Материалы к занятию

Для примера работы нашего тестового бота мы используем сервис Heroku. Для начала давайте вспомним шаблон для работы с вебхуками и модернизируем его под эхо бота. После чего разберем новые для нас понятия.

```
import logging
import os
from aiogram import Bot
from aiogram.dispatcher import Dispatcher
from aiogram.utils.executor import start_webhook
from aiogram import Bot, types


TOKEN = os.getenv('BOT_TOKEN')
bot = Bot(token=TOKEN)
dp = Dispatcher(bot)

HEROKU_APP_NAME = os.getenv('HEROKU_APP_NAME')

# webhook settings
WEBHOOK_HOST = f'https://{HEROKU_APP_NAME}.herokuapp.com'
WEBHOOK_PATH = f'/webhook/{TOKEN}'
WEBHOOK_URL = f'{WEBHOOK_HOST}{WEBHOOK_PATH}'

# webserver settings
WEBAPP_HOST = '0.0.0.0'
WEBAPP_PORT = os.getenv('PORT', default=8000)


async def on_startup(dispatcher):
    await bot.set_webhook(WEBHOOK_URL, drop_pending_updates=True)


async def on_shutdown(dispatcher):
    await bot.delete_webhook()


@dp.message_handler()
async def echo(message: types.Message):
    await message.answer(message.text)


 if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO)
    start_webhook(
        dispatcher=dp,
        webhook_path=WEBHOOK_PATH,
        skip_updates=True,
        on_startup=on_startup,
        on_shutdown=on_shutdown,
        host=WEBAPP_HOST,
        port=WEBAPP_PORT,
    )
```

```

```

Разбор выгрузки на сервис мы разберем позднее. Давайте для начала кратко разберем, что у нас в коде.

TOKEN, HEROKU_APP_NAME – мы считываем из переменных окружения, которые скоро добавим в наш проект.

WEBHOOK_HOST – доменное имя нашего приложения

WEBHOOK_PATH – часть пути, на который мы будем принимать запросы. Его следует придумать таким, чтобы не было возможности его угадать, во избежание фальсификации запросов. В нашем случае используется токен бота, так как его, также, следует держать в секрете.

WEBHOOK_URL – полный url адрес, на который будут принимать запросы.

WEBAPP_HOST – хост нашего приложения, оставляем локальный.

WEBAPP_PORT – порт, на котором работает наше приложение, так же считывается с переменных окружения, которое предоставляет Heroku, его мы не заполняем.

Асинхронная функция on_startup устанавливает вебхук для нашего телеграм бота, на который будут отсылаться уведомления о получении новых сообщений.
on_shutdown, наоборот, удаляет этот вебхук при выключении.
Далее мы переключаем вывод логов только на вывод только чисто информативной информации. И, собственно, запускаем наш диспетчер, при этом при запуске опускаются все сообщения, которые были получены в то время, когда бот не работал, что указано в параметре «skip_updates».

Почти всё готово, но чтобы дать инструкции Heroku, как именно развернуть наше приложение, нужно создать файл «Procfile» и вставляем туда следующий код:

```
web: python main.py
```

Здесь: web – значит, что наше приложение будет web приложением, а то, что идёт после «:» это строка, которую необходимо выполнить в первую очередь. Запустить наш файл main.py с помощью питона.

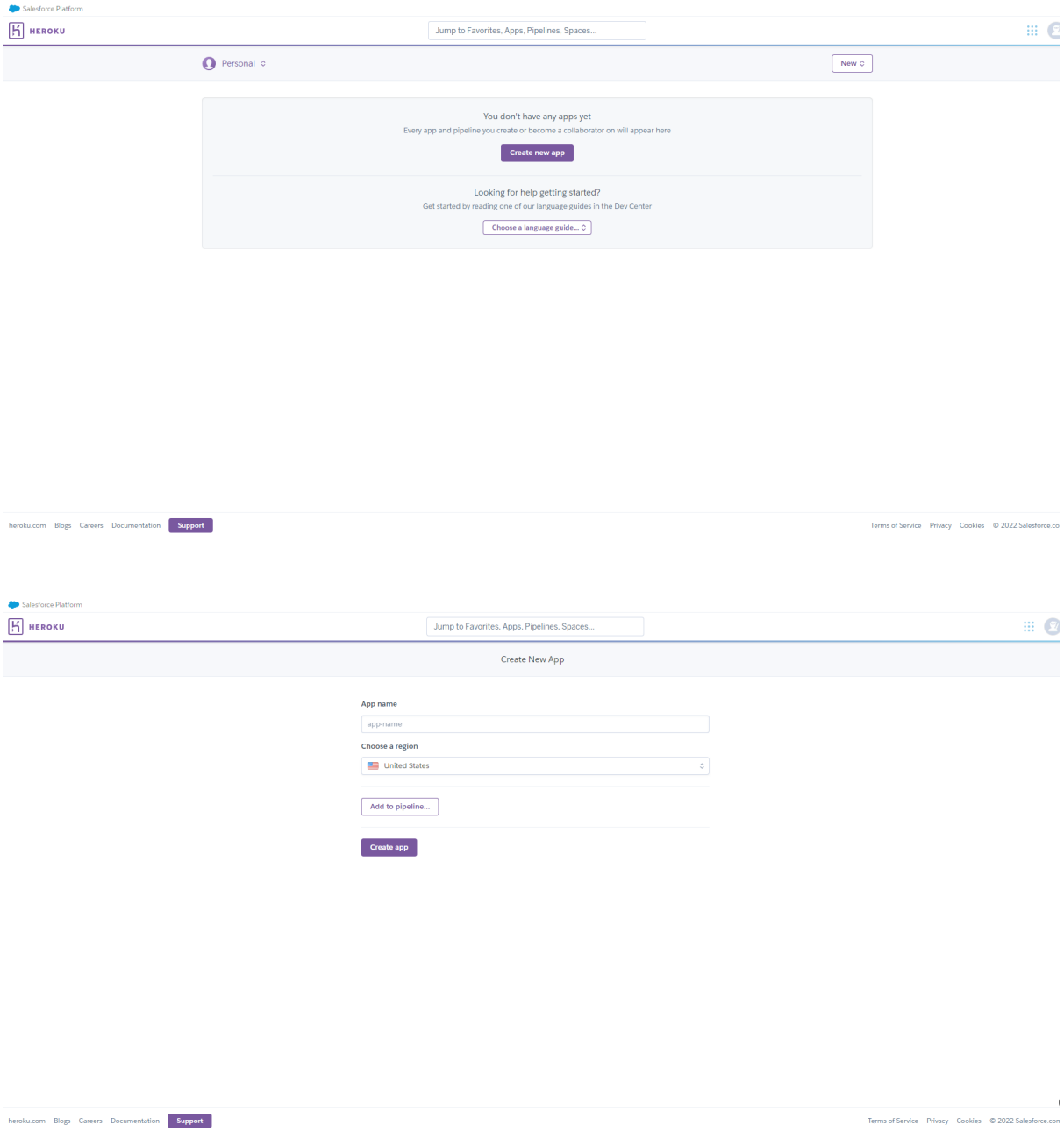
И ещё один файл, который необходим для запуска, это requirements.txt, в котором мы указываем все зависимости нашего проекта. Его создаём, выполнив команду

pip freeze > requirements.txt.

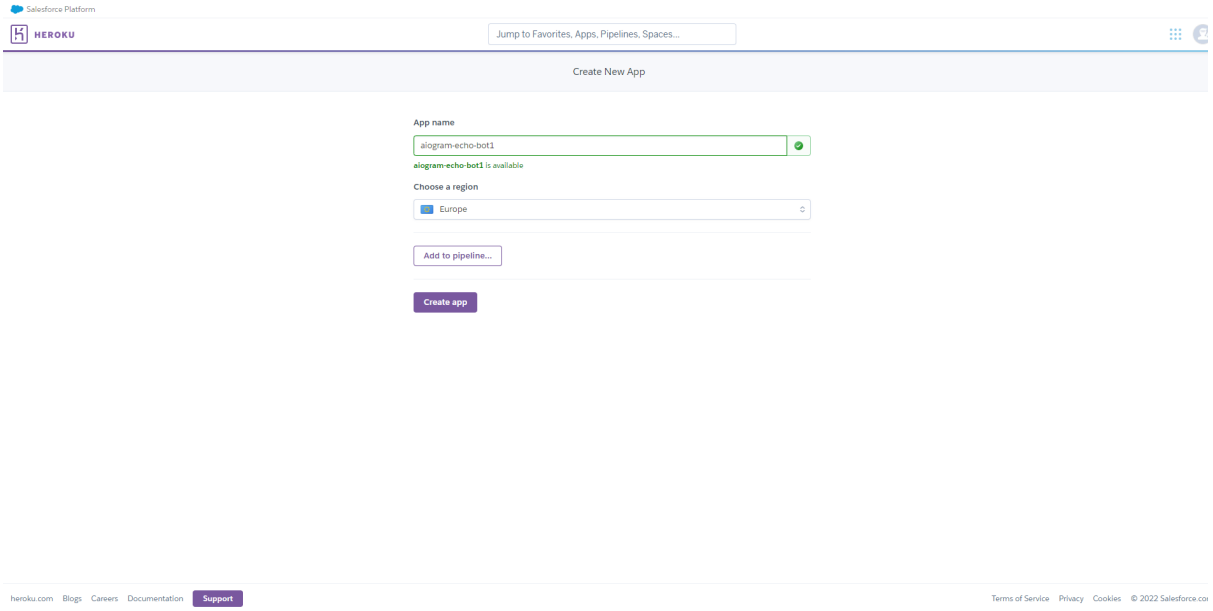
Также можно указать, какую конкретную версию питона использовать: для этого создадим файл «runtime.txt» и впишем туда версию питона по шаблону «python-3.9.7»

Далее необходимо:

1. Зарегистрироваться на сайте https://dashboard.heroku.com/
2. На странице Personal создаём новое приложение.

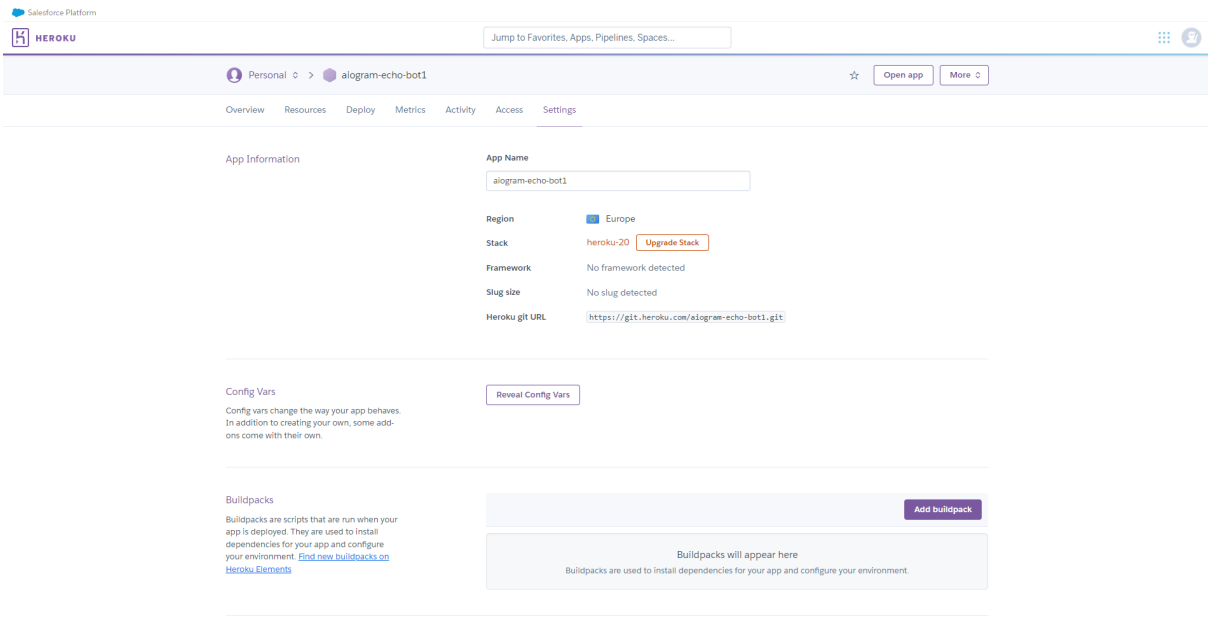


3. Выбираем имя нашего приложения (у меня это «aiogram-echo-bot1» - запомним его, оно нам ещё понадобится!), меняем сервер на Europe и нажимаем кнопку «create app»



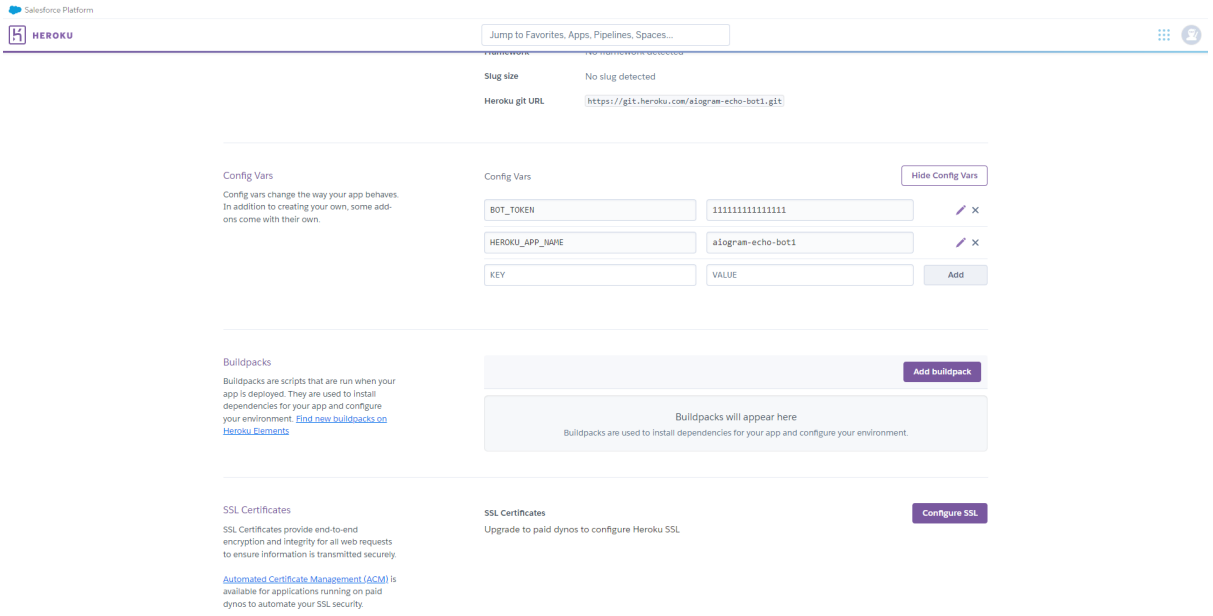
Отлично, мы подготовили контейнер для нашего приложения! Передать туда код самого приложения можно несколькими способами, например через Heroku CLI или через GitHub.

Теперь подготовим переменные среды на Heroku: для этого переходим на вкладку «Settings» и жмём кнопку «Reveal Config Vars»

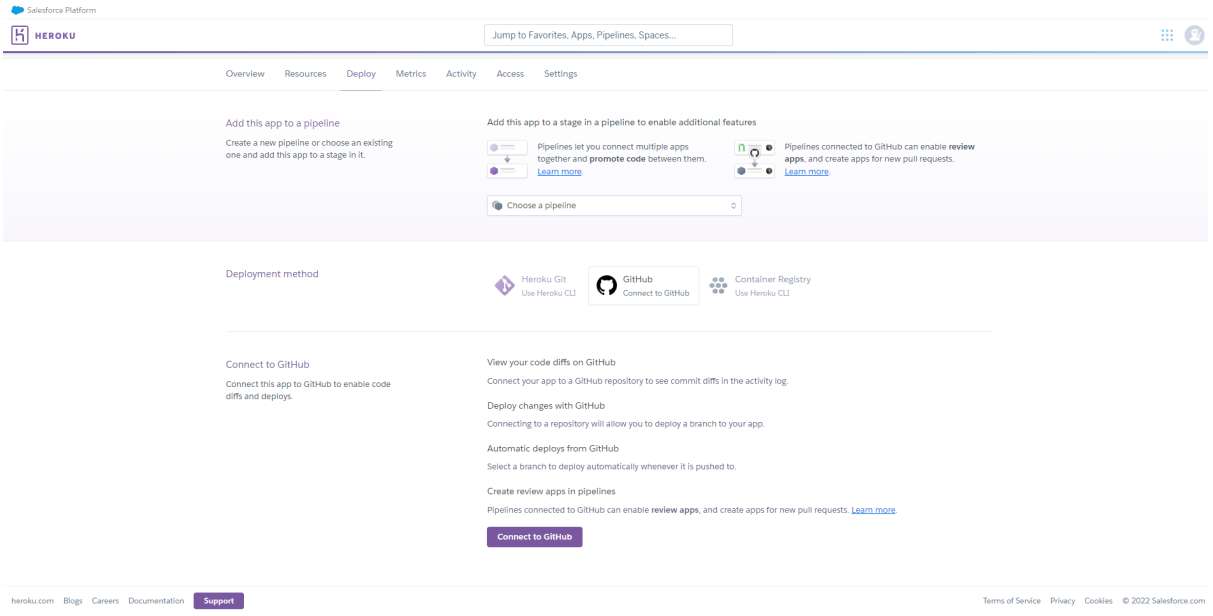


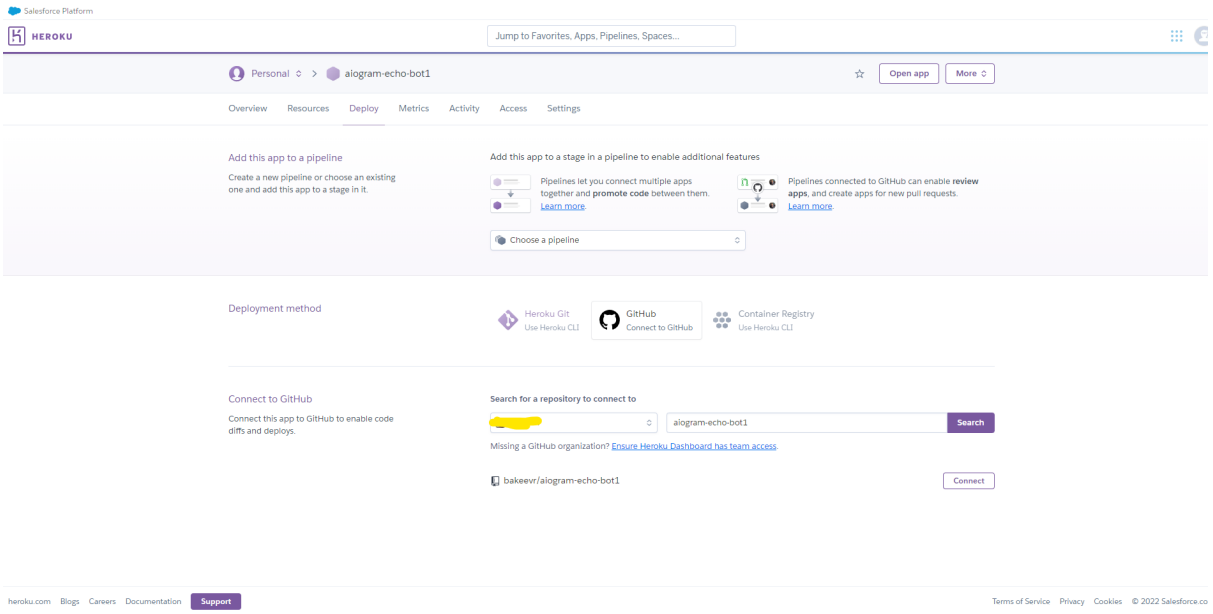
Здесь добавляем два поля:
BOT_TOKEN – токен, полученный у BotFather

HEROKU_APP_NAME – имя приложения созданного на heroku

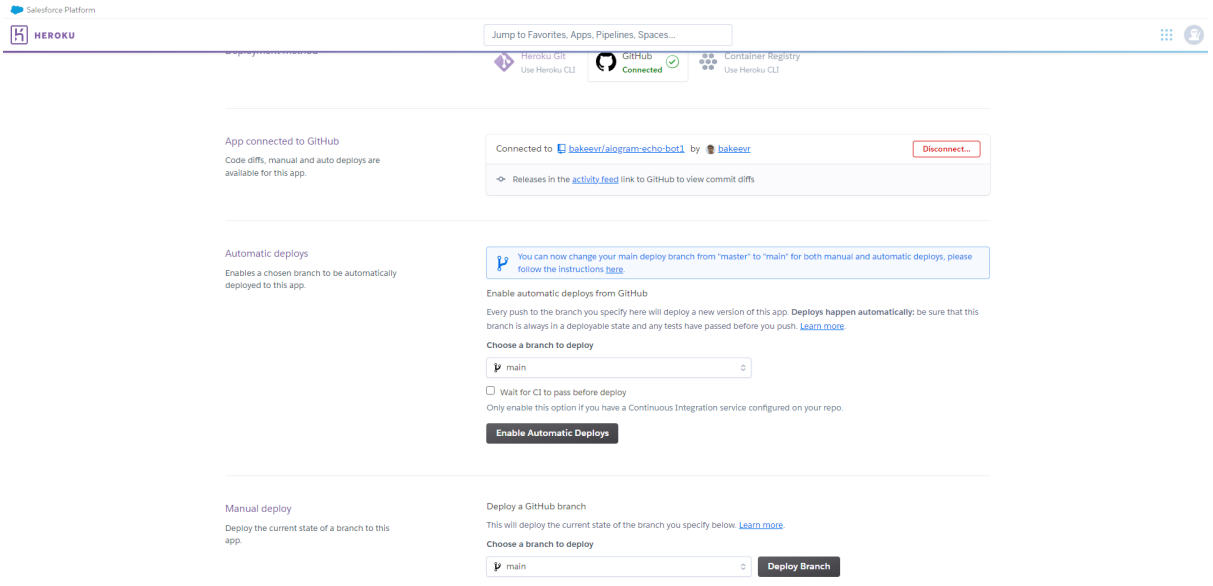


Далее мы регистрируемся на сайте github. После подключаем свой аккаунт github к Heroku

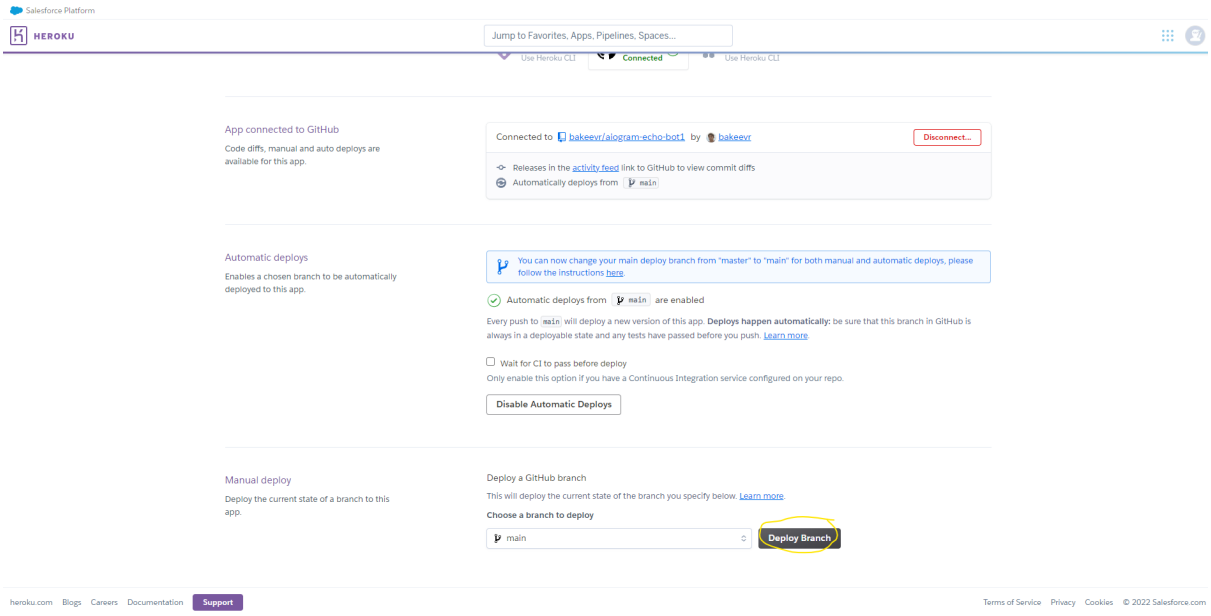




Для того, чтобы наше приложение обновлялось каждый раз, как мы заливаем новые изменения в ветку «master», можем нажать кнопку «Enable Automatic Deploys»



первый раз, всё-таки придётся деплоить самим, для этого нажимаем кнопку ниже:



Переходим в наш бот и отправляем ему пару сообщений, если бот отвечает, значит всё в порядке, если нет, то переходим в логи и смотрим, в чём может быть ошибка.

Вот таким образом мы реализовали деплой нашего простого бота в сервис Heroku. Как мы видим это занимает больше времени, но наш бот может работать 24 часа на 7, что уже совсем неплохо, ведь он не занимает ресурсы нашего пк