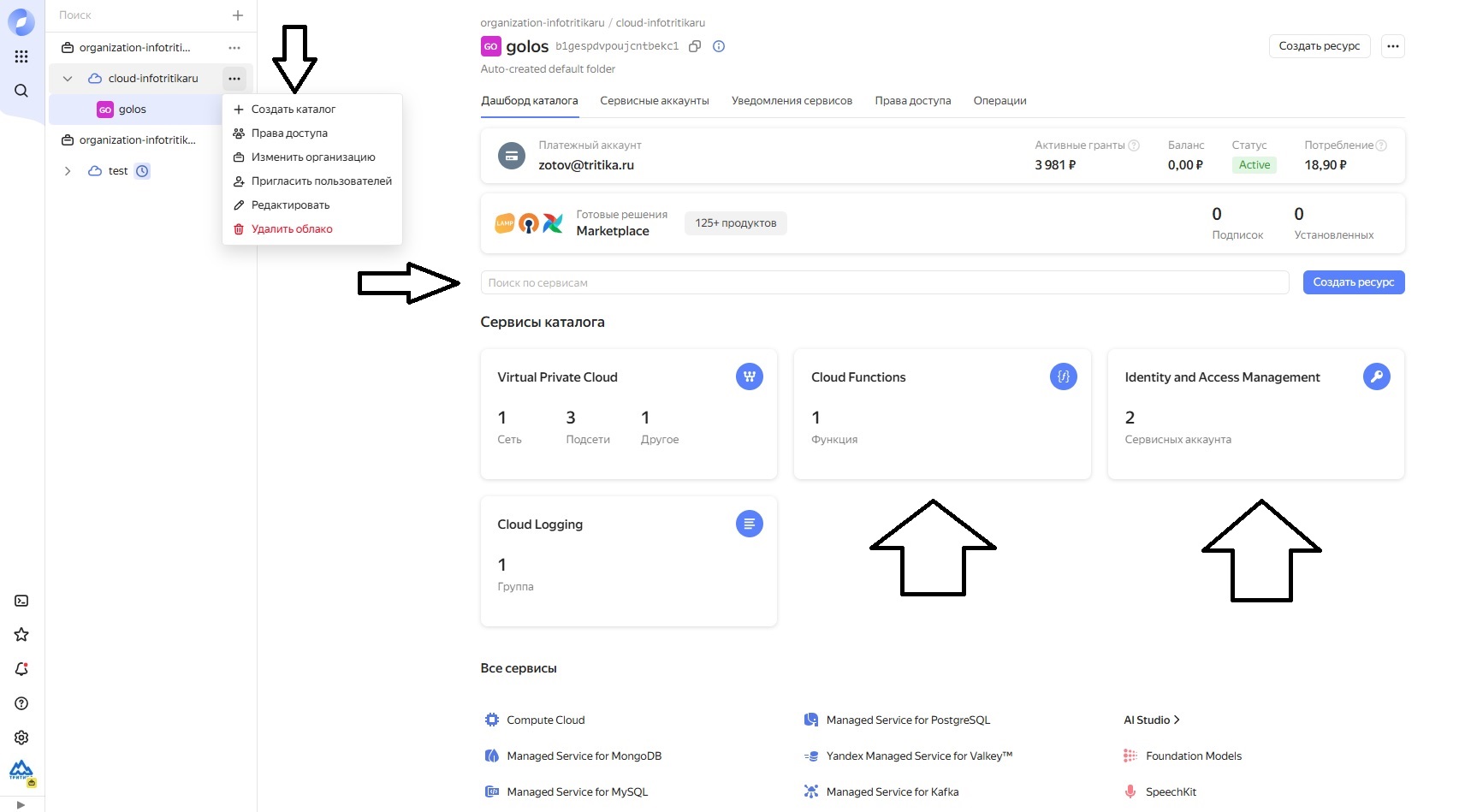
**Полное Руководство: Создание Голосового Telegram-Бота с Yandex Cloud**

**Шаг 1: Создание Telegram-бота и получение API-токена**

1. **Найдите BotFather**: Откройте приложение Telegram на вашем устройстве. В строке поиска введите @BotFather и начните диалог с этим официальным ботом.
2. **Создайте нового бота**: Отправьте команду /start, чтобы увидеть меню. Затем отправьте команду /newbot.
3. **Задайте имя бота**: BotFather попросит вас ввести имя для вашего бота. Это **имя**, которое будут видеть пользователи (например, "Голосовой Помощник Алексея").
4. **Задайте username бота**: Затем вам нужно придумать уникальный username для бота. Он должен заканчиваться на bot (например, MySuperVoiceHelperBot).
5. **Получите API-токен**: Если username свободен, BotFather поздравит вас с созданием бота и выдаст **секретный API-токен**. Он выглядит примерно так: 123456789:AAFdC1bB2cC3dD4eE5fF6gG7hH8iI9jJkK.
6. **Сохраните токен: Скопируйте и сохраните этот токен в надежном месте**. Он понадобится вам на шаге 4. Этот токен — это пароль для управления вашим ботом через Telegram Bot API. В коде нашей функции мы будем использовать его как переменную TELEGRAM\_TOKEN.

**Шаг 2: Регистрация и настройка Yandex Cloud**



1. **Регистрация аккаунта:**

* Перейдите на сайт https://cloud.yandex.ru/ и нажмите кнопку «**Войти**» или «**Начать работу**».
* Вы можете войти с помощью своего аккаунта Яндекса (почта на @yandex.ru) или создать новый.
* **Бесплатный период**: Yandex Cloud предоставляет новый пользователям **бесплатный грант** (обычно 3000-5000 рублей) на первые 60 дней. Привязка банковской карты **не обязательна** для начала работы и использования гранта. Вы можете добавить карту позже, если исчерпаете грант и захотите продолжить пользоваться платными сервисами.
* Выберите тариф «**Бесплатный**» или «**Пробный**» при регистрации, если такая опция будет предложена.

1. **Создание Облака (Cloud):**

* После входа в консоль Yandex Cloud вы попадете на главную страницу.
* Нажмите кнопку «**Создать облако**».
* Введите **имя** для вашего облака (например, MyFirstCloud) и нажмите «**Создать облако**».
* Облако — это контейнер для всех ваших ресурсов (сервисов, виртуальных машин, функций и т.д.).

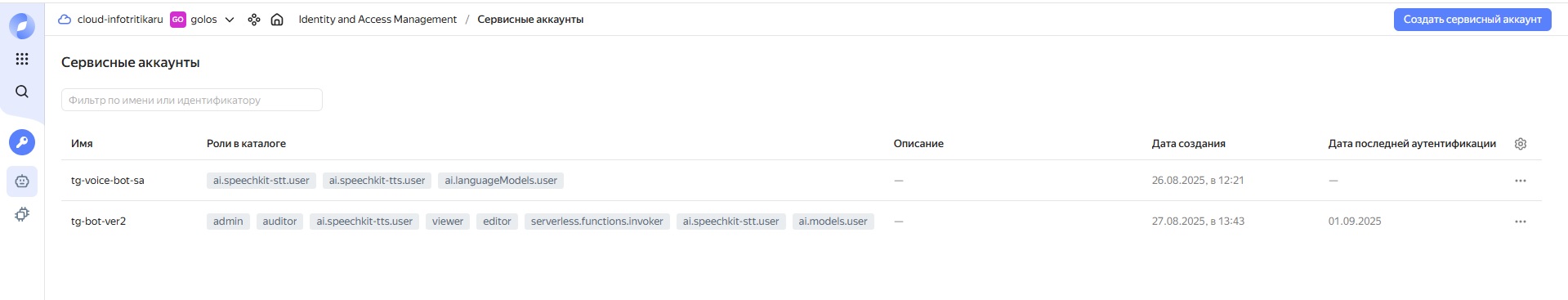
1. **Создание Каталога (Folder):**

* После создания облака вы автоматически попадете в него.
* Нажмите кнопку «**Создать каталог**».
* Введите **имя** для каталога (например, VoiceBotProject) и нажмите «**Создать каталог**».
* Каталог — это еще один уровень организации внутри облака. Все сервисы, которые мы будем создавать (функции, сервисные аккаунты), будут находиться в этом каталоге. Запомните или сохраните **ID каталога** — он понадобится позже. Найти его можно в настройках каталога или в URL-адресе консоли, когда вы находитесь внутри каталога (это длинная строка цифр и букв).

1. **Настройка доступа (IAM — Identity and Access Management):**

* В левом верхнем углу консоли Yandex Cloud есть строка поиска. Введите в нее IAM.
* Выберите сервис «**IAM**» (Identity and Access Management). Это система управления доступом, которая определяет, кто и что может делать в вашем облаке и каталоге.
* Перейдите на вкладку «**Роли**» или «**Права доступа**» для вашего каталога.
* Нажмите кнопку «**Назначить роли**».
* В поле «**Выберите принципал**» (или «Выберите пользователя/сервисный аккаунт») мы пока ничего не выбираем — мы создадим сервисный аккаунт на следующем шаге. Сейчас важно понять, где находится этот раздел.

**Шаг 3: Создание сервисного аккаунта и назначение ролей**



Сервисный аккаунт — это виртуальный пользователь, от имени которого будет работать наша Cloud Function, чтобы иметь доступ к другим сервисам Yandex Cloud, таким как SpeechKit.

1. **Создание сервисного аккаунта:**

* В консоли Yandex Cloud перейдите в раздел «Сервисные аккаунты». Его можно найти через строку поиска или в меню слева.
* Нажмите кнопку «**Создать сервисный аккаунт**».
* Введите **имя** (например, voice-bot-sa).
* Нажмите «**Создать сервисный аккаунт**».

1. **Назначение ролей:**

* После создания аккаунта вы попадете на его страницу. Нажмите кнопку «Назначить роли».
* В поле «**Роль**» начните вводить название роли. Нам нужны следующие роли для работы с SpeechKit:
* ai.speechkit-stt.user — для распознавания речи (Speech-to-Text).
* ai.speechkit-tts.user — для синтеза речи (Text-to-Speech), если вы планируете, чтобы бот отвечал голосом.
* Выберите нужные роли из выпадающего списка и нажмите «**Сохранить**».
* Эти роли дают нашему сервисному аккаунту право использовать соответствующие API SpeechKit.

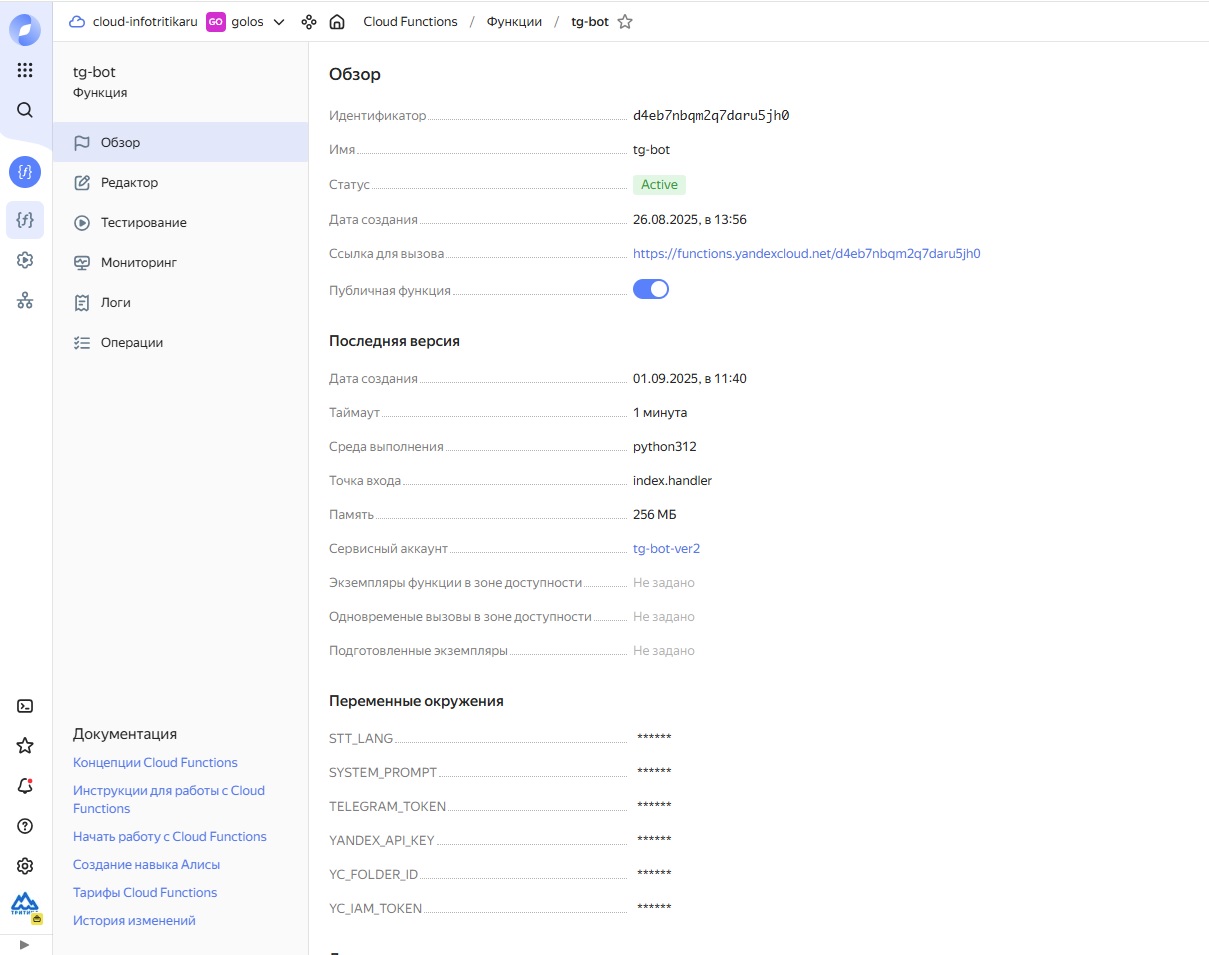
1. **Создание API-ключа для аутентификации:**

* На странице созданного сервисного аккаунта перейдите на вкладку «**API-ключи**».
* Нажмите кнопку «**Создать API-ключ**».
* Скопируйте и **сохраните сгенерированный ключ**. Он понадобится нам в коде функции для аутентификации запросов к SpeechKit.
* Этот ключ — это "пароль" для нашего сервисного аккаунта. Без него, даже с правильными ролями, функция не сможет авторизоваться в SpeechKit.

1. **Привязка сервисного аккаунта к Cloud Function (важно!):**

* Этот шаг будет выполнен на следующем этапе при создании функции. При настройке Cloud Function вы сможете выбрать, от чьего имени она будет запускаться. **Обязательно выберите только что созданного сервисного аккаунта (voice-bot-sa)**. Это даст функции все права, которые есть у аккаунта.

**Шаг 4: Создание и настройка Cloud Function**



Cloud Function — это наш серверный код, который будет обрабатывать запросы от Telegram-бота.

1. **Создание функции:**

* В консоли Yandex Cloud найдите сервис «**Cloud Functions**» через строку поиска.
* Убедитесь, что вы находитесь в нужном каталоге (например, VoiceBotProject).
* Нажмите кнопку «**Создать функцию**».

1. **Настройка параметров функции:**

* **Имя**: Введите имя функции (например, telegram\_voice\_bot).
* **Среда выполнения**: Выберите Python 3.9.
* **Точка входа**: Введите handler. Это имя функции в вашем коде, которую Yandex Cloud будет вызывать при каждом запросе.
* **Учетная запись службы: Очень важно**! Выберите из выпадающего списка созданный вами сервисный аккаунт (например, voice-bot-sa). Это обеспечит доступ функции к SpeechKit.
* **Ресурсы**: Для начала можно оставить значения по умолчанию (например, 128 МБ памяти, таймаут 5 секунд).
* **Триггер**: Выберите тип триггера «HTTP-вызов» и установите галочку «Вызов функции по HTTP». Это создаст публичный URL для вашей функции.
* **Секреты (Переменные окружения):** Это очень удобный способ хранить секретные данные, такие как токены. Нажмите «**Добавить секрет**» или «**Добавить переменную**» и создайте следующие переменные:

TELEGRAM\_TOKEN: Вставьте сюда API-токен вашего Telegram-бота, полученный от BotFather.

FOLDER\_ID: Вставьте сюда ID вашего каталога в Yandex Cloud.

API\_KEY: Вставьте сюда API-ключ, созданный для сервисного аккаунта на предыдущем шаге.

* Нажмите кнопку «**Создать функцию**».

1. **Установка вебхука для Telegram-бота:**

* После создания функции вы увидите ее **публичный HTTPS URL**. Скопируйте его.
* Чтобы Telegram знал, куда отправлять обновления (новые сообщения, голосовые и т.д.), нужно установить **вебхук**.
* Откройте браузер и перейдите по следующей ссылке, заменив <YOUR\_TELEGRAM\_TOKEN> на токен вашего бота, а <YOUR\_FUNCTION\_URL> на URL вашей функции:

https://api.telegram.org/bot<YOUR\_TELEGRAM\_TOKEN>/setWebhook?url=<YOUR\_FUNCTION\_URL>

Пример:

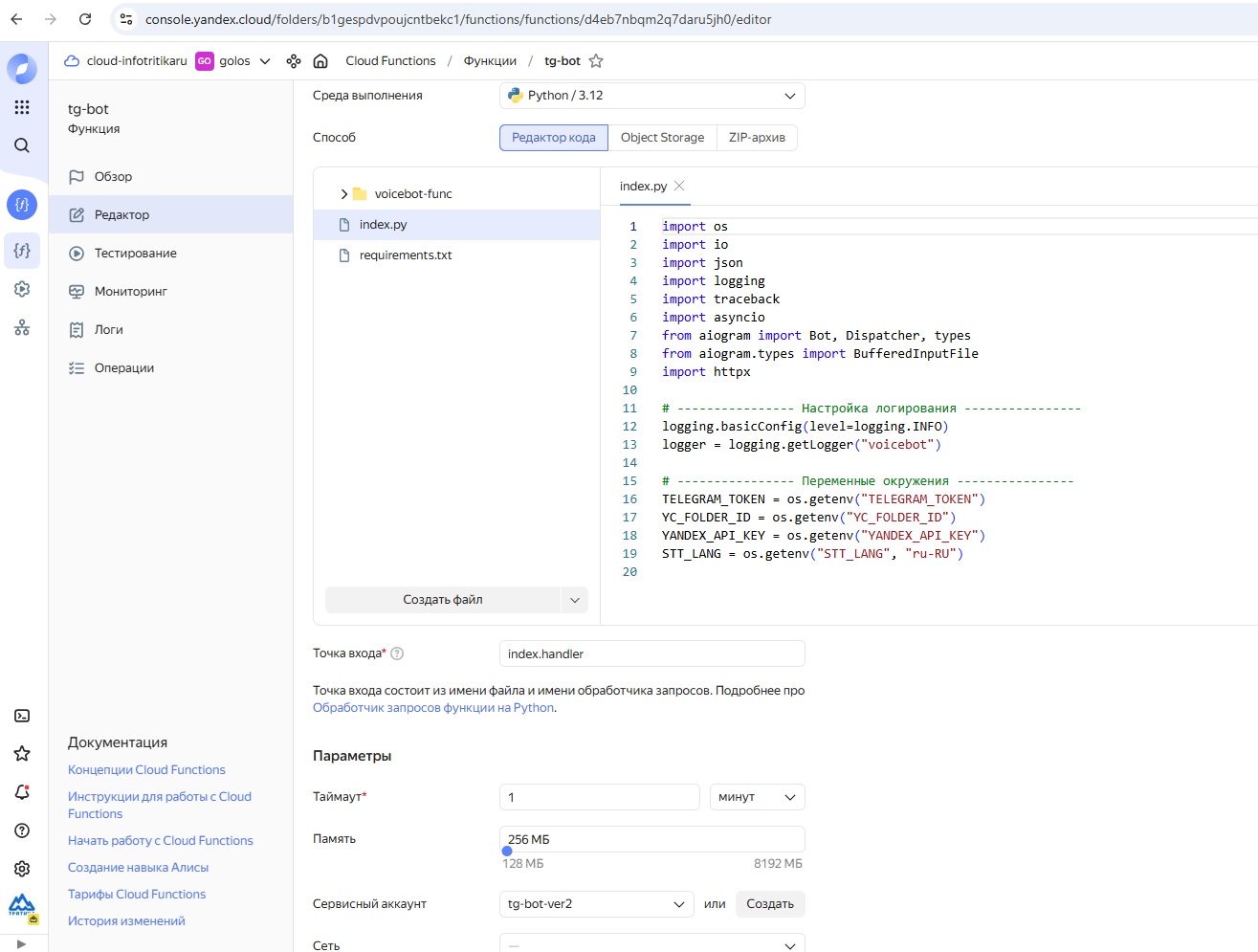
<https://api.telegram.org/bot123456789:AAFdC1bB2cC3dD4eE5fF6gG7hH8iI9jJkK/setWebhook?url=https://functions.yandexcloud.net/d5d5d5d5d5d5d5d5>

* Если все сделано правильно, браузер покажет ответ в формате JSON: {"ok":true,"result":true,"description":"Webhook was set"}. Это означает, что Telegram теперь будет отправлять все обновления для вашего бота на URL вашей Cloud Function.

1. **Тестирование функции:**

* В консоли Yandex Cloud на странице вашей функции перейдите на вкладку «**Тестирование**».
* Вы можете отправить тестовый запрос. Для начала можно оставить тело запроса пустым или ввести {}.
* Нажмите «**Запустить тест**».
* Перейдите на вкладку «**Логи**», чтобы увидеть результат выполнения функции и возможные ошибки. Это основной инструмент для отладки.

**Шаг 5: Написание и развертывание кода бота**



Теперь напишем код, который будет внутри Cloud Function. Этот код будет получать запросы от Telegram, скачивать голосовые сообщения, отправлять их в SpeechKit для распознавания и отправлять результат обратно пользователю.

1. **Подготовка кода:**

* Перейдите на вкладку «**Код**» вашей Cloud Function в консоли Yandex Cloud.
* Убедитесь, что выбран Python 3.9.
* В поле «**Точка входа**» должно быть указано handler.
* В разделе «**Зависимости**» вставьте следующий код, который описывает, какие библиотеки нужно установить:

requests==2.31.0

1. **Основной код функции:**

* Вставьте следующий код в редактор кода. Этот код является законченным примером голосового помощника.

import json

import os

import requests

# Получаем переменные окружения

TELEGRAM\_TOKEN = os.environ.get('TELEGRAM\_TOKEN')

FOLDER\_ID = os.environ.get('FOLDER\_ID')

API\_KEY = os.environ.get('API\_KEY')

# Базовые URL для API

TELEGRAM\_API\_URL = f'https://api.telegram.org/bot{TELEGRAM\_TOKEN}'

SPEECHKIT\_STT\_URL = 'https://stt.api.cloud.yandex.net/speech/v1/stt:recognize'

def handler(event, context):

"""

Точка входа для Cloud Function.

Обрабатывает вебхук от Telegram.

"""

try:

# Парсим JSON из тела запроса

update = json.loads(event['body'])

print(f"Получено обновление: {update}")

# Проверяем, есть ли в обновлении сообщение и голосовое сообщение

if 'message' not in update or 'voice' not in update['message']:

print("Нет голосового сообщения.")

return {'statusCode': 200, 'body': 'OK'}

message = update['message']

chat\_id = message['chat']['id']

voice\_file\_id = message['voice']['file\_id']

# Шаг 1: Получаем информацию о файле от Telegram

file\_info\_url = f"{TELEGRAM\_API\_URL}/getFile?file\_id={voice\_file\_id}"

file\_info\_response = requests.get(file\_info\_url)

file\_info = file\_info\_response.json()

if not file\_info['ok']:

raise Exception(f"Ошибка получения информации о файле: {file\_info}")

file\_path = file\_info['result']['file\_path']

file\_url = f"https://api.telegram.org/file/bot{TELEGRAM\_TOKEN}/{file\_path}"

# Шаг 2: Скачиваем аудиофайл

audio\_response = requests.get(file\_url)

if audio\_response.status\_code != 200:

raise Exception("Ошибка скачивания аудиофайла")

audio\_content = audio\_response.content

# Шаг 3: Отправляем аудио в Yandex SpeechKit для распознавания

headers = {

'Authorization': f'Api-Key {API\_KEY}',

}

params = {

'folderId': FOLDER\_ID,

'lang': 'ru-RU' # Язык распознавания

}

stt\_response = requests.post(

SPEECHKIT\_STT\_URL,

headers=headers,

params=params,

data=audio\_content

)

if stt\_response.status\_code != 200:

error\_text = f"Ошибка SpeechKit: {stt\_response.text}"

print(error\_text)

send\_telegram\_message(chat\_id, "Произошла ошибка при распознавании речи.")

return {'statusCode': 500, 'body': error\_text}

recognized\_text = stt\_response.json().get('result', 'Текст не распознан')

# Шаг 4: Отправляем распознанный текст обратно в Telegram

send\_telegram\_message(chat\_id, f"Вы сказали: {recognized\_text}")

return {

'statusCode': 200,

'body': 'OK'

}

except Exception as e:

print(f"Ошибка: {str(e)}")

return {

'statusCode': 500,

'body': f'Ошибка: {str(e)}'

}

def send\_telegram\_message(chat\_id, text):

"""

Функция для отправки текстового сообщения в Telegram.

"""

url = f"{TELEGRAM\_API\_URL}/sendMessage"

payload = {

'chat\_id': chat\_id,

'text': text

}

response = requests.post(url, json=payload)

return response.json()

1. **Как работает код:**

* handler — это главная функция, которую вызывает Yandex Cloud при каждом HTTP-запросе (т.е. при каждом новом сообщении от Telegram).
* Код получает JSON-данные от Telegram, извлекает chat\_id (чтобы знать, кому отвечать) и file\_id голосового сообщения.
* Затем он запрашивает у Telegram прямую ссылку на аудиофайл и скачивает его.
* Скачанный аудиофайл (в формате OGG OPUS, который Telegram использует по умолчанию) отправляется напрямую в API Yandex SpeechKit для распознавания. SpeechKit умеет работать с этим форматом, поэтому конвертация не требуется.
* Полученный текст отправляется обратно пользователю в Telegram как обычное текстовое сообщение.

1. **Развертывание кода:**

* После вставки кода и зависимостей нажмите кнопку «**Сохранить версию**» или «**Создать версию**».
* Дождитесь, пока статус функции не станет «**Опубликована**».

**Шаг 6: Тестирование бота**

1. Откройте Telegram.
2. Найдите своего бота по его username (например, @MySuperVoiceHelperBot).
3. Нажмите «**Start**» или отправьте любое текстовое сообщение, чтобы активировать бота.
4. Нажмите на значок микрофона и запишите голосовое сообщение.
5. Отправьте его.
6. Через несколько секунд бот должен ответить вам текстом, содержащим распознанные слова из вашего голосового сообщения.

**Архитектура решения (Общая схема)**

Для полного понимания, как все компоненты работают вместе:

1. **Пользователь** отправляет **голосовое сообщение** боту в **Telegram**.
2. **Telegram** автоматически отправляет данные об этом сообщении (вебхук) на **публичный URL вашей Cloud Function в Yandex Cloud**.
3. **Cloud Function** запускается:

* Скачивает аудиофайл голосового сообщения из Telegram.
* Отправляет этот аудиофайл в сервис **Yandex SpeechKit (STT)** для распознавания речи, используя **API-ключ** сервисного аккаунта.
* Получает от SpeechKit распознанный текст.
* Отправляет этот текст обратно **пользователю в Telegram**, используя **API-токен** бота.

1. **Пользователь** получает **текстовый/голосовой ответ** от бота.

**Что дальше?**

- **Добавление синтеза речи (TTS):** Чтобы бот отвечал голосом, а не текстом, вы можете модифицировать код. Вместо send\_telegram\_message используйте API SpeechKit TTS для генерации аудио из текста, а затем метод Telegram Bot API sendVoice для отправки аудиосообщения.

- **Добавление "ИИ":** Для более "умных" ответов вы можете интегрировать дополнительные сервисы, например, Yandex GPT (для генерации ответов на основе распознанного текста) или любой другой API для обработки естественного языка (NLP). Для этого нужно будет добавить соответствующую роль (ai.foundation-models.user) сервисному аккаунту и модифицировать код для отправки запросов к новому API.

**Поздравляем! Вы создали своего первого голосового помощника в Telegram.**