**Управление качеством программных систем. Супрунов Матвей, ПИ21-3**

**ЛАБОРАТОРНАЯ РАБОТА №3**

Тема лабораторной работы: создание тестовой документации.

**Цель работы:**

Целью данной лабораторной работы является написание тестовой документации для реализованной системы микробатчевой обработки для преобразования потоковых данных ресторана в качестве контейнеризованных микросервисов на Python. Тестирование включает в себя проверку основного функционала приложения, а также его надежности, безопасности и удобства использования с помощью smoke-тестов. Помимо этого будут рассмотрены положительные и отрицательные сценарии тестирования.

**Описание тестируемого приложения:**

Система подразумевает собой чтение сообщений из Kafka, наполнение stg-слоя, обогащение данными из Redis и отправка обогащённых сообщений в новый топик. Система, которая состоит из обработчиков сообщений. Каждый сервис — это отдельный обработчик, который получает и отдаёт сообщение. При этом в процессе обработки сервис порождает новые артефакты, оседающие в системе, — например, заполняет таблицы разных слоёв данных в БД.

Формат входного сообщения для STG-слоя из Kafka - JSON. Выходное сообщение должно содержать всю необходимую информацию о передаваемых объектах. Части этой информации нет во входном сообщении, а именно: данных о покупателях, ресторане (лишь id) и продукте, т. е. заказанном блюде. Эту информацию необходимо будет доложить.

С заданной периодичностью планировщик будет запускать функцию run, внутри которой вычитываются и обрабатываются сообщения из Kafka. Таким образом, будет реализован micro-batch processing — обработка небольших пакетов сообщений.

**Тест-план:**

* что надо тестировать – своевременное чтение сообщений из топика, их корректную структуру и наполненность, целостность данных в dds слое.
* график тестирования – при каждой новой пересборке контейнеризованных сервисов.
* необходимое оборудование и программные средства – Kafka, Redis, PostgreSQL, Kubernetes, Docker
* возможные риски и пути их решения – падение парсера по истечении таймаута или конца сообщений в очереди. Необходимо предпринять обработчик ошибок для данного сценария, который позволяет «мягко» прерывать выполнение программы.

**Таблица 1. Список тест-сьютов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Автор** | **Приоритет** | **Заголовок** | **Список тест-кейсов** |
| 1 | developer | 1 | Очередь в Кафке | * 1. Корректная обработка неполноценных сообщений   2. Обогащение пропущенных данных в сообщении |
| 2 | developer | 2 | Процесс обновления данных в PostgreSQL | * 1. Проверка на корректную вставку имеющихся значений   2. Проверка на целостность через констрэинты |
| 3 | developer | 3 | Контейнеризация | * 1. Проверка на совместимость с другими операционками   2. Отработка автоматизировнного дебага контейнеров в Kubernetes |

**Таблица 2. Список тест-кейсов**

Очередь в Кафке

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Описание** | **Предусловия** | **Шаги** | **Ожидаемый результат** |
| 1.1 | Корректная обработка неполноценных сообщений (позитивный) | Бегущий топик | Запущен консьюмер сообщений топика, бежит контейнер | Неполноценный сообщения заполняются Null значениями. Если пустых значений больше трех, то сообщение скипается |
| 1.2 | Обогащение пропущенных данных в сообщении | Бегущий топик, запущенный сервис | Подрубается key-value хранилище Redis, данные джойнятся по product\_id | Данные пишутся в новый топик в обогащенном виде |

Процесс обновления данных в PostgreSQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Описание** | **Предусловия** | **Шаги** | **Ожидаемый результат** |
| 2.1 | Проверка на корректную вставку имеющихся значений | Схема для таблиц в детальном слое прописана | Изначальная запись в таблицу была изменена на upsert-метод путем обновления данных по схожему ключу. | Данные перезаписываются с новыми данных по атрибутах. |
| 2.2 | Проверка на целостность через констрэинты | Схема для таблиц в детальном слое прописана | Добавлены констрэинты в схему по значимым колонкам | С помощью чекеров, ограничений на primary ключ и not null |

Контейнеризация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Описание** | **Предусловия** | **Шаги** | **Ожидаемый результат** |
| 3.1 | Проверка на совместимость с другими операционками | Контейнер спокойно запускается на локальной машине | Прописать в манифестах (деплойменте) совместимость с другими платформами | При развертывании контейнера на другой машине сервис спокойно отрабатывает сырые данные из сообщений топиков |
| 3.2 | Отработка автоматизировнного дебага контейнеров в Kubernetes | Пакетный менеджер хелм спокойно менеджирует запущенные конейнеры | Отрубить интернет | Парсинг сообщений прерывается, при появлении интернета пишутся несчитанные сообщения |

**Выводы по работе:**

В результате данной лабораторной работы была составлена тестовая документация, включающая списки тест-сьютов и тест-кейсов для тестирования развернутых контейнеризованных сервисов, работающих с потоковыми данными. С помощью отработки тестовых сценариев была повышена отказоустойчивость и эффективность системы, были найдены некоторые недочеты, которые впоследствии были устранены.

**Список использованных источников:**

1. Официальная документация по Кафке: <https://kafka.apache.org/>
2. Руководства по Докере: <https://www.docker.com/>
3. Проектная работа на GitHub: <https://github.com/MatthewS-M/Stream-Cloud-Containerization-project/blob/main/StreamProject.md>