрузки, или это может быть сделано в соответствии с какой-либо семантической функцией разбиения (например, на основе некоторого ключа в записи).

Потребители относят себя к группе потребителей, и каждая запись, опубликованная в топике, доставляется каждому инстансу потребителя, группа которого подписана на данный топик. При этом инстансы потребителя могут находиться на отдельных процессах или на отдельных машинах. Если все инстансы потребителя имеют одну и ту же группу, то записи эффективно балансируются. А в случае если инстансы потребителя имеют разные группы, то каждая запись передается во все потребительские процессы (Рис.1.4.).

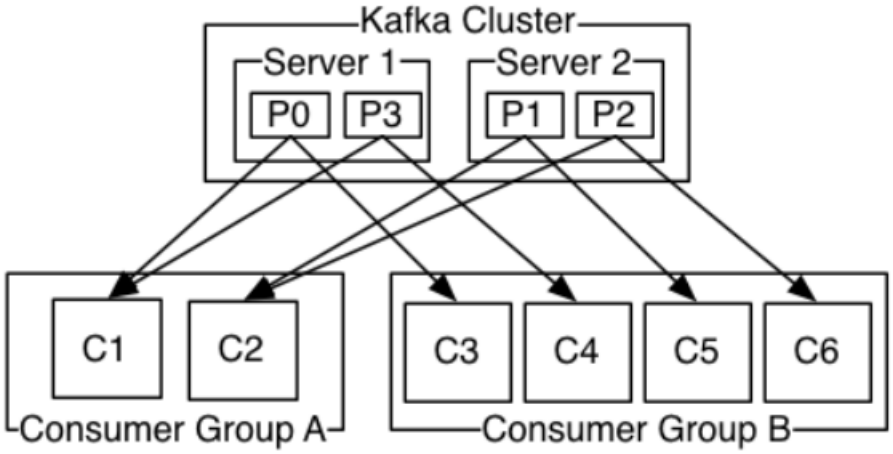


Рис.1.4.: Группы потребителей

На рисунке приведен пример двухсерверного кластера **ADS** с четырьмя партициями (*P0-P3*) и с двумя группами потребителей. Группа потребителей *A* имеет два экземпляра потребителей, группа *B* – четыре.

Чаще всего топики имеют небольшое количество групп потребителей – по одной для каждого “логического подписчика”. Каждая группа состоит из множества инстансов потребителей для обеспечения масштабируемости и отказоустойчивости. Это не что иное, как семантика “издатель-подписчик”, где подписчик представляет собой кластер потребителей, а не один процесс.

Реализация способа считывания в **ADS** заключается в разделении на партиции в журнале исходя из экземпляров потребителя, чтобы каждый экземпляр был исключительным потребителем “изрядной доли” партиций в любой момент времени. Процесс поддержания членства в группе динамически обрабатывается

протоколом **ADS**. Если к группе присоединяются новые экземпляры, они принимают некоторые партиции от других членов группы; если экземпляр удаляется, его партиции распределяются по остальным экземплярам.

ADS предоставляет только общий порядок записей внутри партиции, а не между партициями в топике. Упорядочивание по разделам в сочетании с возможностью разбиения данных по ключам для большинства приложений является достаточным. Однако если требуется полный порядок по записям, это может быть достигнуто с помощью топика, имеющего только одну партицию, хотя это будет означать только один потребительский процесс для каждой группы потребителей.

ADS можно развернуть как *multi-tenant* решение. Мультиарендный режим включается путем настройки, в какой топик могут записывать данные, а из какого считываться. Существует также операционная поддержка квот. Администраторы могут определять и применять квоты на запросы для управления ресурсами брокера, которые используются клиентами.

1.3 Гарантии

На высоком уровне **ADS** дает следующие гарантии:

- Сообщения, отправленные поставщиком в определенную партицию топика, добавляются в таком же порядке. То есть, если записи *M1* и *M2* отправляются одним поставщиком, и сначала отправляется *M1*, тогда *M1* имеет меньшее смещение, чем *M2*, и появляется в журнале раньше;
- Инстанс потребителя видит записи в том же порядке, в котором они хранятся в журнале;
- Для топика с коэффициентом репликации *N* допустимо до *N-1* сбоев сервера без потери записей, зафиксированных в журнале

1.4 Рекомендации по использованию

1.4.1 ADS как Messaging System

Как понятие о потоках **ADS** сравнивается с традиционной корпоративной системой обмена сообщениями?

Обмен сообщениями традиционно имеет две модели: “очередь” и “публикация-подписка”. В очереди группа потребителей может читать с сервера, и каждая запись переходит к одному из них; в модели “публикация-подписка” запись передается всем потребителям.

Каждая из этих двух моделей имеет свои плюсы и минусы. Преимущество организации очереди заключается в том, что она допускает разделение обработки данных на несколько инстансов потребителей, что позволяет масштабировать обработку. Но при этом очереди не являются многопотребительскими – как только один процесс считывает данные, они удаляются. Модель обмена сообщениями “публикация-подписка” позволяет передавать данные нескольким процессам, но при этом не имеет возможности масштабирования, так как каждое сообщение отправляется каждому подписчику.

Концепция группы потребителей в **ADS** обобщает эти две модели. Как и в случае с очередью, группа потребителей позволяет разделять обработку по совокупности процессов (по членам группы потребителей). А как в случае с моделью “публикация-подписка”, **ADS** позволяет передавать сообщения нескольким группам потребителей.

Преимущество модели **ADS** заключается в том, что каждый топик имеет оба этих свойства – он может масштабировать обработку, а также является многопотребитительским – необходимость выбора той или иной модели отпадает.

К тому же **ADS** имеет более сильные гарантии упорядочения, чем традиционная система обмена сообщениями.

Традиционная очередь сохраняет записи на сервере по порядку, и передача записей с сервера осуществляется в порядке их хранения на нем. Однако если считывание из очереди производится несколькими потребителями, то, несмотря на то, что сервер ведет записи по порядку, данные доставляются потребителям асинхронно. Фактически это означает, что при параллельном считывании упорядоченность записей теряется. Системы обмена сообщениями часто обходят эту проблему, используя понятие "исключительный потребитель", которое позволяет только одному процессу считывать из очереди, но это говорит о том, что параллелизм в обработке отсутствует.

ADS эта проблема решена. Имея понятие "параллелизм – партиция – в рамках топика", **ADS** может обеспечить как гарантии упорядоченности, так и балансировку нагрузки над группами потребительских процессов. Это достигается путем назначения партиций в топике для потребителей соответствующей группы, чтобы каждая партиция считывалась ровно одним потребителем в группе. При этом гарантируется, что потребитель является единственным читателем этой партиции и потребляет данные по порядку. Поскольку существует много партиций, нагрузка балансируется по всем экземплярам потребителя. Также следует обратить внимание, что в группе не может быть экземпляров потребителей больше, чем партиций.

1.4.2 ADS как Storage System

Любая очередь, которая позволяет публиковать сообщения, не связанные с их потреблением, эффективно может быть использована как система хранения *на лету* поступающих сообщений. Отличие платформы **ADS** заключается в том, что она является не только брокером сообщений, но и очень хорошей системой хранения данных.

Помещаемые в **ADS** данные записываются на диск и реплицируются для обеспечения отказоустойчивости. **ADS** позволяет поставщикам дождаться подтверждения операции, так что запись не считается полной, пока она не будет полностью реплицирована и гарантированно сохранена, даже если сервер вышел из строя.

Дисковые структуры **ADS** хорошо масштабируются: платформа функционирует одинаково, не зависимо от объема постоянных данных на сервере – будь то *50 КБ* или *50 ТБ*.

С целью серьезного использования **ADS** в качестве хранилища и предоставления клиентам возможности контролирования позиции чтения, платформу можно рассматривать как своего рода распределенную файловую систему специального назначения, предназначенную для высокопроизводительного хранения данных с низкой задержкой коммитов в журнал, а так же для их репликации и распространения.

1.4.3 ADS – Stream Processing

Недостаточно просто читать, записывать и хранить потоки данных, важно обеспечить обработку потоков в реальном времени.

В **ADS** потоковый процессор – это все, что принимает непрерывные потоки данных из входных топиков, выполняет некоторую обработку и создает непрерывные потоки данных в выходные топики.

Например, приложение розничной торговли может принимать входные потоки по продажам и отгрузкам и выводить поток с корректировками цен, вычисленными на основе входных данных.

Простую обработку можно выполнять напрямую, используя API-интерфейсы поставщика и потребителя. Однако для более сложных преобразований **ADS** предоставляет полностью интегрированный Streams API. Он позволяет создавать приложения с нетривиальной обработкой, выполняющей агрегацию потоков или объединяющей их вместе. Это помогает решить сложные проблемы, с которыми сталкивается подобный тип приложений: обработка неупорядоченных данных, переработка входных данных при изменении кода, выполнение вычислений с учетом состояния и т.д.

Streams API построен на основных примитивах **ADS**: он использует API-интерфейс поставщика и потребителя для ввода, использует Kafka для хранения состояний и применяет механизм группы для обеспечения отказоустойчивости среди инстансов потокового процессора.

Глава 2

Установка Arenadata Streaming с помощью Ambari 2.7.0

Установка сервера **Ambari 2.7.0** проходит в несколько этапов:

- Настройка удаленного репозитория YUM;
- Установка сервера Ambari;
- Настройка сервера Ambari;
- Запуск сервера Ambari.

2.1 Настройка удаленного репозитория YUM

Настройка удаленного репозитория не отличается от настройки любого дополнительного репозитория **YUM**. Для добавления репозитория необходимо выполнить от имени *root* команду:

```
yum-config-manager --add-repo http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.7.0.0/a
```

2.1.1 Установка сервера Ambari

Сервер **Ambari** устанавливается из RPM-пакета по команде **YUM**:

```
yum install ambari-server
```

Данная команда устанавливает сервер **Ambari**, являющийся сервером веб-приложений, на порт *8080*. Также устанавливает инстанс сервера **PostgreSQL** на порт *5432*.

2.1.2 Настройка сервера Ambari

Сервер **Ambari** необходимо настроить для корректной работы.

В случае если инстанс **PostgreSQL** настроен на порт по умолчанию, следует выполнить следующую команду:

```
ambari-server setup
```

В процессе настройки необходимо указать или принять по умолчанию параметры:

- *Учетная запись пользователя* – для запуска Ambari-сервера можно выбрать любую учетную запись (необязательно выполнять вход от *root*). В случае если пользователя не существует, он создается автоматически;
- *Java JDK* – для загрузки Oracle JDK 1.8 необходимо ввести значение *1* и принять лицензию Oracle JDK для загрузки файлов из Oracle. При этом установка JDK выполняется автоматически;
- *База данных* – выбор базы данных:

  `Enter advanced database configuration`

  В командной строке необходимо ответить *n* или *y*:

  – *n* – для использования с Ambari стандартной встроенной базы данных PostgreSQL. По умолчанию для базы данных PostgreSQL устанавливается имя “ambari” и логин / пароль принимают значения *ambari / bigdata*.

  – *y* – при необходимости использования с Ambari уже существующей базы данных PostgreSQL, MySQL или Oracle вместо предлагаемой по умолчанию. Далее для выбранной базы данных необходимо указать параметры подключения (см. Приложение 1.).

2.1.3 Запуск сервера Ambari

После установки сервера **Ambari** запуск его осуществляется по команде:

```
ambari-server start
```

Для проверки статуса сервера необходимо использовать команду:

```
ambari-server status
```

Для остановки сервера необходимо использовать команду:

```
ambari-server stop
```

Сервер **Ambari** доступен на порту *8080*. По умолчанию для него установлена следующая учетная запись:

User: *admin*

Password: *admin*

Important: Рекомендуется сменить пароль после первого входа в систему

Для входа в веб-интерфейс **Ambari** необходимо в адресной строке браузера указать адрес сервера:

```
http://<адрес сервера>:8080
```

При этом запрашивается логин и пароль. После авторизации открывается веб-интерфейс **Ambari** (Рис.2.1.).

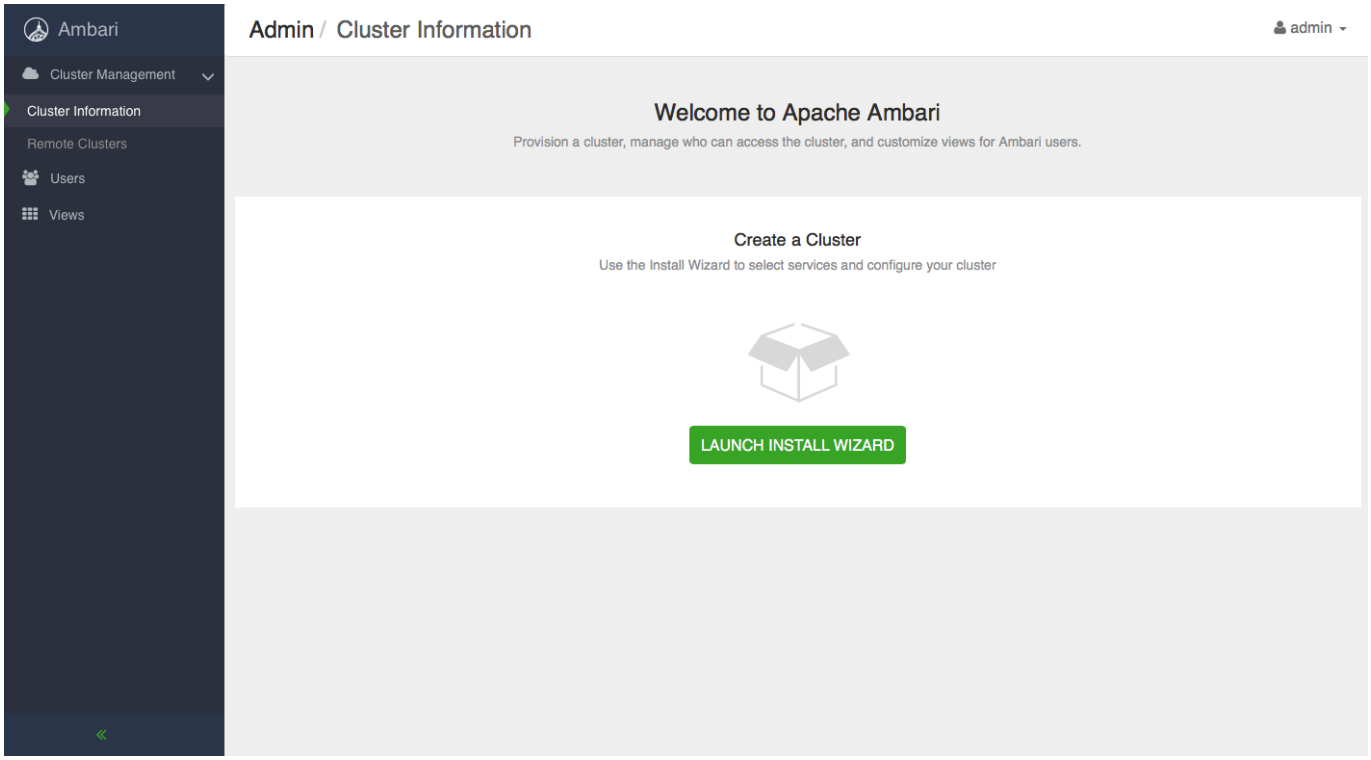


Рис.2.1.: Веб-интерфейс Ambari до настройки кластера

Глава 3

Подготовка к установке Arenadata Streaming MPack

Основные компоненты **ADS** устанавливаются с помощью Ambari Management Pack (MPack)

- Загрузка Arenadata Streaming MPack;
- Импортирование MPack в Ambari стек.

Необходимо загрузить и распаковать следующие архивы в выделенном для них месте (при этом следует избегать использования каталога */tmp*):

- *ads-ambari-mpack-{version}.tar.gz* – набор конфигураций севрвисов и скриптов для развертывания инастройки платформы через Ambari;

В случае если архивы загружены в каталог */tmp*, то для их распаковки в каталоге, например, */staging* необходимо выполнить следующую команду:

```
tar -xzvf /tmp/ads-ambari-mpack-{version}.tar.gz -C /staging/
```

Для испортирование стека ADS в **Ambari** необходимо выполнить следующие комманды:

1. Сохраните текущую папку Ambari Resources для возможного восстановления системы

   ```
   cp -r /var/lib/ambari-server/resources /var/lib/ambari-server/resources.backup
   ```

2. Запустите комманду импорта стека ADS

   ```
   ambari-server install-mpack --mpack=/staging/ads-ambari-mpack-{version}.tar.gz --purge
   --verbose
   ```

3. Перезааустите ambari

   ```
   ambari-server restart
   ```

Глава 4

Архитектурные особенности

4.1 Обзор

Разработанная компанией **Arenadata** платформа **ADS** имеет возможность выступления в качестве единой платформы для обработки всех потоков данных, которые может иметь крупная компания, в реальном времени. С этой целью продуман достаточно широкий набор вариантов использования:

- **ADS** должна иметь высокую пропускную способность для поддержки потоков событий большого объема, таких как агрегация журнала в режиме реального времени.
- **ADS** должна грамотно справляться с большими объемами данных, чтобы иметь возможность поддержки периодических загрузок данных из автономных систем.
- **ADS** должна обрабатывать передачу данных с низкой задержкой с целью обработки большого количества традиционных случаев использования сообщений.

Партицирование и потребительская модель были мотивированы желанием **Arenadata** поддерживать секционированную, распределенную обработку в реальном времени для создания новых унаследованных лент.

Наконец, в тех случаях, когда поток подается в сторонние системы данных для обслуживания, специалисты **Arenadata** понимали, что система должна быть в состоянии гарантировать отказоустойчивость при наличии сбоев в работе машины.

Поддержка перечисленных критериев использования привела **Arenadata** к разработке с несколькими уникальными элементами, более похожими на журнал базы данных, чем на традиционную систему обмена сообщениями.

4.2 Персистентность

ADS в значительной степени опирается на файловую систему для хранения и кэширования сообщений. Существует общее представление о том, что “диски медленные”, и это заставляет людей скептически относиться к тому, что персистентная структура может предложить конкурентоспособную производительность. На самом деле диски и намного медленнее и намного быстрее в зависимости от того, как они используются. И правильно разработанная структура диска часто может быть такой же быстрой, как и сеть.

Ключевым фактом о производительности диска является то, что пропускная способность жестких дисков отличается от латентности диска в течение последнего десятилетия. В результате производительность линейных записей в конфигурации **JBOD** с шестью массивами **SATA RAID-5** размером *7200 об/мин* составляет около *600 МБ/с*, но производительность операции случайной записи составляет всего около *100 к/сек* – разница более *6000X*. Эти линейные операции чтения и записи являются наиболее предсказуемыми для всех моделей

использования и сильно оптимизированы операционной системой. Современная ОС предоставляет технологии опережающего чтения и записи, которые обеспечивают предварительную выборку данных в больших кратных блоках и группируют меньшие логические записи в большие физические записи. Дальнейшее обсуждение этой проблемы можно найти в статье ACM Queue, где показывается, что последовательный доступ к диску может в некоторых случаях быть быстрее, чем случайный доступ к памяти.

Для компенсации расхождения в производительности, современные операционные системы становятся все более агрессивными в использовании основной памяти для кэширования диска. Современная ОС с радостью перенаправит всю свободную память на кэширование диска с небольшим снижением производительности при восстановлении памяти. Все операции чтения и записи на диск будут проходить через этот единый кэш. Данная функция не может быть легко отключена без использования прямого ввода-вывода, поэтому, даже если процесс поддерживает встроенный кэш данных, эти данные, вероятно, будут дублироваться в pagecache операционной системы, эффективно сохраняя все дважды.

Платформа **ADS** строится поверх **JVM**, и каждый, кто проводил какое-либо время с использованием памяти **Java**, знает две вещи:

- Накладные расходы памяти на объектах очень высоки, что часто удваивает размер хранимых данных;
- Сбор мусора Java становится все более неудобным и медленным по мере увеличения объема данных in-heap.

В результате этих факторов, используя файловую систему и полагаясь на pagecache, лучше поддерживать кэш in-memory или другую структуру – **Arenadata**, по крайней мере, удваивает доступный кэш, имея автоматический доступ ко всей свободной памяти и, примерно удваивает повторно, сохраняя компактную структуру байтов, а не отдельные объекты. Это приводит к кэшу до *28-30 ГБ* на машине *32 Гб* без ограничений сборщиков мусора. Кроме того, кэш остается теплым, даже при перезагрузке сервиса. В то время как кэш in-process необходимо перестраивать в памяти (что для кэша *10 ГБ* может занять до *10 минут*), иначе ему придется стартовать с полностью холодного кэша (что означает ужасную начальную производительность). К тому же значительно упрощается код, поскольку вся логика поддержания согласованности между кэшем и файловой системой теперь находится в ОС, которая имеет тенденцию делать это более эффективно и правильно, чем однократные попытки in-process. Если использование диска способствует линейному чтению, то опережающее чтение эффективно заполняет этот кэш полезными данными на каждом диске.

Предлагается очень простая структура: вместо того, чтобы содержать как можно больше данных in-memory, и потом в панике полностью очищать файловую систему, когда заканчивается пространство, мы инвертируем это – все данные сразу же записываются в постоянный журнал файловой системы без необходимости сброса на диск. По сути это просто означает, что данные переносятся в pagecache ядра.

Такой ориентированный на pagecache стиль описывается в статье Notes from the Architect.

Персистентная структура данных, используемая в системах обмена сообщениями, часто представляет собой очередь для каждого потребителя со связанным BTree или другими структурами данных произвольного доступа общего назначения для поддержки метаданных о сообщениях. BTrees являются наиболее универсальной структурой данных и позволяют поддерживать широкий спектр транзакционной и нетранзакционной семантики в системе обмена сообщениями. Они имеют довольно высокую стоимость, однако: Btree операции – $O\ (log\ N)$. Обычно $O\ (log\ N)$ считается эквивалентным постоянному времени, но это не относится к дисковым операциям. Диск выполняет поиск за *10 мс*, и каждый диск может делать только один поиск за раз, поэтому параллелизм ограничен. Следовательно, даже небольшое количество запросов на диск приводит к очень высоким накладным расходам. Поскольку системы хранения данных сочетают очень быстрые кэшированные операции с очень медленными физическими дисковыми операциями, наблюдаемая производительность древовидных структур часто суперлинейна по мере увеличения данных с фиксированным кэшем, то есть дублирование данных – это намного хуже по сравнению с вдвое медленной скоростью.

Интуитивно персистентная очередь может быть построена на простых операциях чтения и добавления к файлам, как это обычно бывает с решениями по ведению журнала. Такая структура имеет преимущество в том, что все операции – $O\ (1)$, и операции чтения не блокируют операции записи или друг друга. Это имеет очевидные преимущества в производительности, так как она полностью отделена от размера данных. Один сервер теперь может в полной мере использовать ряд дешевых и низкоскоростных дисков *1+TB SATA*. Хотя

у них низкая производительность поиска, она приемлема для многочисленных операций чтения и записи и достигает 1/3 цены и 3x емкости.

Имея доступ к практически неограниченному дисковому пространству без какого-либо снижения производительности **Arenadata** может предоставить некоторые функции, которые обычно не встречаются в системах обмена сообщениями. Например, в **ADS** вместо того, чтобы удалять сообщения сразу после их считывания, можно сохранять их в течение относительно длительного периода. Это приводит к большой гибкости для потребителей.

4.3 Производительность

Специалисты **Arenadata** приложили значительные усилия для повышения производительности платформы. Одним из основных вариантов использования является обработка данных веб-активности, которая очень велика: каждый просмотр страницы может генерировать десятки записей. Кроме того, каждое опубликованное сообщение читается как минимум одним потребителем (а часто – многими), поэтому есть стремление сделать потребление как можно более дешевым.

По опыту разработки и эксплуатации ряда аналогичных систем выяснилось, что производительность является ключом к эффективным многопользовательским операциям. Если нижестоящий сервис инфраструктуры может легко стать узким местом из-за незначительного увеличения по использованию приложения, то такие небольшие изменения часто создают проблемы. В результате приложение выйдет из строя под нагрузкой перед инфраструктурой. Это особенно важно при попытке запуска централизованного сервиса, поддерживающего десятки или сотни приложений в централизованном кластере, поскольку изменения в шаблонах использования происходят почти ежедневно.

После устранения неудачных шаблонов доступа к диску остается две общие причины неэффективности в подобном типе системы: слишком много мелких операций ввода-вывода и избыточное копирование байтов.

Проблема множества мелких операций ввода-вывода происходит как между клиентом и сервером, так и в собственных персистентных операциях сервера. Во избежание этого протокол **Arenadata** построен вокруг абстракции “набор сообщений”, группирующей сообщения. Это позволяет сетевым запросам объединять сообщения вместе, амортизировав накладные расходы в сети, а не отправлять каждый раз по одному сообщению. Сервер единоразово добавляет фрагменты сообщений в свой журнал, а потребитель извлекает большие линейные фрагменты за раз.

Такая простая оптимизация на порядок увеличивает скорость работы. Пакетирование приводит к увеличению сетевых пакетов, последовательных операций с дисками и смежными блоками памяти и т.д., что позволяет платформе **ADS** превращать поток случайных сообщений в линейные записи, которые поступают потребителям.

Другая непродуктивность заключается в копировании байтов. При низких скоростях передачи сообщений это не проблема, но под нагрузкой влияние значительно. Во избежание этого **ADS** использует стандартный бинарный формат сообщений, который совместно применяется поставщиком, брокером и потребителем (таким образом, блоки данных могут передаваться без изменений).

Журнал сообщений, поддерживаемый брокером, сам по себе является просто каталогом файлов, каждый из которых заполняется рядом наборов сообщений, которые были записаны на диск в том же формате, который используется поставщиком и потребителем. Поддержка общего формата позволяет оптимизировать наиболее важную операцию – сетевую передачу персистентных блоков журнала. Современные операционные системы unix предлагают высоко оптимизированный путь кода для передачи данных из pagecache в сокет; в Linux это делается с помощью системного вызова sendfile.

Путь передачи данных из файла в сокет заключается в следующем:

1. Операционная система считывает данные с диска в pagecache в пространстве ядра.
2. Приложение считывает данные из пространства ядра в буфер пространства пользователя.

3. Приложение записывает данные в пространство ядра в буфер сокета.
4. Операционная система копирует данные из буфера сокета в буфер сетевого адаптера по сети.

Такой метод явно неэффективен – он предполагает четыре операции копирования и два системных вызова. А при использовании sendfile повторное копирование исключается (zero-copy), позволяя ОС напрямую отправлять данные из pagecache в сеть. Таким образом, из оптимизированного пути требуется только последняя копия в буфер сетевого адаптера.

Как правило, общий вариант использования состоит из нескольких потребителей топика. При приведенной выше оптимизации данные копируются в pagecache только один раз и при каждом потреблении используются повторно, а не хранятся в памяти и копируются в пространство пользователя при каждом чтении. Это позволяет считывать сообщения со скоростью, приближенной к пределу сетевого подключения.

Такая комбинация pagecache и sendfile означает, что в кластере **ADS** нет активности чтения на дисках, поскольку данные полностью обслуживаются из кэша.

Дополнительные сведения о поддержке sendfile и zero-copy в **Java** приведены в статье.

4.4 Поставщики данных

4.4.1 Балансировка нагрузки

Поставщик отправляет данные непосредственно брокеру, являющемуся лидером партиции, без какого-либо промежуточного уровня маршрутизации. При этом все узлы **ADS** могут ответить на запрос метаданных, какие серверы активны, и где лидеры партиций топика находятся в любой момент времени, чтобы позволить поставщику соответствующим образом направлять свои заявки.

Клиент контролирует, в какую партицию публикуются сообщения. Публикация данных может осуществляться произвольным образом, реализуя некую случайную балансировку нагрузки, либо с помощью некоторой функции семантического разбиения, предоставленной интерфейсом **ADS**, которая позволяет пользователю указывать ключ для секционирования, и использует этот хэш в партиции (при необходимости функцию можно переопределить). Например, если выбранный ключ является id пользователя, то все данные по конкретному пользователю публикуются в эту партицию. В свою очередь, это позволяет потребителям делать выводы относительно их потребления. Данный стиль партицирования явно разработан таким образом, чтобы разрешить локально-чувствительную обработку.

4.4.2 Асинхронная передача

Пакетирование является одним из главных факторов эффективности, и для ее обеспечения поставщик накапливает данные в памяти и затем отправляет их большими партиями в одном запросе. Пакетная обработка может быть сконфигурирована таким образом, чтобы накапливание не превышало фиксированного количества сообщений, и/или время ожидания было не дольше указанного (например, *64k* или *10 мс*). Это позволяет накапливать больше байтов для передачи и большее количество операций ввода-вывода на серверах. Буферизация является настраиваемой и дает возможность обмена небольшой дополнительной задержки по времени на лучшую пропускную способность.

4.5 Потребители данных

Потребитель **ADS** передает “выборки” запросов брокерам-лидерам необходимых ему партиций и задает свое смещение в журнале. Таким образом, потребитель имеет контроль над своей позицией в журнале и при необходимости может повторно считывать данные.

Изначально вопрос заключался в том, должны ли потребители получать данные от брокеров или брокеры должны передавать данные потребителю. В этом отношении **ADS** следует более традиционному способу, разделяемому большинством систем обмена сообщениями, где данные отправляются в брокер поставщиком и получаются из брокера потребителем. Некоторые системы, ориентированные на ведение журнала, такие как **Scribe** и **Apache Flume**, следуют совершенно другому принципу – модели "push". У обоих подходов есть плюсы и минусы. Система push сталкивается с трудностями при работе с несколькими потребителями, поскольку брокер контролирует скорость передачи данных. А цель, как правило, заключается в том, чтобы потребитель считывал данные с максимально возможной скоростью; к сожалению, в системе push потребитель имеет тенденцию к перегрузке, когда его уровень потребления падает ниже уровня поставки данных. Система "pull" имеет лучшие свойства, при которых потребитель просто "отстает и догоняет". При этом можно использовать протокол, в котором потребитель указывает свою перегрузку, но получение повышенной скорости передачи данных для потребителя сложнее, чем кажется. Предыдущие попытки построения систем привели **ADS** к использованию традиционной модели pull.

Другим преимуществом системы pull является то, что она поддается агрессивной пакетной обработке данных, отправляемых потребителю. В то время как система push должна либо немедленно отправить запрос, либо накопить больше данных, а затем отправить их, не зная, сможет ли нижестоящий потребитель немедленно их обработать. А при настроенной низкой задержке оправка сообщений осуществляется по одному за раз и в любом случае буферизуется, что является расточительным. В модели pull потребитель всегда получает все доступные ему данные в зависимости от его текущего положения в журнале (или до установленного максимального размера). Таким образом, можно получить оптимальное дозирование без лишней задержки.

Недостаток системы pull заключается в том, что если у брокера нет данных, потребитель может выполнять запрос в жестком цикле, фактически находясь в режиме ожидания поступления данных. Во избежание этого в **ADS** есть параметры запроса pull, которые позволяют заявке потребителя блокировать "длинный опрос" ("long poll") до поступления данных (и при необходимости до тех пор, пока не будет достигнуто заданное количество байтов).

4.5.1 Позиция потребителя

Отслеживание считываемых данных, на удивление, является одной из ключевых точек производительности системы обмена сообщениями.

Большинство систем обмена сообщениями содержат метаданные о том, какие сообщения были считаны в брокер. То есть, при передаче сообщения потребителю брокер либо немедленно записывает его локально, либо ожидает подтверждения от потребителя. Это довольно интуитивный выбор, и действительно, серверу неясно, куда еще может потребоваться заявка. Поскольку структуры данных, используемые для хранения во многих системах обмена сообщениями, плохо масштабируются, это прагматичный выбор – поскольку брокер знает, какие данные считаны, и он может тут же удалить их, сохраняя размер данных небольшим.

Что, возможно, не очевидно, так это то, что заставить брокера и потребителя прийти к соглашению о считанных данных не является тривиальной задачей. Если брокер каждый раз немедленно записывает сообщение как считанное при его передаче по сети, то если потребитель не обрабатывает сообщение (например, по причине сбоя или времени ожидания запроса), сообщение теряется. Для решения этой проблемы многие системы обмена сообщениями добавляют функцию подтверждения, которая помечает сообщения как отправленные, но не считанные, пока брокер не дождется подтверждения от потребителя, чтобы записать сообщение как считанное. Такая стратегия устраняет проблему потери данных, но создает новые проблемы. Прежде всего, если потребитель обрабатывает сообщение, но не может отправить подтверждение, сообщение считывается дважды. Вторая проблема связана с производительностью – теперь брокер должен хранить несколько состояний о каждом сообщении (сначала заблокировать, чтобы не было повторного считывания, затем отметить как явно считанное, чтобы можно было удалить). Подобные замысловатые проблемы необходимо решать, как, например, что делать с сообщениями, которые отправляются, но никогда не подтверждаются.

ADS справляется с этим по-другому. Топик разделяется на набор полностью упорядоченных партиций, каждая из которых потребляется в любой момент времени ровно одним потребителем из подписавшейся группы потребителей. Это означает, что позиция потребителя в каждой партиции, то есть смещение от следующего

сообщения – это всего лишь целое число. Это говорит о том, что состояние о считывании данных очень малоемкое – всего лишь одно число для каждой партиции. Текущее состояние можно проверять. Все это делает подобный эквивалент функции подтверждения о считывании сообщений очень дешевым.

К тому же данное решение имеет побочное преимущество – потребитель имеет возможность намеренно вернуть назад свое смещение и повторно считать данные. Это нарушает общий порядок очереди, но при этом является существенным плюсом для многих потребителей.

4.5.2 Автономная загрузка данных

Масштабируемая персистентность позволяет потребителям, которые только периодически считывают пакетные данные, время от времени загружать данные в автономную систему, такую как, например, **Hadoop** или реляционное хранилище данных.

В случае **Hadoop** нагрузка данных распараллеливается на отдельные задачи, по одной для каждой комбинации узел/топик/партиция. **Hadoop** обеспечивает управление задачами, а задачи, потерпевшие неудачу, могут перезапускаться без риска дублирования данных – они просто возобновляются с исходной позиции.

4.5.3 Отслеживание смещений потребителя

Потребитель высокого уровня отслеживает свое максимальное смещение при считывании данных в каждой партиции и периодически фиксирует вектор смещения для возможности возобновления работы с необходимых позиций. **ADS** предоставляет возможность хранения всех смещений группы потребителей в определенном брокере (для каждой группы) – в менеджере смещений, то есть любой инстанс потребителя группы должен передавать свои смещения и запрашивать выборки через менеджер смещения (брокер). У потребителей высокого уровня эта операция выполняется автоматически, в то время как простой потребитель управляет своими смещениями вручную, так как в настоящее время автоматическая передача смещений простых потребителей не поддерживается **Java**, им доступно только ручное фиксирование или извлечение смещений в **ZooKeeper**. При использовании **Scala** простой потребитель может явно зафиксировать или получить смещения в менеджере.

Поиск менеджера смещения осуществляется по запросу *GroupCoordinatorRequest* в любой брокер **ADS** с последующим ответом *GroupCoordinatorResponse*, в котором содержится менеджер смещения. После чего потребитель может перейти к фиксации или извлечению смещений. В случае перемещения менеджера смещений потребителю необходимо повторно выполнить его поиск.

При получении менеджером смещения запроса на фиксацию *OffsetCommitRequest*, он добавляет смещение в специальный сжатый топик **ADS** с именем *__consumer_offsets*. Менеджер смещения отправляет успешный ответ фиксации смещения потребителю только после получения смещений всеми репликами топика. В случае если смещения не могут реплицироваться в пределах настренного времени ожидания, фиксация смещения завершается неудачей, и потребитель может повторить ее после отмены действия (у потребителей высокого уровня процедура выполняется автоматически). Брокеры периодически сжимают топик смещения, так как ему требуется поддерживать только последнее смещение на партицию. Менеджер смещения также упорядоченно кэширует все смещения в таблице in-memory для возможности их быстрой обработки.

Когда менеджер смещения получает запрос на извлечение смещения, он просто возвращает последний зафиксированный вектор смещения из кэша. В случае если менеджер был только что запущен или стал менеджером смещения для нового набора групп потребителей (став лидером для партиции топика смещения), может потребоваться загрузка партиции топика смещений в кэш. В это время операции по извлечению смещения завершаются ошибкой *OffsetsLoadInProgress*, и потребитель может повторить запрос *OffsetFetchRequest* после окончания копирования (у потребителей высокого уровня процедура выполняется автоматически).

4.5.4 Миграция смещений из ZooKeeper в ADS

Для миграции потребителей и смещений из **ZooKeeper** в **ADS** необходимо выполнить следующие действия:

1. Установить *offsets.storage=ads* и *dual.commit.enabled=true* в настройках потребителя.
2. Выполнить резкий переход к потребителям и убедиться в их исправности.
3. Установить *dual.commit.enabled=false* в настройках потребителя.
4. Выполнить резкий переход к потребителям и убедиться в их исправности.

Обратный переход (с **ADS** в **ZooKeeper**) также может выполняться с помощью вышеуказанных шагов при установке *offsets.storage=zookeeper*.

4.6 Механизм доставки сообщений

После введения в принципы работы поставщиков и потребителей данных следует изучить семантический анализ гарантий между ними, которые обеспечивает **ADS**. Существует несколько возможных гарантий по доставке сообщений:

- *At most once* – сообщения могут быть потеряны, но никогда не будут повторно отправлены.
- *At least once* – сообщения никогда не теряются, но могут быть повторно отправлены.
- *Exactly once* – наиболее востребованное – каждое сообщение доставляется один и только один раз.

Все сводится к двум проблемам: гарантии долговечности при публикации данных и гарантии при их считывании.

Многие системы утверждают, что обеспечивают гарантию “Exactly once”, но при этом большинство подобных утверждений ошибочно (т.к. ими не предусмотрены случаи, когда потребители или поставщики могут потерпеть неудачу; при многопотребительских процессах; когда данные, записанные на диск, могут быть потеряны).

Семантика **ADS** прямолинейна. При публикации сообщения, оно фиксируется в журнале. Как только опубликованное сообщение зафиксировано, оно не может быть потеряно, пока брокер, реплицирующий партицию, на которую было написано это сообщение, остается “живым”. Определения зафиксированного сообщения, живой партиции, а также описание типов сбоев приведено подробно в следующем разделе. Сейчас рассмотрим идеальный брокер без потерь и попытаемся понять гарантии поставщика и потребителя. Если поставщик пытается опубликовать сообщение и возникает сетевая ошибка, он не может быть уверен в том, когда произошла ошибка – до или после фиксации сообщения. Аналогично семантике операции вставки в таблицу базы данных с автогенерацией ключа.

Поставщик **ADS** поддерживает опцию идемпотентной доставки, гарантирующую, что повторная отправка не приводит к дублированию записей в журнале. Для достижения этого брокер назначает каждому поставщику идентификатор и дедуплицирует данные, используя порядковый номер, который отправляется поставщиком вместе с каждым сообщением. Так же поставщику предоставляется возможность отправки сообщений в несколько партиций топика, используя подобную транзакционной семантику: то есть либо успешно записаны все сообщения, либо не записано ни одно из них. Основным вариантом использования этого метода является однократная обработка между топиками **ADS** (описание приведено ниже).

Не все варианты использования требуют таких строгих гарантий. Для чувствительных к задержкам случаев у поставщика есть возможность определить желаемый уровень устойчивости (время ожидания фиксации сообщения может занять порядка *10 мс*). Однако поставщик может также указать, что он хочет выполнить запись данных полностью асинхронно, или что он хочет ждать только до тех пор, пока лидер (но не обязательно последователи) получит сообщение.

Теперь рассмотрим семантику с точки зрения потребителя. Все реплики имеют один и тот же журнал с одинаковыми смещениями. Потребитель контролирует свое положение в этом журнале. Если у потребителя не было сбоев, то он просто хранит эту позицию в памяти. Но в случае если потребитель терпит неудачу и необходимо, чтобы партиция этого топика была перехвачена другим процессом, новому процессу нужно выбрать соответствующую позицию, с которой следует продолжить обработку. К примеру, потребитель считывает данные – он имеет два варианта обработки данных и обновления своей позиции:

1. Потребитель может считывать сообщения, затем сохранять свою позицию в журнале и потом выполнять обработку. В этом случае существует вероятность того, что процесс выйдет из строя после сохранения позиции потребителя и до сохранения результатов обработки данных. Тогда другой процесс, перехвативший на себя обработку, начинается с сохраненной позиции, даже если какие-то сообщения до этой позиции не были обработаны. Это соответствует семантике “At most once”, так как в случае сбоя потребителя данные могут быть не обработаны.

2. Потребитель может считывать сообщения, затем обрабатывать их и потом сохранять свою позицию. В этом случае существует вероятность того, что процесс выйдет из строя после обработки данных, но до сохранения позиции потребителя. Тогда другой процесс, перехвативший на себя обработку, обрабатывает уже обработанные данные. В случае сбоя потребителя это соответствует семантике “At least once”. При этом во многих случаях сообщения имеют первичный ключ, поэтому обновления являются идемпотентными (повторное получение одного и того же сообщения просто перезаписывает его другой копией самого себя).

Что по поводу семантики “Exactly once” (то, что действительно желаемо)? При потреблении данных из топика **ADS** и их записи в другой топик, можно использовать транзакционную семантику поставщика, которая была упомянута выше. Позиция потребителя сохраняется как сообщение в топике, поэтому можно записать смещение в **ADS** в той же транзакции, что и топики смещения, получающие обработанные данные. Если транзакция прерывается, позиция потребителя возвращается к своему старому значению, и записанные в топик смещения данные доступны другим потребителям в зависимости от их “уровня изоляции”. На установленном по умолчанию уровне изоляции “read_uncommitted” все сообщения видны для потребителей, даже если они были частью прерванной транзакции, а на уровне “read_committed” потребитель возвращает только зафиксированные транзакцией сообщения (и любые данные, которые не являлись частью транзакции).

При записи данных во внешнюю систему существует оговорка, которая заключается в необходимости координировать позицию потребителя с тем, что фактически хранится в качестве выходных данных. Классический способ достижения этого – ведение двухфазной фиксации между хранением потребительской позиции и хранением потребительских данных. Но также этого можно добиться и проще – позволяя потребителю хранить смещение в том же месте, что и выходные данные. Например, коннектор **ADS Connect** записывает данные в **HDFS** вместе со смещениями считанных данных для гарантии их обновлений. Аналогичный шаблон используется для многих других систем данных, требующих более сильную семантику, и у которых сообщения не имеют первичного ключа для дедупликации. Такой метод лучше, потому что многие выходные системы, в которые потребитель может записывать данные, не поддерживают двухфазную фиксацию.

Так что **ADS** эффективно поддерживает гарантию “Exactly once”, и транзакции поставщик/потребитель могут использоваться для ее обеспечения при передаче и обработке данных между топиками. “Exactly once” гарантия в других системах обычно требует взаимодействия с такими же системами, а в **ADS** обеспечивается смещение, которое это реализует. В иных случаях **ADS** по умолчанию гарантирует как минимум доставку “At least once” и дает возможность пользователю реализовывать доставку по принципу “At most once” путем отключения повторных попыток записи у поставщика и фиксации смещений у потребителя перед обработкой пакета данных.

4.7 Репликация

Платформа **ADS** реплицирует журнал для каждой партиции топика через настраиваемое число серверов (коэффициент репликации можно задать по принципу “topic-by-topic”). Это реализует постоянный доступ к данным, выполняя автоматический переход к репликам при сбое сервера в кластере.

Другие системы обмена сообщениями предоставляют некоторые связанные с репликацией функции, но, по нашему (полностью предвзятому) мнению, они не сильно востребованы и при этом имеют большие минусы: неактивность подчиненных процессов, сильное отрицательное влияние на производительность, ручная настройка fiddly и т.д. Платформа **ADS** подразумевает репликацию по умолчанию – фактически нереплицируемые топики реализуются как реплицированные с коэффициентом репликации – один.

Единицей репликации является партиция топика. В условиях отсутствия сбоев каждая партиция в **ADS** имеет одного лидера и ноль или более последователей. Коэффициент репликации составляет общее количество реплик, включая лидера, которому направляются все операции чтения и записи. Как правило, существует гораздо больше партиций, чем брокеров, и лидеры равномерно распределяются между брокерами. Журналы подписчиков идентичны журналу лидера – все имеют одинаковые смещения и данные, расположенные в одном и том же порядке (хотя, конечно, в любой момент времени лидер может иметь несколько пока еще нереплицированных сообщений в конце своего журнала).

Последователи считывают данные от лидера, как обычный потребитель, и применяют их к собственному журналу. Наличие у лидера последователей имеет хорошее свойство, позволяющее последователям пакетировать записи используемых ими журналов в своем журнале.

Как и в большинстве распределенных систем для автоматической обработки сбоев требуется точное определение термина “живой” узел. Для такого узла у **ADS** есть два условия:

1. Узел должен иметь возможность поддержки сессии с **ZooKeeper** (через механизм **Heartbeat ZooKeeper**).
2. Если узел является подчиненным, он должен реплицировать записи, происходящие на лидере, и не отставать “слишком далеко”.

В **ADS** узлы, удовлетворяющие этим двум условиям, называются “синхронизированными” во избежание неопределенности “живых” или “неисправных” узлов. Лидер отслеживает набор синхронизированных узлов и, если последователь терпит неудачу, зависает или отстает, лидер удаляет его из списка синхронизирующих реплик. Выявление зависающих и отстающих реплик контролируется конфигурацией *replica.lag.time.max.ms*.

В терминологии распределенных систем **ADS** старается обрабатывать модель “сбой/восстановление”, когда узлы внезапно прекращают работу, а затем восстанавливаются (возможно, не зная, что они потерпели неудачу). **ADS** не обрабатывает так называемые “византийские” сбои, когда узлы выдают произвольные или вредоносные ответы.

Теперь можно более точно определить, что сообщение считается зафиксированным, когда все синхронизированные реплики для этой партиции применены к своему журналу. Потребителем могут быть считаны только зафиксированные данные. С другой стороны, у поставщиков есть возможность либо дожидаться фиксации сообщения, либо нет – в зависимости от предпочтения в отношении компромисса между задержкой и устойчивостью. Это предпочтение управляется настройкой *acks*, используемой поставщиком. Важно так же обратить внимание, что в топиках есть параметр минимального количества (*minimum number*) синхронизированных реплик, которое проверяется при запросе поставщиком подтверждения, что сообщение записано в полный набор синхронизированных реплик. Если у поставщика менее строгий запрос подтверждения, то сообщение может быть зафиксировано и считано, даже если количество синхронизированных реплик ниже минимального (например, оно может быть таким же низким, как у лидера).

Important: Гарантия, которую оказывает ADS, заключается в том, что зафиксированное сообщение не может быть потеряно, если хотя бы одна из синхронизированных реплик остается “живой”

Платформа **ADS** остается доступной при наличии сбоев узла после короткого периода отказа, но может быть недоступной при сетевых разделениях.

4.8 Сжатие журналов

Сжатие журналов гарантирует, что платформа **ADS** всегда сохраняет в журнале партиции топика по крайней мере последнее известное значение для каждого ключа сообщения. При этом учитываются варианты использования и сценарии, такие как перезагрузка кэшей после перезапуска приложения во время эксплуатационного обслуживания, восстановление состояния после сбоя приложения или системы.

До настоящего момента был описан только наиболее простой подход к хранению данных – когда старые данные удаляются из журнала через установленный период времени, или при достижении журналом определенного размера. Это хорошо работает для временных данных таких событий, как ведение журнала, где каждая запись фиксируется отдельно. Однако важным классом потоков данных является журнал изменений ключей и переменных (например, изменения в таблице базы данных).

Рассмотрим конкретный пример. Имеется топик, содержащий адреса электронной почты пользователей. Каждый раз, когда пользователь обновляет адрес своей электронной почты, в топик отправляется сообщение с использованием идентификатора пользователя в качестве первичного ключа. Теперь допустим, что за определенный период времени отправляются следующие сообщения для пользователя с id *123*, каждое сообщение соответствует изменению адреса электронной почты (сообщения для других идентификаторов опущены):

```
1  123 => bill@microsoft.com
2          .
3          .
4          .
5  123 => bill@gatesfoundation.org
6          .
7          .
8          .
9  123 => bill@gmail.com
```

Сжатие журнала дает более гранулированный механизм хранения, поэтому **ADS** гарантирует сохранение по крайней мере последнего обновления для каждого первичного ключа (например, *bill@gmail.com*). Так же гарантируется, что журнал содержит полный снапшот конечного значения для каждого ключа, а не только недавно измененные. Это означает, что нижестоящие потребители могут восстановить свое собственное состояние с этого топика без необходимости ведения полного журнала всех изменений.

Далее приведены несколько вариантов использования:

1. *Подписка на изменение базы данных.* Часто необходимо иметь набор данных в нескольких системах, и при этом одна из этих систем представляет собой некоторую базу данных (либо РСУБД, либо, возможно, новомодное хранилище ключ-значение). Например, может быть база данных, кэш, кластер поиска и кластер **Hadoop**. Каждое изменение БД должно отражаться в кэше, кластере поиска и, в конечном итоге, в **Hadoop**. В случае если идет обработка обновлений только в реальном времени, то нужен последний актуальный журнал. Но в случае необходимости перезагрузки кэша или восстановления отказавшего узла поиска, может понадобиться полный набор данных.

2. *Поиск событий.* Это стиль разработки приложений, который совмещает обработку запросов с дизайном приложения и использует журнал изменений в качестве основного хранилища для приложения.

3. *Ведение журнала для обеспечения высокой работоспособности.* Процесс, выполняющий локальные вычисления, можно сделать отказоустойчивым, фиксируя вносимые в локальном состоянии изменения, с целью, чтобы другой процесс мог перезагрузить эти изменения и продолжить работу в случае сбоя первого. Конкретным примером является обработка счетчиков, агрегатов и других операций типа “group by” в потоковой системе запросов. **Samza**, платформа потоковой обработки данных в реальном времени, использует эту функцию именно для данной цели.

В каждом из перечисленных случаев в первую очередь необходимо обрабатывать поток изменений в реальном времени, но иногда, когда машина выходит из строя или данные должны быть повторно загружены

или переработаны, необходимо выполнить их полную загрузку. Сжатие журнала позволяет использовать в этих целях один и тот же топик резервного копирования.

Общая идея довольно проста. Имея бесконечную сохраняемость журнала, каждое изменение в вышеуказанных случаях регестрировалось бы в него, и тогда состояние системы фиксировалось бы каждый раз с момента ее первого запуска. Используя этот полный журнал, была бы возможность восстановить систему в любой момент времени, воспроизведя первые N записей журнала. Но этот гипотетический полный журнал не очень практичен для систем, многократно обновляющих одну запись, поскольку журнал имел бы безграничный рост даже для стабильного набора данных. Простой же механизм ведения журнала, отбрасывающий устаревшие обновления, ограничен в пространстве, но и не является способом восстановления текущего состояния – восстановление системы с момента запуска не осуществляется посредством журнала, поскольку старые обновления могут быть не зафиксированы.

Сжатие журнала – это механизм, обеспечивающий гранулированное хранение каждой записи, а не крупнозернистое хранение во времени. Идея заключается в выборочном удалении записей, имеющих более новое обновление с тем же самым первичным ключом. Таким образом, журнал гарантированно имеет по крайней мере последнее состояние для каждого ключа.

Такая политика хранения может быть настроена для каждого топика, поэтому один кластер может иметь несколько топиков, где сохранение осуществляется по заданному объему данных или по времени ожидания, и другие топики, где сохранение обеспечивает сжатие.

Эта функциональность вдохновлена одним из старейших и наиболее успешных элементов инфраструктуры **LinkedIn** – службой кэширования базы данных, называемой Databus. В отличие от большинства лог-структурированных систем хранения платформа **ADS** построена для подписчиков и организует данные для быстрого линейного чтения и записи. **ADS** действует как хранилище достоверных данных, поэтому она полезна даже в ситуациях, когда вышестоящий источник данных не может быть воспроизведен.

4.8.1 Основы сжатия журналов

Далее приведен рисунок, отображающий логическую структуру ведения журнала **ADS** со смещением для каждого сообщения (Рис.4.1.).

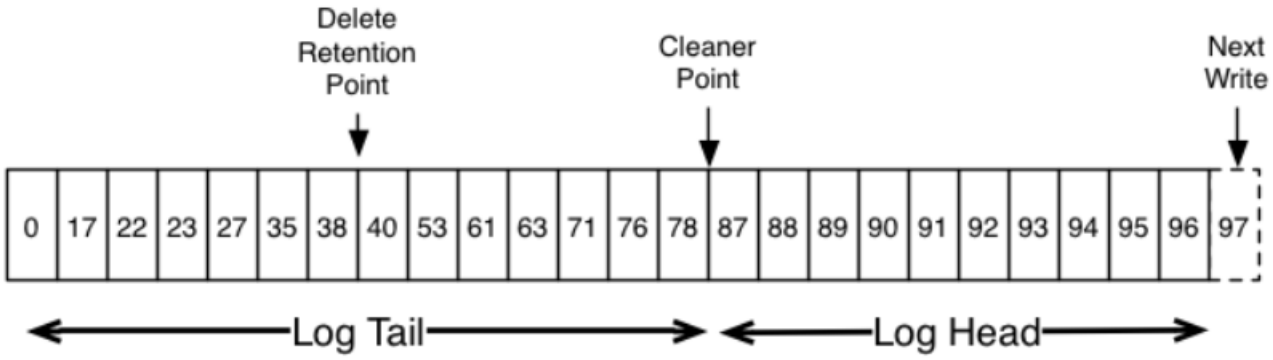


Рис.4.1.: Логическая структура журнала **ADS**

Журнал **ADS** идентичен традиционному журналу. В нем плотно фиксируются последовательные смещения и сохраняются все данные. Сжатие журнала добавляет опцию для обработки его хвоста (на Рис.4.1. показан журнал с уплотненным хвостом). Важно обратить внимание, что данные в хвосте журнала сохраняют исходное смещение, назначенное при первой записи, которое никогда не изменяется. Также важно, что все смещения остаются действующими позициями в журнале, даже если сообщение с этим смещением сжато; в таком случае позиция неотличима от следующего наибольшего смещения, которое появляется в журнале. Например,

на Рис.4.1. смещения *36*, *37* и *38* являются эквивалентными позициями, и начало чтения в любом из этих смещений выдает набор данных, начинающийся с *38*.

Сжатие также может удалять данные. Сообщение с ключом и нулевой полезной нагрузкой рассматривается на удаление из журнала и маркируется. Это приводит к удалению любых предыдущих сообщений с таким ключом (как и любые новые данные с таким же ключом), при этом маркированные на удаление данные являются особенными, поскольку они сами будут удалены из журнала через некоторое время. Момент времени, в который такие данные больше не сохраняются, на приведенном выше рисунке помечен как “Delete Retention Point”.

Сжатие выполняется в фоновом режиме с периодическим копированием сегментов журнала. Очистка не блокирует операции чтения и может регулироваться на настроенный объем пропускной способности ввода-вывода данных во избежание влияния на поставщиков и потребителей. Процесс сжатия сегмента журнала выглядит примерно как показано на Рис.4.2.

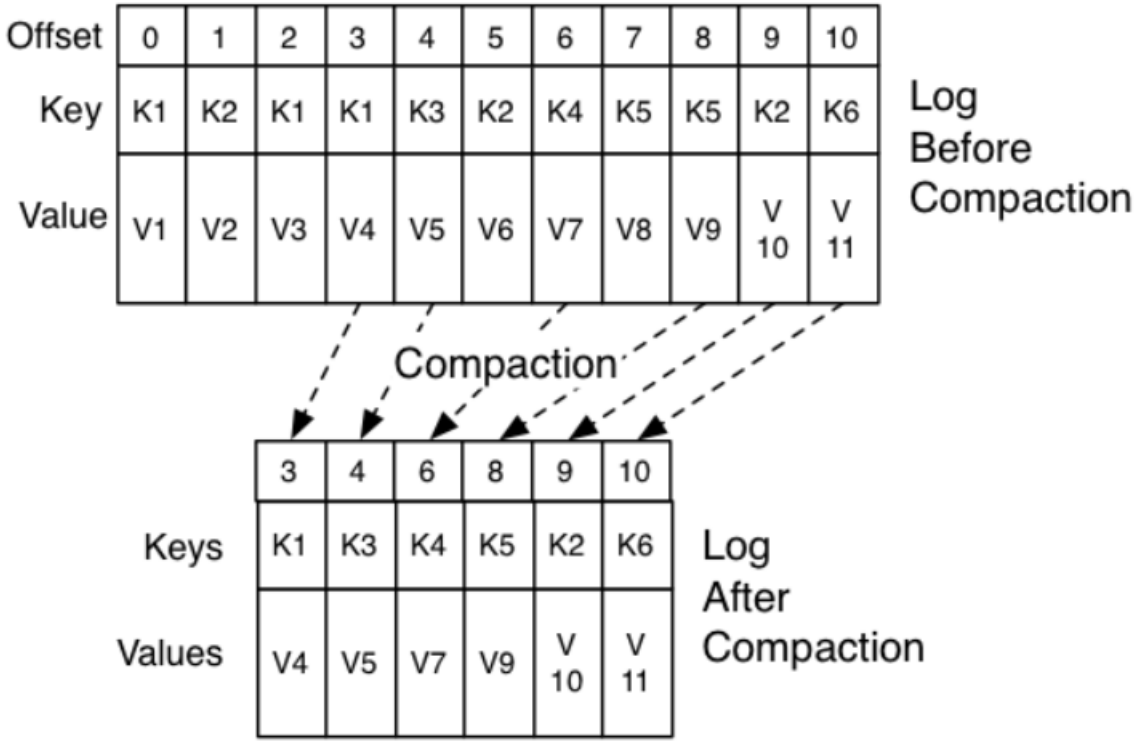


Рис.4.2.: Процесс сжатия сегмента журнала

4.8.2 Обеспечиваемые гарантии

Сжатие журнала обеспечивает следующие гарантии:

1. Любой потребитель, находящийся в голове журнала, видит каждое записанное сообщение; эти сообщения имеют последовательные смещения. Параметр топика *min.compaction.lag.ms* используется для гарантии минимального промежутка времени, затрачиваемого после записи сообщения прежде, чем оно будет сжато. То есть обеспечивается нижняя граница того, как долго каждое сообщение остается в неуплотненной голове журнала.
2. Всегда поддерживается упорядоченность данных. Сжатие никогда не нарушает порядок сообщений, а просто удаляет их.
3. Смещение для сообщения никогда не изменяется. Это постоянный идентификатор позиции в журнале.

4. Любому идущему от начала журнала потребителю видны, по крайней мере, окончательные состояния всех данных в порядке их записи. Кроме того, видны все маркированные под удаление данные при условии, что потребитель достигает головы журнала в течение меньшего периода времени, чем установлено в топике в параметре *delete.retention.ms* (по умолчанию *24 часа*). Другими словами, поскольку удаление маркированных данных происходит одновременно с операцией чтения, потребитель может пропустить эти данные при отставании более, чем на установленное в параметре *delete.retention.ms* время.

4.8.3 Детали сжатия журнала

Сжатие журнала выполняется посредством очистки log cleaner – объединением фоновых потоков, которые перезаписывают файлы сегментов журнала, и удалением данных, ключ которых отображается в голове журнала. Средство очистки работает следующим образом:

1. Выбирает журнал с самым высоким отношением между головой и хвостом.
2. Создает краткую сводку последнего смещения для каждого ключа в голове журнала.
3. Перезаписывает журнал от начала до конца, удаляя ключи, которые имеют более позднее появление в журнале. Новые, чистые сегменты немедленно подставляются в журнал, поэтому дополнительное пространство на диске требуется всего лишь для одного дополнительного сегмента журнала (а не для полной копии журнала).
4. Суммарно, голова журнала – это просто компактная хэш-таблица, использующая *24 байта* на запись. В результате с *8 ГБ* пространства под очистку одна итерация log cleaner может освободить около *366 ГБ* головы журнала (принимая данные равными *1 Кб*).

Глава 5

Настройки платформы

Платформа **ADS** использует пары ключ-значение в формате файла свойств для конфигурации. Данные значения могут быть поставлены либо из файла, либо программно.

- *Настройка брокера*
- *Настройка на уровне топика*
- *Конфигурирование Producer*
- *Конфигурирование Consumer*
- *Конфигурирование Connect*
- *Конфигурирование Streams*

5.1 Настройка брокера

Основные конфигурации брокера:

- *broker.id*
- *log.dirs*
- *zookeeper.connect*

Далее приведен список настроек с описанием и указанием их типа, значений по умолчанию и действительных, их важностью и режимом обновления.

zookeeper.connect – Строка хоста Zookeeper

- TYPE – string
- DEFAULT – high
- DYNAMIC UPDATE MODE – read-only

advertised.host.name – Применяется при неустановленном параметре *advertised.listeners* или *listeners*. Рекомендуется использовать *advertized.listeners*. Обозначает имя хоста для публикации в ZooKeeper для использования клиентами. В средах IaaS может отличаться от интерфейса, к которому привязывается брокер. Если параметр не задан, используется настроенное значение для *host.name*. В противном случае – значение из *java.net.InetAddress.getCanonicalHostName()*

- TYPE – string

- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

advertised.listeners – Слушатели для публикации в ZooKeeper для использования клиентами (если есть отличие от свойства *listeners*). В средах IaaS может отличаться от интерфейса, к которому привязывается брокер. Если параметр не задан, используется значение для *listeners*. В отличие от *listeners* значение "0.0.0.0" недопустимо

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – per-broker

advertised.port – Применяется при неустановленном параметре *advertised.listeners* или *listeners*. Рекомендуется использовать *advertized.listeners*. Обозначает имя хоста для публикации в ZooKeeper для использования клиентами. В средах IaaS может отличаться от интерфейса, к которому привязывается брокер. Если параметр не задан, публикуется тот же порт, к которому привязан брокер

- TYPE – int
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

auto.create.topics.enable – Включение автоматического создания топика на сервере

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

auto.leader.rebalance.enable – Включение автоматической балансировки лидера. Балансировка лидера в фоновом режиме через регулярные промежутки времени

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

background.threads – Количество потоков для различных задач фоновой обработки

- TYPE – int
- DEFAULT – 10
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

broker.id – Идентификатор брокера для сервера. Если значение не установлено, создается уникальный идентификатор брокера. Чтобы избежать конфликтов между id брокера, созданными с помощью zookeeper, и id

брокера, настроенными пользователем, генерация идентификаторов брокера начинается с *reserved.broker.max.id + 1*

- TYPE – int
- DEFAULT – - 1
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

compression.type – Конечный тип сжатия для топика. Конфигурация принимает стандартные кодеки сжатия (“gzip”, “snappy”, “lz4”). Так же возможно “uncompressed”, что эквивалентно отсутствию сжатия; и “producer”, что означает сохранение исходного кодека сжатия, установленного поставщиком

- TYPE – string
- DEFAULT – producer
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

delete.topic.enable – Включение возможности удаления топика. При отключенном параметре удаление топика через администратора не имеет результата

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

host.name – Применяется только когда параметр *listeners* не установлен. Рекомендуется использовать *listeners*. Обозначает имя хоста брокера. Если параметр задан, то привязка выполняется только к данному адресу. Если параметр не задан, привязка выполняется ко всем интерфейсам

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

leader.imbalance.check.interval.seconds – Частота, с которой контроллер запускает проверку балансировки партиции

- TYPE – long
- DEFAULT – 300
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

leader.imbalance.per.broker.percentage – Коэффициент дисбаланса лидера, допустимый для каждого брокера. Контроллер запускает балансировку лидера, если он превышает данное значение для брокера. Указывается в процентах

- TYPE – int
- DEFAULT – 10
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

listeners – Listener List – Разделенный запятыми список URI, которые прослушиваются, и имена слушателей сети. Если имя слушателя не является протоколом безопасности, необходимо установить *listener.security.protocol.map*. Для привязки ко всем интерфейсам указать имя хоста “0.0.0.0”. Если имя хоста не указано, привязка осуществляется к интерфейсу по умолчанию. Примеры списков слушателей сети: PLAINTEXT://myhost:9092,SSL://:9091,CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – per-broker

log.dir – Каталог хранения данных журнала (дополнительный для свойства log.dirs)

- TYPE – string
- DEFAULT – /tmp/kafka-logs
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.dirs – Каталоги хранения данных журнала. Если параметр не установлен, используется значение свойства *log.dir*

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – high

log.flush.interval.messages – Количество накопленных в партиции журнала данных перед их сбросом на диск

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.flush.interval.ms – Максимальное время хранения данных в любом топике в памяти до их сброса на диск. Указывается в миллисекундах. Если параметр не установлен, используется значение *log.flush.scheduler.interval.ms*

- TYPE – long
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.flush.offset.checkpoint.interval.ms – Частота обновления постоянной записи последнего сброса, который действует как точка восстановления журнала

- TYPE – int
- DEFAULT – 60000
- VALID VALUES – [0,...]

- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.flush.scheduler.interval.ms – Частота log flusher проверки на необходимость сброса какого-либо журнала на диск. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 9223372036854775807
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.flush.start.offset.checkpoint.interval.ms – Частота обновления постоянной записи смещения начала журнала

- TYPE – int
- DEFAULT – 60000
- VALID VALUES – [0,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.retention.bytes – Максимальный размер журнала перед его удалением

- TYPE – long
- DEFAULT – - 1
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.retention.hours – Количество часов для хранения файла журнала перед его удалением, третично по отношению к свойству *log.retention.ms*. Указывается в часах

- TYPE – int
- DEFAULT – 168
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.retention.minutes – Количество минут для хранения файла журнала перед его удалением, вторично по отношению к свойству *log.retention.hours*. Указывается в минутах

- TYPE – int
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.retention.ms – Количество миллисекунд для хранения файла журнала перед его удалением. Указывается в миллисекундах. Если параметр не установлен, используется значение *log.retention.minutes*

- TYPE – long
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.roll.hours – Максимальное время до развертывания нового сегмента журнала, вторично по отношению к свойству *log.roll.ms*. Указывается в часах

- TYPE – int
- DEFAULT – 168
- VALID VALUES – [1,...]
- IMPORTANCE – [1,...]
- DYNAMIC UPDATE MODE – read-only

log.roll.jitter.hours – Максимально допустимое значение джиттера для вычитания из *logRollTimeMillis*, вторично по отношению к свойству *log.roll.jitter.ms*. Указывается в часах

- TYPE – int
- DEFAULT – int
- VALID VALUES – [0,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

log.roll.jitter.ms – Максимально допустимое значение джиттера для вычитания из *logRollTimeMillis*. Указывается в миллисекундах. Если параметр не установлен, используется значение *log.roll.jitter.hours*

- TYPE – long
- DEFAULT – long
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.roll.ms – Максимальное время до развертывания нового сегмента журнала. Указывается в миллисекундах. Если параметр не установлен, используется значение *log.roll.hours*

- TYPE – long
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.segment.bytes – Максимальный размер одного файла журнала

- TYPE – int
- DEFAULT – 1073741824
- VALID VALUES – [14,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

log.segment.delete.delay.ms – Время ожидания перед удалением файла из файловой системы

- TYPE – long
- DEFAULT – 60000
- VALID VALUES – [0,...]
- IMPORTANCE – high

- DYNAMIC UPDATE MODE – cluster-wide

message.max.bytes – Наибольший размер пакета данных, разрешенный ADS. При увеличении параметра следует также увеличить размер выборки для потребителей с целью обеспечения возможности получения пакета данных установленного размера. Параметр можно настроить для каждого топика с помощью поуровневой конфирурации топика *max.message.bytes*

- TYPE – int
- DEFAULT – 1000012
- VALID VALUES – [0,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

min.insync.replicas – При установленном поставщиком подтверждении acks на “all” или “-1”, *min.insync.replicas* задается на минимальное количество реплик для подтверждения записи. Если этот минимум не может быть удовлетворен, то поставщик задает исключение (либо *NotEnoughReplicas*, либо *NotEnoughReplicasAfterAppend*). Совместное использование *min.insync.replicas* и acks обеспечивает более высокую гарантию к устойчивости. Типичным сценарием является создание топика с коэффициентом репликации *3*, параметром *min.insync.replicas* равным *2* и acks установленным на “all”. Это гарантирует, что поставщик задает исключение, если большинство реплик не принимает запись

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

num.io.threads – Число потоков, используемых сервером для обработки запросов, которые могут включать дисковые операции ввода-вывода

- TYPE – int
- DEFAULT – 8
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

num.network.threads – Количество потоков, используемых сервером для получения запросов от сети и отправки ответов в сеть

- TYPE – int
- DEFAULT – 3
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

num.recovery.threads.per.data.dir – Число потоков в каталоге данных, используемых для восстановления журнала при запуске или при сбросе по прекращению работы

- TYPE – int
- DEFAULT – 1

- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

num.replica.alter.log.dirs.threads – Число потоков, которые могут перемещать реплики между каталогами журналов, включая дисковые операции ввода-вывода

- TYPE – int
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

num.replica.fetchers – Количество потоков выборки, используемых для репликации данных от исходного брокера. Увеличение этого значения может увеличить степень параллелизма ввода-вывода в брокере-подписчике

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – cluster-wide

offset.metadata.max.bytes – Максимальный размер для записи метаданных с учетом фиксации смещения

- TYPE – int
- DEFAULT – 4096
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.commit.required.acks – Принятие необходимых подтверждений acks перед фиксацией данных. Значение по умолчанию "-1" не следует переопределять

- TYPE – short
- DEFAULT – - 1
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.commit.timeout.ms – Фиксация смещения откладывается до тех пор, пока все реплики для топика смещения не получат коммит или данный установленный таймаут не будет достигнут. Аналогично времени ожидания запроса поставщика

- TYPE – int
- DEFAULT – 5000
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.load.buffer.size – Размер пакета для чтения из сегментов смещений при загрузке смещений в кэш

- TYPE – int
- DEFAULT – 5242880

- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.retention.check.interval.ms – Частота проверки устаревших смещений

- TYPE – long
- DEFAULT – 600000
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.retention.minutes – Сброс смещений старше установленного срока хранения

- TYPE – int
- DEFAULT – 1440
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.topic.compression.codec – Кодек сжатия для топика смещения. Сжатие может использоваться для достижения "атомных" коммитов

- TYPE – int
- DEFAULT – 0
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.topic.num.partitions – Количество партиций для коммита топика смещения (не следует изменять после развертывания)

- TYPE – int
- DEFAULT – 50
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.topic.replication.factor – Коэффициент репликации для топика смещения (устанавливается выше с целью обеспечения доступности). Создание внутреннего топика невозможно, пока размер кластера не соответствует данному требованию коэффициента репликации

- TYPE – short
- DEFAULT – 3
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

offsets.topic.segment.bytes – Размер сегмента топика смещений в байтах. Значение должно быть относительно небольшим с целью ускорения сжатия журнала и загрузку кэша

- TYPE – int
- DEFAULT – 104857600
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

port – Применяется при неустановленном параметре *listeners*. Рекомендуется использовать *listeners*. Обозначает порт для прослушивания и приема подключений

- TYPE – int
- DEFAULT – 9092
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

queued.max.requests – Количество запросов в очереди до блокировки сетевых потоков

- TYPE – int
- DEFAULT – 500
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

quota.consumer.default – Применяется при неустановленном параметре динамических квот по умолчанию в Zookeeper. Любой потребитель группы customerId/consumer дросселируется при получении большего количества байтов, чем данное установленное значение в секунду

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

quota.producer.default – Применяется при неустановленном параметре динамических квот по умолчанию в Zookeeper. Любой поставщик с известным clientId дросселируется при получении большего количества байтов, чем данное установленное значение в секунду

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

replica.fetch.min.bytes – Минимальное количество байт, ожидаемое для каждого ответа на выборку. При недостаточном объеме срабатывает параметр *replicaMaxWaitTimeMs*

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – high

- DYNAMIC UPDATE MODE – read-only

replica.fetch.wait.max.ms – Максимальное время ожидания для каждого запроса на выборку с последующей публикацией реплик. Значение всегда должно быть меньше параметра *replica.lag.time.max.ms* для предотвращения частого сжатия ISR низкопроизводительных топиков

- TYPE – int
- DEFAULT – 500
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

replica.high.watermark.checkpoint.interval.ms – Верхний предел частоты сохранения на диск (Частота сохранения высокого водяного знака на диск)

- TYPE – long
- DEFAULT – 5000
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

replica.lag.time.max.ms – Удаление подписчика лидером из isr в случае, если подписчик не отправил ни одного запроса на выборку или не считал конечное смещение журнала лидеров

- TYPE – long
- DEFAULT – 10000
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

replica.socket.receive.buffer.bytes – Буфер приема сокетов для сетевых запросов

- TYPE – int
- DEFAULT – 65536
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

replica.socket.timeout.ms – Время ожидания сокета для сетевых запросов. Значение должно быть не менее установленного параметра *replica.fetch.wait.max.ms*

- TYPE – int
- DEFAULT – 30000
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

request.timeout.ms – Максимальное время ожидания клиентом ответа на запрос. Если ответ не получен до истечения установленного значения, клиент повторно отправляет запрос при необходимости

- TYPE – int
- DEFAULT – 30000
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

socket.receive.buffer.bytes – Буфер SO_RCVBUF сокета сервера сокетов. При значении параметра “-1” используется ОС по умолчанию

- TYPE – int
- DEFAULT – 102400
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

socket.request.max.bytes – Максимальное количество байт в запросе сокета

- TYPE – int
- DEFAULT – 104857600
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

socket.send.buffer.bytes – Буфер SO_SNDBUF сокета сервера сокетов. При значении параметра “-1” используется ОС по умолчанию

- TYPE – int
- DEFAULT – 102400
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transaction.max.timeout.ms – Максимально допустимое время ожидания для транзакций. Если запрошенное клиентом время транзакции превышает установленное значение, тогда брокер выдает ошибку в *InitProducerIdRequest*. Это предотвращает чрезмерное превышение времени ожидания для клиента, которое может тормозить чтение данных потребителями из топиков, включенных в транзакцию

- TYPE – int
- DEFAULT – 900000
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transaction.state.log.load.buffer.size – Размер пакета для чтения из сегментов журнала транзакций при загрузке в кэш идентификаторов поставщиков и транзакций

- TYPE – int
- DEFAULT – 5242880
- VALID VALUES – [1,...]
- IMPORTANCE – [1,...]
- DYNAMIC UPDATE MODE – read-only

transaction.state.log.min.isr – Переопределение конфигурации *min.insync.replicas* для топика транзакции

- TYPE – int
- DEFAULT – 2

- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transaction.state.log.num.partitions – Количество партиций для топика транзакции (после развертывания параметр должен остаться неизменным)

- TYPE – int
- DEFAULT – 50
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transaction.state.log.replication.factor – Коэффициент репликации для топика транзакции (задается выше для обеспечения доступности). Создание внутреннего топика завершится ошибкой, пока размер кластера не будет соответствовать данному требованию к фактору репликации

- TYPE – short
- DEFAULT – 3
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transaction.state.log.segment.bytes – Байты сегмента топика транзакции должны быть относительно небольшими для ускорения сжатия журнала и загрузки кэша

- TYPE – int
- DEFAULT – 104857600
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

transactional.id.expiration.ms – Максимальное время ожидания для координатора транзакций прежде, чем предварительно истечет срок действия идентификатора транзакции поставщика без получения обновлений состояния транзакции. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 604800000
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

unclean.leader.election.enable – Указывает, следует ли включить не входящие в набор ISR реплики и установка последнего средства в качестве лидера, даже если это может привести к потере данных

- TYPE – boolean
- DEFAULT – false
- IMPORTANCE – high

- DYNAMIC UPDATE MODE – cluster-wide

zookeeper.connection.timeout.ms – Максимальное время ожидания клиентом установки соединения с Zookeeper. Если параметр не задан, используется значение для *zookeeper.session.timeout.ms*. Указывается в миллисекундах

- TYPE – int
- DEFAULT – null
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

zookeeper.max.in.flight.requests – Максимальное количество неподтвержденных запросов, отправленных клиентом в Zookeeper, перед блокировкой

- TYPE – int
- DEFAULT – 10
- VALID VALUES – [1,...]
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

zookeeper.session.timeout.ms – Тайм-аут сеанса Zookeeper. Указывается в миллисекундах

- TYPE – int
- DEFAULT – int
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

zookeeper.set.acl – Настройка клиента для использования безопасных списков управления доступом ACL

- TYPE – boolean
- DEFAULT – boolean
- IMPORTANCE – high
- DYNAMIC UPDATE MODE – read-only

broker.id.generation.enable – Автоматическое создание идентификатора брокера на сервере. При включенном параметре значение, настроенное для *reserved.broker.max.id*, должно быть пересмотрено

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

broker.rack – Стойка брокера. Используется при назначении репликации в стойке для отказоустойчивости. Примеры: “RACK1”, “us-east-1d”

- TYPE – string
- DEFAULT – string
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

connections.max.idle.ms – Время ожидания бездействующих соединений: потоки процессора сокета сервера закрывают соединения, которые простаивают больше установленного значения. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 600000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

controlled.shutdown.enable – Включение контролируемого завершения работы сервера

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

controlled.shutdown.max.retries – Контролируемое выключение может завершиться ошибкой по нескольким причинам: параметр определяет количество повторных попыток подключения при возникновении таких сбоев

- TYPE – int
- DEFAULT – 3
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

controlled.shutdown.retry.backoff.ms – Перед каждой повторной попыткой подключения системе требуется время для восстановления состояния, вызвавшего предыдущий сбой (сбой контроллера, задержка реплики и т.д.). Параметр определяет время ожидания перед повторной попыткой. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 5000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

controller.socket.timeout.ms – Время ожидания сокета для каналов контроллер-брокер. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 30000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

default.replication.factor – Коэффициенты репликации по умолчанию для автоматически создаваемых топиков

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

delegation.token.expiry.time.ms – Время действия токена перед его обновлением. Значение по умолчанию 1 день. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 86400000
- VALID VALUES – [1,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

delegation.token.master.key – Мастер/секретный ключ для создания и проверки делегированных токенов. Один и тот же ключ должен быть настроен для всех брокеров. Если ключ не установлен или задана пустая строка, брокеры отключают поддержку делегированных токенов

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

delegation.token.max.lifetime.ms – Максимальный срок действия токена, по истечении которого он больше не может быть обновлен. Значение по умолчанию 7 дней. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 604800000
- VALID VALUES – [1,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

delete.records.purgatory.purge.interval.requests – Интервал очистки записей на удаление. Значение указывается в количестве запросов

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

fetch.purgatory.purge.interval.requests – Интервал очистки запросов выборки. Значение указывается в количестве запросов

- TYPE – int
- DEFAULT – 1000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

group.initial.rebalance.delay.ms – Время, в течение которого координатор группы ожидает присоединения большего числа потребителей к новой группе перед выполнением первой перебалансировки. Более длительная задержка означает потенциально меньшее количество перебалансировок, но увеличивает время до начала обработки. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 3000

- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

group.max.session.timeout.ms – Максимально допустимое время ожидания сессии для зарегистрированных потребителей. Более длительные тайм-ауты дают потребителям больше времени для обработки данных между heartbeat-сообщениями за счет большего времени для выявления сбоев. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 300000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

group.min.session.timeout.ms – Минимально допустимое время ожидания сессии для зарегистрированных потребителей. Более короткие тайм-ауты приводят к более быстрому обнаружению сбоев за счет более частых heartbeat-сообщений, которые могут перегружать ресурсы брокера. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 6000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

inter.broker.listener.name – Имя слушателя для связи между брокерами. Если параметр не задан, имя слушателя определяется свойством *security.inter.broker.protocol*. Одновременная установка параметров *inter.broker.listener.name* и *security.inter.broker.protocol* вызывает ошибку

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

inter.broker.protocol.version – Версия межброкерского протокола. Обычно параметр задается после обновления всех брокеров до новой версии. Пример некоторых допустимых значений: “0.8.0”, “0.8.1”, “0.8.1.1”, “0.8.2”, “0.8.2.0”, “0.8.2.1”, “0.9.0.0”, “0.9.0.1”. Необходимо проверить ApiVersion для полного списка

- TYPE – string
- DEFAULT – 1.1-IV0
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

log.cleaner.backoff.ms – Время спящего режима при отсутствии журналов для очистки. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 15000
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.dedupe.buffer.size – Общая память, используемая для дедупликации журнала во всех чистых потоках

- TYPE – long
- DEFAULT – 134217728
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.delete.retention.ms – Длительность хранения удаленных записей. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 86400000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.enable – Включение процесса очистки журналов для запуска на сервере. Параметр должен быть включен, если используются какие-либо топики с помощью *cleanup.policy=compact*, включая топик внутренних смещений. Если параметр отключен, данные топики не сжимаются и постоянно растут в объеме

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

log.cleaner.io.buffer.load.factor – Коэффициент загрузки буфера дедуплирования журнала очистки – процент заполнения буфера дедуплирования. Более высокое значение позволит очистить больше журнала, но приведет к большему количеству хэш-конфликтов

- TYPE – double
- DEFAULT – 0.9
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.io.buffer.size – Общая память, используемая для ввода-вывода буферов журнала очистки через все чистые потоки

- TYPE – int
- DEFAULT – 524288
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.io.max.bytes.per.second – Очистка журнала дросселируется таким образом, чтобы сумма операций чтения и записи была меньше установленного значения

- TYPE – double
- DEFAULT – 1.7976931348623157E308
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.min.cleanable.ratio – Минимальное отношение грязного журнала к общему журналу для журнала, пригодного для очистки

- TYPE – double
- DEFAULT – 0.5
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.min.compaction.lag.ms – Минимальное время, в течение которого сообщение остается несжатым в журнале. Применяется только для журналов с функцией сжатия. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 0
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleaner.threads – Количество фоновых потоков для очистки журнала

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.cleanup.policy – Политика очистки по умолчанию для сегментов, превышающих период хранения. Допустимые политики: “delete” и “compact”

- TYPE – list
- DEFAULT – delete
- VALID VALUES – [compact, delete]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.index.interval.bytes – Интервал добавления записи в индекс смещения

- TYPE – int
- DEFAULT – 4096
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.index.size.max.bytes – Максимальный размер индекса смещения. Указывается в байтах

- TYPE – int
- DEFAULT – 10485760
- VALID VALUES – [4,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.message.format.version – Версия формата сообщений, которую брокер использует для добавления данных в журналы. Значение должно быть действительным ApiVersion. Некоторые примеры: “0.8.2”, “0.9.0.0”, “0.10.0”. Необходимо проверить ApiVersion для получения более подробной информации. Установив версию формата сообщений, пользователь подтверждает, что все существующие данные на диске меньше или равны указанной версии. Неправильное задание параметра приводит к тому, что потребители с более старыми версиями получают данные в нечитаемом формате

- TYPE – string
- DEFAULT – 1.1-IV0
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

log.message.timestamp.difference.max.ms – Максимальное допустимое различие между отметкой времени, когда брокер получает сообщение, и отметкой времени, указанной в сообщении. При *log.message.timestamp.type=CreateTime* сообщение отклоняется, если разница в отметке времени превышает указанный порог. Конфигурация игнорируется, если *log.message.timestamp.type=LogAppendTime*. Максимально допустимое различие временных отметок должно быть не больше, чем *log.retention.ms*. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 9223372036854775807
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.message.timestamp.type – Определить, является ли отметка времени в сообщении временем создания сообщения или временем добавления журнала. Параметр может принимать значение “CreateTime” либо “LogAppendTime”

- TYPE – string
- DEFAULT – CreateTime
- VALID VALUES – [CreateTime, LogAppendTime]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.preallocate – Предварительное выделение файла при создании нового сегмента. При испольовании платформы ADS в Windows рекомендуется установить значение “true”

- TYPE – boolean
- DEFAULT – false
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – cluster-wide

log.retention.check.interval.ms – Частота проверки журналом очистки на наличие какого-либо журнала на удаление. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 300000
- VALID VALUES – [1,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

max.connections.per.ip – Максимальное количество подключений с каждого IP-адреса

- TYPE – int
- DEFAULT – 2147483647
- VALID VALUES – [1,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

max.connections.per.ip.overrides – Ip или hostname переопределяет максимальное количество подключений по умолчанию

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

max.incremental.fetch.session.cache.slots – Максимальное количество сессий инкрементной выборки

- TYPE – int
- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

num.partitions – Число партиций по умолчанию для каждого топика

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

password.encoder.old.secret – Старый секрет для кодирования динамически настроенных паролей. Установка параметра требуется только при обновлении секрета. Если параметр задан, все динамически закодированные пароли декодируются и перекодируются с помощью *password.encoder.secret* при запуске брокера

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

password.encoder.secret – Секрет для кодирования динамически настроенных паролей для данного брокера

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

principal.builder.class – Полное имя класса, реализующего интерфейс ADSPrincipalBuilder, который используется для создания объекта ADSPrincipal во время авторизации. Конфигурация также поддерживает устаревший интерфейс PrincipalBuilder, который ранее использовался для аутентификации клиентов по протоколу SSL. Если параметр не задан, действие по умолчанию зависит от используемого протокола безопасности. Для аутентификации SSL имя принципала отличается от имени из сертификата клиента, если он предоставлен; в противном случае, если аутентификация клиента не требуется, имя принципала задается “ANONYMOUS”. Для аутентификации SASL принципал задается на основании правил, определенных в *sasl.kerberos.principal.to.local.rules* с использованием GSSAPI и идентификатора аутентификации SASL для других механизмов. Для PLAINTEXT имя принципала – “ANONYMOUS”

- TYPE – class
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

producer.purgatory.purge.interval.requests – Интервал очистки запросов поставщика. Значение указывается в количестве запросов

- TYPE – int
- DEFAULT – 1000
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

queued.max.request.bytes – Разрешенное число байтов в очереди до того, как запросы не будут прочитаны

- TYPE – long
- DEFAULT – - 1
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

replica.fetch.backoff.ms – Длительность спящего режима при возникновении ошибки партиции. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

replica.fetch.max.bytes – Количество байтов сообщений, получаемых каждой партицией. Параметр не является абсолютным максимумом. Если первый пакет записей в первой непустой партиции выборки больше установленного значения, пакет данных все равно будет возвращен для обеспечения гарантии возможности выполнения. Максимальный размер пакета записей, принятый брокером, определяется через *message.max.bytes* (конфигурация брокера) или *max.message.bytes* (конфигурация топика)

- TYPE – int
- DEFAULT – 1048576
- VALID VALUES – [0,...]
- IMPORTANCE – medium

- DYNAMIC UPDATE MODE – read-only

replica.fetch.response.max.bytes – Максимальное количество байтов, ожидаемое для полного ответа на выборку. Параметр не является абсолютным максимумом. Записи извлекаются пакетами, и если первый пакет записей в первой непустой партиции выборки больше установленного значения, пакет данных все равно будет возвращен для обеспечения гарантии возможности выполнения. Максимальный размер пакета записей, принятый брокером, определяется через *message.max.bytes* (конфигурация брокера) или *max.message.bytes* (конфигурация топика)

- TYPE – int
- DEFAULT – 10485760
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

reserved.broker.max.id – Максимальное число, которое можно использовать для broker.id

- TYPE – int
- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

sasl.enabled.mechanisms – Список механизмов SASL, включенных на сервере ADS. Список может содержать любой механизм, для которого обеспечивается безопасность. По умолчанию включен только GSSAPI

- TYPE – list
- DEFAULT – GSSAPI
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.jaas.config – Параметры контекста входа JAAS для соединений SSL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан по ссылке. Формат значения: "(=)*;"

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.kinit.cmd – Путь команд Kerberos kinit

- TYPE – string
- DEFAULT – /usr/bin/kinit
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.min.time.before.relogin – Время ожидания авторизации потока между попытками обновления

- TYPE – long
- DEFAULT – 60000

- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.principal.to.local.rules – Список правил для сопоставления имен принципалов с короткими именами (обычно с именами пользователей операционной системы). Правила оцениваются по порядку, и первое правило, совпадающее с именем принципала, используется для сопоставления его с коротким именем. Все последующие правила в списке игнорируются. По умолчанию имена принципалов формы {username}/{hostname}@{REALM} сопоставляются с именем {username}. Важно обратить внимание, что данная конфигурация игнорируется, если расширение ADSPrincipalBuilder обеспечивается настройкой *main.builder.class*

- TYPE – list
- DEFAULT – DEFAULT
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.service.name – Имя принципала Kerberos, которое запускает ADS. Значение можно определить в конфигурации ADS JAAS либо в конфигурации ADS

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.ticket.renew.jitter – Процент случайного джиттера по отношению к времени возобновления

- TYPE – double
- DEFAULT – 0.05
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.kerberos.ticket.renew.window.factor – Время ожидания авторизации потока до тех пор, пока не будет достигнут указанный коэффициент времени от последнего обновления до истечения срока действия тикета, и попытка возобновления тикета за этот период времени

- TYPE – double
- DEFAULT – 0.8
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

sasl.mechanism.inter.broker.protocol – Механизм SASL для взаимодействия между брокерами. По умолчанию используется GSSAPI

- TYPE – string
- DEFAULT – GSSAPI
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

security.inter.broker.protocol – Протокол безопасности для связи между брокерами. Допустимые значения: “PLAINTEXT”, “SSL”, “SASL_PLAINTEXT”, “SASL_SSL”. Одновременная установка параметров *security.inter.broker.protocol* и *inter.broker.listener.name* вызывает ошибку

- TYPE – string
- DEFAULT – PLAINTEXT
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – read-only

ssl.cipher.suites – Список наборов шифров. Именованная комбинация аутентификации, шифрования, MAC и ключей обмена алгоритма для согласования параметров безопасности для сетевого подключения с использованием протокола TLS или SSL. По умолчанию поддерживаются все доступные варианты шифрования

- TYPE – list
- DEFAULT – “”
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.client.auth – Конфигурация брокера ADS для запроса аутентификации клиента. Следующие настройки являются общими:

- *ssl.client.auth=required* – требование проверки подлинности клиента;
- *ssl.client.auth=request* – аутентификация клиента является необязательной;
- *ssl.client.auth=none* – аутентификация клиента не требуется
- TYPE – string
- DEFAULT – none
- VALID VALUES – [required, requested, none]
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.enabled.protocols – Список протоколов, включенных для соединений SSL

- TYPE – list
- DEFAULT – TLSv1.2,TLSv1.1,TLSv1
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.key.password – Пароль закрытого ключа в файле хранилища ключей. Необязательный параметр для клиента

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.keymanager.algorithm – Алгоритм службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – SunX509
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.keystore.location – Расположение файла хранилища ключей. Необязательный параметр для клиента, может использоваться для двусторонней аутентификации клиента

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.keystore.password – Пароль хранилища для файла хранения ключей. Необязательный параметр для клиента, требуется только при настройке *ssl.keystore.location*

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.keystore.type – Формат файла хранилища ключей. Необязательный параметр для клиента

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.protocol – Протокол SSL для генерации SSLContext. Значение по умолчанию – “TLS”, что подходит для большинства случаев. Допустимыми значениями в последних JVM являются “TLS”, “TLSv1.1” и “TLSv1.2”. Протоколы “SSL”, “SSLv2” и “SSLv3” могут поддерживаться в более старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности

- TYPE – string
- DEFAULT – TLS
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.provider – Имя поставщика безопасности для соединений SSL. Значением по умолчанию является поставщик безопасности по умолчанию для JVM

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.trustmanager.algorithm – Алгоритм доверенной службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – PKIX
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.truststore.location – Расположение файла хранилища trust store

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.truststore.password – Пароль для файла хранилища trust store. При неустановленном пароле доступ к хранилищу есть, но осуществляется с отключенной проверкой надежности

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

ssl.truststore.type – Формат файла хранилища trust store

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium
- DYNAMIC UPDATE MODE – per-broker

alter.config.policy.class.name – Класс политики изменяемых конфигураций для их валидации. Класс осуществляет интерфейс *org.apache.kafka.server.policy.AlterConfigPolicy*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

alter.log.dirs.replication.quota.window.num – Количество выборок для сохранения в памяти для квот репликации изменяемых журналов

- TYPE – int
- DEFAULT – 11
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

alter.log.dirs.replication.quota.window.size.seconds – Временной интервал каждой выборки для квот репликации изменяемых журналов. Указывается в секундах

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

authorizer.class.name – Класс используемой авторизации

- TYPE – string

- DEFAULT – “”
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

create.topic.policy.class.name – Создание класса политики топика для его валидации. Класс осуществляет интерфейс *org.apache.kafka.server.policy.CreateTopicPolicy*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

delegation.token.expiry.check.interval.ms – Интервал сканирования для удаления делегированных токенов с истекшим сроком действия. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 3600000
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

listener.security.protocol.map – Сопоставление имен слушателей и протоколов безопасности. Параметр должен быть определен для того, чтобы один и тот же протокол безопасности мог использоваться в нескольких портах или IP-адресах. Например, внутренний и внешний трафик могут быть разделены, даже если для обоих требуется SSL. То есть, пользователь может определить слушателей с именами “INTERNAL” и “EXTERNAL” свойством: “INTERNAL:SSL, EXTERNAL:SSL”, где ключ и значение разделяются двоеточием, а записи карты разделяются запятыми (без пробелов). Каждое имя слушателя должно отображаться на карте только один раз. Различные настройки безопасности (SSL и SASL) могут быть настроены для каждого слушателя путем добавления стандартизированного префикса (имя слушателя в нижнем регистре) к имени конфигурации. Например, чтобы установить другое хранилище ключей для внутреннего слушателя, будет установлена конфигурация с именем *listener.name.internal.ssl.keystore.location*. Если конфигурация для имени слушателя не задана, используется общая конфигурация (то есть *ssl.keystore.location*)

- TYPE – string
- DEFAULT – PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – per-broker

metric.reporters – Список классов для использования в качестве репортеров метрик. Реализация интерфейса *org.apache.kafka.common.metrics.MetricsReporter* позволяет подключать классы, которые будут уведомлены о создании новой метрики. JmxReporter всегда включен в реестр статистических данных JMX

- TYPE – list
- DEFAULT – “”
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – cluster-wide

metrics.num.samples – Количество выборок, поддерживаемых для вычисления метрик

- TYPE – int
- DEFAULT – 2

- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

metrics.recording.level – Самый высокий уровень записи для метрик

- TYPE – string
- DEFAULT – INFO
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

metrics.sample.window.ms – Время ожидания вычисления метрик выборки. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 30000
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

password.encoder.cipher.algorithm – Алгоритм шифрования, используемый для кодирования динамически настроенных паролей

- TYPE – string
- DEFAULT – AES/CBC/PKCS5Padding
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

password.encoder.iterations – Число итераций для кодирования динамически настроенных паролей

- TYPE – int
- DEFAULT – 4096
- VALID VALUES – [1024,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

password.encoder.key.length – Длина ключа, используемая для кодирования динамически настроенных паролей

- TYPE – int
- DEFAULT – 128
- VALID VALUES – [8,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

password.encoder.keyfactory.algorithm – Алгоритм SecretKeyFactory, используемый для кодирования динамически настроенных паролей. По умолчанию используется “PBKDF2WithHmacSHA512”, если имеется, и “PBKDF2WithHmacSHA1” в противном случае

- TYPE – string

- DEFAULT – null
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

quota.window.num – Количество выборок, сохраняемых в памяти для квот клиента

- TYPE – int
- DEFAULT – 11
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

quota.window.size.seconds – Временной интервал каждой выборки для квот клиента. Указывается в секундах

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

replication.quota.window.num – Количество выборок, сохраняемых в памяти для квот репликации

- TYPE – int
- DEFAULT – 11
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

replication.quota.window.size.seconds – Временной интервал каждой выборки для квот репликации. Указывается в секундах

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

ssl.endpoint.identification.algorithm – Алгоритм идентификации конечных точек для валидации имени хоста сервера с использованием сертификата сервера

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – per-broker

ssl.secure.random.implementation – Реализация SecureRandom PRNG, используемая для операций шифрования SSL

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – per-broker

transaction.abort.timed.out.transaction.cleanup.interval.ms – Интервал, в течение которого выполняются отложенные транзакции. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 60000
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

transaction.remove.expired.transaction.cleanup.interval.ms – Интервал удаления транзакций, срок действия которых истекает по установленному параметру *transactional.id.expiration.ms passing*. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 3600000
- VALID VALUES – [1,...]
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

zookeeper.sync.time.ms – Удаленность последователя Zookeeper от лидера Zookeeper. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 2000
- IMPORTANCE – low
- DYNAMIC UPDATE MODE – read-only

Более подробную информацию о конфигурации брокера можно найти в классе scala *kafka.server.KafkaConfig*.

5.2 Настройка на уровне топика

Настройки топиков имеют как сервер по умолчанию, так и опциональное переопределение каждого топика. Если топику не задана конфигурация, то используется сервер по умолчанию. Переопределение можно установить во время создания топика, указав один или несколько параметров *–config*.

Далее приведены конфигурации на уровне топика. Настройка сервера по умолчанию указана в Server Default Property. Заданное значение по умолчанию для сервера относится только к топику, если у него нет явного переопределения конфигурации.

cleanup.policy – Определение политики хранения старых сегментов журнала. Политика по умолчанию “delete” отбрасывает старые сегменты при достижении их срока хранения или предельного размера. Значение “compact” включает сжатие журнала по топику

- TYPE – list

- DEFAULT – delete
- VALID VALUES – [compact, delete]
- SERVER DEFAULT PROPERTY –log.cleanup.policy
- IMPORTANCE – medium

compression.type – Окончательный тип сжатия топика. Конфигурация принимает стандартные кодеки сжатия (“gzip”, “snappy”, “lz4”). Также принимает “uncompressed”, что эквивалентно отсутствию сжатия; и “producer”, что означает сохранение исходного кодека сжатия, установленного поставщиком

- TYPE – string
- DEFAULT – producer
- VALID VALUES – [uncompressed, snappy, lz4, gzip, producer]
- SERVER DEFAULT PROPERTY – compression.type
- IMPORTANCE – medium

delete.retention.ms – Время хранения маркированных на удаление данных с целью сжатия топиков журнала. Параметр также дает ограничение на время, в течение которого потребитель должен выполнить чтение, если данные начинаются со смещения *0*, для гарантии валидности снапшота заключительного этапа (в противном случае удаление маркированных данных может произойти до завершения их сканирования). Указывается в миллисекундах

- TYPE – long
- DEFAULT – 86400000
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.cleaner.delete.retention.ms
- IMPORTANCE – medium

file.delete.delay.ms – Время ожидания перед удалением файла из файловой системы. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 60000
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.segment.delete.delay.ms
- IMPORTANCE – medium

flush.messages – Интервал принудительной синхронизации данных, записанных в журнал. Например, если параметр установлен на *1*, синхронизация выполняется после каждого сообщения; если на значение *5* – после каждых пяти сообщений. Установка данного параметра не рекомендуется, эффективней использовать репликацию для обеспечения устойчивости и возможности фоновой очистки операционной системы. Параметр можно переопределить в базовых настройках каждого топика

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.flush.interval.messages
- IMPORTANCE – medium

flush.ms – Временной интервал принудительной синхронизации данных, записанных в журнал. Например, если параметр установлен на *1000*, синхронизация выполняется по истечении 1000 мс. Установка данного параметра не рекомендуется, эффективней использовать репликацию для обеспечения устойчивости и возможности фоновой очистки операционной системы. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.flush.interval.ms
- IMPORTANCE – medium

follower.replication.throttled.replicas – Список реплик, для которых репликация журнала должна дросселироваться на стороне подписчика. Список должен описывать набор реплик в формате "[PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:..." или можно использовать специальный символ "*" для дросселирования всех реплик в данном топике

- TYPE – list
- DEFAULT – ""
- VALID VALUES – [partitionId],[brokerId]:[partitionId],[brokerId]:...
- SERVER DEFAULT PROPERTY – follower.replication.throttled.replicas
- IMPORTANCE – medium

index.interval.bytes – Частота добавления индексной записи в индекс смещения. Значение по умолчанию гарантирует индексацию сообщения примерно каждые 4096 байт. Большее индексирование позволяет потребителям приближаться к более точному положению в журнале, но увеличивает сам индекс. Рекомендуется значение не менять

- TYPE – int
- DEFAULT – 4096
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.index.interval.bytes
- IMPORTANCE – medium

leader.replication.throttled.replicas – Список реплик, для которых репликация журнала должна дросселироваться на стороне лидера. Список должен описывать набор реплик в формате "[PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:..." или можно использовать специальный символ "*" для дросселирования всех реплик в данном топике

- TYPE – list
- DEFAULT – ""
- VALID VALUES – [partitionId],[brokerId]:[partitionId],[brokerId]:...
- SERVER DEFAULT PROPERTY – leader.replication.throttled.replicas
- IMPORTANCE – medium

max.message.bytes – Наибольший размер пакета данных, разрешенный ADS. При увеличении параметра следует также увеличить размер выборки для потребителей с целью обеспечения возможности получения пакета данных установленного размера

- TYPE – int
- DEFAULT – 1000012

- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – message.max.bytes
- IMPORTANCE – medium

message.format.version – Версия формата сообщений, которую брокер использует для добавления данных в журналы. Значение должно быть действительным ApiVersion. Некоторые примеры: “0.8.2”, “0.9.0.0”, “0.10.0”. Необходимо проверить ApiVersion для получения более подробной информации. Установив версию формата сообщений, пользователь подтверждает, что все существующие данные на диске меньше или равны указанной версии. Неправильное задание параметра приводит к тому, что потребители с более старыми версиями получают данные в нечитаемом формате

- TYPE – string
- DEFAULT – 1.1-IV0
- SERVER DEFAULT PROPERTY – log.message.format.version
- IMPORTANCE – medium

message.timestamp.difference.max.ms – Максимальное допустимое различие между отметкой времени, когда брокер получает сообщение, и отметкой времени, указанной в сообщении. При *message.timestamp.type=CreateTime* сообщение отклоняется, если разница в отметке времени превышает указанный порог. Конфигурация игнорируется, если *message.timestamp.type=LogAppendTime*. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 9223372036854775807
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.message.timestamp.difference.max.ms
- IMPORTANCE – medium

message.timestamp.type – Определить, является ли отметка времени в сообщении временем создания сообщения или временем добавления журнала. Параметр может принимать значение “CreateTime” либо “LogAppendTime”

- TYPE – string
- DEFAULT – CreateTime
- SERVER DEFAULT PROPERTY – log.message.timestamp.type
- IMPORTANCE – medium

min.cleanable.dirty.ratio – Частота очистки журнала (при условии включенного сжатия). По умолчанию избегается очистка, где сжато более 50% журнала. Это ограничивает максимальное пространство, выделенное в журнале на дубликаты (не более 50% журнала могут занимать дубликаты). Более высокое отношение означает меньшее количество дубликатов и более эффективную очистку, но при этом большее количество потерянного пространства в журнале

- TYPE – double
- DEFAULT – 0.5
- VALID VALUES – [0,...,1]
- SERVER DEFAULT PROPERTY – log.cleaner.min.cleanable.ratio
- IMPORTANCE – medium

min.compaction.lag.ms – Минимальное время, в течение которого сообщение остается несжатым в журнале. Применяется только для журналов с функцией сжатия. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 0
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.cleaner.min.compaction.lag.ms
- IMPORTANCE – medium

min.insync.replicas – При установленном поставщиком подтверждении acks на "all" или "-1", *min.insync.replicas* задается на минимальное количество реплик для подтверждения записи. Если этот минимум не может быть удовлетворен, то поставщик задает исключение (*NotEnoughReplicas* или *NotEnoughReplicasAfterAppend*). Совместное использование *min.insync.replicas* и acks обеспечивает более высокую гарантию к устойчивости. Типичным сценарием является создание топика с коэффициентом репликации *3*, параметром *min.insync.replicas* равным *2* и acks установленным на "all". Это гарантирует, что поставщик задает исключение, если большинство реплик не принимает запись

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [1,...]
- SERVER DEFAULT PROPERTY – min.insync.replicas
- IMPORTANCE – medium

preallocate – Предварительное выделение файла на диске при создании нового сегмента журнала

- TYPE – boolean
- DEFAULT – false
- SERVER DEFAULT PROPERTY – log.preallocate
- IMPORTANCE – medium

retention.bytes – Контроль максимального размера партиции (состоящей из сегментов журнала), который может увеличиваться до момента отказа от старых сегментов журнала с целью освобождения места при использовании политики хранения "delete". По умолчанию ограничения по размеру нет, есть только ограничение по времени. Поскольку данный предел применяется на уровне партиции, необходимо умножить значение лимита по времени на количество партиций, чтобы вычислить объем хранения топика в байтах

- TYPE – long
- DEFAULT – - 1
- SERVER DEFAULT PROPERTY – log.retention.bytes
- IMPORTANCE – medium

retention.ms – Контроль максимального времени, в течение которого хранится журнал, прежде чем отбрасываются старые сегменты журнала с целью освобождения места при использовании политики хранения "delete". Параметр представляет собой SLA о том, как скоро потребители должны читать свои данные. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 604800000
- SERVER DEFAULT PROPERTY – log.retention.ms
- IMPORTANCE – medium

segment.bytes – Контроль размера файла сегмента для журнала. Сохранение и очистка файла всегда выполняются единовременно, поэтому больший размер сегмента означает меньшее количество файлов, но при этом менее гранулированный контроль над хранением

- TYPE – int
- DEFAULT – 1073741824
- VALID VALUES – [14,...]
- SERVER DEFAULT PROPERTY – log.segment.bytes
- IMPORTANCE – medium

segment.index.bytes – Контроль размера индекса, который отображает смещения в позициях файла. Предварительно индексный файл выделяется и сокращается только после сжатия журнала. Обычно параметр не требует изменений

- TYPE – int
- DEFAULT – 10485760
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.index.size.max.bytes
- IMPORTANCE – medium

segment.jitter.ms – Максимальный рандомный джиттер. Вычитается из запланированного времени сжатия сегмента во избежание проблемы сегментации thundering herd (огромное количество процессов, ждущих события, в то время как требуется только один процесс). Указывается в миллисекундах

- TYPE – long
- DEFAULT – 0
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.roll.jitter.ms
- IMPORTANCE – medium

segment.ms – Период времени, после которого ADS выполняет сжатие журнала, даже если файл сегмента не заполнен, с целью обеспечения сохранения или сжатия устаревших данных. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 604800000
- VALID VALUES – [0,...]
- SERVER DEFAULT PROPERTY – log.roll.ms
- IMPORTANCE – medium

unclean.leader.election.enable – Указывает, следует ли включить не входящие в набор ISR реплики и установка последнего средства в качестве лидера, даже если это может привести к потере данных

- TYPE – boolean
- DEFAULT – false
- SERVER DEFAULT PROPERTY – unclean.leader.election.enable
- IMPORTANCE – medium

5.3 Конфигурирование Producer

Далее приведены конфигурации Java-поставщика.

key.serializer – Класс Serializer для ключа, реализующего интерфейс *org.apache.kafka.common.serialization.Serializer*

- TYPE – class
- IMPORTANCE – high

value.serializer – Класс Serializer для значения, реализующего интерфейс *org.apache.kafka.common.serialization.Serializer*

- TYPE – class
- IMPORTANCE – high

acks – Количество подтверждений, которые поставщик требует от лидера перед рассмотрением запроса. Параметр контролирует устойчивость отправляемых записей. Возможны следующие настройки:

- *acks=0* – поставщик не ждет подтверждения с сервера. Запись немедленно добавляется в буфер сокета и считается отправленной. Данная настройка не гарантирует получение сервером записи, и конфигурация повторных попыток не вступает в силу (так как клиент обычно не знает о каких-либо сбоях). Смещение, возвращаемое для каждой записи, всегда равно "-1".
- *acks=1* – лидер фиксирует запись в свой локальный журнал, но отвечает, не дожидаясь полного подтверждения от всех подписчиков. В этом случае лидер может выйти из строя сразу после подтверждения записи и до того, как подписчики реплицируют ее, тогда запись теряется
- *acks=all* – лидер ожидает полного набора синхронизированных реплик для подтверждения записи. Данная настройка гарантирует, что запись не будет потеряна, пока хотя бы одна синхронизированная реплика остается в строе. Это наивысшая гарантия. Эквивалентно настройке *acks=-1*
- TYPE – string
- DEFAULT – 1
- VALID VALUES – [all, - 1, 0, 1]
- IMPORTANCE – high

bootstrap.servers – Список пар хост/порт, используемых для установления первоначального подключения к платформе ADS. В дальнейшем клиент будет использовать все сервера, независимо от того, какие указаны в данном параметре – этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Параметр должен быть задан в формате "host1:port1, host2:port2,..." (через запятую и без пробелов). Поскольку данные сервера используются только для первоначального подключения с целью обнаружения полного набора в кластере (который может динамически меняться), списку необязательно содержать полный набор серверов (можно указать более одного, на случай отказа первого)

- TYPE – list
- DEFAULT – ""
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@685cb137
- IMPORTANCE – high

buffer.memory – Общий объем памяти в байтах, которую поставщик может использовать для буферизации записей, ожидающих отправки на сервер. Если записи отправляются быстрее, чем они могут быть доставлены на сервер, поставщик блокирует параметр *max.block.ms*, после чего будет сделано исключение. Параметр должен соответствовать примерно общему объему памяти, которая используется поставщиком, но

не полному объему, так как не вся память используется для буферизации. Некоторый дополнительный объем используется для сжатия (если оно включено), а также для поддержания запросов на лету

- TYPE – long
- DEFAULT – 33554432
- VALID VALUES – [0,...]
- IMPORTANCE – high

compression.type – Тип сжатия для всех данных, созданных поставщиком. По умолчанию используется значение “none” (без сжатия). Допустимые значения: “none”, “gzip”, “snappy” и “lz4”. Сжатие выполняется над полной партией данных, поэтому эффективность дозирования влияет на коэффициент сжатия (более многочисленное порционирование означает лучшее сжатие)

- TYPE – string
- DEFAULT – none
- IMPORTANCE – high

retries – Установка значения больше нуля приводит к тому, что клиент переотправляет любую запись, передача которой завершается с временной ошибкой. Повторная попытка ничем не отличается от повторной отправки записи клиентом при получении ошибки. Повторная отправка данных без установки параметра *max.in.flight.requests.per.connection* в значение “1” потенциально может изменить порядок записей, так как если две партии данных отправляются в одну партицию, при этом первая партия не выполняется и повторно отправляется, а вторая выполняется успешно, то данные второго пакета появляются в партиции первыми

- TYPE – int
- DEFAULT – 0
- VALID VALUES – [0,...,2147483647]
- IMPORTANCE – high

ssl.key.password – Пароль закрытого ключа в файле хранилища ключей. Необязательный параметр для клиента

- TYPE – password
- DEFAULT – null
- IMPORTANCE – high

ssl.keystore.location – Расположение файла хранилища ключей. Необязательный параметр для клиента, может использоваться для двусторонней аутентификации клиента

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high

ssl.keystore.password – Пароль хранилища для файла хранения ключей. Необязательный параметр для клиента, требуется только при настройке *ssl.keystore.location*

- TYPE – password
- DEFAULT – null
- IMPORTANCE – high

ssl.truststore.location – Расположение файла хранилища trust store

- TYPE – string

- DEFAULT – null
- IMPORTANCE – high

ssl.truststore.password – Пароль для файла хранилища trust store. При неустановленном пароле доступ к хранилищу есть, но осуществляется с отключенной проверкой надежности

- TYPE – password
- DEFAULT – null
- IMPORTANCE – high

batch.size – При отправке нескольких записей в одну и ту же партицию поставщик пытается объединить их. Это помогает производительности как на клиенте, так и на сервере. Конфигурация управляет размером пакета в байтах. Пакетирование большего размера, чем задан в параметре, не осуществляется. В таком случае отправленные брокерам запросы содержат несколько пакетов (по одному для каждой партиции) с доступными для отправки данными. Небольшой размер пакета делает его менее востребованным и может снизить пропускную способность (нулевой размер пакета полностью отключает пакетирование). Очень большой размер пакета может использовать память расточительно, так как всегда выделяется буфер указанного размера пакета в ожидании дополнительных записей

- TYPE – int
- DEFAULT – 16384
- VALID VALUES – [0,...]
- IMPORTANCE – medium

client.id – Строка id для передачи на сервер при выполнении запросов. Целью является возможность отслеживания источника запросов за пределами ip/port, позволяя включать логическое имя приложения в журнал запросов на стороне сервера

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – medium

connections.max.idle.ms – Закрытие бездействующих соединений по истечению заданного периода. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 540000
- IMPORTANCE – medium

linger.ms – Поставщик объединяет в один пакет все записи, поступающие между трансмиссиями запросов. Обычно это происходит, когда данные поступают быстрее, чем могут быть отправлены. Однако клиент может уменьшить количество запросов даже при умеренной загрузке. Это реализуется путем добавления небольшого промежутка времени искусственной задержки, то есть вместо немедленной отправки данных поставщик ждет до указанной отметки с целью пакетирования данных. Это можно рассматривать как аналог алгоритма Nagle в TCP. Параметр дает верхнюю границу задержки по времени для пакетной обработки. Но как только достигается установленный размер пакета данных *batch.size* для партиции, пакет немедленно отправляется (независимо от заданного параметра *linger.ms*). Однако, имея меньший объем байт пакета, чем в указаном параметре *batch.size*, осуществляется задержка в течение времени, заданного *linger.ms*, с целью ожидания появления новых данных. По умолчанию параметр *linger.ms* равен “0” (то есть без задержки). Например, установка “linger.ms=5” приведет к уменьшению количества отправленных запросов, но добавит до *5 мс* задержки для данных, отправленных при отсутствии нагрузки. Указывается в миллисекундах

- TYPE – long

- DEFAULT – 0
- VALID VALUES – [0,...]
- IMPORTANCE – medium

max.block.ms – Время блокировки *ADSProducer.send()* и *ADSProducer.partitionsFor()*. Данные методы могут быть заблокированы либо по причине заполненного буфера, либо из-за недоступности метаданных. Блокировка в предоставленных пользователем сериализаторах или разделителе не учитывается по таймауту данного параметра. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 60000
- VALID VALUES – [0,...]
- IMPORTANCE – medium

max.request.size – Максимальный размер запроса в байтах. Параметр ограничивает количество пакетов данных, которые поставщик отправляет в одном запросе во избежание отправки огромных запросов. Параметр также эффективно ограничивает максимальный размер пакета данных. При этом сервер имеет свой собственный предел размера пакета данных, который может отличаться от указанного

- TYPE – int
- DEFAULT – 1048576
- VALID VALUES – [0,...]
- IMPORTANCE – medium

partitioner.class – Класс Partitioner, реализующий интерфейс *org.apache.kafka.clients.producer.Partitioner*

- TYPE – class
- DEFAULT – org.apache.kafka.clients.producer.internals.DefaultPartitioner
- IMPORTANCE – medium

receive.buffer.bytes – Размер буфера приема TCP (SO_RCVBUF) при чтении данных. Если значение равно "-1", используется ОС по умолчанию

- TYPE – int
- DEFAULT – 32768
- VALID VALUES – [-1,...]
- IMPORTANCE – medium

request.timeout.ms – Максимальное время ожидания клиентом ответа на запрос. Если ответ не получен до истечения установленного значения, клиент повторно отправляет запрос при необходимости. Значение параметра должно быть больше, чем *replica.lag.time.max.ms* (конфигурация брокера), с целью сокращения возможного дублирования данных по причине излишних попыток поставщика. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 30000
- VALID VALUES – [0,...]
- IMPORTANCE – medium

sasl.jaas.config – Параметры контекста входа JAAS для соединений SSL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан по ссылке. Формат значения: "(=)*;"

- TYPE – password

- DEFAULT – null
- IMPORTANCE – medium

sasl.kerberos.service.name – Имя принципала Kerberos, которое запускает ADS. Значение можно определить в конфигурации ADS JAAS либо в конфигурации ADS

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium

sasl.mechanism – Механизм SASL для клиентских подключений. Может быть любой механизм, для которого обеспечивается безопасность. По умолчанию используется GSSAPI

- TYPE – string
- DEFAULT – GSSAPI
- IMPORTANCE – medium

security.protocol – Протокол безопасности для связи между брокерами. Допустимые значения: “PLAINTEXT”, “SSL”, “SASL_PLAINTEXT”, “SASL_SSL”

- TYPE – string
- DEFAULT – PLAINTEXT
- IMPORTANCE – medium

send.buffer.bytes – Размер буфера отправки TCP (SO_SNDBUF) при отправке данных. Если значение равно “-1”, используется ОС по умолчанию

- TYPE – int
- DEFAULT – 131072
- VALID VALUES – [-1,...]
- IMPORTANCE – medium

ssl.enabled.protocols – Список протоколов, включенных для соединений SSL

- TYPE – list
- DEFAULT – TLSv1.2,TLSv1.1,TLSv1
- IMPORTANCE – medium

ssl.keystore.type – Формат файла хранилища ключей. Необязательный параметр для клиента

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium

ssl.protocol – Протокол SSL для генерации SSLContext. Значение по умолчанию – “TLS”, что подходит для большинства случаев. Допустимыми значениями в последних JVM являются “TLS”, “TLSv1.1” и “TLSv1.2”. Протоколы “SSL”, “SSLv2” и “SSLv3” могут поддерживаться в более старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности

- TYPE – string
- DEFAULT – TLS
- IMPORTANCE – medium

ssl.provider – Имя поставщика безопасности для соединений SSL. Значением по умолчанию является поставщик безопасности по умолчанию для JVM

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium

ssl.truststore.type – Формат файла хранилища trust store

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium

enable.idempotence – При установленном значении “true” поставщик гарантирует, что ровно одна копия каждого сообщения записывается в поток. При значении “false” в поток могут быть записаны дубликаты сообщений при повторных попытках отправки данных поставщиком из-за сбоев брокера или по другим причинам. Данный параметр требует, чтобы свойство *max.in.flight.requests.per.connection* было меньше или равно “5”, повторные попытки более “0”, и acks установлены на “all”. Если перечисленные настройки явно не заданы пользователем, выбираются подходящие значения. При установке несовместимых значений, выдается ConfigException

- TYPE – boolean
- DEFAULT – false
- IMPORTANCE – low

interceptor.classes – Список классов для использования в качестве интерсепторов. Реализация интерфейса *org.apache.kafka.clients.producer.ProducerInterceptor* позволяет перехватывать (и, возможно, видоизменять) записи, полученные поставщиком до их публикации в кластере ADS. По умолчанию интерсепторы не установлены

- TYPE – list
- DEFAULT – “”
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@6a41eaa2
- IMPORTANCE – low

max.in.flight.requests.per.connection – Максимальное количество неподтвержденных запросов, отправляемых клиентом по одному соединению перед блокировкой. Если параметр имеет значение больше *1*, то в случае сбоев существует риск переупорядочения данных из-за повторных попыток (если они включены)

- TYPE – int
- DEFAULT – 5
- VALID VALUES – [1,...]
- IMPORTANCE – low

metadata.max.age.ms – Период времени, после которого принудительно обновляются метаданные даже при отсутствии видимых изменений в лидере партиции с целью предварительного обнаружения новых брокеров или партиций. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 300000
- VALID VALUES – [0,...]
- IMPORTANCE – low

metric.reporters – Список классов для использования в качестве репортеров метрик. Реализация интерфейса *org.apache.kafka.common.metrics.MetricsReporter* позволяет подключать классы, которые будут уведомлены о создании новой метрики. JmxReporter всегда включен в реестр статистических данных JMX

- TYPE – list
- DEFAULT – “”
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@7cd62f43
- IMPORTANCE – low

metrics.num.samples – Количество выборок, поддерживаемых для вычисления метрик

- TYPE – int
- DEFAULT – 2
- VALID VALUES – [1,...]
- IMPORTANCE – low

metrics.recording.level – Самый высокий уровень записи для метрик

- TYPE – string
- DEFAULT – INFO
- VALID VALUES – [INFO, DEBUG]
- IMPORTANCE – low

metrics.sample.window.ms – Время ожидания вычисления метрик выборки. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 30000
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.max.ms – Максимальный период времени ожидания повторного подключения к брокеру при неоднократных сбоях соединения. Отсрочка на хост увеличивается экспоненциально для каждого последующего сбоя соединения, вплоть до установленного максимума. После расчета увеличения отсрочки к значению добавляется *20%* случайного джиттера во избежание помех связи. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.ms – Базовый период времени ожидания повторного подключения к хосту. Позволяет избегать многократного подключения к узлу в узком цикле. Данная отсрочка применяется ко всем попыткам подключения клиента к брокеру. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 50
- VALID VALUES – [0,...]
- IMPORTANCE – low

retry.backoff.ms – Время ожидания перед повторной попыткой отправки неудавшегося запроса в партицию топика. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 100
- VALID VALUES – [0,...]
- IMPORTANCE – low

sasl.kerberos.kinit.cmd – Путь команд Kerberos kinit

- TYPE – string
- DEFAULT – /usr/bin/kinit
- IMPORTANCE – low

sasl.kerberos.min.time.before.relogin – Время ожидания авторизации потока между попытками обновления

- TYPE – long
- DEFAULT – 60000
- IMPORTANCE – low

sasl.kerberos.ticket.renew.jitter – Процент случайного джиттера по отношению к времени возобновления

- TYPE – double
- DEFAULT – 0.05
- IMPORTANCE – low

sasl.kerberos.ticket.renew.window.factor – Время ожидания авторизации потока до тех пор, пока не будет достигнут указанный коэффициент времени от последнего обновления до истечения срока действия тикета, и попытка возобновления тикета за этот период времени

- TYPE – double
- DEFAULT – 0.8
- IMPORTANCE – low

ssl.cipher.suites – Список наборов шифров. Именованная комбинация аутентификации, шифрования, MAC и ключей обмена алгоритма для согласования параметров безопасности для сетевого подключения с использованием протокола TLS или SSL. По умолчанию поддерживаются все доступные варианты шифрования

- TYPE – list
- DEFAULT – null
- IMPORTANCE – low

ssl.endpoint.identification.algorithm – Алгоритм идентификации конечных точек для валидации имени хоста сервера с использованием сертификата сервера

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low

ssl.keymanager.algorithm – Алгоритм службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – SunX509
- IMPORTANCE – low

ssl.secure.random.implementation – Реализация SecureRandom PRNG, используемая для операций шифрования SSL

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low

ssl.trustmanager.algorithm – Алгоритм доверенной службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – PKIX
- IMPORTANCE – low

transaction.timeout.ms – Максимальный интервал времени, который координатор транзакции ожидает для обновления статуса транзакции от поставщика перед тем, как будет прервана текущая транзакция. Если установленное значение больше, чем значение *transaction.max.timeout.ms* в настройках брокера, запрос завершается ошибкой *InvalidTransactionTimeout*

- TYPE – int
- DEFAULT – 60000
- IMPORTANCE – low

transactional.id – Идентификатор транзакции. Параметр позволяет использовать семантику достоверности, которая охватывает несколько сессий поставщика, и позволяет гарантировать клиенту, что транзакции, использующие тот же TransactionalId, завершены до начала любых новых транзакций. Если TransactionalId не указан, то поставщик ограничивается идемпотентной доставкой. Важно, что параметр *enable.idempotence* должен быть включен при сконфигурированном *TransactionalId*. Значение по умолчанию "null", что означает невозможность использования транзакций. Для транзакций требуется по меньшей мере три брокера по умолчанию, что является рекомендуемым параметром для продуктивной системы; для разработки можно изменить настройки в параметре брокера *transaction.state.log.replication.factor*

- TYPE – string
- DEFAULT – null
- VALID VALUES – non-empty string
- IMPORTANCE – low

5.4 Конфигурирование Consumer

Далее приведены конфигурации для нового потребителя.

key.deserializer – Класс десериализатора для ключа, реализующего интерфейс *org.apache.kafka.common.serialization.Deserializer*

- TYPE – class
- IMPORTANCE – high

value.deserializer – Класс десериализатора для значения, реализующего интерфейс *org.apache.kafka.common.serialization.Deserializer*

- TYPE – class
- IMPORTANCE – high

bootstrap.servers – Список пар хост/порт, используемых для установления первоначального подключения к платформе ADS. В дальнейшем клиент будет использовать все сервера, независимо от того, какие указаны в данном параметре – этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Параметр должен быть задан в формате "host1:port1, host2:port2,..." (через запятую и без пробелов). Поскольку данные сервера используются только для первоначального подключения с целью обнаружения полного набора в кластере (который может динамически меняться), списку необязательно содержать полный набор серверов (можно указать более одного, на случай отказа первого)

- TYPE – list
- DEFAULT – ""
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@7cd62f43
- IMPORTANCE – high

fetch.min.bytes – Минимальный объем данных, которые сервер должен вернуть по запросу на выборку. При недостаточном объеме данных запрос ожидает их накопления до установленного значения. Значение по умолчанию *1 байт* означает, что запросы на выборку отвечают, как только доступен *1 байт* данных, или по истечению времени ожидания запроса. Установка большего значения заставляет сервер ожидать больших объемов данных для накопления, что может немного повысить пропускную способность сервера за счет некоторой дополнительной задержки

- TYPE – int
- DEFAULT – 1
- VALID VALUES – [0,...]
- IMPORTANCE – high

group.id – Уникальная строка, идентифицирующая группу потребителей. Свойство требуется, если потребитель использует функциональность группового управления с помощью подписки (топика) или стратегии управления смещением на основе ADS

- TYPE – string
- DEFAULT – ""
- IMPORTANCE – high

heartbeat.interval.ms – Время ожидания для координатора потребителя между heartbeat-сообщениями при использовании средств группового управления ADS. Heartbeat-сообщения используются для обеспечения активности сессии потребителя и перебалансировки, когда пользователи присоединяются или покидают группу. Значение должно быть установлено ниже, чем параметр *session.timeout.ms* (обычно оно не более *1/3* от этого значения). Его можно настроить еще ниже с целью контроля ожидаемого времени для нормальных перебалансировок. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 3000
- IMPORTANCE – high

max.partition.fetch.bytes – Максимальный объем данных для каждой партиции, который будет возвращен серверу. Из пакетов записи извлекаются потребителями. Если первый пакет записей в первой непустой партиции выборки больше установленного значения, пакет данных все равно будет возвращен для

обеспечения гарантии возможности выполнения. Максимальный размер пакета записей, принятый брокером, определяется через *message.max.bytes* (конфигурация брокера) или *max.message.bytes* (конфигурация топика). Для ограничения размера запроса потребителя используется параметр *fetch.max.bytes*

- TYPE – int
- DEFAULT – 1048576
- VALID VALUES – [0,...]
- IMPORTANCE – high

session.timeout.ms – Время ожидания для выявления сбоев потребителей при использовании группового управления ADS. Потребитель посылает периодические heartbeat-сообщения брокеру с целью подтверждения своей активности. Если брокер не получает ни одного бита до истечения установленного времени сеанса, то удаляет данного потребителя из группы и начинает балансировку. Значение параметра должно быть в допустимом диапазоне конфигурации брокера *group.min.session.timeout.ms* и *group.max.session.timeout.ms*. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 10000
- IMPORTANCE – high

ssl.key.password – Пароль закрытого ключа в файле хранилища ключей. Необязательный параметр для клиента

- TYPE – password
- DEFAULT – null
- IMPORTANCE – high

ssl.keystore.location – Расположение файла хранилища ключей. Необязательный параметр для клиента, может использоваться для двусторонней аутентификации клиента

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high

ssl.keystore.password – Пароль хранилища для файла хранения ключей. Необязательный параметр для клиента, требуется только при настройке *ssl.keystore.location*

- TYPE – password
- DEFAULT – null
- IMPORTANCE – high

ssl.truststore.location – Расположение файла хранилища trust store

- TYPE – string
- DEFAULT – null
- IMPORTANCE – high

ssl.truststore.password – Пароль для файла хранилища trust store. При неустановленном пароле доступ к хранилищу есть, но осуществляется с отключенной проверкой надежности

- TYPE – password
- DEFAULT – null
- IMPORTANCE – null

auto.offset.reset – В случае если в ADS нет начального смещения или текущее смещение больше не существует на сервере (например, так как данные удалены):

- *earliest*: автоматически сбросить смещение до самого раннего смещения;
- *latest*: автоматически сбросить смещение до последнего смещения;
- *none*: исключение для потребителя, если предыдущее смещение для группы потребителей не найдено;
- *anything else*: исключение для потребителя.
- TYPE – string
- DEFAULT – latest
- VALID VALUES – [latest, earliest, none]
- IMPORTANCE – medium

connections.max.idle.ms – Закрытие бездействующих соединений по истечению заданного периода. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 540000
- IMPORTANCE – medium

enable.auto.commit – При значении “true” смещение потребителя периодически фиксируется в фоновом режиме

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – medium

exclude.internal.topics – Предоставление потребителю записей из внутренних топиков (например, смещения). При значении “true” единственным способом получения записей из внутреннего топика является подписка на него

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – medium

fetch.max.bytes – Максимальный объем данных, который сервер должен вернуть для запроса на выборку. Параметр не является абсолютным максимумом. Из пакетов записи извлекаются потребителями. Если первый пакет записей в первой непустой партиции выборки больше установленного значения, пакет данных все равно будет возвращен для обеспечения гарантии возможности выполнения. Максимальный размер пакета записей, принятый брокером, определяется через *message.max.bytes* (конфигурация брокера) или *max.message.bytes* (конфигурация топика). При этом потребитель параллельно выполняет несколько выборок

- TYPE – int
- DEFAULT – 52428800
- VALID VALUES – [0,. . .]
- IMPORTANCE – medium

isolation.level – Контроль транзакционно записанных данных. Если установлено значение “read_committed”, *consumer.poll()* возвращает только совершенные транзакционные сообщения. При значении “read_uncommitted” (по умолчанию), *consumer.poll()* возвращает все сообщения, даже прерванные. Нетранзакционные сообщения возвращаются в любом режиме. Данные всегда возвращаются в порядке смещения. Следовательно, в режиме “read_committed” *consumer.poll()* возвращает сообщения только до

последнего стабильного смещения (LSO), которое меньше, чем смещение первой открытой транзакции. То есть любые данные, появляющиеся после текущей транзакциии, удерживаются до завершения соответствующей транзакции. Потребители “read_committed” не могут считывать высокий водяной знак в процессе транзакции. Так же в режиме “read_committed” метод *seekToEnd* возвращает LSO

- TYPE – string
- DEFAULT – read_uncommitted
- VALID VALUES – [read_committed, read_uncommitted]
- IMPORTANCE – medium

max.poll.interval.ms – Максимальная задержка времени между вызовами *poll()* при управлении группами потребителей. Параметр устанавливает верхнюю границу времени, в течение которого потребитель может бездействовать. Если *poll()* не вызывается до истечения установленного таймаута, потребитель считается неисправным, и группа перебалансируется с целью переназначения партиций. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 300000
- VALID VALUES – [1,...]
- IMPORTANCE – medium

max.poll.records – Максимальное число записей, возвращаемых за один вызов *poll()*

- TYPE – int
- DEFAULT – 500
- VALID VALUES – [1,...]
- IMPORTANCE – medium

partition.assignment.strategy – Класс стратегии назначения партиций, которую клиент использует для распределения принадлежности партиции инстансам потребителя при групповом управлении

- TYPE – list
- DEFAULT – class org.apache.kafka.clients.consumer.RangeAssignor
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@6d4b1c02
- IMPORTANCE – medium

receive.buffer.bytes – Размер буфера приема TCP (SO_RCVBUF) при чтении данных. Если значение равно “-1”, используется ОС по умолчанию

- TYPE – int
- DEFAULT – 65536
- VALID VALUES – [-1,...]
- IMPORTANCE – medium

request.timeout.ms – Максимальное время ожидания клиентом ответа на запрос. Если ответ не получен до истечения установленного значения, клиент повторно отправляет запрос при необходимости. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 305000
- VALID VALUES – [0,...]

- IMPORTANCE – medium

sasl.jaas.config – Параметры контекста входа JAAS для соединений SSL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан по ссылке. Формат значения: "(=)*;"

- TYPE – password
- DEFAULT – null
- IMPORTANCE – medium

sasl.kerberos.service.name – Имя принципала Kerberos, которое запускает ADS. Значение можно определить в конфигурации ADS JAAS либо в конфигурации ADS

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium

sasl.mechanism – Механизм SASL для клиентских подключений. Может быть любой механизм, для которого обеспечивается безопасность. По умолчанию используется GSSAPI

- TYPE – string
- DEFAULT – GSSAPI
- IMPORTANCE – medium

security.protocol – Протокол безопасности для связи между брокерами. Допустимые значения: "PLAINTEXT", "SSL", "SASL_PLAINTEXT", "SASL_SSL"

- TYPE – string
- DEFAULT – PLAINTEXT
- IMPORTANCE – medium

send.buffer.bytes – Буфер SO_SNDBUF сокета сервера сокетов. При значении параметра "-1" используется ОС по умолчанию

- TYPE – int
- DEFAULT – 131072
- VALID VALUES – [-1,...]
- IMPORTANCE – medium

ssl.enabled.protocols – Список протоколов, включенных для соединений SSL

- TYPE – list
- DEFAULT – TLSv1.2,TLSv1.1,TLSv1
- IMPORTANCE – medium

ssl.keystore.type – Формат файла хранилища ключей. Необязательный параметр для клиента

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium

ssl.protocol – Протокол SSL для генерации SSLContext. Значение по умолчанию – "TLS", что подходит для большинства случаев. Допустимыми значениями в последних JVM являются "TLS", "TLSv1.1" и "TLSv1.2". Протоколы "SSL", "SSLv2" и "SSLv3" могут поддерживаться в более старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности

- TYPE – string
- DEFAULT – TLS
- IMPORTANCE – medium

ssl.provider – Имя поставщика безопасности для соединений SSL. Значением по умолчанию является поставщик безопасности по умолчанию для JVM

- TYPE – string
- DEFAULT – null
- IMPORTANCE – medium

ssl.truststore.type – Формат файла хранилища trust store

- TYPE – string
- DEFAULT – JKS
- IMPORTANCE – medium

auto.commit.interval.ms – Частота автофиксации потребительских смещений при включенном параметре *enable.auto.commit* (значение “true”). Указывается в миллисекундах

- TYPE – int
- DEFAULT – 5000
- VALID VALUES – [0,...]
- IMPORTANCE – low

check.crcs – Автоматическая проверка CRC32 считываемых записей. Проверка добавляет некоторые накладные расходы, поэтому она может быть отключена в случаях, требующих высокой производительности

- TYPE – boolean
- DEFAULT – true
- IMPORTANCE – low

client.id – Строка id для передачи на сервер при выполнении запросов. Целью является возможность отслеживания источника запросов за пределами ip/port, позволяя включать логическое имя приложения в журнал запросов на стороне сервера

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – low

fetch.max.wait.ms – Максимальный период времени, в течение которого сервер блокируется, прежде чем ответить на запрос выборки (в случае недостаточного объема данных для незамедлительного ответа, заданного функцией *fetch.min.bytes*). Указывается в миллисекундах

- TYPE – int
- DEFAULT – 500
- VALID VALUES – [0,...]
- IMPORTANCE – low

interceptor.classes – Список классов для использования в качестве интерсепторов. Реализация интерфейса *org.apache.kafka.clients.consumer.ConsumerInterceptor* позволяет перехватывать (и, возможно, видоизменять) полученные потребителем записи. По умолчанию интерсепторы не установлены

- TYPE – list
- DEFAULT – “”
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@6093dd95
- IMPORTANCE – low

metadata.max.age.ms – Период времени, после которого принудительно обновляются метаданные даже при отсутствии видимых изменений в лидере партиции с целью предварительного обнаружения новых брокеров или партиций. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 300000
- VALID VALUES – [0,...]
- IMPORTANCE – low

metric.reporters – Список классов для использования в качестве репортеров метрик. Реализация интерфейса *org.apache.kafka.common.metrics.MetricsReporter* позволяет подключать классы, которые будут уведомлены о создании новой метрики. JmxReporter всегда включен в реестр статистических данных JMX

- TYPE – list
- DEFAULT – “”
- VALID VALUES – org.apache.kafka.common.config.ConfigDef$NonNullValidator@5622fdf
- IMPORTANCE – low

metrics.num.samples – Количество выборок, поддерживаемых для вычисления метрик

- TYPE – int
- DEFAULT – 2
- VALID VALUES – [1,...]
- IMPORTANCE – low

metrics.recording.level – Самый высокий уровень записи для метрик

- TYPE – string
- DEFAULT – INFO
- VALID VALUES – [INFO, DEBUG]
- IMPORTANCE – low

metrics.sample.window.ms – Время ожидания вычисления метрик выборки. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 30000
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.max.ms – Максимальный период времени ожидания повторного подключения к брокеру при неоднократных сбоях соединения. Отсрочка на хост увеличивается экспоненциально для каждого последующего сбоя соединения, вплоть до установленного максимума. После расчета увеличения отсрочки к значению добавляется *20%* случайного джиттера во избежание помех связи. Указывается в миллисекундах

- TYPE – long

- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.ms – Базовый период времени ожидания повторного подключения к хосту. Позволяет избегать многократного подключения к узлу в узком цикле. Данная отсрочка применяется ко всем попыткам подключения клиента к брокеру. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 50
- VALID VALUES – [0,...]
- IMPORTANCE – low

retry.backoff.ms – Время ожидания перед повторной попыткой отправки неудавшегося запроса в партицию топика. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 100
- VALID VALUES – [0,...]
- IMPORTANCE – low

sasl.kerberos.kinit.cmd – Путь команд Kerberos kinit

- TYPE – string
- DEFAULT – /usr/bin/kinit
- IMPORTANCE – low

sasl.kerberos.min.time.before.relogin – Время ожидания авторизации потока между попытками обновления

- TYPE – long
- DEFAULT – 60000
- IMPORTANCE – low

sasl.kerberos.ticket.renew.jitter – Процент случайного джиттера по отношению к времени возобновления

- TYPE – double
- DEFAULT – 0.05
- IMPORTANCE – low

sasl.kerberos.ticket.renew.window.factor – Время ожидания авторизации потока до тех пор, пока не будет достигнут указанный коэффициент времени от последнего обновления до истечения срока действия тикета, и попытка возобновления тикета за этот период времени

- TYPE – double
- DEFAULT – 0.8
- IMPORTANCE – low

ssl.cipher.suites – Список наборов шифров. Именованная комбинация аутентификации, шифрования, MAC и ключей обмена алгоритма для согласования параметров безопасности для сетевого подключения с использованием протокола TLS или SSL. По умолчанию поддерживаются все доступные варианты шифрования

- TYPE – list
- DEFAULT – null
- IMPORTANCE – low

ssl.endpoint.identification.algorithm – Алгоритм идентификации конечных точек для валидации имени хоста сервера с использованием сертификата сервера

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low

ssl.keymanager.algorithm – Алгоритм службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – SunX509
- IMPORTANCE – low

ssl.secure.random.implementation – Реализация SecureRandom PRNG, используемая для операций шифрования SSL

- TYPE – string
- DEFAULT – null
- IMPORTANCE – low

ssl.trustmanager.algorithm – Алгоритм доверенной службы управления ключами для SSL-соединений. Значением по умолчанию является алгоритм, настроенный для Java Virtual Machine

- TYPE – string
- DEFAULT – PKIX
- IMPORTANCE – low

5.5 Конфигурирование Connect

5.6 Конфигурирование Streams

Далее приведены конфигурации для клиентской библиотеки ADS Streams.

application.id – Идентификатор приложения потоковой обработки. Должен быть уникальным в пределах платформы ADS. Используется как: 1) префикс идентификатора клиента по умолчанию, 2) идентификатор группы для управления членством, 3) префикс топика изменений в журнале

- TYPE – string
- IMPORTANCE – high

bootstrap.servers – Список пар хост/порт, используемых для установления первоначального подключения к платформе ADS. В дальнейшем клиент будет использовать все сервера, независимо от того, какие указаны в данном параметре – этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Параметр должен быть задан в формате “host1:port1, host2:port2,...” (через запятую и без пробелов). Поскольку данные сервера используются только для первоначального подключения с целью

обнаружения полного набора в кластере (который может динамически меняться), списку необязательно содержать полный набор серверов (можно указать более одного, на случай отказа первого)

- TYPE – list
- IMPORTANCE – high

replication.factor – Коэффициент репликации для топиков журнала изменений и репартицирования топиков, созданных приложением потоковой обработки

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – high

state.dir – Местоположение каталога хранилища state store

- TYPE – string
- DEFAULT – /tmp/kafka-streams
- IMPORTANCE – high

cache.max.bytes.buffering – Максимальный объем памяти в байтах, используемый для буферизации всех потоков

- TYPE – long
- DEFAULT – 10485760
- VALID VALUES – [0,...]
- IMPORTANCE – medium

client.id – Строка префикса ID, используемая для идентификаторов клиента внутреннего потребителя и поставщика (восстановления потребителя); шаблон "-StreamThread–"

- TYPE – string
- DEFAULT – ""
- IMPORTANCE – medium

default.deserialization.exception.handler – Класс обработки исключений, реализующий интерфейс *org.apache.kafka.streams.errors.DeserializationExceptionHandler*

- TYPE – class
- DEFAULT – org.apache.kafka.streams.errors.LogAndFailExceptionHandler
- IMPORTANCE – medium

default.key.serde – Класс сериализатора/десериализатора по умолчанию для ключа, реализующего интерфейс *org.apache.kafka.common.serialization.Serde*

- TYPE – class
- DEFAULT – org.apache.kafka.common.serialization.Serdes$ByteArraySerde
- IMPORTANCE – medium

default.production.exception.handler – Класс обработки исключений, реализующий интерфейс *org.apache.kafka.streams.errors.ProductionExceptionHandler*

- TYPE – class
- DEFAULT – org.apache.kafka.streams.errors.DefaultProductionExceptionHandler

- IMPORTANCE – medium

default.timestamp.extractor – Класс выделения временных меток по умолчанию, реализующий интерфейс *org.apache.kafka.streams.processor.TimestampExtractor*

- TYPE – class
- DEFAULT – org.apache.kafka.streams.processor.FailOnInvalidTimestamp
- IMPORTANCE – medium

default.value.serde – Класс сериализатора/десериализатора по умолчанию для значения, реализующего интерфейс *org.apache.kafka.common.serialization.Serde*

- TYPE – class
- DEFAULT – org.apache.kafka.common.serialization.Serdes$ByteArraySerde
- IMPORTANCE – medium

num.standby.replicas – Число резервных реплик для каждой задачи

- TYPE – int
- DEFAULT – 0
- IMPORTANCE – medium

num.stream.threads – Количество потоков для выполнения потоковой обработки

- TYPE – int
- DEFAULT – 1
- IMPORTANCE – medium

processing.guarantee – Гарантия на обработку. Возможные значения: “at_least_once” (по умолчанию) и “exact_once”. Для обработки “exact_once” требуется по меньшей мере три брокера по умолчанию, что является рекомендуемой настройкой для продуктивной среды; для разработки можно изменить параметр, при этом переустановив настройку брокера *transaction.state.log.replication.factor*

- TYPE – string
- DEFAULT – at_least_once
- VALID VALUES – [at_least_once, exactly_once]
- IMPORTANCE – medium

security.protocol – Протокол безопасности для связи между брокерами. Допустимые значения: “PLAINTEXT”, “SSL”, “SASL_PLAINTEXT”, “SASL_SSL”

- TYPE – string
- DEFAULT – PLAINTEXT
- IMPORTANCE – medium

application.server – Пара “host:port”, указывающая на встроенную конечную точку пользователя, которая может использоваться для обнаружения местоположений хранилищ state stores в рамках одного приложения

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – low

buffered.records.per.partition – Максимальное количество записей в буфере для каждой партиции

- TYPE – int
- DEFAULT – 1000
- IMPORTANCE – low

commit.interval.ms – Частота сохранения положения процессора. Если параметр *processing.guarantee* установлен на “exactly_once”, значение по умолчанию равно “100”, иначе (при “at_least_once”) значение по умолчанию равно “30000”. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 30000
- IMPORTANCE – low

connections.max.idle.ms – Закрытие бездействующих соединений по истечению заданного периода. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 540000
- IMPORTANCE – low

key.serde – Сериализатор/десериализатор для ключа, реализующего интерфейс *org.apache.kafka.common.serialization.Serde*. Данная конфигурация устарела, вместо нее используется *default.key.serde*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low

metadata.max.age.ms – Период времени, после которого принудительно обновляются метаданные даже при отсутствии видимых изменений в лидере партиции с целью предварительного обнаружения новых брокеров или партиций. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 300000
- VALID VALUES – [0,...]
- IMPORTANCE – low

metric.reporters – Список классов для использования в качестве репортеров метрик. Реализация интерфейса *org.apache.kafka.common.metrics.MetricsReporter* позволяет подключать классы, которые будут уведомлены о создании новой метрики. JmxReporter всегда включен в реестр статистических данных JMX

- TYPE – list
- DEFAULT – “”
- IMPORTANCE – low

metrics.num.samples – Количество выборок, поддерживаемых для вычисления метрик

- TYPE – int
- DEFAULT – 2
- VALID VALUES – [1,...]
- IMPORTANCE – low

metrics.recording.level – Самый высокий уровень записи для метрик

- TYPE – string
- DEFAULT – INFO
- VALID VALUES – [INFO, DEBUG]
- IMPORTANCE – low

metrics.sample.window.ms – Время ожидания вычисления метрик выборки. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 30000
- VALID VALUES – [0,...]
- IMPORTANCE – low

partition.grouper – Класс Partition grouper, реализующий интерфейс *org.apache.kafka.streams.processor.PartitionGrouper*

- TYPE – class
- DEFAULT – org.apache.kafka.streams.processor.DefaultPartitionGrouper
- IMPORTANCE – low

poll.ms – Время блокировки ожидания ввода. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 100
- IMPORTANCE – low

receive.buffer.bytes – Размер буфера приема TCP (SO_RCVBUF) при чтении данных. Если значение равно "-1", используется ОС по умолчанию

- TYPE – int
- DEFAULT – 32768
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.max.ms – Максимальный период времени ожидания повторного подключения к брокеру при неоднократных сбоях соединения. Отсрочка на хост увеличивается экспоненциально для каждого последующего сбоя соединения, вплоть до установленного максимума. После расчета увеличения отсрочки к значению добавляется *20%* случайного джиттера во избежание помех связи. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 1000
- VALID VALUES – [0,...]
- IMPORTANCE – low

reconnect.backoff.ms – Базовый период времени ожидания повторного подключения к хосту. Позволяет избегать многократного подключения к узлу в узком цикле. Данная отсрочка применяется ко всем попыткам подключения клиента к брокеру. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 50
- VALID VALUES – [0,...]

- IMPORTANCE – low

request.timeout.ms – Максимальное время ожидания клиентом ответа на запрос. Если ответ не получен до истечения установленного значения, клиент повторно отправляет запрос при необходимости. Указывается в миллисекундах

- TYPE – int
- DEFAULT – 40000
- VALID VALUES – [0,...]
- IMPORTANCE – low

retries – Установка значения больше нуля приводит к тому, что клиент переотправляет любую запись, передача которой завершается с временной ошибкой

- TYPE – int
- DEFAULT – 0
- VALID VALUES – [0,...,2147483647]
- IMPORTANCE – low

retry.backoff.ms – Время ожидания перед повторной попыткой отправки неудавшегося запроса в партицию топика. Позволяет избежать неоднократной отправки запросов в сжатом цикле. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 100
- VALID VALUES – [0,...]
- IMPORTANCE – low

rocksdb.config.setter – Класс или имя класса установщика конфигурации базы данных Rocks, реализующий интерфейс *org.apache.kafka.streams.state.RocksDBConfigSetter*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low

send.buffer.bytes – Размер буфера отправки TCP (SO_SNDBUF) при отправке данных. Если значение равно "-1", используется ОС по умолчанию

- TYPE – int
- DEFAULT – 131072
- VALID VALUES – [0,...]
- IMPORTANCE – low

state.cleanup.delay.ms – Время ожидания перед удалением state каталогов после перемещения партиции. Удаляются только state каталоги, которые не были изменены. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 600000
- IMPORTANCE – low

timestamp.extractor – Класс выделения временных меток, реализующий интерфейс *org.apache.kafka.streams.processor.TimestampExtractor*. Данная конфигурация устарела, вместо нее используется *default.timestamp.extractor*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low

value.serde – Класс сериализатора/десериализатора для значения, реализующего интерфейс *org.apache.kafka.common.serialization.Serde*. Данная конфигурация устарела, вместо нее используется *ddefault.value.serde*

- TYPE – class
- DEFAULT – null
- IMPORTANCE – low

windowstore.changelog.additional.retention.ms – Добавление *maintainMs* с целью исключения риска преждевременного удаления данных из журнала. Позволяет осуществлять отставание часов. По умолчанию устанавливается 1 день. Указывается в миллисекундах

- TYPE – long
- DEFAULT – 86400000
- IMPORTANCE – low

zookeeper.connect – Соединение строки для управления топиками ADS через Zookeeper. Конфигурация устарела и игнорируется, поскольку Streams API больше не использует Zookeeper

- TYPE – string
- DEFAULT – “”
- IMPORTANCE – low

Глава 6

Новые функциональности

Новые функциональности компонентов описаны по следующим ссылкам:

- Apache Zookeeper;
- Apache Kafka;
- Apache NiFi;
- Apache NiFi Registry;