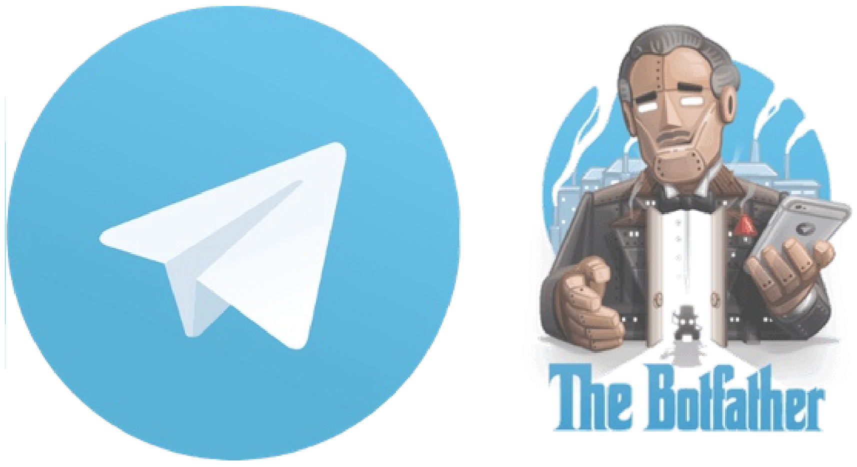
How to make a responsive telegram bot

September 21, 2016

This tutorial will go through a straightforward set of steps to get a responsive telegram bot up and running from scratch



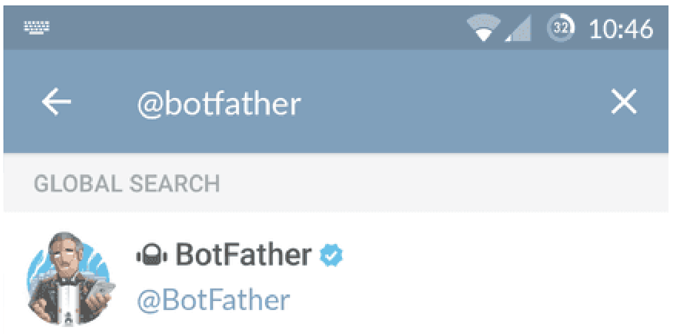
I spent a considerable amount of time figuring out how to make a functional telegram bot. I mean sure, the official introduction is good, but theres a lot of stuff about what bots are, and a few scattered instructions about the API, but not enough of structure for a beginner to get up and running quickly.

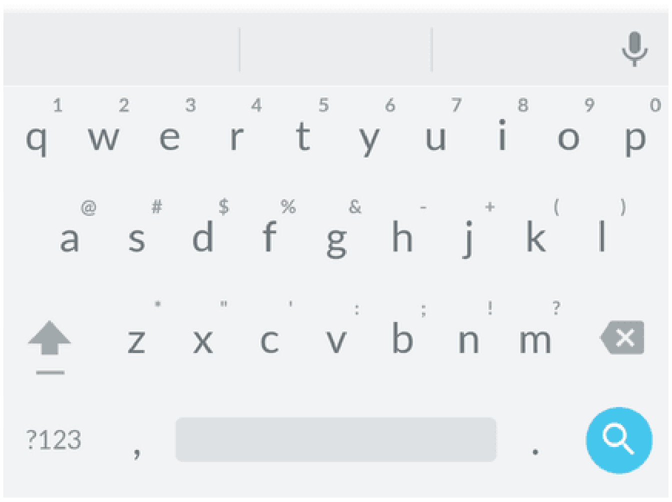
So, heres how to make a responsive telegram bot with the least amount of hassle :

Set up your bot

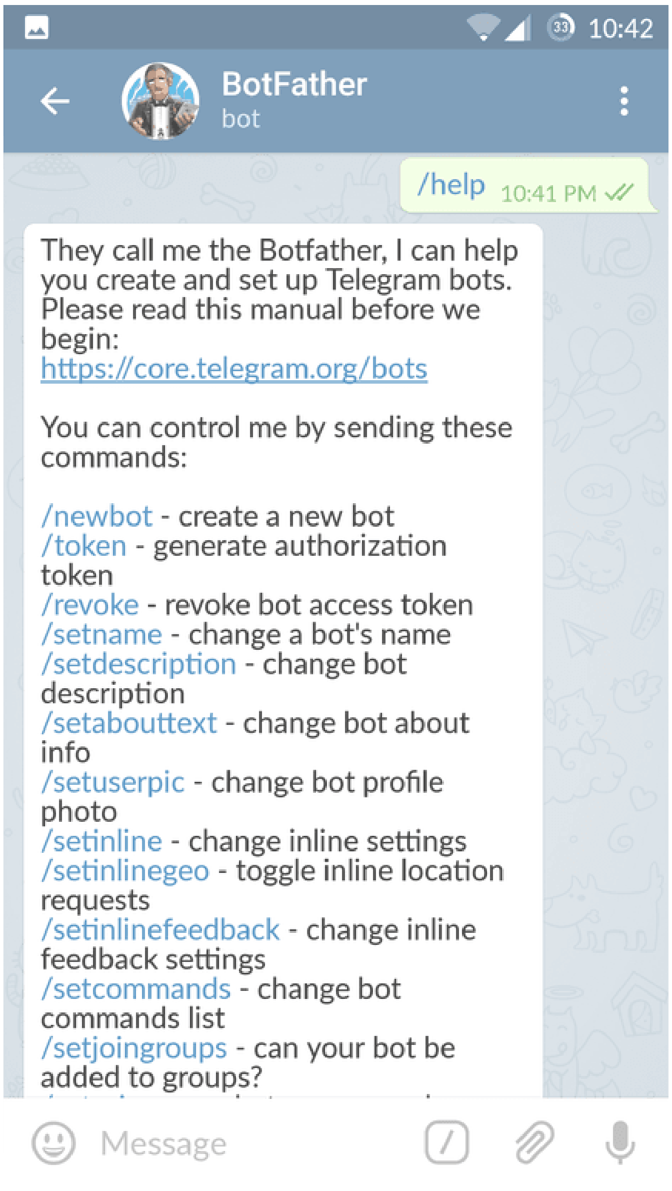
You don't need to write any code for this. In fact, you don't even need your computer! Go to the telegram app on your phone and...

1. Search for the "botfather" telegram bot (he's the one that'll assist you with creating and managing your bot)





2. Type `/help` to see all possible commands the botfather can handle

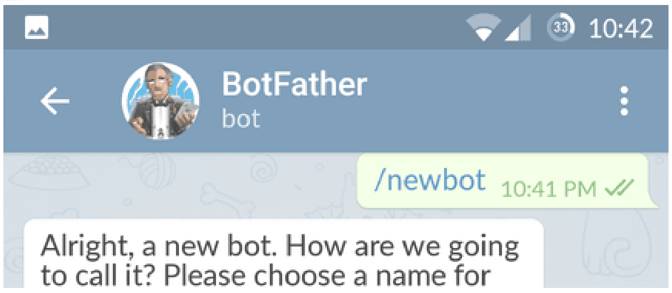


They call me the Botfather, I can help you create and set up Telegram bots. Please read this manual before we begin:
https://core.telegram.org/bots

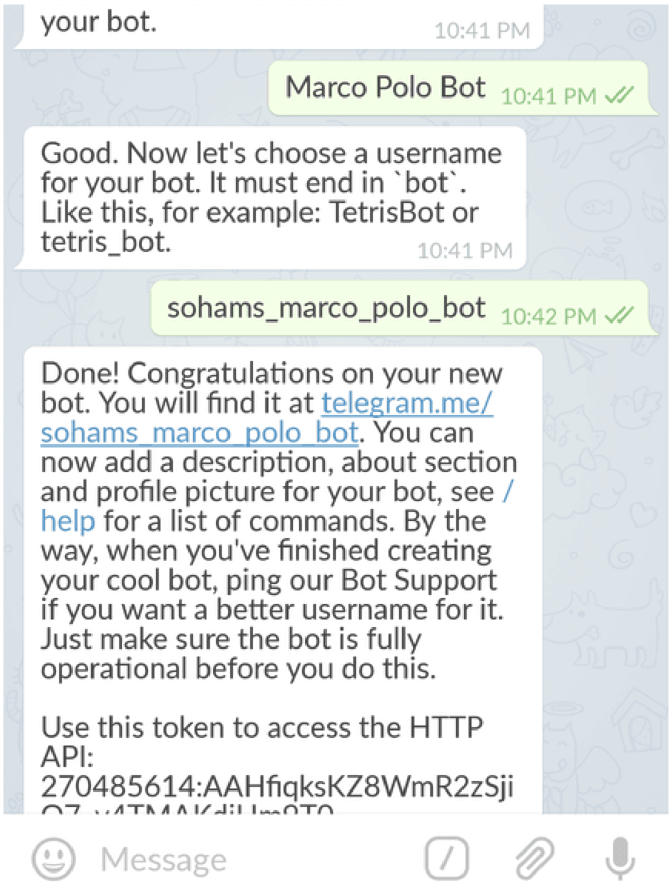
You can control me by sending these commands:

/newbot - create a new bot
/token - generate authorization token
/revoke - revoke bot access token
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setinline - change inline settings
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setcommands - change bot commands list
/setjoingroups - can your bot be added to groups?

3. Click on or type `/newbot` to create a new bot.



Alright, a new bot. How are we going to call it? Please choose a name for


your bot. 10:41 PM

Marco Polo Bot 10:41 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 10:41 PM

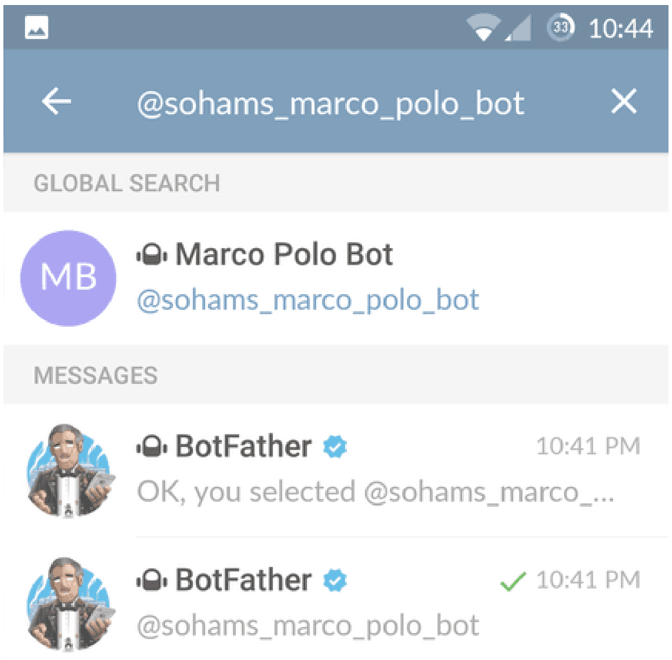
sohams_marco_polo_bot 10:42 PM

Done! Congratulations on your new bot. You will find it at telegram.me/sohams_marco_polo_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
270485614:AAHfiqksKZ8WmR2zSji

Follow instructions and make a new name for your bot. If you are making a bot just for experimentation, it can be useful to namespace your bot by placing your name before it in its username, since it has to be a unique name. Although, its screen name can be whatever you like. I have chosen "Marco Polo Bot" as the screen name and "sohams*marco*polo_bot" as its username.

4. Congratulations! You have created your first bot. You should see a new API token generated for it (for example, in the previous picture, you can see my newly generated token is `270485614:AAHfiqksKZ8WmR2zSjiQ7_v4TMAKdiHm9T0`). Now you can search for your newly created bot on telegram :

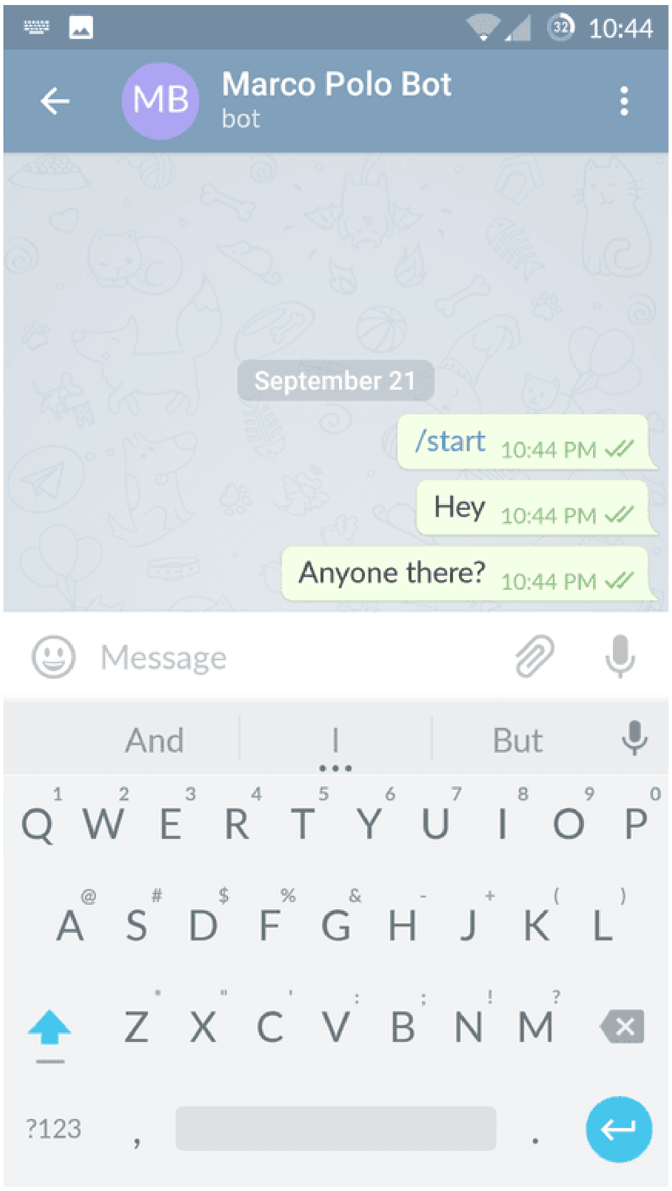


MB Marco Polo Bot
@sohams_marco_polo_bot

BotFather 10:41 PM
OK, you selected @sohams_marco_...

BotFather 10:41 PM
@sohams_marco_polo_bot

5. Go ahead and start chatting with your bot!



Marco Polo Bot
bot

September 21

/start 10:44 PM
Hey 10:44 PM
Anyone there? 10:44 PM

Well, that’s pretty disappointing. Our bot seems to be stupid, in the sense that it can’t really reply or say anything back. Let’s take care of that by building our bot server which runs on the back end.

Set up you bot server

Every time you message a bot, it forwards your message in the form of an API call to a server. This server is what processes and responds to all the messages you send to the bot.

There are two ways we can go about receiving updates whenever someone sends messages to our bot :

1. Long polling : Periodically scan for any messages that may have appeared. Not recommended.

2. Webhooks : Have the bot call an API whenever it receives a message. Much faster and more responsive.

We are going to go with webhooks for this tutorial. Each webhook is called with anupdate object. Lets create our server to handle this update.

We will be creating our server usingnode.js, but you can use whatever suits you to make your server. Once you have node and npm installed :

First, initialize your project

```
## Create a new directory and enter it
mkdir my-telegram-bot
cd my-telegram-bot

## Initialize your npm project
npm init
```

After following the instructions, you will end up with a `package.json` file.

Next, install you dependencies by running :

```
npm install --save express axios body-parser
```

- `express` is our application server
- `axios` is an http client
- `body-parser` will help us parse the response body received from each request

Make a new file `index.js` :

```
var express = require('express')
var app = express()
var bodyParser = require('body-parser')
const axios = require('axios')

app.use(bodyParser.json()) // for parsing application/json
app.use(
  bodyParser.urlencoded({
    extended: true
  })
) // for parsing application/x-www-form-urlencoded

//This is the route the API will call
app.post('/new-message', function(req, res) {
  const { message } = req.body

  //Each message contains "text" and a "chat" object, which has an "id" which is the chat id

  if (!message || message.text.toLowerCase().indexOf('marco') < 0) {
    // In case a message is not present, or if our message does not have the word marco in it, do nothing and return an empty re
    return res.end()
  }

  // If we've gotten this far, it means that we have received a message containing the word "marco".
  // Respond by hitting the telegram bot API and responding to the approprite chat_id with the word "Polo!!"
  // Remember to use your own API toked instead of the one below  "https://api.telegram.org/bot<your_api_token>/sendMessag
  axios
    .post(
      'https://api.telegram.org/bot270485614:AAHfiqksKZ8WmR2zSjiQ7_v4TMAKdiHm9T0/sendMessage',
      {
        chat_id: message.chat.id,
        text: 'Polo!!'
      }
    )
    .then(response => {
      // We get here if the message was successfully posted
      console.log('Message posted')
      res.end('ok')
    })
    .catch(err => {
      // ...and here if it was not
      console.log('Error :', err)
      res.end('Error :' + err)
    })
})

// Finally, start our server
app.listen(3000, function() {
  console.log('Telegram app listening on port 3000!')
})
```

You can run this server on your local machine by running `node index.js`

If all goes well, you should see the message “Telegram app listening on port 3000!” printed on your console.

But, this is not enough. The bot cannot call an API if it is running on your local machine. It needs a public domain name. This means we have to deploy our application.

Deploy your service

You can deploy your server any way you want, but I find it really quick and easy to use a service called now.

Install now on your system :

```
npm install -g now
```

Add a start script to your `package.json` file.

My original `package.json` file looks like :

```
{
  "name": "telegram-bot",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Soham Kamani <sohamkamani@gmail.com> (http://sohamkamani.com)",
  "license": "ISC"
}
```

Add a start script, to get :

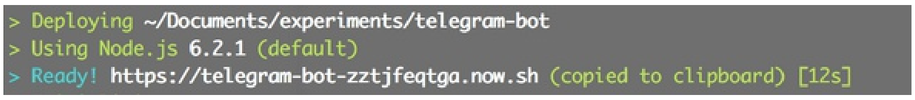
```
{
  "name": "telegram-bot",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start" : "node index.js"
  },
  "author": "Soham Kamani <sohamkamani@gmail.com> (http://sohamkamani.com)",
  "license": "ISC"
}
```

Once you've added the script, run the command :

```
now
```

(remember to run in in the root of your project folder, wherever the `package.json` file is located)

If this is your first time using "now", you will see some instructions for signing in, but after that you should see something like this :



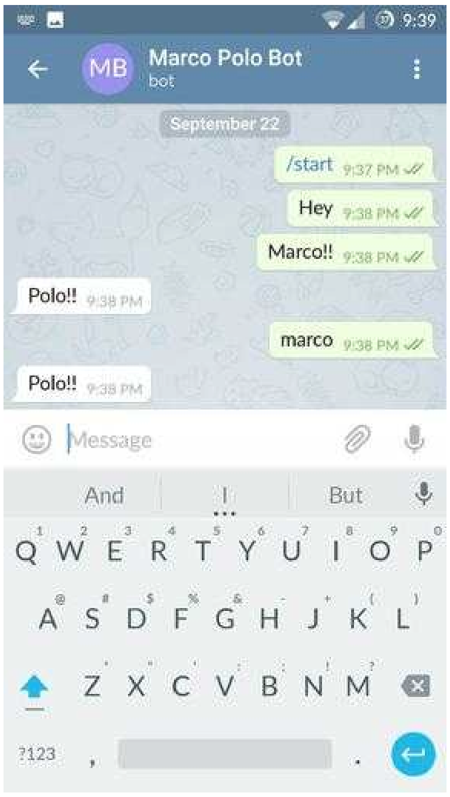
```
> Deploying ~/Documents/experiments/telegram-bot
> Using Node.js 6.2.1 (default)
> Ready! https://telegram-bot-zztjfeqtga.now.sh (copied to clipboard) [12s]
```

Great! This means your server is deployed on `https://telegram-bot-zztjfeqtga.now.sh` (or whatever link you see instead), and your API would be present on `https://telegram-bot-zztjfeqtga.now.sh/new-message` (as defined in `index.js`)

Now, all we need to do is let telegram know that our bot has to talk to this url whenver it receives any message. We do this through the telegram API. Enter this in your terminal :

```
curl -F "url=https://telegram-bot-zztjfeqtga.now.sh/new-message"  https://api.telegram.org/bot<your_api_token>/setWebhook
```

...and you're pretty much done! Try chatting with your newly made bot and see what happens!



Marco Polo Bot
bot
September 22
/start 9:37 PM
Hey 9:38 PM
Marco!! 9:38 PM
Polo!! 9:38 PM
marco 9:38 PM
Polo!! 9:38 PM

Comments

Written by **Soham Kamani**, an author,and a full-stack developer who has extensive experience in the JavaScript ecosystem, and building large scale applications in Go. He is an open source enthusiast and an avid blogger. You shouldfollow him on Twitter

Social stuff:

- Follow me on Twitter
- Fork me on Github
- Connect on LinkedIn

Related Posts:

Implementing OAuth 2.0 with Node.js ▯

How express.js works - Understanding the internals of the express library ⚙

Using enums (enumerations) in javascript ▯

React is a framework

Make your node server faster by caching responses with redis ▯

An incremental tutorial on promises in javascript ▯

Object oriented programming in javascript ▯

How to communicate between Python and NodeJs ▯