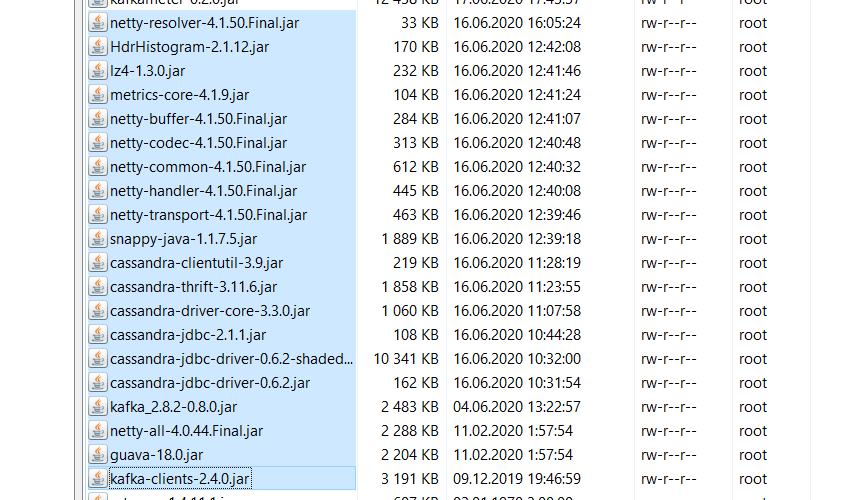
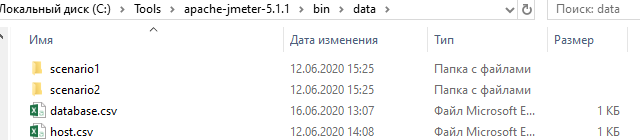
1. В библиотеки jmeter поместите файлы, обеспечивающие работу продюсера Kafka и работу с БД Cassandra (список со скрина – итог экспериментов с разными подходами, не все из них нужны для финального решения)

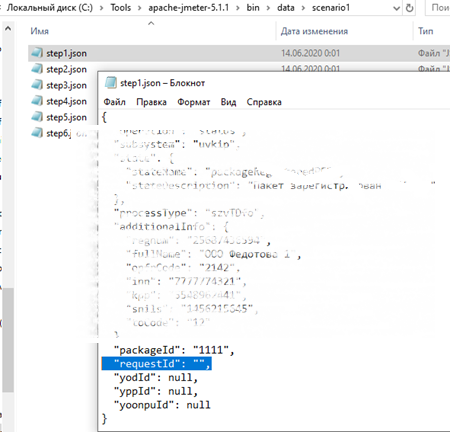


1. Подготовьте входные данные о сервере и сценариях. Они должны быть размещены в папке data внутри bin c jmeter

Структура папки:



Внутри папок scenario – набор файлов, каждый их которых соответствует шагу сценария:

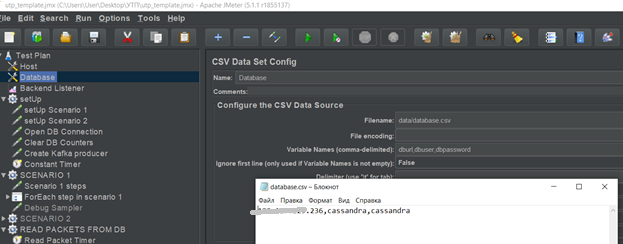


requestId во всех шагах должно быть пустое, параметризуется при выполнении.

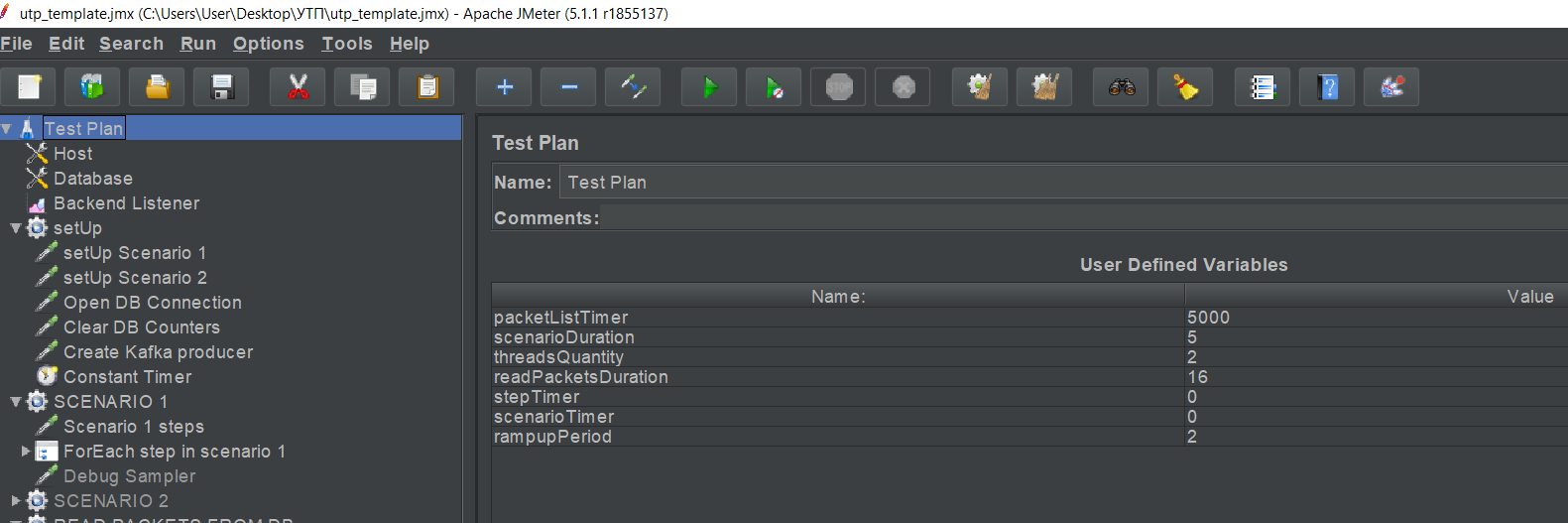
1. Внесите данные вашего стенда в файлы:

database.csv – адрес БД Cassandra, логин и пароль для доступна к БД

host.csv – адрес сервера Kafka, куда отправляются сообщения



1. Настройте профиль нагрузки



packetListTimer (ms) – период, с которым обращаемся к БД для чтения результатов

scenarioDuration (s) – не используется в общем случае, продолжительность нагрузки

threadsQuantity – кол-во виртуальных пользователей

readPacketsDuration (s) – продолжительность чтения результатов, позволяет читать данные из БД после завершения нагрузки

stepTimer (ms) – не используется в общем случае, ожидание между отправкой квитанций

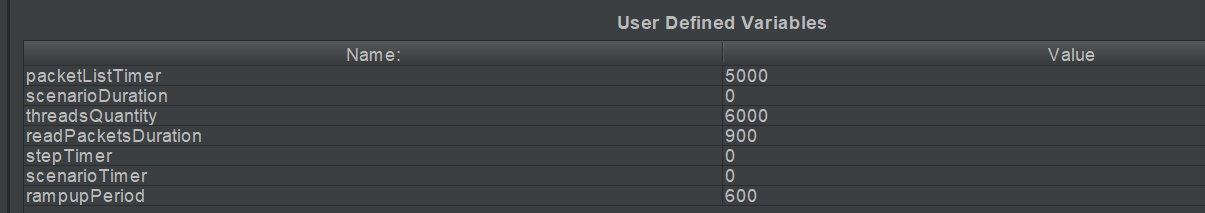
scenarioTimer (ms) – не используется в общем случае, ожидание между отправкой пакетов

rampupPeriod (s) – распределение подачи нагрузки.

Например, есть сценарий из 6 квитанций.

Нагрузка: 10 пакетов (60 квитанций/сек) на протяжении 10 минут.

Читаем из БД раз в 5 секунд. После окончании нагрузки нужно продолжать читать результаты из БД 5 минут. Настройки следующие:

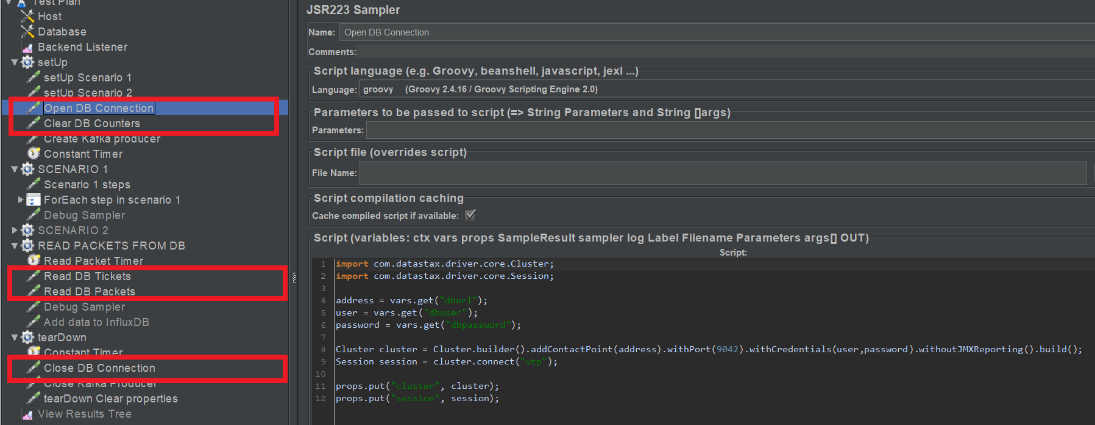


1. Работа с БД идёт в 3 этапа – создание и закрытие соединения вынесены из нагрузки, делаются 1 раз. Также в начале тесте очищаются таблицы в БД со счётчиками.

setup – создаём соединение, очищаем счётчики

при нагрузке – читаем счётчики с заданным интервалом, чтобы оценить результат нагрузки, кол-во созданных квитанций и пакетов

teardown – закрываем соединение



1. Работа с Kafka-producer’ом идёт в 3 этапа – инициализация и закрытие продюсера вынесены из нагрузки, делаются 1 раз.

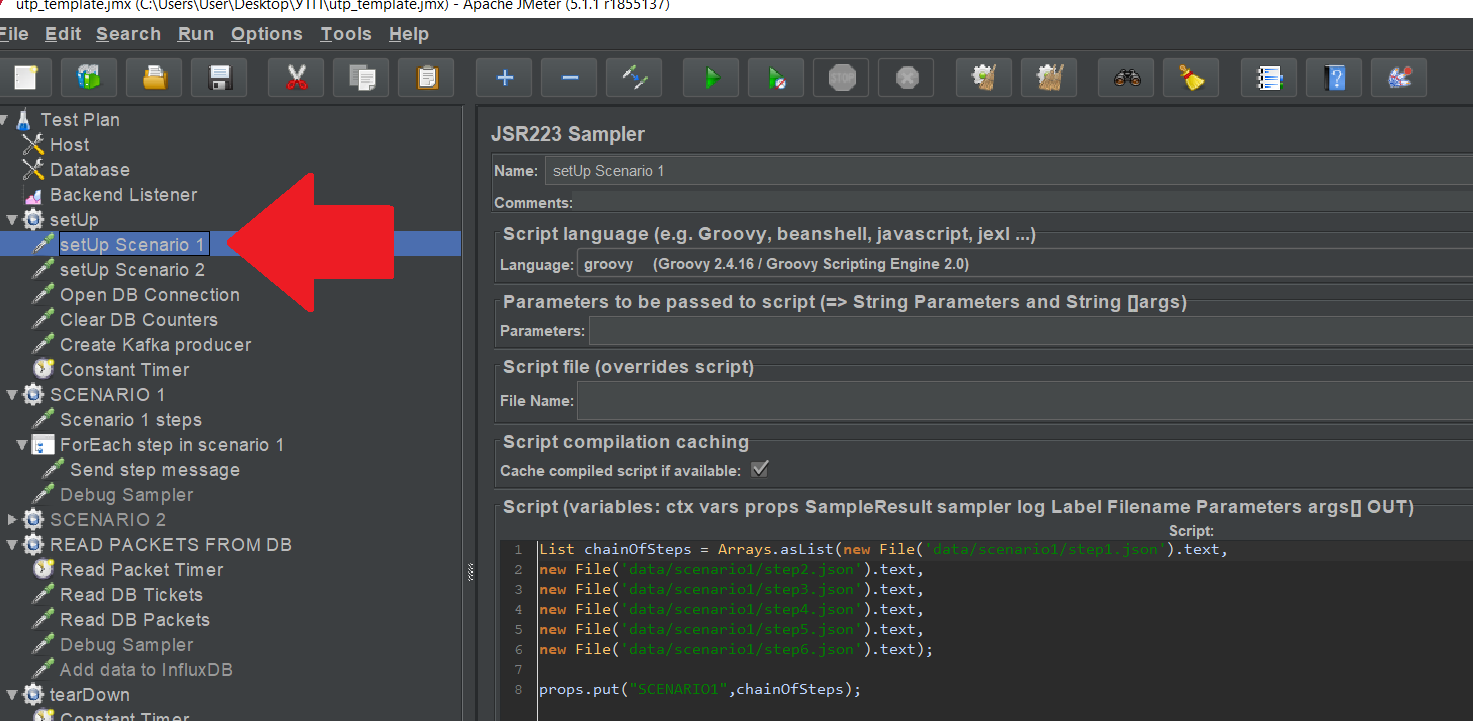
setup – создаём producer

при нагрузке – отправляем сообщение

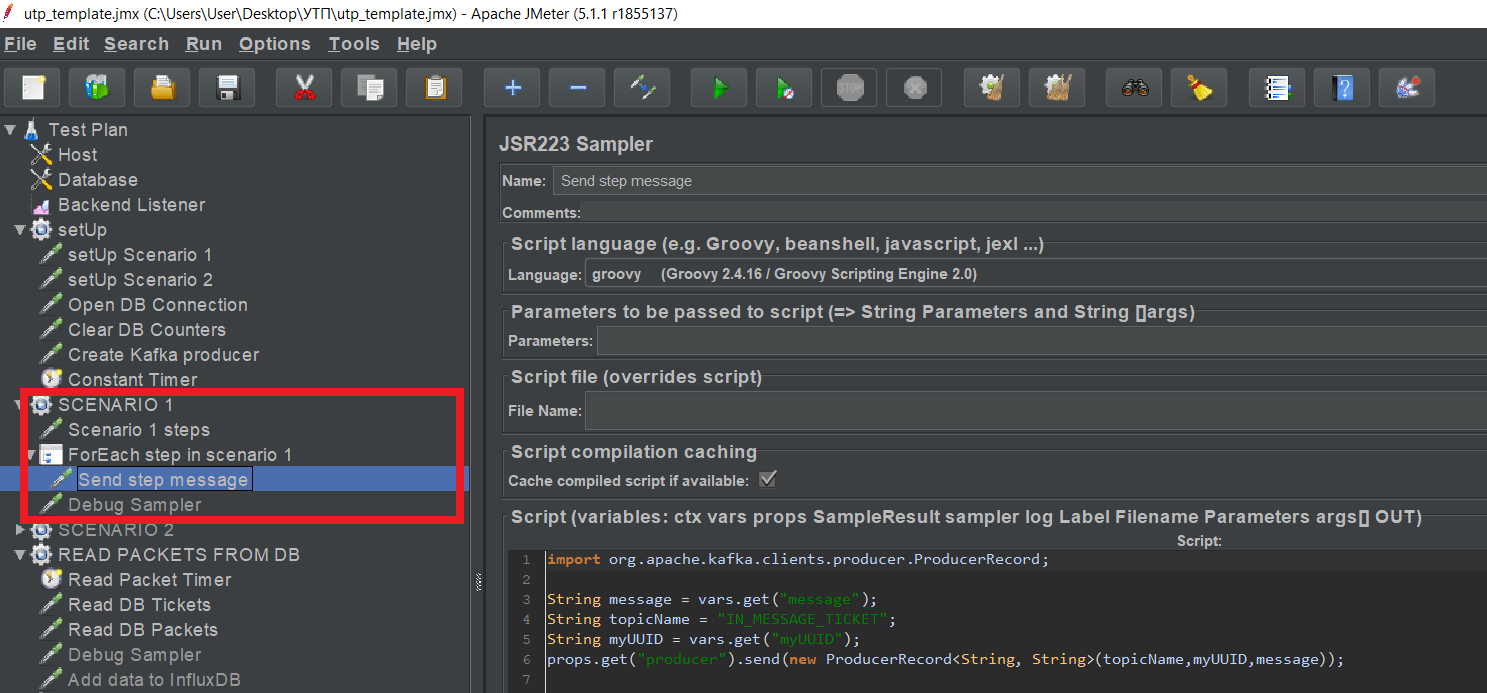
teardown – закрываем producer

Примечание: от решений kafkameter и pepperbox отказались, т.к. первый не позволяет разделить создание продюсера и отсылку (создается продюсер при каждой отсылке), а второй не позволяет использовать переменные в сообщениях.

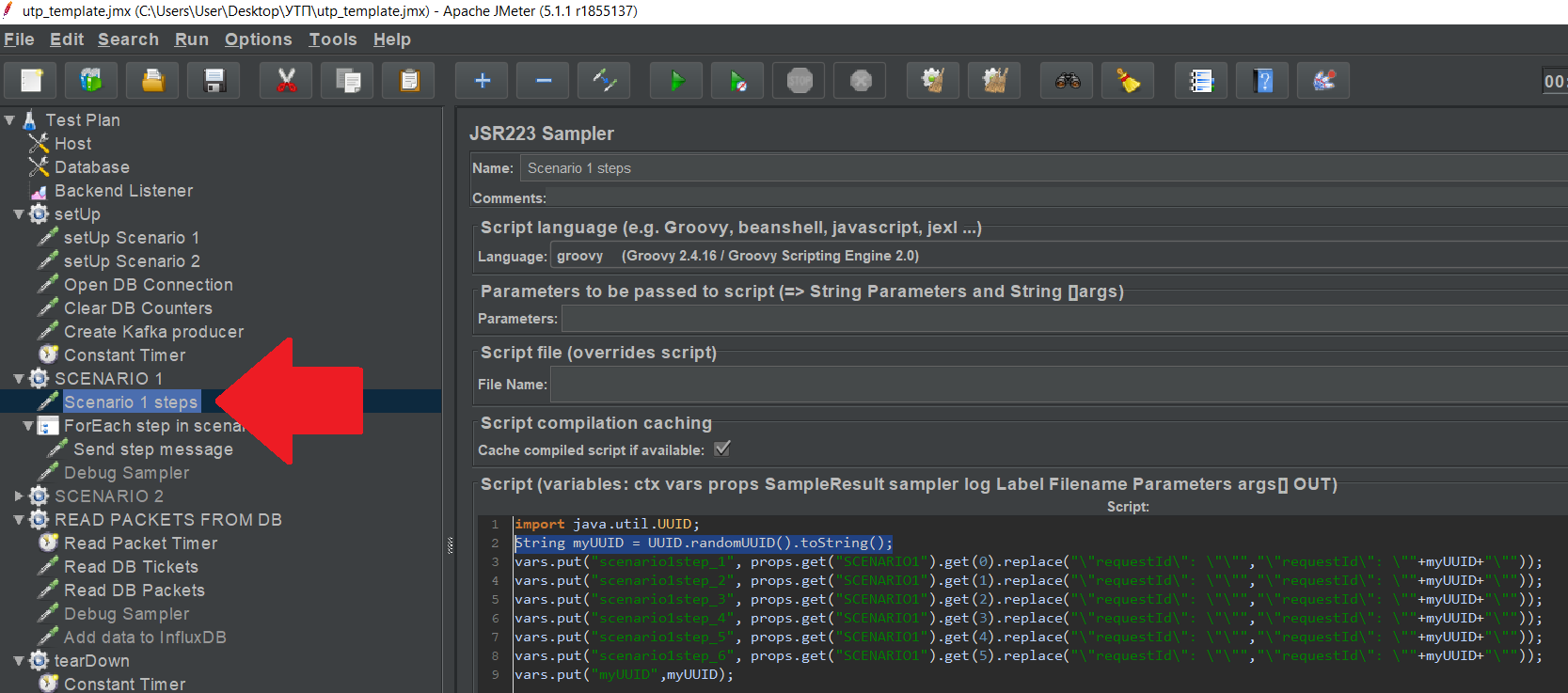
1. Данные сценариев считываются из файлов json 1 раз в setup



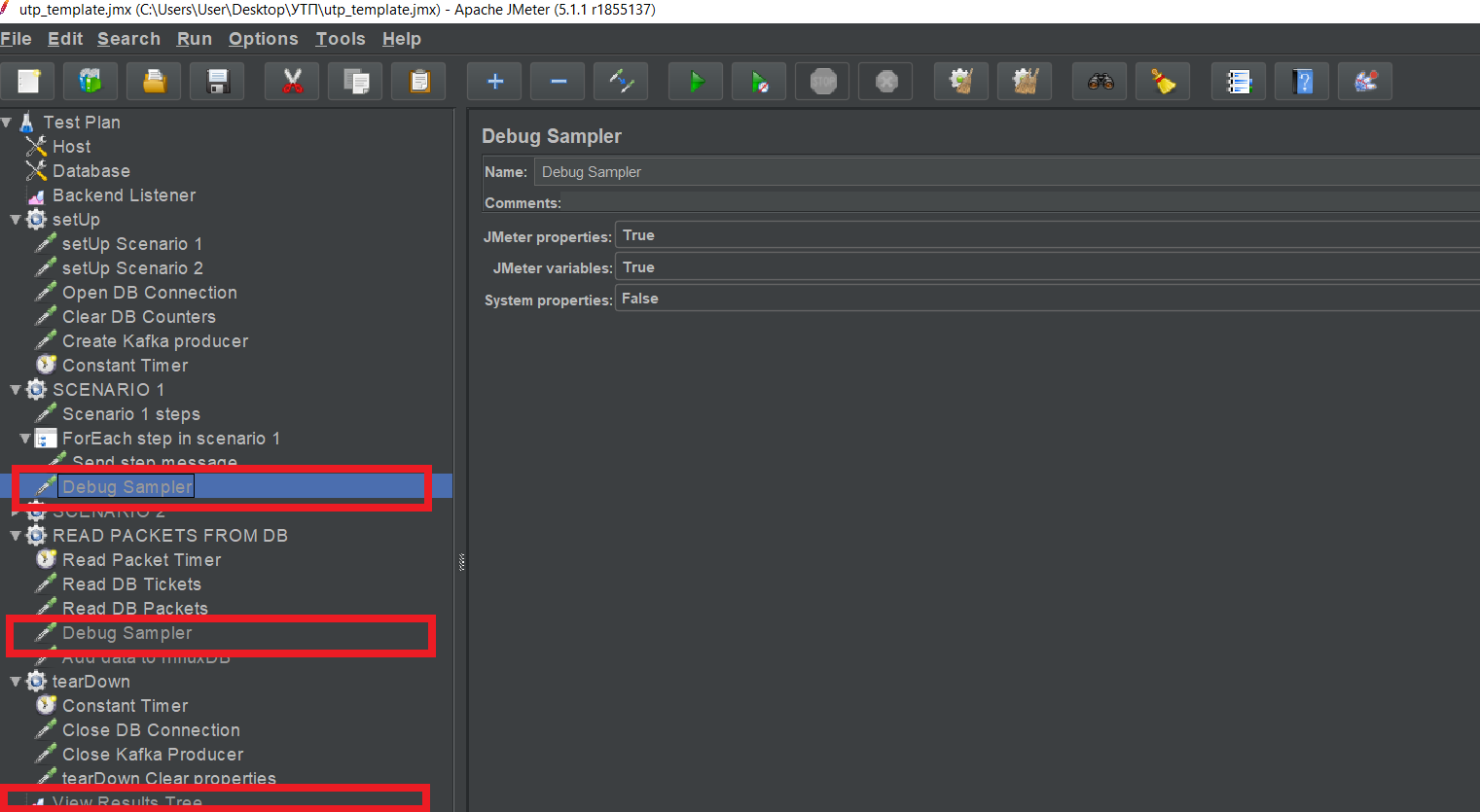
1. Отсылка квитанций из пакета осуществляется в итераторе, каждый итератор – пакет, каждый шаг итератора – квитанция.



1. Внутри сценария параметризуется requestId – создаётся GUID, единый для всех квитанций одного пакета.



1. Вспомогательные элементы для отладки должны быть закомментированы при создании нагрузки.



1. Настройки Kafka-producer (могут быть изменены под частные цели)

Для обеспечения надежности отправки рекомендуется:

kprops.put("acks", "all"); - получаем подтверждение от брокера об отправке

kprops.put("retries", 1); - попытаться ещё раз при неудаче

kprops.put("linger.ms", 1); - ожидание между отправкой пакета и формированием следующего

kprops.put("max.in.flight.requests.per.connection",2); - для обеспечения порядка отправок

Для обеспечения скорости отправки рекомендуется:

kprops.put("producer.type","async");

kprops.put("batch.size", 5000); - уменьшен стандартный размер пакета (чаще отправка)

1. Приложен json файл для вывода данных в Grafana