

Поделитесь своим мнением о Яндекс.Лицее и помогите нам в его развитии

[Пройти опрос →](#)

Урок Облако

Разворачиваем проект в облаке. Дорешка

- 1 Введение
- 2 Туннель в Интернет
- 3 Heroku
- 4 Яндекс.Облако

Аннотация

Сегодня мы посмотрим, как разместить свой проект в Интернете, чтобы к нему могли получить доступ все желающие, а также изучим несколько сервисов, которые смогут нам в этом помочь. Этот урок не содержит задач.

1. Введение

Разрабатывать веб-приложения, которые «живут» только на нашем локальном компьютере, согласитесь, не очень весело. Само слово «веб» подразумевает, что к результатам нашей работы должны иметь доступ все пользователи сети Интернет, ну или те, которым мы даем на это право. Процесс размещения ресурсов в Интернете называется deploy.

Пока наше flask-приложение размещено на локальном компьютере, и мы помним, что каждый компьютер в мире думает про себя, что его IP-адрес 127.0.0.1. Как же тогда пользовательский запрос из Интернета может добраться до нашего компьютера и получить нужную веб-страницу или json-ответ от сервиса? Ведь таких машин с адресом 127.0.0.1 миллионы!

Для решения этой задачи есть несколько способов и множество разных сервисов. С некоторыми из них мы сегодня познакомимся.

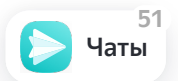
Давайте сделаем простейшее веб-приложение, на примере которого мы будем проверять различные способы организации доступа к веб-приложению через Интернет.

```
import os
```

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route("/")  
def index():
```



```

def index():
    return "Привет от приложения Flask"

if __name__ == '__main__':

    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)

```

Из нового: `os.environ` пытается получить значение переменной окружения с именем `PORT`, а если не получается, то выставляет порт 5000. Это нам пригодится позднее.

`host = 0.0.0.0` означает, что мы разрешаем подключения из любой сети по любому интерфейсу.

2. Туннель в Интернет

Очень часто при отладке веб-приложений хочется посмотреть, что происходит, когда в приложение приходят запросы из Интернета, или продемонстрировать результаты работы другу/заказчику. Не всегда удобно ради тестирования или ради одноразовой демонстрации арендовать и настраивать сервер в Интернете.

Согласитесь, было бы удобно получить какой-то временный адрес в Интернете «в один клик» для таких целей. К счастью, для этого есть сервисы, которые организуют виртуальный туннель из Интернета на ваш локальный компьютер. Один из них — **ngrok**.

Важная его особенность: у **ngrok** есть бесплатный план, который не требует даже регистрации на сервисе. Но даже этой ограниченной версии хватит для наших целей.

Ngrok не требует практически никакой настройки. Необходимо скачать исполняемый файл приложения для нужной операционной системы со **страницы загрузки** и запустить его. Нас встретит такое окно:

```

C:\Users\crja72\Downloads\ngrok-stable-windows-amd64\ngrok.exe

EXAMPLES:
  ngrok http 80                        # secure public URL for port 80 web server
  ngrok http -subdomain=baz 8080      # port 8080 available at baz.ngrok.io
  ngrok http foo.dev:80               # tunnel to host:port instead of localhost
  ngrok http https://localhost       # expose a local https server
  ngrok tcp 22                        # tunnel arbitrary TCP traffic to port 22
  ngrok tls -hostname=foo.com 443     # TLS traffic for foo.com to port 443
  ngrok start foo bar baz             # start tunnels from the configuration file

VERSION:
  2.3.35

AUTHOR:
  inconshreveable - <alan@ngrok.com>

COMMANDS:
  authtoken  save authtoken to configuration file
  credits    prints author and licensing information
  http       start an HTTP tunnel
  start      start tunnels by name from the configuration file
  tcp        start a TCP tunnel
  tls        start a TLS tunnel
  update     update ngrok to the latest version
  version    print the version string
  help       Shows a list of commands or help for one command

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\crja72\Downloads\ngrok-stable-windows-amd64>

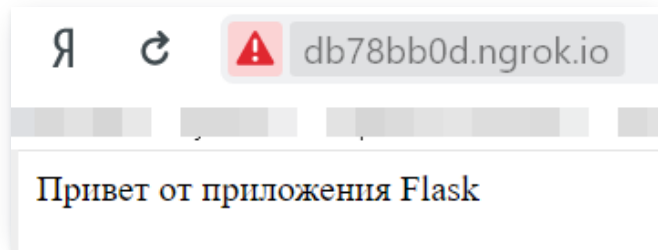
```

Теперь запустим наше тестовое приложение в PyCharm или через командную строку. Выполним в окне команду:

```
ngrok http 5000
```

После этого **ngrok** создаст временный адрес в Интернете вида `набор_символов.ngrok.io` и виртуальный туннель

с этого адреса на пятидесятый порт нашего локального компьютера. Теперь все запросы по протоколу http и https, которые придут на созданный временный адрес, будут перенаправляться на наш компьютер. Давайте проверим. Перейдем по этому адресу (скопируем и вставим его в строку запроса браузера) и посмотрим на результат.



Все работает! А в консоли ngrok появилось сообщение, что GET-запрос успешно обработан:

```
HTTP Requests
-----

GET /favicon.ico           404 NOT FOUND
GET /favicon.ico           404 NOT FOUND
GET /                       200 OK
```

Таким образом можно организовывать тестирование приложения в Интернете. Особенно такой подход удобен, когда для проверки функциональности приложения надо получить запрос извне. В дальнейшем мы будем использовать эту методику при отладке приложений для голосового помощника Алисы.

Кроме ручного запуска ngrok есть возможность автоматизировать этот процесс с помощью модуля flask-ngrok. Установим модуль:

```
pip install flask-ngrok
```

Чтобы все заработало как надо, необходимо внести небольшие изменения в код нашего примера:

```
from flask import Flask
from flask_ngrok import run_with_ngrok

app = Flask(__name__)
run_with_ngrok(app)

@app.route("/")
def index():
    return "Привет от приложения Flask"

if __name__ == '__main__':
    app.run()
```

После создания объекта класса Flask необходимо передать его в функцию run_with_ngrok библиотеки flask-ngrok. Все предельно просто. Однако обратите внимание: при использовании этой библиотеки при вызове метода run объекта класса Flask пока нельзя передавать дополнительные параметры (хост, порт, и т. д.).

После запуска программы сначала запустится прослушивание локального порта, а затем и туннель через ngrok.

```
* Serving Flask app "main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
```

```
Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
* Running on http://d58732a9.ngrok.io  
* Traffic stats available on http://127.0.0.1:4040
```

3. Heroku

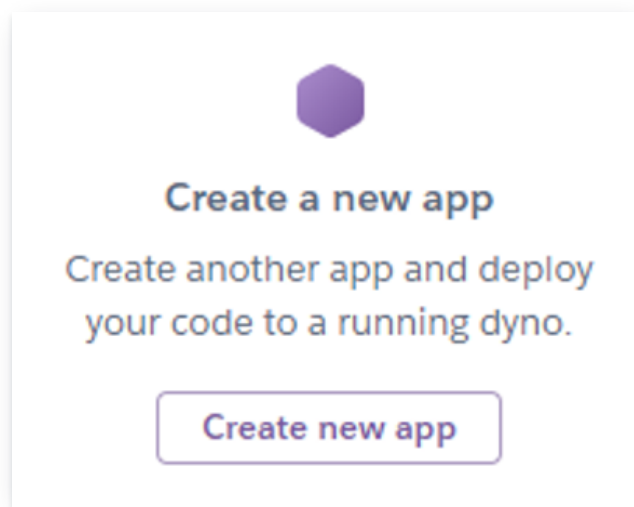
Более продвинутый способ, который (при использовании платных планов) подойдет для «боевого» разворачивания приложения в Интернете (деплой) — аренда контейнеров для приложений на одном из множества сервисов, которые предоставляют такие услуги.

Сервисов действительно очень много, и все они разнятся как в стоимости (некоторые предлагают бесплатный тариф, который подойдет для приложения с небольшим количеством пользователей), так и в сложности настроек для запуска.

В качестве одного из самых легких стоит привести в пример **pythonanywhere.com**, который содержит ряд преднастроенных шаблонов для различных приложений, разработанных на Python. К недостатку этого сервиса можно отнести то, что приложения, размещенные на нем, могут обращаться только к тем ресурсам, которые были внесены в «белый список» доступных адресов.

Немного более сложным в обращении является сервис **Heroku**. Он поддерживает приложения не только на Python, но и на других языках программирования, различные базы данных и разные преднастроенные сторонние приложения.

Зарегистрируйтесь на Heroku. После чего создайте новое приложение по адресу <https://dashboard.heroku.com/apps>



Придумайте приложению имя и выберите расположение.

A screenshot of the Heroku "Create New App" form. The form has a title "Create New App" at the top. Below it, there is a section "App name" with a text input field containing "yl-flask-test" and a green checkmark icon to its right. Below the input field, the text "yl-flask-test is available" is displayed in green. Underneath, there is a section "Choose a region" with a dropdown menu showing "Europe" and a globe icon. The form is light blue and white.

51
Чаты

 Add to pipeline...

Create app

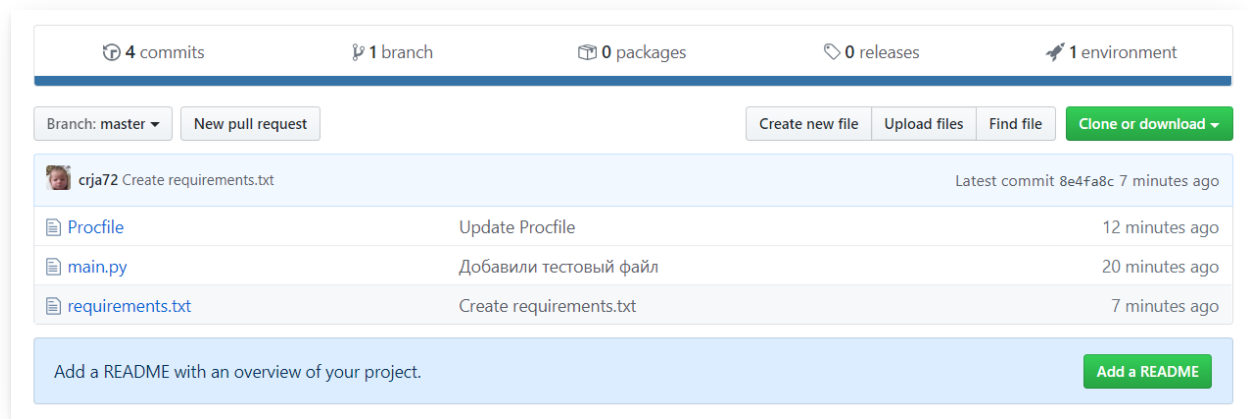
Есть несколько способов разместить свое приложение на Heroku:

- Через консольный клиент Heroku CLI
- Подключив свой репозиторий на GitHub

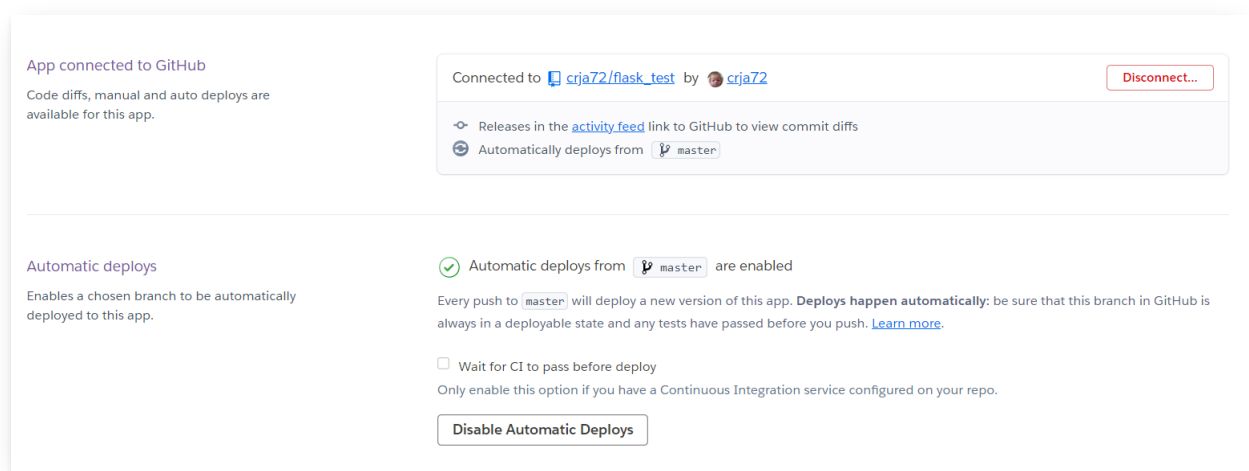
Давайте пойдем по второму пути и создадим репозиторий (не имеет значения публичный или приватный), в который **запушим** файл с нашим простейшим веб-приложением, а также еще несколько файлов. Первый из них — служебный текстовый файл Procfile (без расширения), в котором для Heroku будет указано, какой файл запускать, с текстом:

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

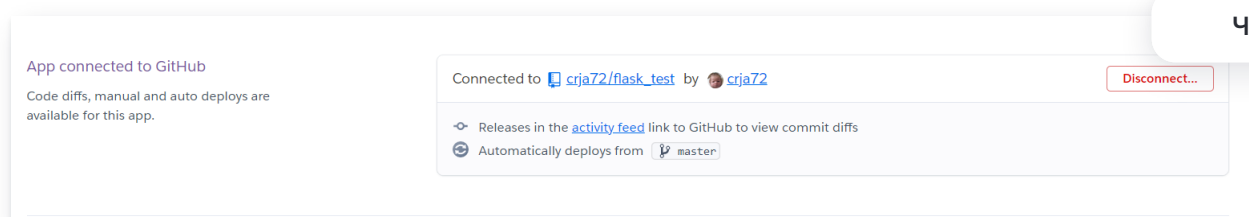
Второй — уже знакомый нам из раздела PyQT requirements.txt, в котором необходимо указать все используемые сторонние модули (flask).



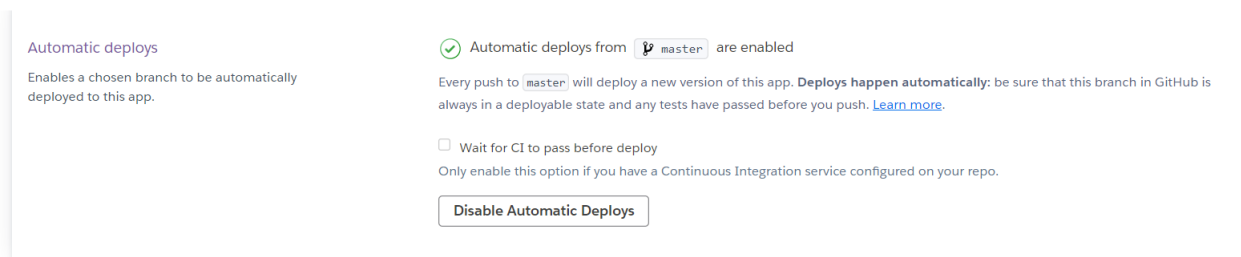
Подключим GitHub к Heroku и выберем нужный репозиторий, а затем и ветку, с которой будем забирать для размещения код. Это можно сделать автоматически или в ручном режиме.



Если не включено автоматическое размещение кода, надо нажать на кнопку Deploy branch.



51
Чаты



Все. Можно переходить по адресу `имя_приложения.herokuapp.com` и наслаждаться результатом.

При размещении приложений на Heroku очень важно указывать порт, на котором будет работать ваше приложение именно так, как мы сделали в первом примере этого урока.

```
if __name__ == '__main__':
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)
```

Когда приложение разворачивается в Heroku, для него будет выделен порт и установлен в переменную окружения. Важно, чтобы ваше приложение слушало именно этот порт, иначе оно не заработает.

4. Яндекс.Облако

Самым сложным, но одновременно самым гибким и продвинутым вариантом является размещение приложения на облачной виртуальной машине с использованием сервиса Яндекс.Облако или аналогов.

Создание и настройка таких виртуальных машин требует определенных усилий и несколько сложнее, чем сервисы, которые мы рассматривали ранее, но зато вы сможете полностью контролировать результат и реализовывать разные продвинутые сценарии разворачивания вашего приложения.

Для начала необходимо зайти на **Яндекс.Облако** и нажать кнопку «Подключиться» в правом верхнем углу экрана. Далее алгоритм работы можно описать так:

1. Необходимо создать виртуальную машину
2. Выбрать и установить образ операционной системы, которую вы планируете использовать. Можно использовать как «чистый» образ (только операционная система), так и подготовленный командой Яндекс.Облака или сторонними разработчиками
3. Установить все необходимое ПО: интерпретатор Python, все библиотеки, системные утилиты, которые обеспечат работу при большой нагрузке и т. д.
4. Загрузить разработанный вами сервер и ...
5. Поддерживать работу вашего ресурса

Поверьте, это далеко не просто.

Поэтому мы немного вам помогли. Мы подготовили для вас видео, в котором рассказываем, как сделать базовую настройку сервера и установить на него ваше приложение.

Справка

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках проекта «Яндекс.Лицей», принадлежат АНО ДПО «ШАД». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «ШАД».

[Пользовательское соглашение.](#)

© 2018 – 2021 ООО «Яндекс»