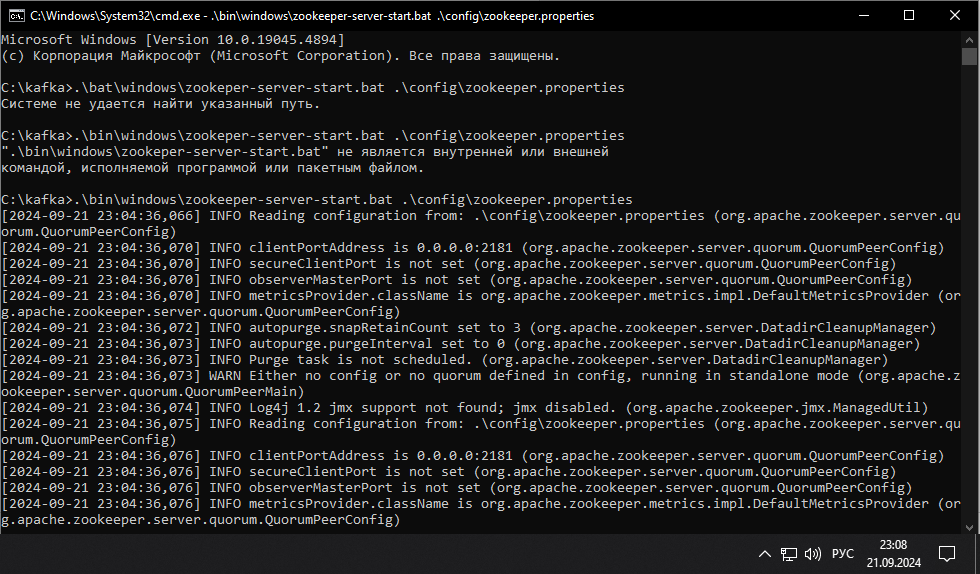
Аюб Эль-Шаварби Адхам Осама

ИСП222

Практическая 13

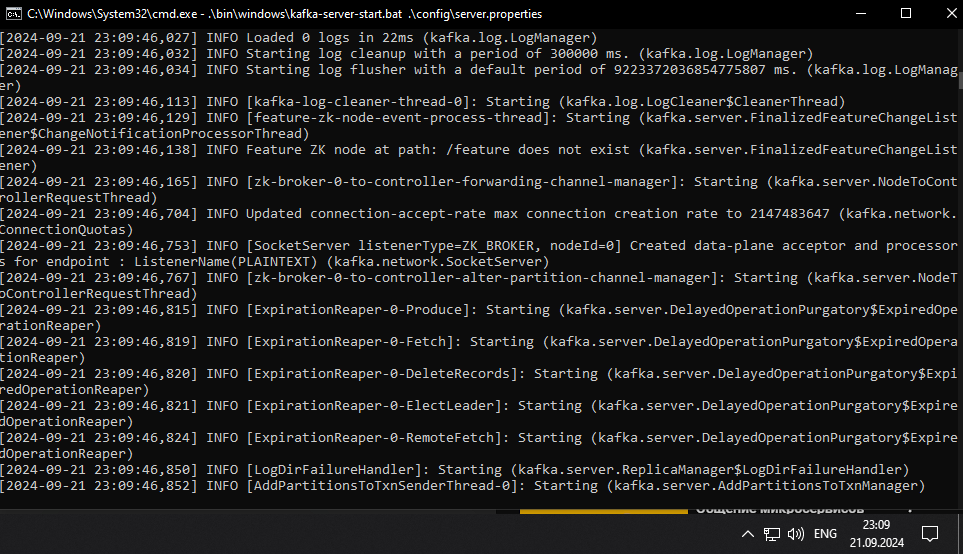
1. Запускаем Zookeeper через командную строку

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

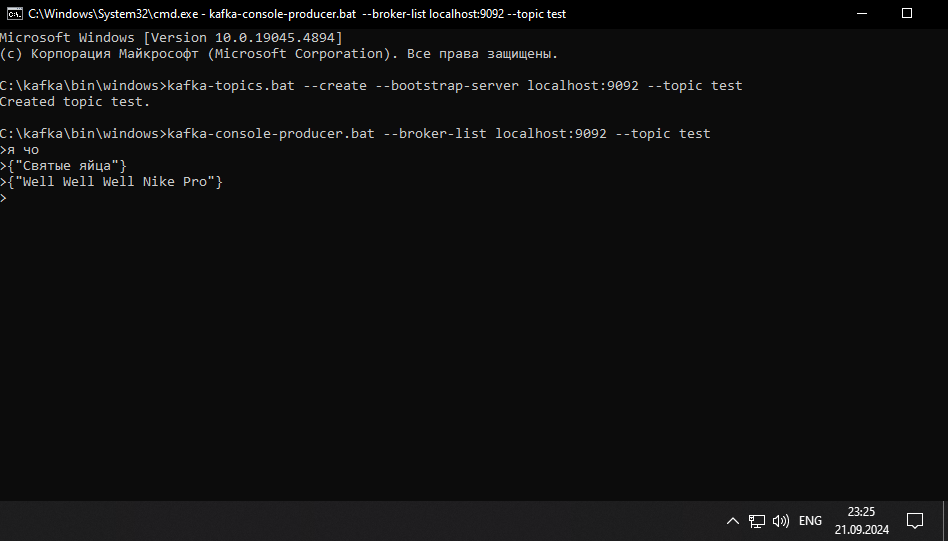


2. Запускаем Kafka через командную строку

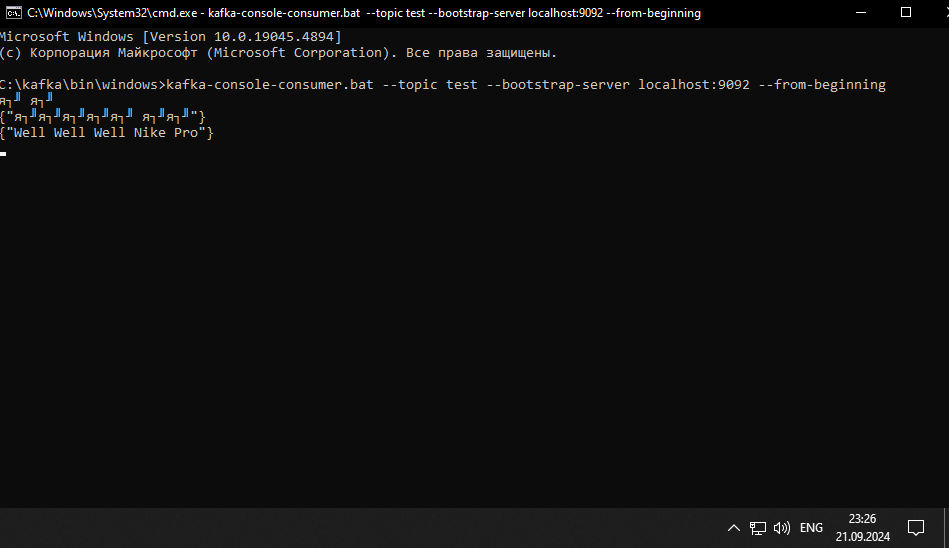
.\bin\windows\kafka-server-start.bat .\config\server.properties



3. Подрубаем Kafka Producer



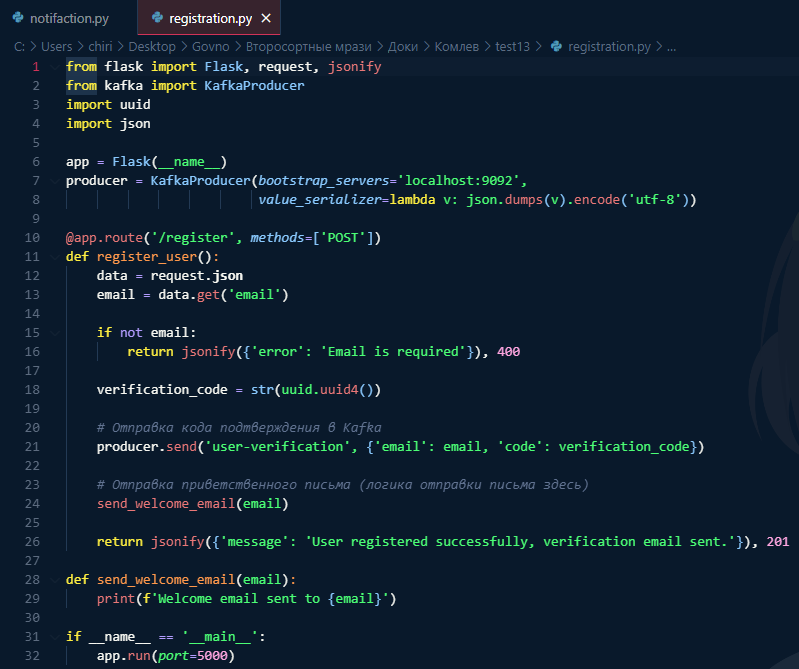
4. Подрубаем Kafka Consumer который будет получать данные с Producer

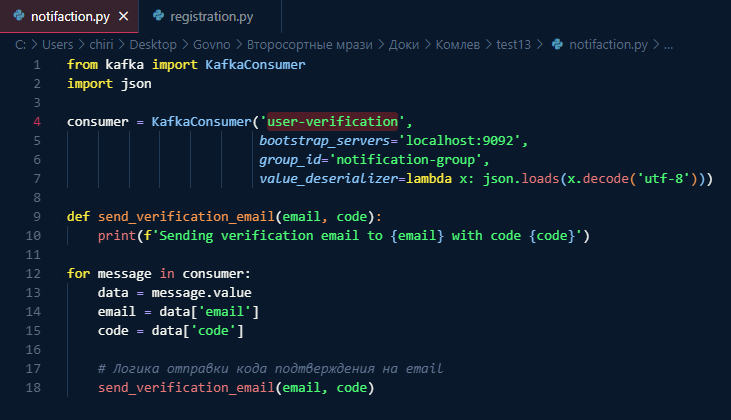


В пунктах 3 и 4 также видна проверка того что они связаны и пункт 4 получает данные из пункта 3.

Пишем два микросервиса

Один для регистрации, другой для отправки кода подтверждения

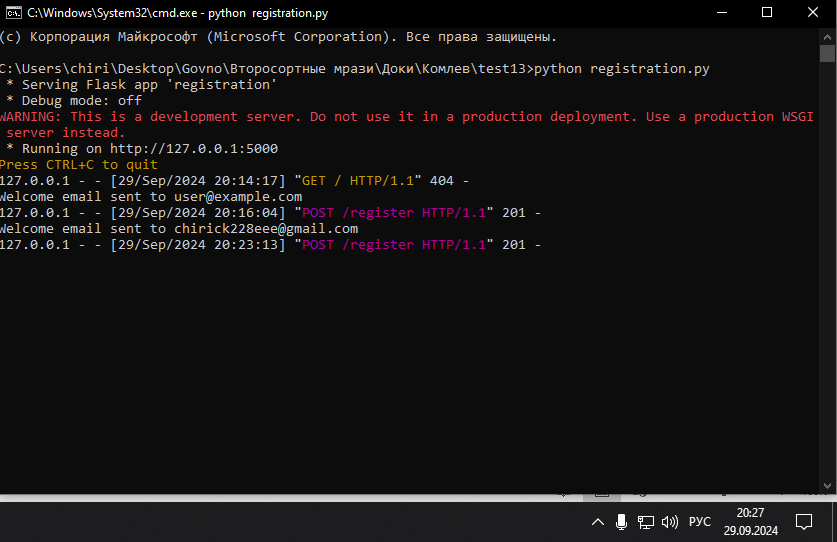




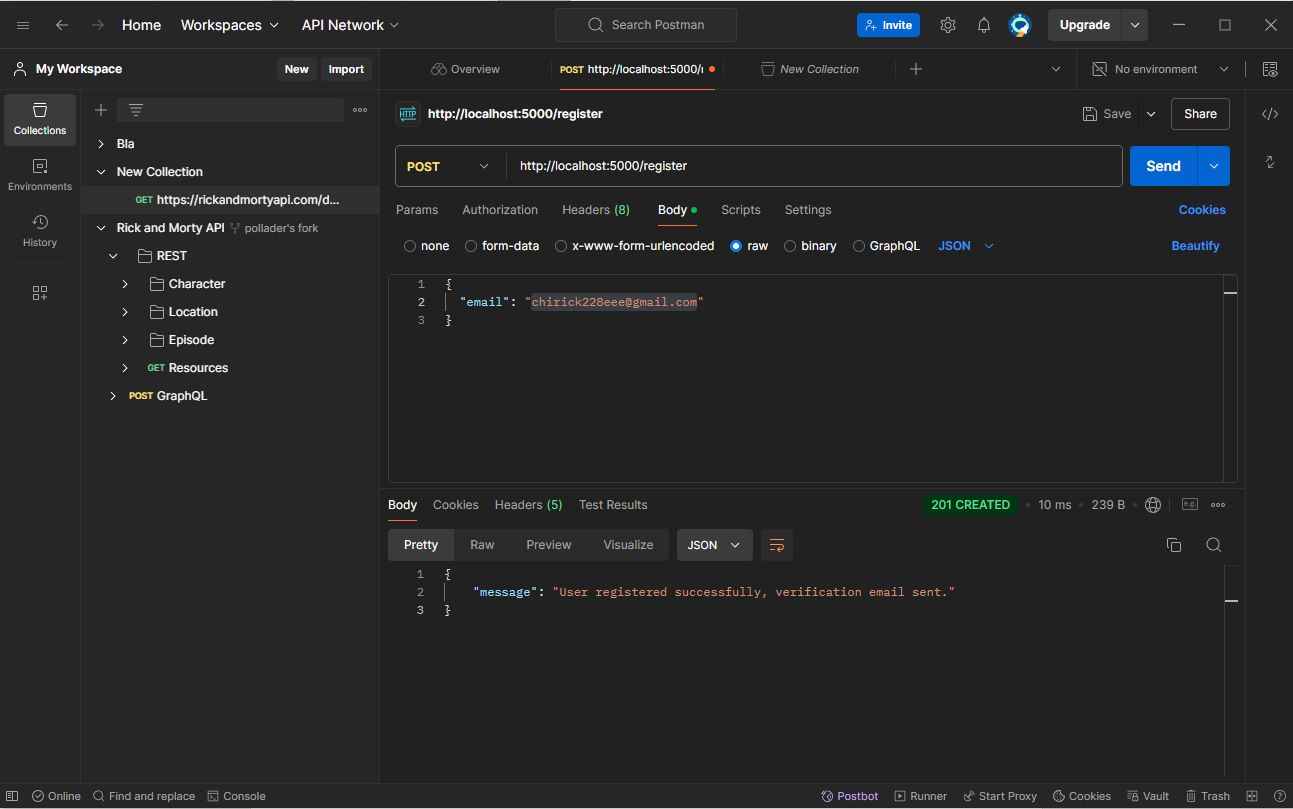
Дальше по факту они подключены к KafkaConsumer и KafkaProducer. На скриншотах видно.

Вот на этом скриншоте появился лог того что проги заработали корректно

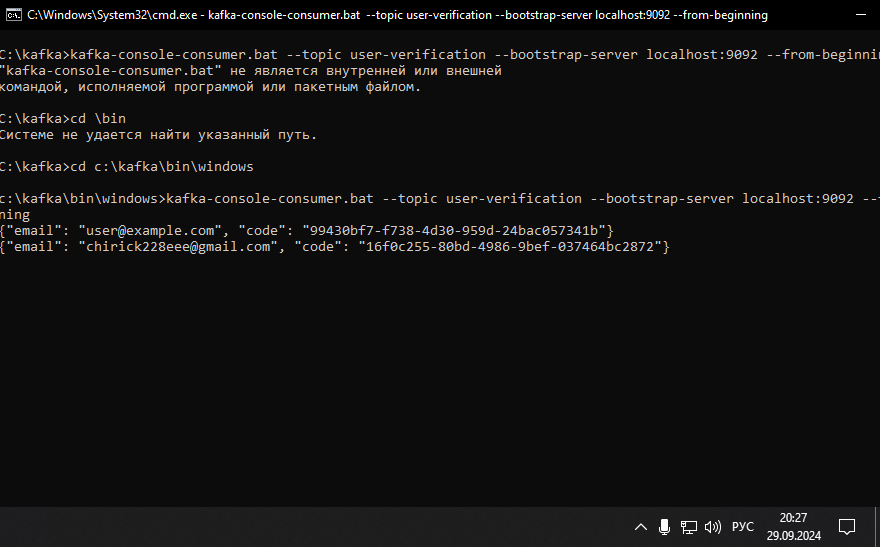




Через Postman отправить POST-запрос с JSON телом



И после этого мне в KafkaConsumer придет код



Для оборачивания Kafka и микросервисов в Docker Compose, нужно создать файл docker-compose.yml, который будет содержать определения сервисов

для Kafka, Zookeeper (если требуется) и ваших микросервисов (регистрация и

| **Характеристика** | **Apache Kafka** | **RabbitMQ** |
| --- | --- | --- |
| **Архитектура** | Распределенная платформа для потоковой передачи событий | Сообщений брокер с поддержкой сложной маршрутизации |
| **Подход к потребителям** | Pull-модель (потребители запрашивают сообщения) | Push-модель (брокер отправляет сообщения) |
| **Производительность** | Может обрабатывать миллионы сообщений в секунду | Может обрабатывать десятки тысяч сообщений в секунду |
| **Устойчивость к сбоям** | Высокая устойчивость, данные могут храниться долго | Устойчивость зависит от конфигурации очередей |
| **Гарантии доставки** | Поддержка «точно один раз» | Не гарантирует атомарность при обработке |
| **Модели маршрутизации** | Ограниченные возможности маршрутизации | Расширенные возможности маршрутизации через ключи и типы обмена |
| **Хранение сообщений** | Сообщения могут храниться неограниченно | Сообщения удаляются после обработки |
| **Сложность разработки** | Простая архитектура брокера, сложные потребители | Сложная архитектура брокера, простые потребители |

уведомления).

1. Что такое Apache Kafka и для чего он используется?

Apache Kafka — это **распределенная платформа для потоковой передачи событий** и обработки данных в реальном времени. Она предназначена для обработки высокоскоростных и объемных потоков данных, позволяя приложениям публиковать, подписываться и обрабатывать данные в реальном времени. Kafka часто используется для построения систем обработки данных, таких как мониторинг, анализ пользовательской активности и интеграция микросервисов.

2. Какие компоненты включает в себя архитектура Kafka?

Архитектура Kafka состоит из следующих основных компонентов:

* **Producers**: приложения, которые отправляют сообщения в Kafka.
* **Consumers**: приложения, которые получают сообщения из Kafka.
* **Brokers**: серверы, которые хранят и управляют сообщениями.
* **Topics**: категории или группы сообщений.
* **Partitions**: подмножества тем, которые позволяют разделить данные для повышения производительности и отказоустойчивости.
* **Zookeeper**: компонент, который управляет метаданными и координирует брокеры.

3. Что такое Topic в Kafka?

**Topic** — это логическая категория или группа сообщений, в которую отправляются данные. Каждый topic может иметь множество подписчиков (consumers), которые получают сообщения из него. Topics позволяют организовать данные по категориям, например, "регистрация пользователей" или "события покупок".

4. Какие роли выполняют Producer и Consumer в системе Kafka?

* **Producer**: отвечает за отправку сообщений в определенные topics. Producers могут отправлять данные асинхронно, что позволяет им не ждать подтверждения от брокеров перед отправкой следующих сообщений.
* **Consumer**: получает сообщения из topics и обрабатывает их. Consumers могут работать независимо друг от друга и могут быть организованы в группы для совместной обработки данных.

5. Какие гарантии по доставке сообщений предоставляет Kafka?

Kafka предлагает несколько уровней гарантии доставки сообщений:

* **At most once**: сообщения могут быть потеряны, но не дублированы.
* **At least once**: сообщения не потеряются, но могут быть дублированы.
* **Exactly once**: гарантирует, что каждое сообщение будет доставлено точно один раз (реализуется с помощью дополнительных механизмов).

6. Что такое Partition в Kafka?

**Partition** — это наименьшая единица хранения данных в Kafka. Каждый topic может быть разделен на несколько partitions, что позволяет распределить нагрузку между брокерами и обеспечить параллельную обработку сообщений. Каждое сообщение внутри partition хранится в порядке его поступления и имеет уникальный смещение (offset).

7. Какую роль играет Zookeeper в архитектуре Kafka?

Zookeeper управляет метаданными Kafka-кластера, включая информацию о topics, partitions и brokers. Он также отвечает за координацию брокеров, выбор лидеров для partitions и управление списками контроля доступа (ACL) для безопасности.

8. Что такое Consumer Group в Kafka и для чего он используется?

**Consumer Group** — это группа consumers, которые совместно обрабатывают сообщения из одного или нескольких topics. Каждый consumer в группе получает уникальные partitions, что позволяет эффективно распределять нагрузку и обеспечивать масштабируемость.

9. Как можно обеспечить отказоустойчивость в Kafka?

Отказоустойчивость обеспечивается через:

* **Репликацию partitions**: каждая partition может иметь несколько реплик на разных брокерах. Если один брокер выходит из строя, другой может взять на себя его задачи.
* **Использование Zookeeper** для управления состоянием кластера и выбора лидеров.

10. Что такое Kafka Stream?

**Kafka Streams** — это библиотека для обработки потоков данных в реальном времени с использованием данных из Kafka topics. Она позволяет разрабатывать приложения для анализа потоковых данных с возможностью выполнения сложных операций над ними (например, фильтрация, агрегация).

11. Какие протоколы передачи данных могут использоваться в Kafka?

Kafka использует собственный бинарный TCP-протокол для передачи данных между producers, consumers и brokers. Этот протокол оптимизирован для высокой производительности и минимизации задержек.

12. Какие уровни надежности предоставляет Kafka для хранения сообщений?

Kafka предлагает несколько уровней надежности:

* Сообщения могут храниться временно (с заданным временем хранения) или постоянно (без ограничения времени).
* Поддержка компактных topics позволяет хранить только последние сообщения по каждому ключу.

13. Как можно управлять конфигурацией Kafka?

Конфигурацией Kafka можно управлять через:

* Файлы конфигурации (например, server.properties).
* Admin API для динамического управления темами, брокерами и другими объектами.

14. Как обеспечивается масштабируемость Kafka?

Масштабируемость достигается за счет:

* Разделения topics на partitions, которые могут храниться на разных brokers.
* Добавления новых brokers в кластер без остановки работы системы.
* Использования consumer groups для распределенной обработки данных.