

КАК СТАТЬ АВТОРОМ

Show must go on! 12 историй про работу в российском IT



853.76

Рейтинг

Timeweb Cloud

Облачная платформа для разработчиков и бизнеса

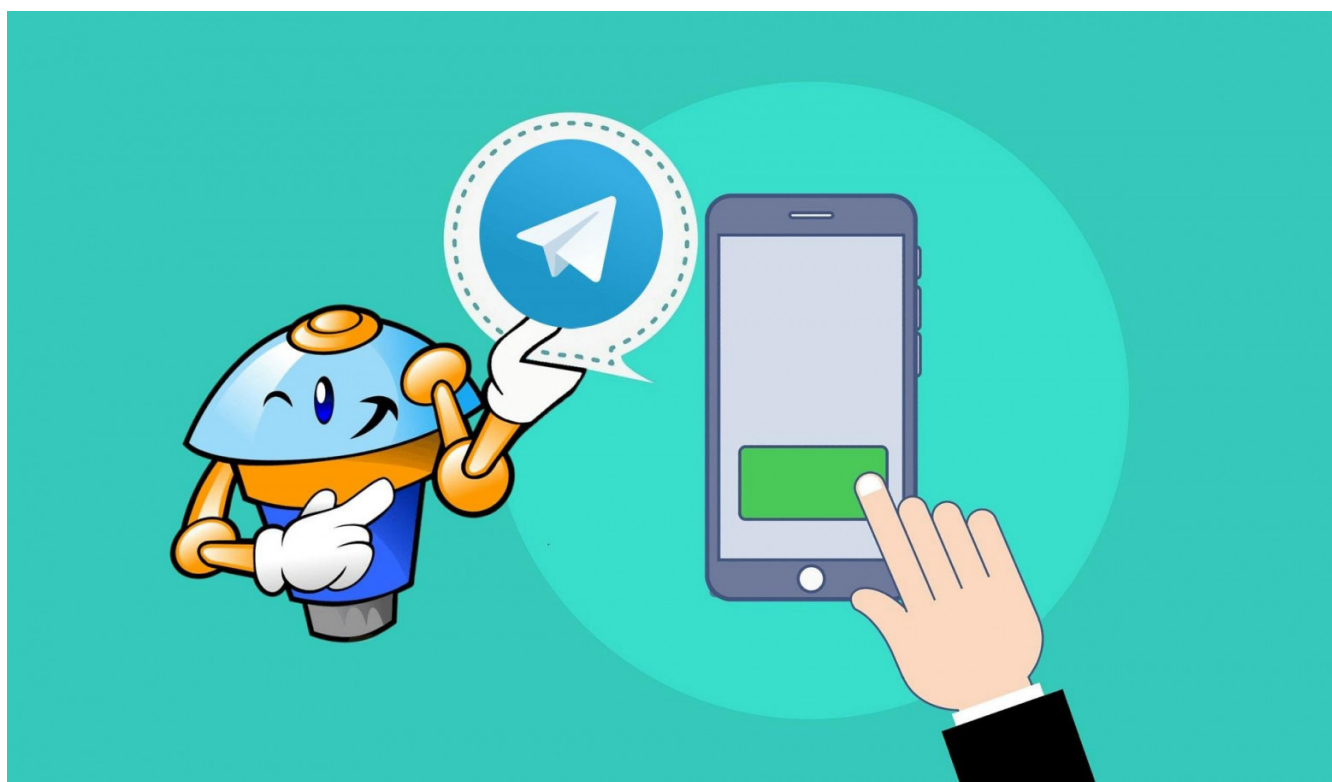


aio350 12 мая в 15:44

Node.js: разрабатываем бота для Telegram

Блог компании Timeweb Cloud, JavaScript*, Node.JS*

Tutorial



Привет, друзья!

В данном tutorialе мы разработаем простого бота для [Telegram](#). Сначала зарегистрируем и кастомизируем бота с помощью *BotFather*, затем напишем для него сервер на [Express](#), развернем сервер на [Heroku](#) и подключим бота к серверу с помощью веб-хука.

Функционал бота будет следующим:

- в ответ на сообщение *joke* возвращается программистская шутка, например: "Algorithm: a word used by programmers when they don't want to explain how their code works." (Алгоритм — это слово, используемое программистами, когда они не хотят объяснять, как работает их код));
- в ответ на сообщение, представляющее собой дату в формате *ДД.ММ*, возвращается либо

список дел, запланированных на эту дату в таблице Google (массив объектов), либо фраза "You have nothing to do on this day.", если на эту дату не запланировано никаких дел;

- в ответ на любое другое сообщение возвращается фраза "I have nothing to say."

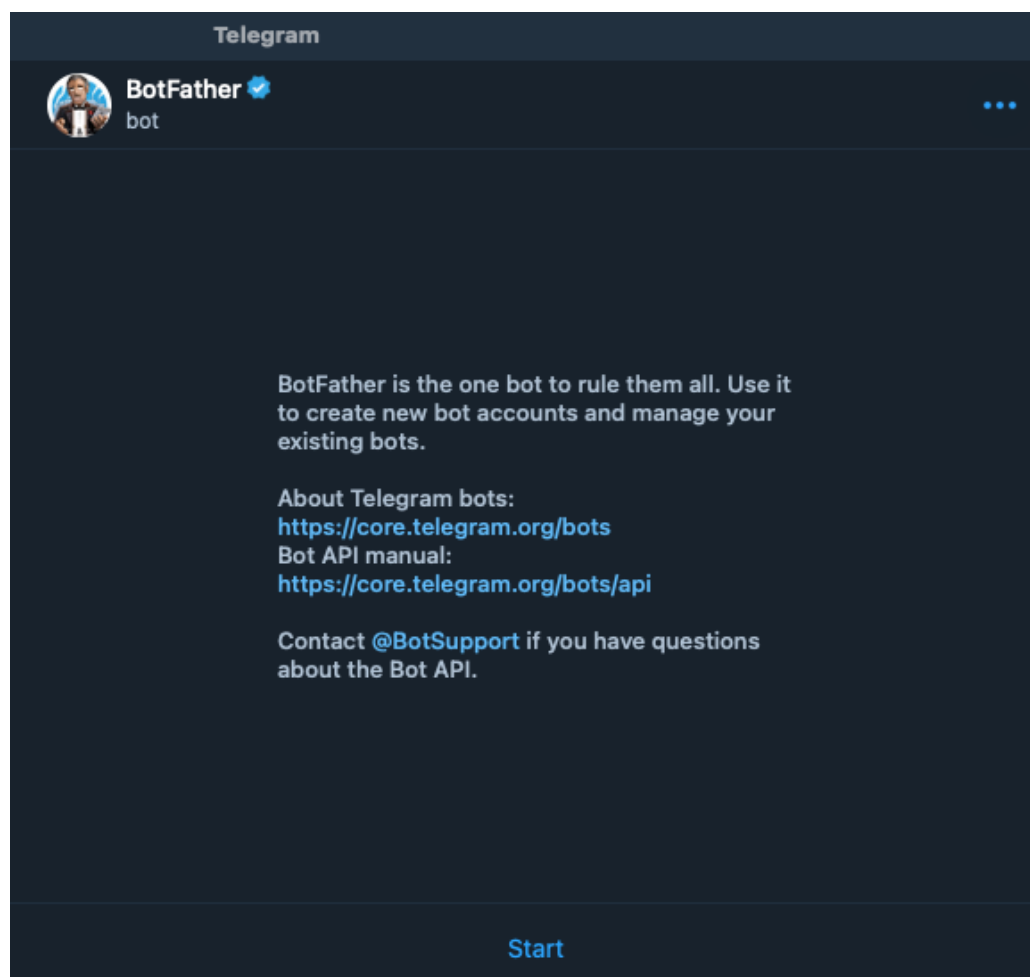
При разработке бота я буду опираться в основном на [официальную документацию](#).

- Репозиторий с кодом сервера для бота.
- Бот — @aio350_reminder_bot.

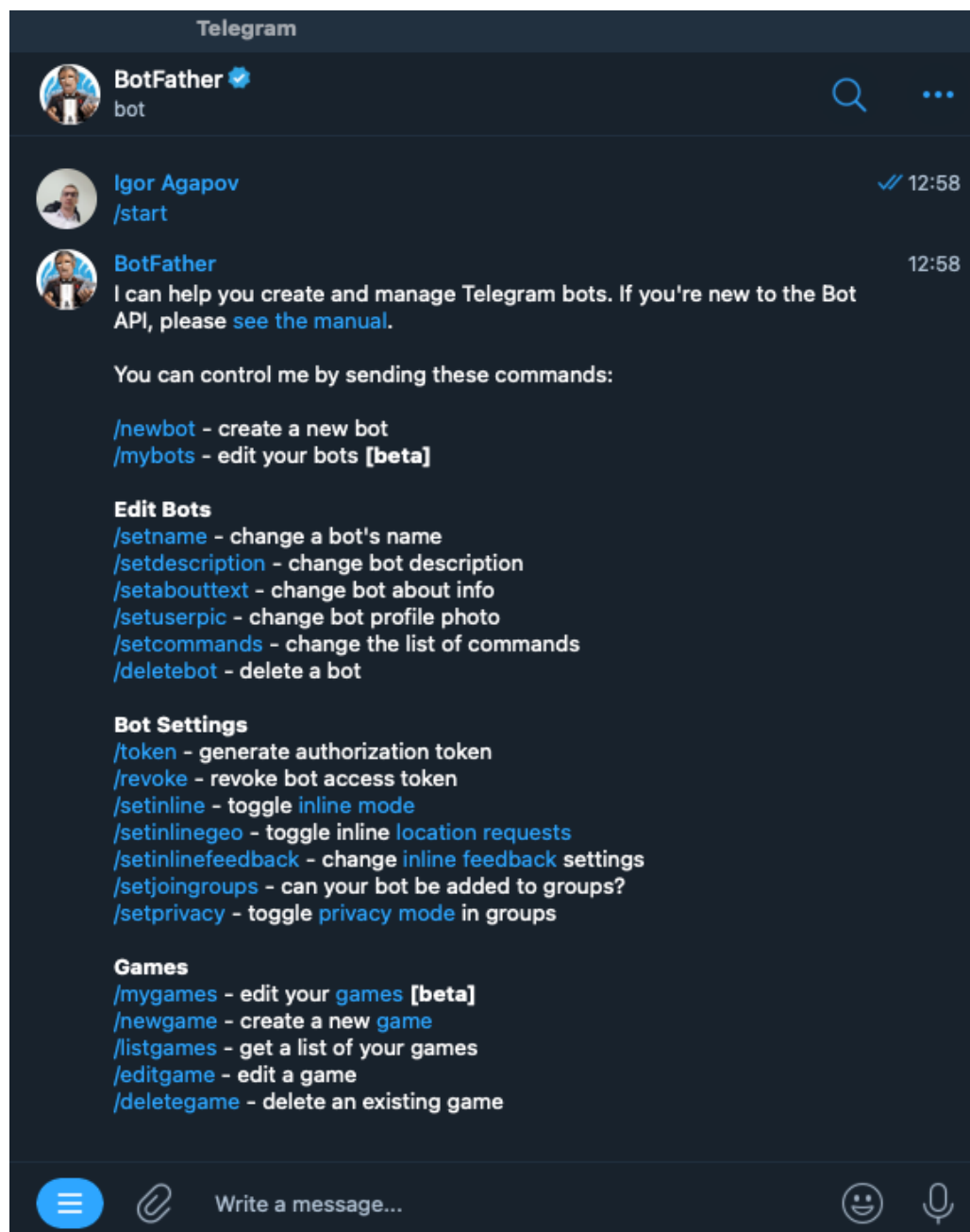
Если вам это интересно, прошу под кат.

Регистрация и кастомизация бота

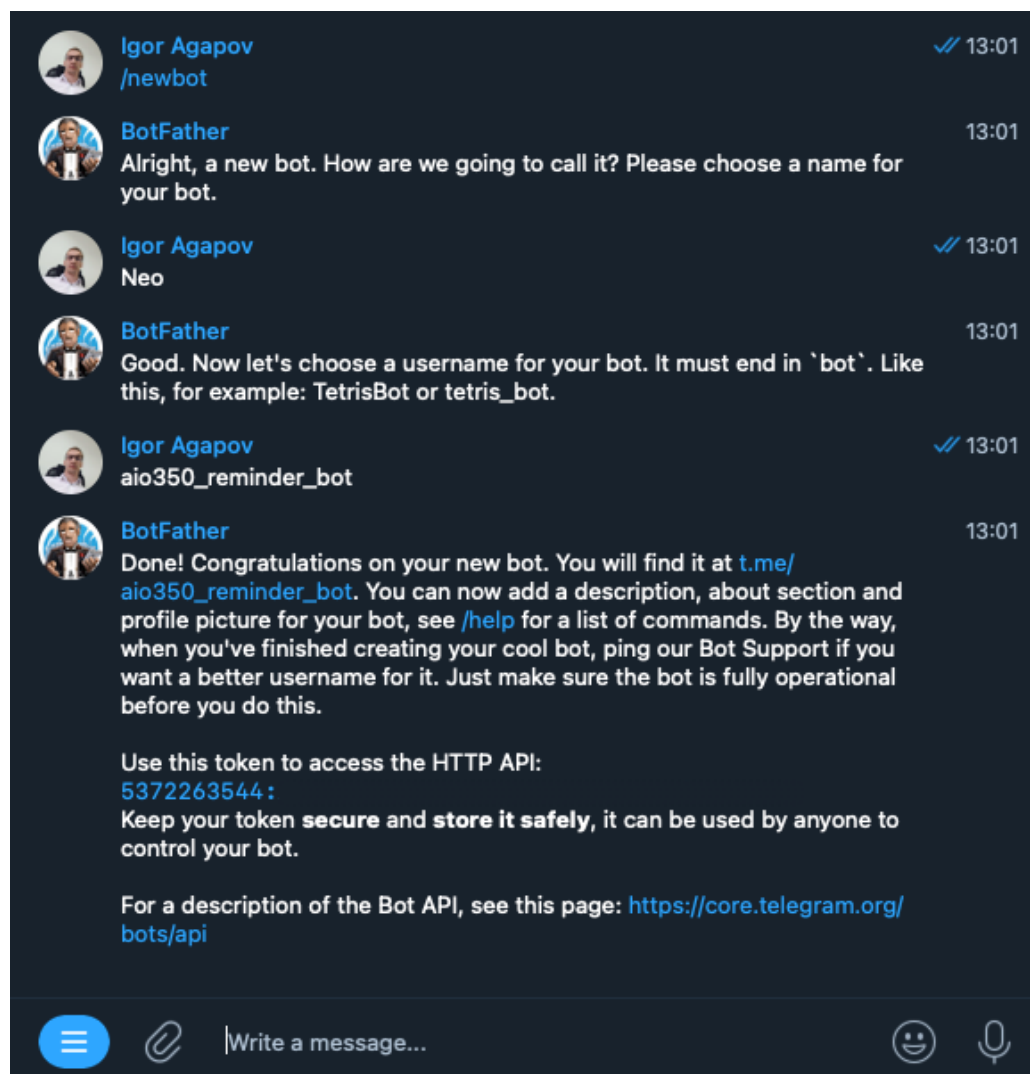
Для регистрации бота нужен только Telegram (я буду использовать десктопную версию). Находим в нем *BotFather* (@BotFather):



Нажимаем на *Start* и получаем список команд:

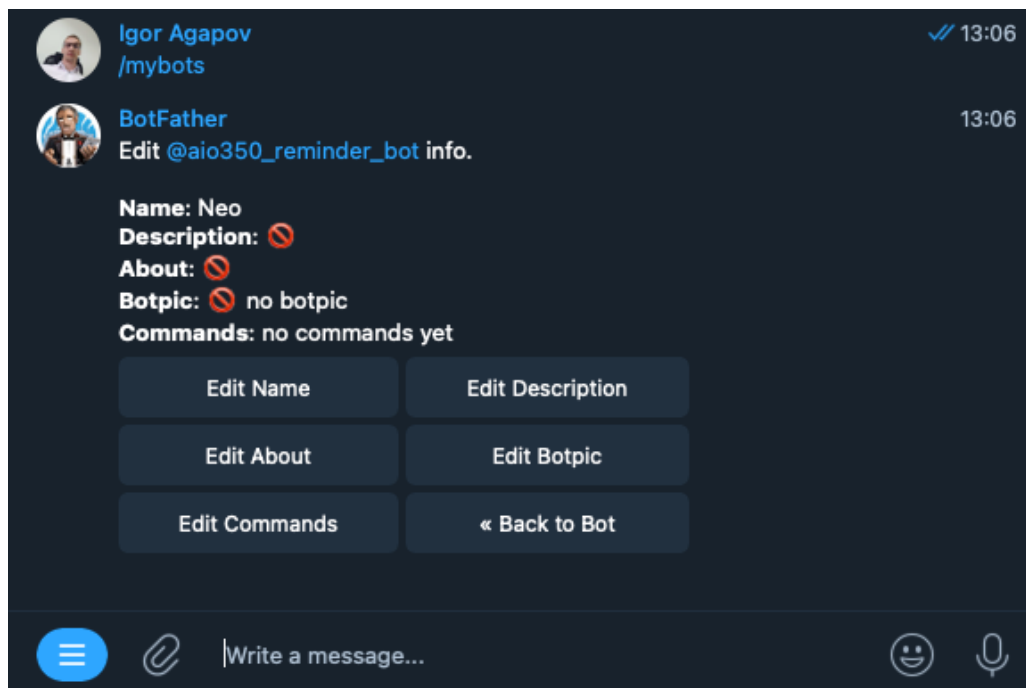


Выполняем команду `/newbot` для создания бота, указываем имя и `username` бота (`username` должно быть уникальным в пределах *Telegram*), например: "Neo" и "aio350_reminder_bot".

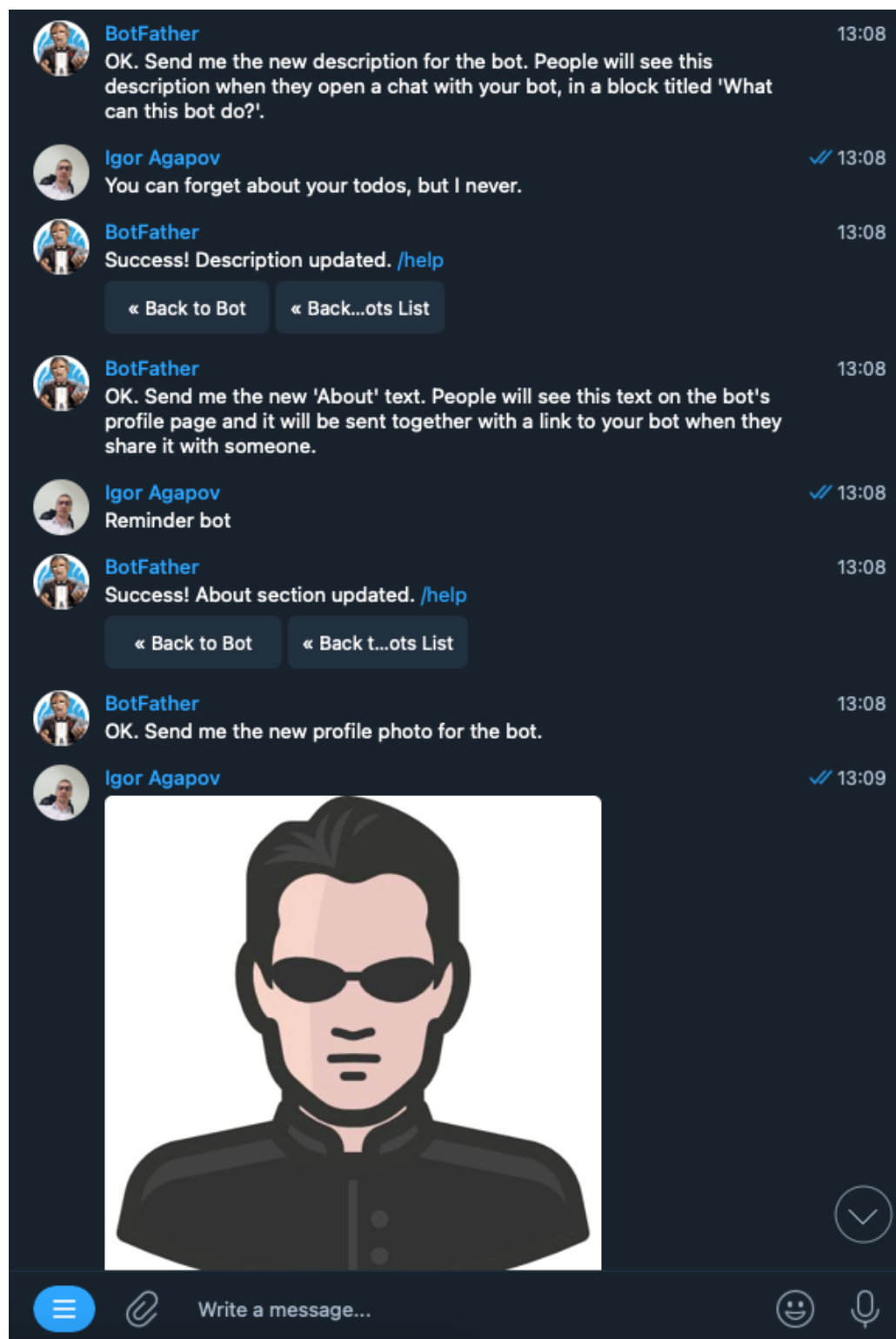


Получаем токен доступа: 5372263544:...

Выполняем команду `/mybots` для получения списка наших ботов, выбираем только что созданного бота и нажимаем *Edit Bot*:



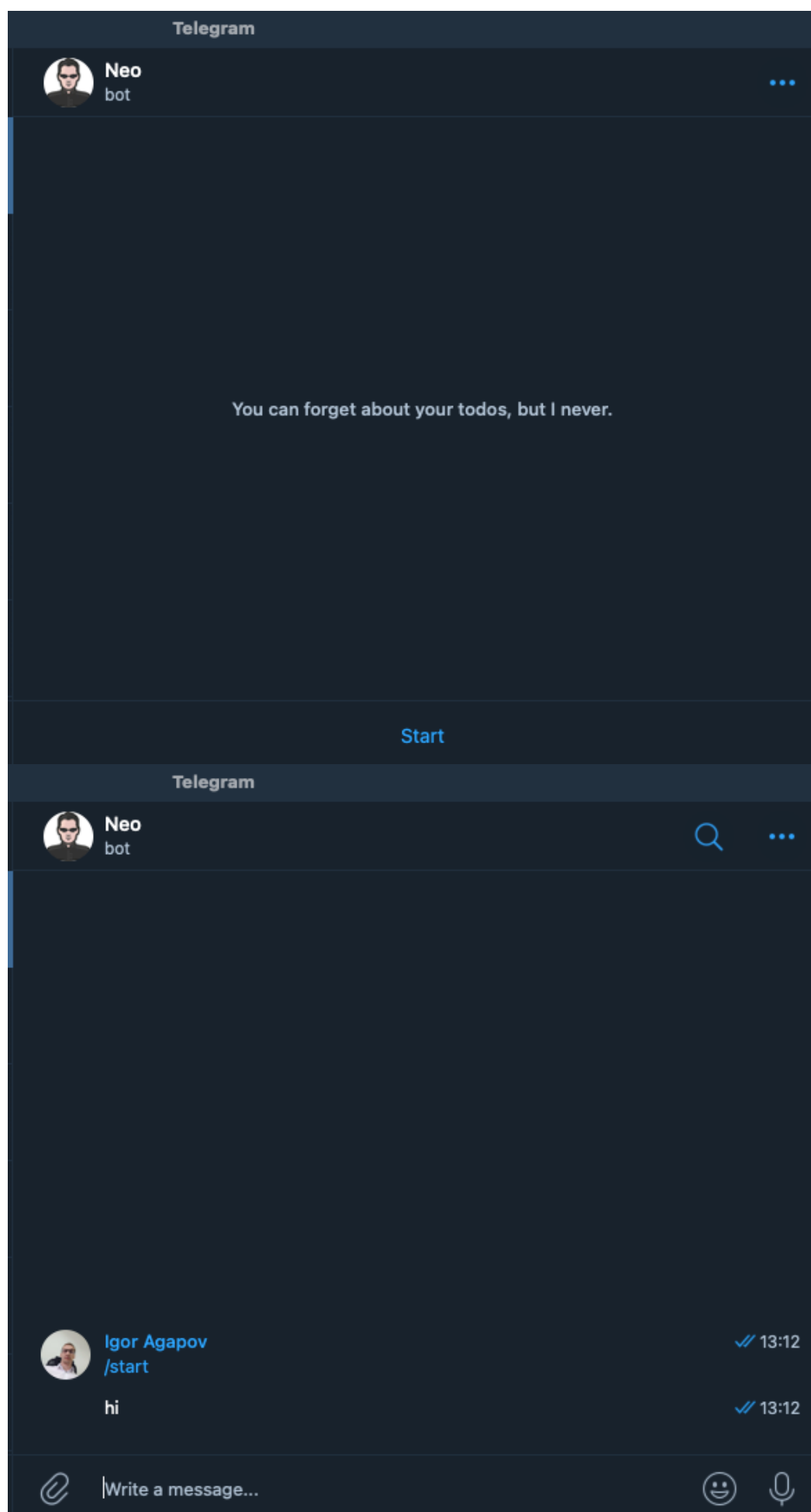
Добавляем боту описание (*Edit Description*), характеристику (*Edit About*) и аватар (*Edit Botpic*):



Аватар (спасибо FlatIcon):



Отлично, мы зарегистрировали бота в Telegram и кастомизировали его. Поздороваемся с ним:



Бот молчит, потому что у него пока нет "мозгов") Давайте это исправим.

Разработка и деплой сервера для бота

Создаем директорию, переходим в нее и инициализируем *Node.js-проект*:

```
mkdir telegram-bot-server
cd telegram-bot-server

yarn init -y
# or
npm init -y
```

Устанавливаем зависимости:

```
# производственные зависимости
yarn add axios dotenv express fs-extra google-spreadsheet

# зависимость для разработки
yarn add -D nodemon
```

- **axios**: клиент-серверная утилита для выполнения *HTTP-запросов*;
- **dotenv**: утилита для работы с переменными среды окружения;
- **express**: *Node.js-фреймворк* для разработки веб-серверов;
- **fs-extra**: расширенный *Node.js-модуль fs*;
- **google-spreadsheet**: пакет для работы с гугл-таблицами;
- **nodemon**: утилита для запуска сервера для разработки.

Определяем тип кода сервера (модуль) и команды для запуска сервера в производственном режиме (*start*) и режиме для разработки (*dev*) в файле *package.json*:

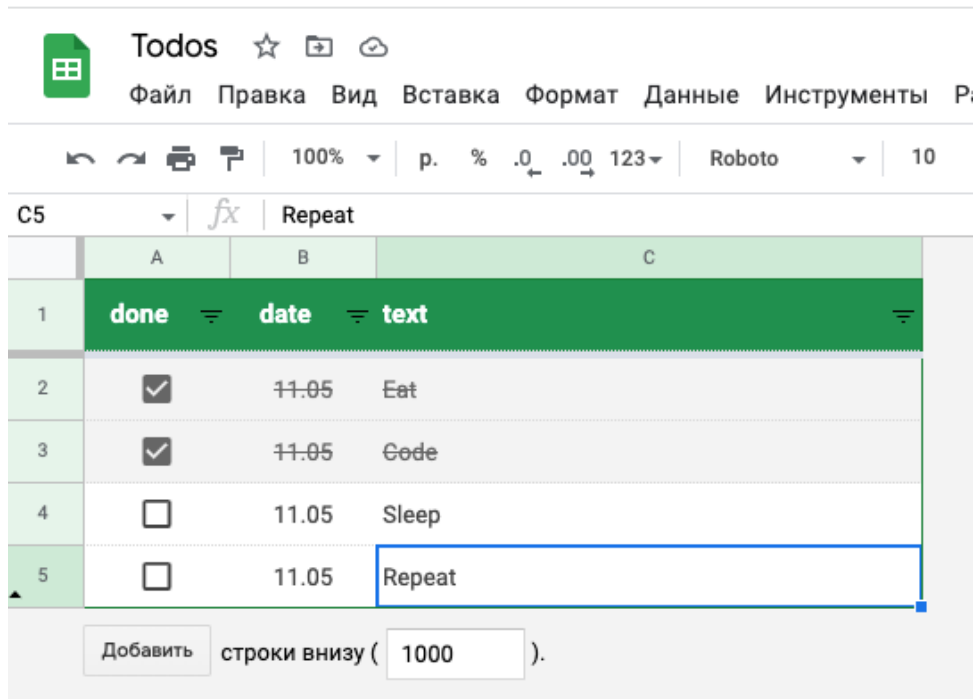
```
"type": "module",
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
},
```

Создаем файл *.env* и записываем туда токен доступа:

```
TELEGRAM_API_TOKEN=5348751300:...
```

Прежде, чем приступить к разработке сервера, необходимо настроить гугл-таблицу, в которой будут храниться наши задачи.

Создаем таблицу в [Google Sheets](#) следующего содержания:



The screenshot shows a Google Sheet titled "Todos" with the following structure:

	A	B	C
1	done	date	text
2	<input checked="" type="checkbox"/>	11.05	Eat
3	<input checked="" type="checkbox"/>	11.05	Code
4	<input type="checkbox"/>	11.05	Sleep
5	<input type="checkbox"/>	11.05	Repeat

At the bottom, there is a button "Добавить" (Add) and a text input "строки внизу (" followed by a value of "1000" and a closing parenthesis "))."

Извлекаем идентификатор таблицы из адресной строки (набор символов между `d/` и `/edit`) и записываем его в `.env`:

```
GOOGLE_SPREADSHEET_ID=1HG60...
```

Идем в [Google Cloud Platform](#), переходим в раздел *IAM & Admin* -> *Service Accounts* и создаем сервис-аккаунт (*Create Service Account*), например: *Telegram Bot Spreadsheet*.

The screenshot shows the Google Cloud Platform (GCP) console interface. On the left, the navigation menu is open, highlighting 'IAM & Admin'. A dropdown menu is visible, listing various IAM-related services, with 'Service Accounts' selected. The main content area displays the 'Create service account' wizard, which is currently on the first step: 'Service account details'. The wizard includes input fields for the service account name (set to 'Telegram Bot Spreadsheet'), the service account ID (set to 'telegram-bot-spreadsheet-1000'), and a description. The email address for the service account is also shown. A 'CREATE AND CONTINUE' button is visible at the bottom of the first step. The second and third steps of the wizard are also visible: 'Grant this service account access to project (optional)' and 'Grant users access to this service account (optional)'. At the bottom of the wizard, there are 'DONE' and 'CANCEL' buttons. The top of the console shows the 'Google Cloud Platform' header with a search bar and a user profile icon.

Google Cloud Platform

Home

View all products

PINNED

Pin your top products here

MORE PRODUCTS

Marketplace

Billing

APIs & Services

Support

IAM & Admin

Getting started

IAM

- Identity & Organization
- Policy Troubleshooter
- Policy Analyzer
- Organization Policies
- Service Accounts**
- Workload Identity Federation
- Labels
- Tags
- Settings
- Privacy & Security
- Identity-Aware Proxy
- Roles
- Audit Logs
- Manage Resources
- Create a Project
- Asset Inventory

Google Cloud Platform Telegram Bot Spreadsheet

Search Produ...

HELP ASSISTANT

Create service account

1 Service account details

Service account name

Telegram Bot Spreadsheet

Display name for this service account

Service account ID *

telegram-bot-spreadsheet-1000

Email address: telegram-bot-spreadsheet-1000@telegram-bot-spreadsheet.iam.gserviceaccount.com

Service account description

Describe what this service account will do

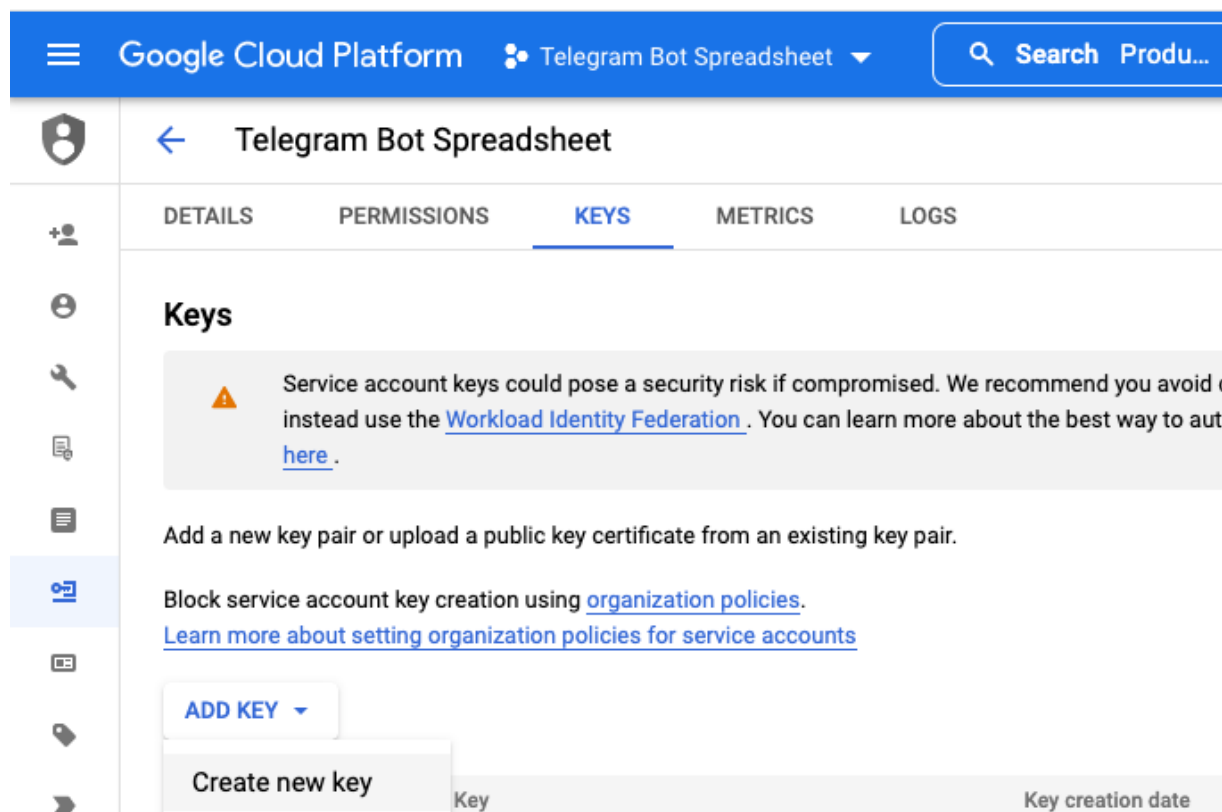
CREATE AND CONTINUE

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

DONE CANCEL

Выбираем созданный сервис-аккаунт, переходим в раздел *Keys* и генерируем ключ (*Add Key* -> *Create new key*) в формате *JSON*:



Create private key for "Telegram Bot Spreadsheet"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

☒ JSON

Recommended

☐ P12

For backward compatibility with code using the P12 format

CANCEL

CREATE

В скачанном *JSON-файле* нас интересуют поля *client_email* и *private_key*. Записываем значения этих полей в *.env*:

```
GOOGLE_SERVICE_ACCOUNT_EMAIL=telegram-bot-spreadsheet@telegram-bot-spreadsheet.iam.g
GOOGLE_PRIVATE_KEY=-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----
```

Переходим в раздел *APIs & Services* -> *Enabled APIs & services* и подключаем интерфейс таблиц с сервис-аккаунту (*Enable APIs and Services*):

The screenshot shows the Google Cloud Platform API Library interface. At the top, there's a blue header with the Google Cloud Platform logo and a search bar. Below the header, the search results for 'sheets' are displayed. The results are filtered by 'sheets' and show 1 result. The result is the Google Sheets API, which is a Google Enterprise API that reads and writes Google Sheets.

API API Library


API Library > "sheets"

Filter by "sheets"

CATEGORY


- Google Enterprise APIs (1)
- CRM (1)
- Google Workspace (1)

1 result


 **Google Sheets API**
Google Enterprise API ?
Reads and writes Google Sheets.

The screenshot shows the detail page for the Google Sheets API. The page has a blue header with the Google Cloud Platform logo and a search bar. Below the header, there's a back arrow. The main content area shows the Google Sheets API icon, the title 'Google Sheets API', and the description 'Google Enterprise API'. Below this, there's a paragraph describing the API: 'The Sheets API gives you full control over the content and appearance of your spreadsheet data.' At the bottom, there are two buttons: 'MANAGE' and 'TRY THIS API'. To the right of these buttons, there's a green checkmark and the text 'API Enabled'. Below the main content area, there's a navigation bar with three tabs: 'OVERVIEW', 'DOCUMENTATION', and 'SUPPORT'.

←

 **Google Sheets API**
Google Enterprise API

The Sheets API gives you full control over the content and appearance of your spreadsheet data.

[MANAGE](#) [TRY THIS API](#)  API Enabled

[OVERVIEW](#) [DOCUMENTATION](#) [SUPPORT](#)

Обратите внимание: аналогичным образом можно настроить доступ к гугл-календарю (только для работы с ним потребуется другой пакет, например, [@googleapis/calendar](#)).

Теперь можно вернуться к разработке сервера.

Создаем файл *index.js* и импортируем зависимости:

```
import axios from 'axios'
import { config } from 'dotenv'
import express from 'express'
import { GoogleSpreadsheet } from 'google-spreadsheet'
```

Получаем доступ к переменным среды окружения, создаем экземпляр приложения *Express* и определяем пути к шуткам и телеграмму:

```
config()
const app = express()

const JOKE_API = 'https://v2.jokeapi.dev/joke/Programming?type=single'
const TELEGRAM_URI = `https://api.telegram.org/bot${process.env.TELEGRAM_API_TOKEN}/`
```

Подключаем посредников (middleware) *Express* и инициализируем таблицу:

```
app.use(express.json())
app.use(
  express.urlencoded({
    extended: true
  })
)

const doc = new GoogleSpreadsheet(process.env.GOOGLE_SPREADSHEET_ID)
await doc.useServiceAccountAuth({
  client_email: process.env.GOOGLE_SERVICE_ACCOUNT_EMAIL,
  private_key: process.env.GOOGLE_PRIVATE_KEY.replace(/\n/g, '\n')
})
```

Определяем роут для *POST-запроса* к */new-message*:

```
app.post('/new-message', async (req, res) => {
  // ...
})
```

Извлекаем сообщение из тела запроса и проверяем, что сообщение содержит текст и идентификатор чата:

```
const { message } = req.body
```

```
const messageText = message?.text?.toLowerCase()?.trim()
const chatId = message?.chat?.id
if (!messageText || !chatId) {
  return res.sendStatus(400)
}
```

Получаем данные из таблицы и формируем данные для ответа:

```
await doc.loadInfo()
const sheet = doc.sheetsByIndex[0]
const rows = await sheet.getRows()
const dataFromSpreadsheet = rows.reduce((obj, row) => {
  if (row.date) {
    const todo = { text: row.text, done: row.done }
    obj[row.date] = obj[row.date] ? [...obj[row.date], todo] : [todo]
  }
  return obj
}, {})
```

Формируем текст ответа:

```
let responseText = 'I have nothing to say.'
if (messageText === 'joke') {
  try {
    const response = await axios(JOKE_API)
    responseText = response.data.joke
  } catch (e) {
    console.log(e)
    res.send(e)
  }
} else if (/^d\d\d\d\d\d$/.test(messageText)) {
  responseText =
    dataFromSpreadsheet[messageText] || 'You have nothing to do on this day.'
}
```

И отправляет ответ:

```
try {
  await axios.post(TELEGRAM_URI, {
    chat_id: chatId,
    text: responseText
  })
  res.send('Done')
} catch (e) {
  console.log(e)
}
```

```
res.send(e)
}
```

Наконец, определяем порт и запускаем сервер:

```
const PORT = process.env.PORT || 3000
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`)
})
```

Бот не сможет взаимодействовать с сервером, запущенным локально, поэтому сервер необходимо где-нибудь развернуть, например, на *Heroku*.

Создаем там новое приложение, например: *my-telegram-bot-server* (название должно быть уникальным в пределах *Heroku*):

The screenshot shows the Heroku web interface. At the top, there's a navigation bar with the Heroku logo and a search bar. Below that, a sidebar on the left contains a 'Personal' dropdown and a search bar for apps and pipelines. The main content area is titled 'Create New App'. It features a form with two main sections: 'App name' and 'Choose a region'. The 'App name' field contains the text 'my-telegram-bot-server' and has a red border with a red exclamation mark icon, indicating an error. Below this field, a red message states 'my-telegram-bot-server is not available'. The 'Choose a region' section has a dropdown menu currently showing 'United States'. At the bottom of the form, there are two buttons: 'Add to pipeline...' and 'Create app'.

Глобально устанавливаем *Heroku CLI* и авторизуемся:

```
yarn global add heroku
```



```
heroku login
# далее следуем инструкциям
```

Находясь в корневой директории проекта, выполняем инициализацию *Git-репозитория*, подключаемся к *Heroku*, добавляем и фиксируем изменения и отправляем их в *Heroku* (не забудьте создать файл *.gitignore* с *node_modules* и *.env*):

```
git init

# у вас название проекта будет другим
heroku git:remote -a my-telegram-bot-server

git add .
git commit -m "create app"
git push heroku master
```

После деплоя получаем *URL* приложения, например: <https://my-telegram-bot-server.herokuapp.com/> (он нам еще пригодится).

Открываем вкладку *Settings* на странице приложения и добавляем переменные среды окружения в разделе *Config Vars (Reveal Config Vars)*:

Personal > my-telegram-bot-server

Overview Resources Deploy Metrics Activity Access Settings

App Information

App Name: my-telegram-bot-server

Region: United States

Stack: heroku-20

Framework: Node.js

Slug size: 35.4 MiB of 500 MiB

Heroku git URL: <https://git.heroku.com/my-telegram-bot-server.git>

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

Hide Config Vars

GOOGLE_PRIVATE_KEY	-----BEGIN PRIVATE KEY-----\nMIIIEvgIBw	✎ ✕
GOOGLE_SERVICE_ACCOUNT_EMAIL	telegram-bot-spreadsheet@telegram-bot-	✎ ✕
GOOGLE_SPREADSHEET_ID	1HG60Jv14AS4o1mUV20T5Xhu4e8wee_-F635Z	✎ ✕
TELEGRAM_API_TOKEN	5348751300:AAEZ_0M26-ULI41A1Q6yA1mFUs;	✎ ✕
KEY	VALUE	Add

Отлично, на этом с разработкой и деплоем сервера мы закончили.

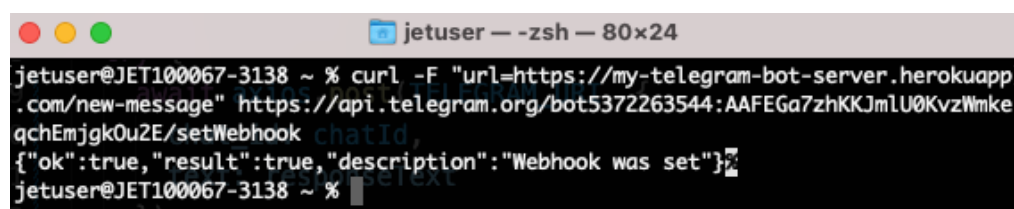
Осталось подключить к нему бота.

Подключение бота к серверу

Существует несколько способов подключения бота к серверу, но самым простым является использование веб-хука.

Открываем терминал и выполняем следующую команду:

```
# в строке ("url=...") указываем `URL` проекта + `/new-message`  
# после `bot` указываем токен доступа из переменной `TELEGRAM_API_TOKEN`  
curl -F "url=https://my-telegram-bot-server.herokuapp.com/new-message" https://api.t
```

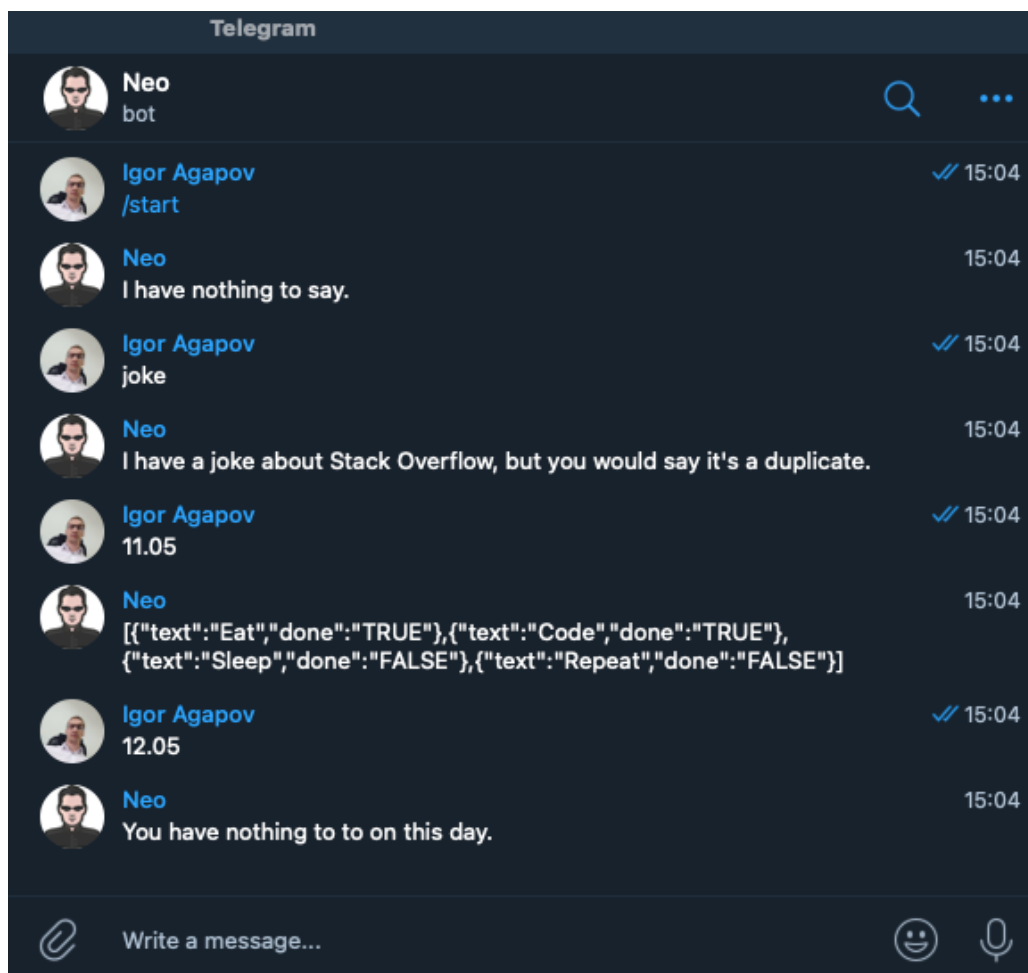


```
jetuser@JET100067-3138 ~ % curl -F "url=https://my-telegram-bot-server.herokuapp.com/new-message" https://api.telegram.org/bot5372263544:AAFEga7zhKKJmLU0KvzWmkeqchEmjgkOuZE/setWebhook  
{\"ok\":true,\"result\":true,\"description\":\"Webhook was set\"}  
jetuser@JET100067-3138 ~ %
```

Получаем сообщение об успешной установке хука.

Возвращаемся к боту.

Нажимаем `/start`, получаем от бота сообщение "I have nothing to say.". Отправляем сообщение "joke", получаем шутку. Отправляем "11.05", получаем задачи в виде массива объектов. Отправляем "12.05", получаем "You have nothing to do on this day."



Примечание: при разработке бота я немного опечтался (вместо "to do" написал "to to"), но уже исправился.

It's alive!)

Пожалуй, это все, чем я хотел поделиться с вами в этой статье.

Надеюсь, вы нашли для себя что-то интересное и не зря потратили время.

Благодарю за внимание и happy coding!

An advertisement for "timeweb > cloud". The background is dark blue. On the right, a white robotic hand is holding a glowing, multi-colored sphere. On the left, the text "timeweb > cloud" is in white. Below it, "Облачный сервер за 10 рублей" is in large white font. Underneath that, in smaller white font, is "С безлимитным трафиком и гарантированной долей CPU 100%". At the bottom left, there is a blue button with the white text "Перейти".

Теги: javascript, node.js, nodejs, telegram bot, telegram, bot, tutorial, бот для телеграмма, телеграмм-бот, бот, tutorial

Хабы: Блог компании Timeweb Cloud, JavaScript, Node.JS

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



Timeweb Cloud

Облачная платформа для разработчиков и бизнеса



113

Карма

60


Рейтинг

Igor Agapov @aio350

JavaScript Developer

Комментарии 3



 **gnome2_terminal_is_best**

13.05.2022 в 14:12

Ссылку на репозиторий с кодом, нужно пофиксить, нужно указать конкретную ветку, после названия репозитория.



0

Ответить



 **aio350**

14.05.2022 в 02:54

Спасибо, друг.



0

Ответить



 **igs_knure**

30.05.2022 в 18:35

Можно реализовать бота с помощью привязанного к той же гугл-таблице приложения, написанного на **Google App Scripts** - вместо реализации сервера на ноде/экспрессе и разворачивания на хероку. GAS-приложение бы и возьмёт на себя приём и обработку запросов от бота, работу с данными... С данными, кстати, так вообще работа сильно упростится - т.к. организовать доступ к ним из приложения, которое к таблице с данными и привязано, всяко проще, чем из своего отдельного приложения.



+1

Ответить



Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

6 мая в 14:16

Node.js: разрабатываем сборщик модулей

◆ +14

👁 2.3K

🔖 38

💬 4 +4

25 апреля в 19:09

Node.js: разрабатываем пакетный менеджер

◆ +7

👁 3.1K

🔖 37

💬 2 +2

9 декабря 2021 в 17:32

Node.js: документирование и визуализация API с помощью Swagger

◆ +9

👁 8.8K

🔖 71

💬 10 +10

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 13:50

Вымой руки. Радиоактивный инцидент в Гоянии

◆ +85

👁 18K

🔖 41

💬 58 +58

вчера в 17:00

Вспоминаем Apple Newton 30 лет спустя

◆ +20

👁 3K

🔖 13

💬 4 +4

сегодня в 00:16

Как россиянину получить ВНЖ в Армении?

◆ +17

👁 2.3K

🔖 18

💬 11 +11

вчера в 19:50

Моя инновационная технология 3Д звука для наушников

◆ +17

👁 3.2K

🔖 15

💬 13 +13

вчера в 23:09

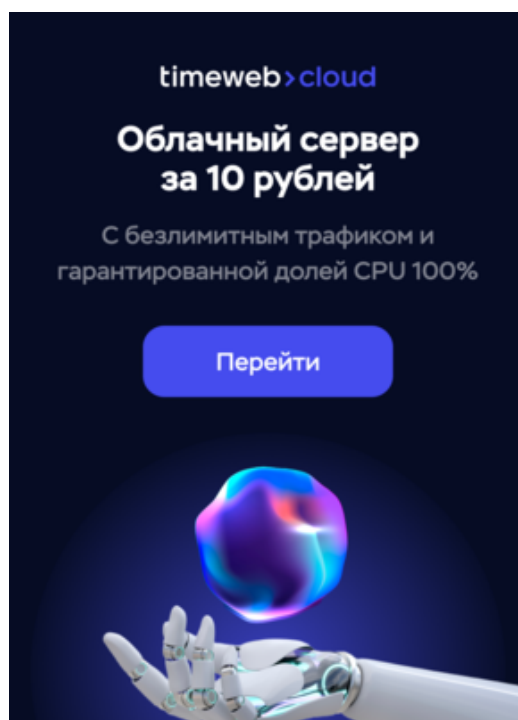
Дайджест недели от Apple Pro Weekly News (30.05 – 05.06.22)

 +16 763 1 0

ИНФОРМАЦИЯ

Дата основания	25 мая 2006
Местоположение	Россия
Сайт	cloud.timeweb.com
Численность	201–500 человек
Дата регистрации	11 августа 2011

ВИДЖЕТ



ВИДЖЕТ



Ваш аккаунт

Войти
Регистрация

Разделы

Публикации
Новости
Хабы
Компании
Авторы

Информация

Устройство сайта
Для авторов
Для компаний
Документы
Соглашение

Услуги

Корпоративный блог
Медийная реклама
Нативные проекты
Мегапроекты

[Песочница](#)[Конфиденциальность](#)[Настройка языка](#)[Техническая поддержка](#)[Вернуться на старую версию](#)

© 2006–2022, Habr

[cloud.timeweb.com](#)[Партнерская программа](#)[timeweb.com](#)[Timeweb News — актуальные новости и скидки](#)[t.me](#)[«Релиз в пятницу» — подкаст от команды Timeweb Cloud](#)[www.youtube.com](#)[Craftum — конструктор сайтов](#)[craftum.com](#)

НОВОСТИ

[Переменные оболочки и окружения в Linux](#)

3 июня

[Май: VPN на WireGuard, свои домены на S3 и отчет с HighLoad](#)

3 июня

[Миграции и сидеры Laravel: настройка базы данных](#)

3 июня

[Привязка собственных доменов к бакетам в S3, отключение доступа к БД по публичному IP и другие изменения](#)

2 июня

[Как установить Webmin на Ubuntu Server 20.04](#)

2 июня

[Как установить Node.js в Ubuntu 20.04](#)

1 июня

[Логические выражения и операторы в Python 3](#)

31 мая

[Установка MariaDB в Ubuntu 20.04](#)

30 мая


Установка Anaconda Python в Ubuntu

30 мая

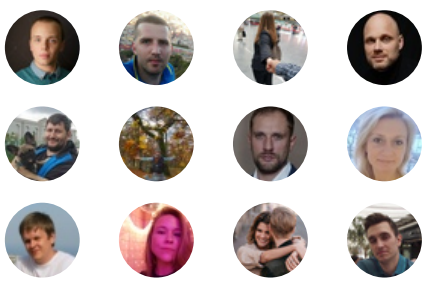
Вывод и удаление правил брандмауэра iptables: инструкция


27 мая

ВКОНТАКТЕ

 **Timeweb: всё про хостинг, IT ...**

54,886 followers



 Follow on VK

ПРИЛОЖЕНИЯ



Приложение для VDS Evo

Управляйте VDS Evo со своего мобильного в удобное для вас время.

Информация о работе ваших VDS и консультация со службой поддержки теперь доступны на мобильном устройстве.

[Android](#) [iOS](#)



Приложение для хостинга

Управляйте виртуальным хостингом прямо со смартфона. Все данные о хостинге и возможность связаться со Службой поддержки внутри одного приложения для Android и iOS.

[Android](#) [iOS](#)

БЛОГ НА ХАБРЕ

вчера в 13:50

Вымой руки. Радиоактивный инцидент в Гоянии

 18K  58 **+58**

3 июня в 16:55

JavaScript: заметка о побитовых операторах и числах с плавающей точкой

 2.1K  3 **+3**

2 июня в 16:00

Automation bias или побочные эффекты автоматизации. Доверяй, но проверяй

 3.4K  10  +10

1 июня в 14:13

Запускаем .NET nanoFramework на Raspberry Pi Pico

 3.5K  23  +23

31 мая в 15:10

Dependency Injection и Full state сервер

 2K  19  +19