Домашнее задание

Построение простейших пайплайнов переливки данных с помощью DIT

**Цель:**

Реализовать любой из представленных двух вариантов практической работы, поработать с БД и инструментами по переливке данных, отработать навыки развертывания инструментов и взаимодействия с облаком Яндекса и Докером. Подготовиться к выполнению более сложных домашних работ в дальнейшем.

**Описание/Пошаговая инструкция выполнения домашнего задания:**

**Вариант №1**

1. Развернуть БД Postgres
2. Развернуть БД ClickHouse
3. Развернуть инструмент для выгрузки данных Airbyte.
4. Залить тестовые данные в каждую из БД с помощью консольных клиентов или через IDE
5. С помощью инструмента для переливки данных Airbyte произвести обмен данными между БД. Из PG в CH и наоборот

Установку и развертывание всех инструментов и БД можно реализовать как локально, так и на виртуальных машинах в облаке Yandex.  
Инструкции те же, что и рассмотренные в рамках вебинара. Доступны по ссылке - <https://github.com/AlexeyFerum/teaching_time/wiki>

**Вариант №2**

1. Скачиваем себе образ Apache NiFi (если еще не сделали этого на занятии):

docker pull apache/nifi:latest

1. Собираем контейнер с mapping-ом 2-х портов: через который будем локально смотреть в интерфейс и через который будем общаться через curl:

docker run --name nifi -p 9090:9090 -p 8081:8081 -d -e NIFI\_WEB\_HTTP\_PORT='9090' apache/nifi:latest

1. Создаем стартовую точку pipeline-а:

ListeHTTP-процессор.  
Порт: 8081 (его мы заранее мэппили для docker-контейнера)  
Base Path: loglistener

1. Второй узел. Он будет перенаправлять данные в соответствии с тем, что пришло на наш веб-сервер извне:

RouteOnContent-процессор.  
Подключаем его связью "success" к ListeHTTP.  
Match Requirement: "content must contain match"  
Добавляем ему новое свойство errortext со значением "ERROR".  
Теперь из узла способно выходить 2 потока: для flowfiles с ключевым значением (под названием errortext) и второй: unmatched

1. Проверяем работоспособность 1-го узла:

Нажимаем "Play" для ListeHTTP-процессора  
Из командной строки делаем: " curl --data "someRandomText" 127.0.0.1:8081/loglistener"  
Жмем Refresh (после клика ПКМ)  
Видим, что в него пришли данные

1. Добавляем узел, который будет объединять несколько файлов в один:

Добавляем MergeContent-процессор.  
Создаем 2 связи из RouteOnContent в MergeContent: errortext и unmatched  
Настраиваем узел (ПКМ => configure):

* Scheduling -> Run Schedule: 10 sec
* Merge Strategy: Bin-Packing Algorithm
* Correlation Attribute Name: RouteOnContent.Route - этот параметр позволит нам писать файлы в разные директории в дальнейшем
* Merge Format: Binary Concatenation
* Maximum Number of Entries: 500
* Maximum Bin Age: 90s (максимальное время жизни flow-файла, после которого его принудительно выведет из pipeline)

Мы будем склеивать текстовые файлы, чтобы не плодить их в большом количестве.  
Поэтому укажем формат склейки:

* Delimiter Strategy: Text
* Demarcator: открываем и в поле нажимаем Shift+Enter (это аналог \n- перевод каретки на новую строку)

1. Сохраняем полученные артефакты в облачный s3 Yandex.Cloud:

PutS3Object-процессор

* Directory: nifiHW/${RouteOnContent.Route}/${now()}.txt

Создаем бакет nifiHW через интерфейс.  
У нас будет 2 папки в s3: merged и unmatched  
Делаем связи от узла MergeContent вида "merged" и "unmatched"

Дополнительно требуется разобраться с тем, как начать писать в бакет через API.

1. Запускаем весь pipeline

1. Шлем много разных запросов через cli: содержащих текст ERROR внутри и нет.

Примеры:

curl --data "adbrsgbndt ERROR fevrtb" 127.0.0.1:8081/loglistener  
  
curl --data "uiebveovne" 127.0.0.1:8081/loglistener

1. Наблюдаем за тем, что происходит внутри pipeline-а.

1. Заходим через интерфейс в s3 Yandex.Cloud и смотрим, какие новые папки и файлы появились внутри.

**Результатом выполнения любой из вариантов практической работы является отчет со скриншотами, описанием выполненных действий и ошибок, которые удалось решить.**