

# Prompt2Pic

Проект выполнили студенты групп М8О-406Б-21, М8О-407Б-21

Мезенин Олег

Меркулов Фёдор

Чапкин Владислав

# Бизнес цель проекта

- **Бизнес-цель:** создание удобного сервиса, позволяющего пользователям генерировать изображения по текстовым описаниям через Telegram, что упрощает доступ к возможностям генеративных моделей.

# Задачи проекта

- Разработать надёжную архитектуру, обеспечивающую:
  - масштабируемость: поддержка большого числа пользователей.
  - производительность: минимальные задержки при обработке запросов.
- Настроить взаимодействие между компонентами через Apache Kafka как надёжную систему очередей.
- Упаковать приложение в Docker-контейнеры для простоты развертывания.
- Реализовать защищённый процесс деплоя на облачной платформе cloud.ru.
- Развернуть рабочую демо-версию, доступную через Telegram.

# Цель МО

Рассмотрим три возможные цели для системы машинного обучения:

- Максимизация качества генерируемых изображений.
- Минимизация времени обработки запросов.
- Оптимизация баланса между качеством и скоростью.

# Вариант 1: Максимизация качества изображений

Для улучшения качества генерируемых изображений необходимо использование мощных генеративных моделей, которые способны создавать реалистичные изображения высокого разрешения.

## Преимущества:

- Реалистичность и детализация создаваемых изображений.
- Соответствие изображений текстовому описанию.

## Недостатки:

- Высокая потребность в вычислительных ресурсах.
- Потенциальное увеличение времени обработки запроса.

# Вариант 2: Минимизация времени обработки

Минимизация времени обработки достигается за счёт оптимизации модели и её настройки для работы на ограниченных ресурсах (в нашем случае CPU, вместо GPU).

## Преимущества:

- Быстрая генерация изображений.
- Увеличение количества пользователей, способных одновременно пользоваться сервисом.

## Недостатки:

- Возможное снижение качества генерируемых изображений.
- Ограничения на сложность текстовых описаний.

# Вариант 3: Оптимизация баланса между качеством и скоростью

Этот подход сочетает оба предыдущих, предоставляя возможность гибко настраивать модель в зависимости от нагрузки на систему. Например, при высокой нагрузке акцент делается на скорость, а при низкой — на качество.

## Преимущества:

- Гибкость в настройке под разные условия.
- Возможность адаптации под бизнес-цели и ожидания пользователей.

## Недостатки:

- Сложность реализации, связанная с необходимостью учёта множества факторов.

# Вывод: Какой вариант выбрать? 🤔

Мы выбирали комбинированный подход (Вариант 3)

Поскольку он позволяет учитывать потребности пользователей и особенности инфраструктуры.

Такой подход даёт больше возможностей для масштабирования сервиса и поддержания высокого качества изображений при минимальных задержках.

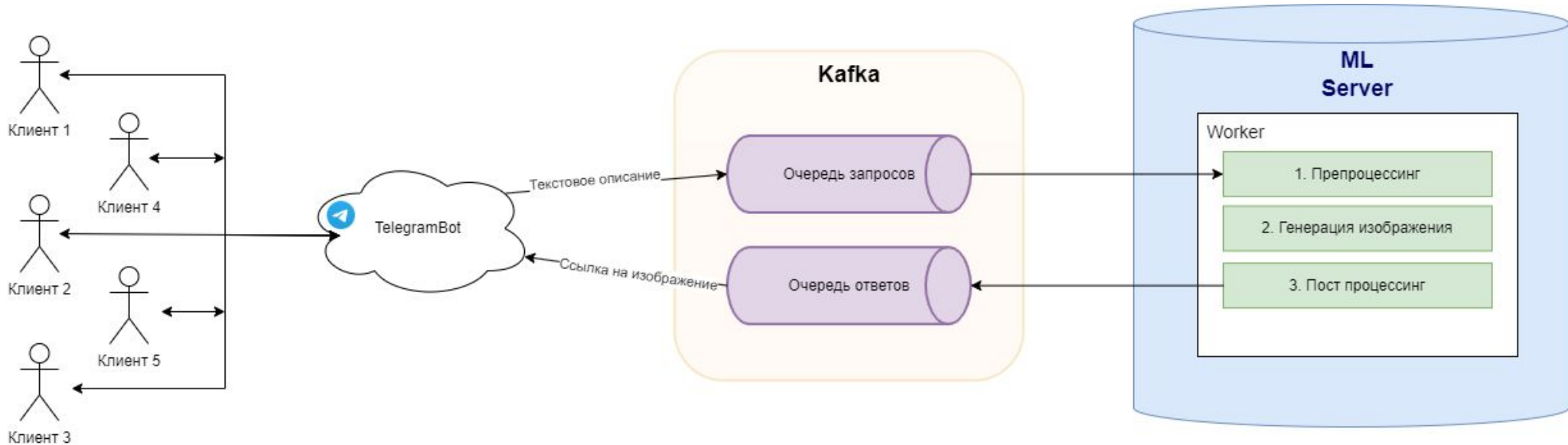


# Определение входных и выходных данных системы

Как показано на схеме ниже, система генерации изображений принимает на вход текстовое описание от пользователя и возвращает ссылку на готовое изображение:

**Вход:** текстовое описание, переданное через Telegram-бота.

**Выход:** ссылка на сгенерированное изображение, доступное пользователю.



# Выбор категории МО

Для генерации изображений используется диффузионная модель, а именно **Würstchen**, оптимизированная для работы на ограниченных ресурсах.

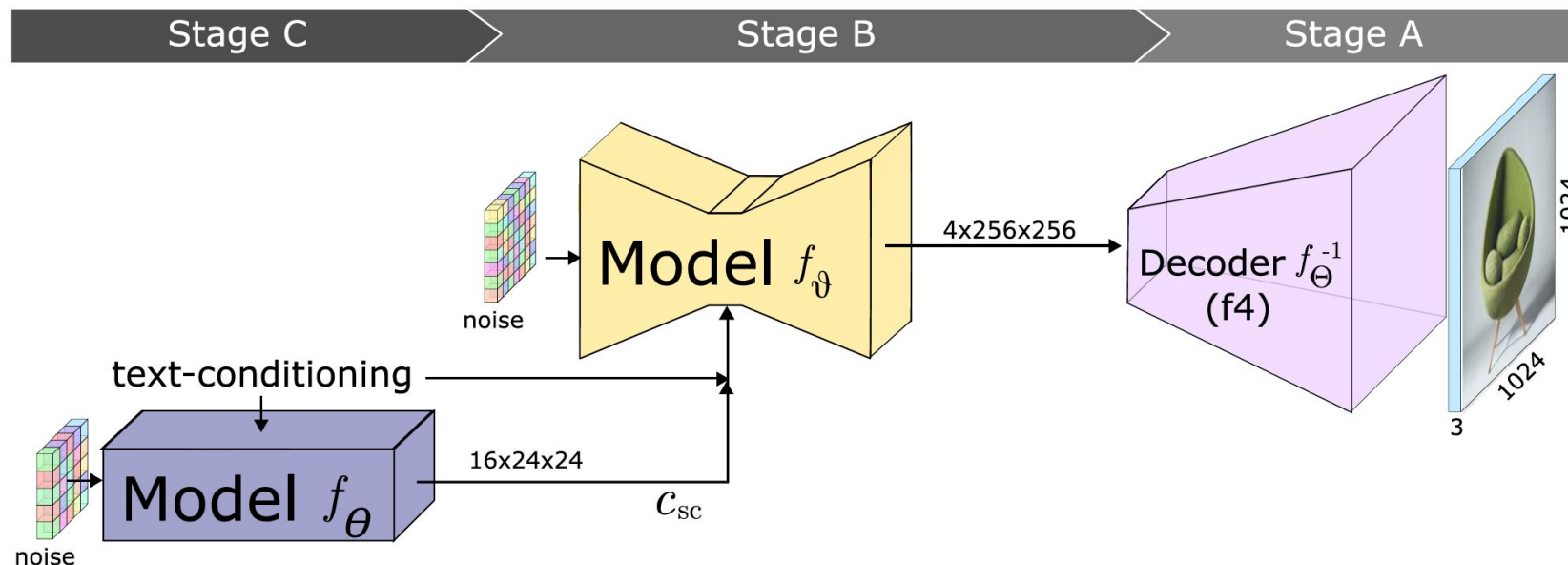
# Как работает модель?

## Подача текстового описания:

Пользователь вводит текстовое описание изображения.

## Текстовая обработка:

Текстовое описание преобразуется в эмбединги с использованием предварительно обученной модели Stage C (Prior), который работает в сверхкомпрессированном латентном пространстве.



# Как работает модель?

## Генерация латентных представлений:

Stage C создает сжатое латентное представление изображения на основе текстовых эмбеддингов. Этот процесс происходит в пространстве с 42-кратным сжатием.

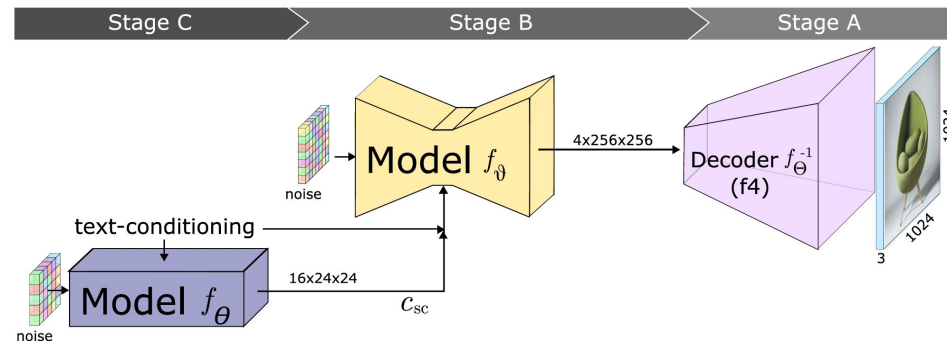
## Декодирование в изображение:

Stage B (Diffusion Autoencoder) декодирует латенты из Stage C в промежуточное представление в пространстве VQGAN.

Stage A (VQGAN) далее декодирует это представление в изображение пиксельного формата.

## Выдача результата:

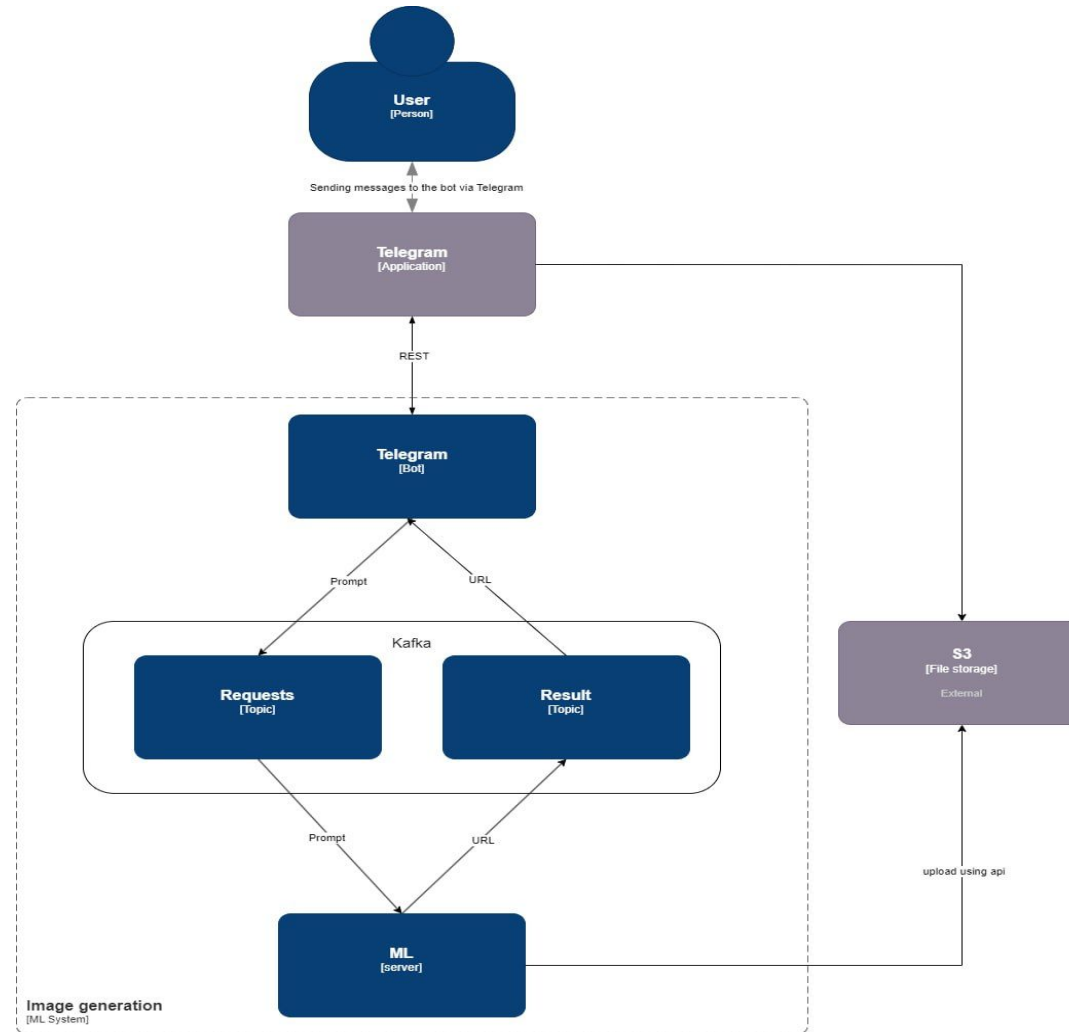
Сгенерированное изображение сохраняется в хранилище и возвращается пользователю в подходящем формате.



# Эксплуатация

После обучения модели и подготовки системы начинается этап эксплуатации, где система обрабатывает запросы пользователей, предоставляя сгенерированные изображения в реальном времени. Рассмотрим ключевые аспекты эксплуатации, включая архитектуру системы и мониторинг её производительности.

# Общий дизайн системы генерации изображений



# Генеративный пайплайн


Генеративный пайплайн выполняет обработку текстовых запросов пользователей и генерирует изображения. Он включает два ключевых компонента.

# Сервис обработки запросов (bot)

Этот компонент принимает текстовые описания, вводимые пользователями через Telegram-бота, передаёт их в сервис с моделью генерации через очередь сообщений, и читает очередь сообщений с ссылками на изображения, которые сгенерировала модель, после чего выдаёт ответ (картинку) пользователю.

## Как работает:

- Принимает текстовый запрос через API бота.
- Передаёт текст в топик Requests (Apache Kafka) для последующей обработки моделью.
- Читает из топика Result ссылку на сгенерированное изображение и по готовности выдаёт его пользователю.

 **Пример:** Пользователь вводит запрос: "Закат над океаном". Сервис отправляет текст в очередь, затем читает из очереди ссылку на изображение и выдаёт картинку пользователю.




# Сервис с моделью генерации изображений (server)

Этот компонент читает запросы из очереди сообщений, с помощью генеративного пайплайна **Würstchen** генерирует изображение и загружает его на файлообменник, ссылку на файл кладёт в очередь сообщений.

## Как работает:

- Читает из топика Requests промпт.
- Модель создаёт изображение на основе промпта, учитывая параметры (разрешение, негативный промпт).
- Изображение загружается на файлообменник.
- Ссылка на изображение отправляется в топик Result.

 **Пример:** Из очереди поступил промпт "Горный пейзаж на рассвете". На его основе создаётся реалистичное изображение. Изображение сохраняется в файлообменнике. Ссылка на изображение в файлообменнике поступает в очередь.

# Мониторинг и поддержка системы



Для успешной эксплуатации системы важно обеспечить её стабильность и эффективность. Это достигается через мониторинг ключевых метрик.

## Мониторинг производительности

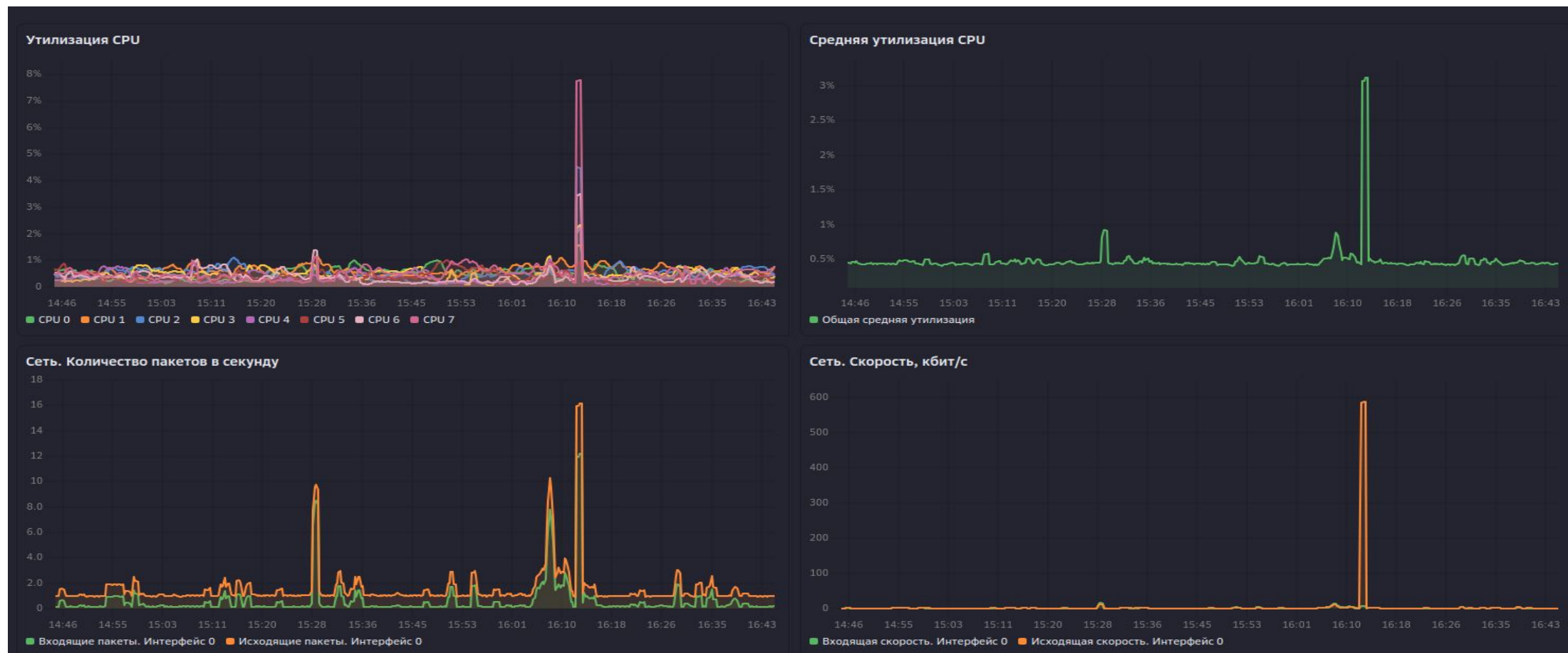
### Технические метрики:

- Утилизация CPU (CPU utilization).
- Сеть. Количество пакетов в секунду (packets per second).

### Пользовательские метрики:

- Время генерации изображений.

# Пример технических мониторингов



# Проблемы эксплуатации и их решения

## Высокая нагрузка

При большом количестве запросов может возникать задержка в обработке.

### Решения:

- Горизонтальное масштабирование системы: увеличение количества подов.
- Оптимизация кода модели и инфраструктуры.
- Вертикальное масштабирование: например, использование GPU вместо CPU.

# Почему такая архитектура?

- Использование Apache Kafka позволяет распределять нагрузку и обеспечивать надёжную передачу данных.
- Контейнеризация с Docker упрощает развертывание и масштабирование.
- Telegram обеспечивает простой доступ для пользователей через привычный интерфейс.

# Почему мы выбрали Telegram?

- Широкая доступность: Telegram используется на всех типах устройств и имеет простую интеграцию через Telegram API.
- Скорость разработки: создание бота не требует больших затрат времени, благодаря удобной библиотеке Python (например, aiogram).
- Удобство для пользователей: интерфейс Telegram интуитивно понятен, что снижает порог входа.

# Почему Apache Kafka?

- Производительность: Kafka может обрабатывать миллионы сообщений в секунду, что важно для масштабируемых приложений.
- Распределённость: позволяет распределять нагрузку между несколькими серверами, обеспечивая отказоустойчивость.
- Гибкость: поддерживает как асинхронное, так и синхронное взаимодействие между компонентами.
- Надёжность: все сообщения сохраняются в логах, что предотвращает потерю данных при сбоях.


# Пример использования

- Пользователь вводит текстовое описание через команду /txt2img.
- Telegram-Bot отправляет запрос в Kafka.
- ML-сервер обрабатывает запрос, генерирует изображение через Stable Diffusion и сохраняет его во внешнем хранилище.
- Ссылка на изображение возвращается пользователю через Telegram.




11 января


ФМ /start 17:26 ✓

 Привет Фёдор Меркулов!  
Бот активирован 17:26


ФМ /help 17:26 ✓

 Привет Фёдор Меркулов!  
Этот бот создаст для тебя картинку по твоему описанию.  
Для этого воспользуйся командой `/txt2img`.  
Помни, что делать свой запрос нужно на английском языке 17:26

ФМ /txt2img 17:27 ✓

 Введите описание изображения 17:27

ФМ beautiful art, high resolution, many details 17:27 ✓

 Ваш запрос обрабатывается 17:27

Меню



Написать сообщение...



# Доступ к боту

- Вы можете найти бота в Telegram по имени:  
`@MAINeuroML_bot`