СБЕР УНИВЕРСИТЕТ

Школа Java Middle Developer

Kafka

Реализация конвейера данных

Содержание

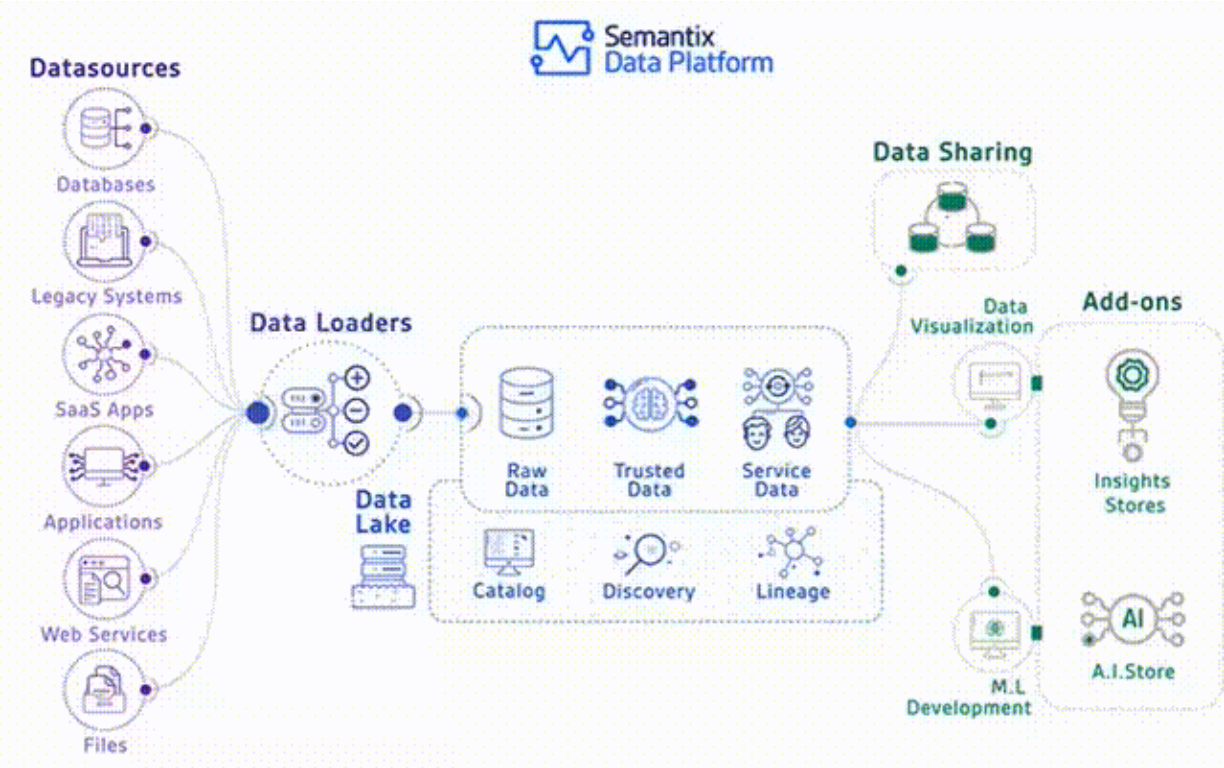
1. Создание конвейера данных;

2. Kafka Connect.

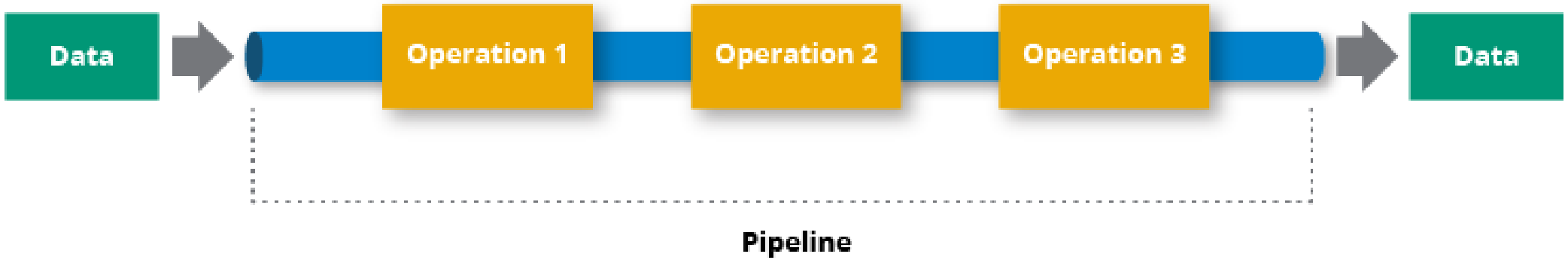
Конвейер данных

Конвейеры данных (Data Pipelines) состоят из трех основных компонентов:

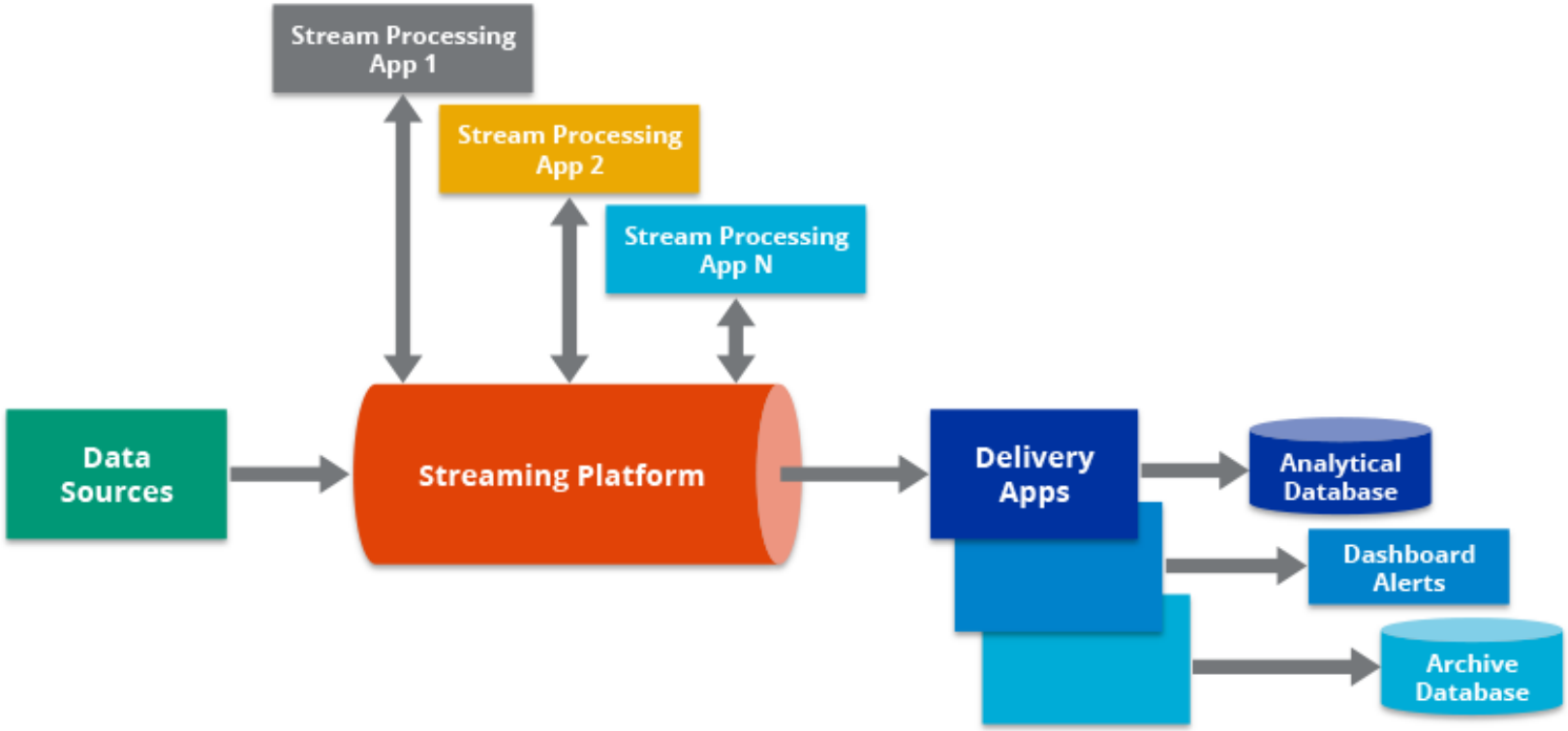
- ✓ Источник;
- ✓ Один или нескольких этапов обработки;
- ✓ Пункт назначения, который также может называться стоком.



Конвейер данных



Конвейер данных



Конвейер данных на основе Kafka

Сценарии использования Kafka при создании конвейера данных:

- Создание конвейера данных, в котором Apache Kafka представляет собой одну из двух конечных точек;
- Создание конвейера данных между двумя различными системами с Kafka в качестве промежуточной.

Требования при создании конвейеров данных

Требования: своевременность

- Kafka как платформу потоковой обработки с масштабируемым и надежным хранилищем можно использовать для поддержки от конвейеров, работающих в режиме реального времени, до пакетов, поступающих раз в час;
- Возможна реализация пакетного режима работы потребителей с запуском раз в час.

Требования: надёжность

- Kafka способна обеспечить как минимум однократную доставку;
- Конвейер на основе Kafka можно сделать строго однократным;

Требования: высокая и переменная нагрузка

- Конвейеры данных должны масштабироваться до очень высокой производительности;
- Умение Kafka масштабироваться за счёт независимого добавления производителей и потребителей даёт возможность динамически и независимо масштабировать любую из сторон конвейера, чтобы приспособиться к меняющимся требованиям;
- Kafka также поддерживает несколько типов сжатия.

Требования: форматы данных

- Одна из важнейших задач конвейеров данных - согласование их форматов и типов;
- Kafka и API Kafka Connect форматы данных совершенно не важны (применяются сериализаторы и десериализаторы);
- Вне зависимости от задействованного формата данных Kafka не ограничивает выбор преобразователей.

Требования: преобразование данных

Существуют две парадигмы создания конвейеров данных:

- ETL - конвейер данных отвечает за изменение проходящих через него данных;
- ELT - конвейер лишь минимально преобразует данные с тем, чтобы попадающие по месту назначения данные как можно меньше отличались от исходных;

Требования: безопасность

В терминологии конвейеров данных основные проблемы безопасности состоят в следующем:

- Можем ли мы гарантировать шифрование проходящих через конвейер данных?
- Кому разрешено вносить в конвейер изменения?
- Может ли конвейер при чтении им данных из мест с контролируемым доступом обеспечить должную аутентификацию?

Требования: обработка сбоев

Важно заранее предусмотреть обработку сбоев:

- Можно ли сделать так, чтобы дефектные записи никогда не попадали в конвейер?
- Можно ли восстановить работу системы после обработки не поддающихся разбору записей?
- Можно ли исправить «плохие» записи и обработать их заново?
- Что если «плохая» запись выглядит точно так же, как нормальная, и проблема вскроется лишь через несколько дней?

Требования: связывание и быстрота адаптации

Одна из важнейших задач конвейеров данных - независимость источников и приемников данных. Связывание может возникнуть множеством способов:

- ❖Узкоспециализированные конвейеры;
- ❖Потери метаданных;
- ❖Чрезмерная обработка.

Выбор между Kafka Connect и обычными производителями и потребителями Kafka

- Используйте клиенты Kafka тогда, когда у вас есть возможность модифицировать код приложения, к которому вы хотите подключиться, и когда вы хотели бы поместить данные в Kafka или извлечь их из нее;
- Kafka Connect же можете задействовать для подключения Kafka к хранилищам данных, созданным не вами, код которых вы не можете или не должны менять;
- Если же нужно подключить Kafka к хранилищу данных, для которого ещё не существует коннектора, можно написать приложение, задействующее как клиенты Kafka, так и Kafka Connect.

Kafka Connect

- Фреймворк Kafka Connect — часть Apache Kafka, обеспечивающая масштабируемый и гибкий способ перемещения данных между Kafka и другими хранилищами данных;
- Плагины-коннекторы (connector plugins) - исполняемые Kafka Connect библиотеки, отвечающие за перемещение данных;
- Kafka Connect выполняется в виде кластера процессов-исполнителей (worker processes);
- Для настройки коннекторов используется REST API;
- Коннекторы запускают дополнительные задачи (tasks), которые могут выполняться параллельно;
- Kafka Connect использует преобразователи формата (convertors).

Запуск Kafka Connect

- Kafka Connect поставляется вместе с Apache Kafka;
- Запуск исполнителя Kafka Connect:

  bin/connect-distributed.sh config/connect-distributed.properties
- Основные настройки исполнителей Connect:
  - bootstrap.servers - список брокеров Kafka, с которыми будет работать Connect;
  - group.id - все исполнители с одним идентификатором группы образуют один кластер Connect;
  - key.converter и value.converter - Connect может работать с несколькими форматами данных, хранимых в Kafka;
- key.converter.schema.enable и value.converter.schema.enable – включение схемы в сообщение (true/false);

Запуск Kafka Connect

- Сообщения в формате Avro также содержат схему, но необходимо задать местоположение реестра схем с помощью свойств key.converter.schema.registry.url и value.converter.schema.registry.url;
- Вы можете задать конкретный порт для API REST через настройки rest.host.name и rest.port;
- Настроив исполнителей, можно убедиться, что ваш кластер работает, с помощью REST API, например через команду:
  - curl http://localhost: 8083/
- Просмотреть доступные плагины коннекторов:
  - curl http://localhost: 8083/connector-plugins

Автономный режим

- У Kafka Connect имеется автономный режим:
  - bin/connect-standalone.sh
- Файл настроек коннекторов можно передать в командной строке, а не через REST API;
- В автономном режиме все коннекторы и задачи выполняются на одном автономном исполнителе;
- Connect в автономном режиме обычно удобнее использовать для разработки и отладки, а также в случаях, когда коннекторы и задачи должны выполняться на конкретной машине.

Пример коннектора: файловый источник и файловый приемник

- Запуск распределенного исполнителя Connect:
  - bin/connect-distributed.sh config/connect-distributed.properties &

- Удалить коннектор можно с помощью команды:
  - curl -X DELETE http://localhost: 8083/connectors/dump-kafka-config

Создание своих собственных коннекторов

- API коннекторов общедоступен, так что каждый может создать новый коннектор;

- Можно опубликовать свой коннектор, чтобы другие инженеры его увидели и смогли переиспользовать.

Как работает Connect

Чтобы понять, как работает Connect, необходимо разобраться с тремя основными его понятиями:

- Коннекторы и задачи;
- Исполнители;
- Преобразователи форматов и модель данных Connect.

Коннекторы и задачи

Плагины коннекторов реализуют API коннекторов, состоящее из двух частей:

- Коннекторы – отвечают за выполнение трёх важных вещей:
  - определение числа задач для коннектора
  - разбиение работы по копированию данных между задачами;
  - получение от исполнителей настроек для задач и передачу их далее;
- Задачи - отвечают за получение данных из Kafka и вставку данных в неё.

Исполнители

- Отвечают за обработку HTTP-запросов с описанием коннекторов и их настроек, а также за хранение настроек коннекторов, запуск коннекторов и их задач, включая передачу соответствующих настроек;
- Исполнители отвечают также за автоматическую фиксацию смещений для коннекторов как источника, так и приемника и за выполнение повторов в случае генерации задачами исключений;
- Главное преимущество API Connect состоит в разделении обязанностей между коннекторами и исполнителями.

Преобразователи форматов и модель данных Connect

- API Kafka Connect включают API данных, который в свою очередь включает как объекты данных, так и описывающие эти данные схемы;
- Пользователи, настраивая исполнитель (или коннектор), выбирают преобразователь формата, который будет применяться для сохранения данных в Kafka;
- API Connect может поддерживать различные типы хранимых в Kafka данных вне зависимости от реализации коннекторов.

Управление смещениями

- Управление смещениями - один из удобных сервисов, предоставляемых исполнителями коннекторам;
- Записи, возвращаемые коннектором исполнителям Connect, включают информацию о логическом разделе и логическом смещении (это не партиции и смещения в Kafka, а разделы и смещения в том виде, в каком они нужны в системе источника);
- Отслеживание смещений самим фреймворком должно облегчить разработчикам задачу написания коннекторов и до определенной степени гарантировать согласованное поведение при использовании различных коннекторов.

Альтернативы Kafka Connect

- Не во всех проектах предпочитают именно решения на основе Kafka;
- Многие разработчики закладывают в основу своих архитектур данных такие системы, как Hadoop или Elasticsearch;
- В некоторых системах есть собственные утилиты ввода и обработки данных - Flume для Hadoop и Logstash или Fluentd для Elasticsearch;
- Будет правильным использовать API Kafka Connect в случаях, когда Kafka является неотъемлемой частью архитектуры, а цель состоит в соединении большого числа источников и приемников.

ETL-утилиты на основе GUI

- Множество классических систем наподобие Informatica, а также их альтернатив с открытым исходным кодом, например Talend и Pentaho, и даже более новых вариантов, таких как Apache NiFi и StreamSets, поддерживают использование Apache Kafka в качестве как источника данных, так и целевой системы;
- Основной целью интеграции данных должна быть добросовестная передача сообщений при любых условиях, в то время как большинство ETL-утилит привносят ненужную сложность;
- Kafka может послужить прекрасной заменой для ETL-утилиты, которая занимается только интеграцией хранилищ данных.

Фреймворки потоковой обработки

- Практически все фреймворки потоковой обработки могут читать данные из Kafka и записывать их в некоторые другие системы;
- Если ваша целевая система поддерживается в Kafka и вы все равно собираетесь использовать конкретный фреймворк потоковой обработки для обработки поступающих событий из Kafka, имеет смысл задействовать его же для интеграции данных.

Спасибо за внимание