

Разворачиваем проект в облаке. Дорешка

- 1 Введение
- 2 Туннель в Интернет
- 3 Glitch.com
- 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():
    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, но даже этой ограниченной версии хватит для наших целей. Сначала надо зарегистрироваться и залогиниться на **сайте** сервиса.

На странице "**Setup & Installation**" приведена подробная инструкция по настройке и запуску утилиты.

1. Необходимо скачать исполняемый файл приложения для нужной операционной системы со **страницы загрузки**.

2. Распаковать архив и в окне терминала перейти в папку с распакованной утилитой.

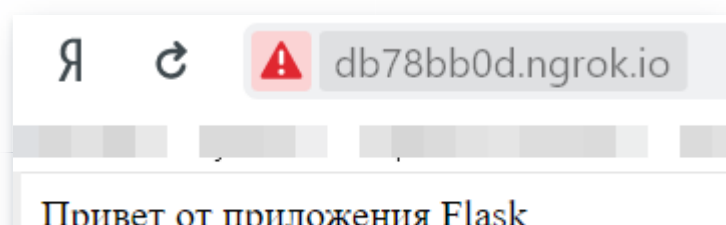
3. Скопировать со страницы "Setup & Installation" команду для настройки работы ngrok с вашей учетной записью и запустить ее в терминале. Выглядит она так `ngrok authtoken 25srTxReQIIwsPMX67ZiY1E5TMH_7KbpfVGLcg9rRDd2hwfk9`

4. Настройка завершена.

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

```
ngrok http 5000
```

После этого ngrok создаст временный адрес в Интернете вида `набор_символов.ngrok.io` и виртуальный туннель с этого адреса на пяти тысячный 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.

3. Glitch.com

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 **Glitch**. Он поддерживает приложения не только на Python, но и на других языках программирования.

Зарегистрируйтесь на Glitch. После чего можно приступить к созданию нового приложения. Сначала надо его подготовить к размещению на сервисе.

1. Переименуйте основной запускаемый файл вашего проекта и назовите его `"server.py"`
2. В корне проекта создайте файл `".gitignore"` со следующим содержимым:

```
.idea
venv
```

Это необходимо, чтобы файл из этих служебных папок игнорировались системой контроля версий.

3. Создайте файл `"requirements.txt"` со списком библиотек, которые необходимы для вашего приложения.
4. В корне проекта создайте файл `"start.sh"` со следующим содержимым:

```
#!/bin/bash

# Exit early on errors
set -eu
```

```
# Python buffers stdout. Without this, you won't see what you "print" in the Activity Logs
export PYTHONUNBUFFERED=true

# Install Python 3 virtual env
VIRTUALENV=./venv

if [ ! -d $VIRTUALENV ]; then
    python3 -m venv $VIRTUALENV
fi

# Install pip into virtual environment
if [ ! -f $VIRTUALENV/bin/pip ]; then
    curl --silent --show-error --retry 5 https://bootstrap.pypa.io/get-pip.py | $VIRTUALENV/bin/
fi

# Install the requirements
$VIRTUALENV/bin/pip install -r requirements.txt

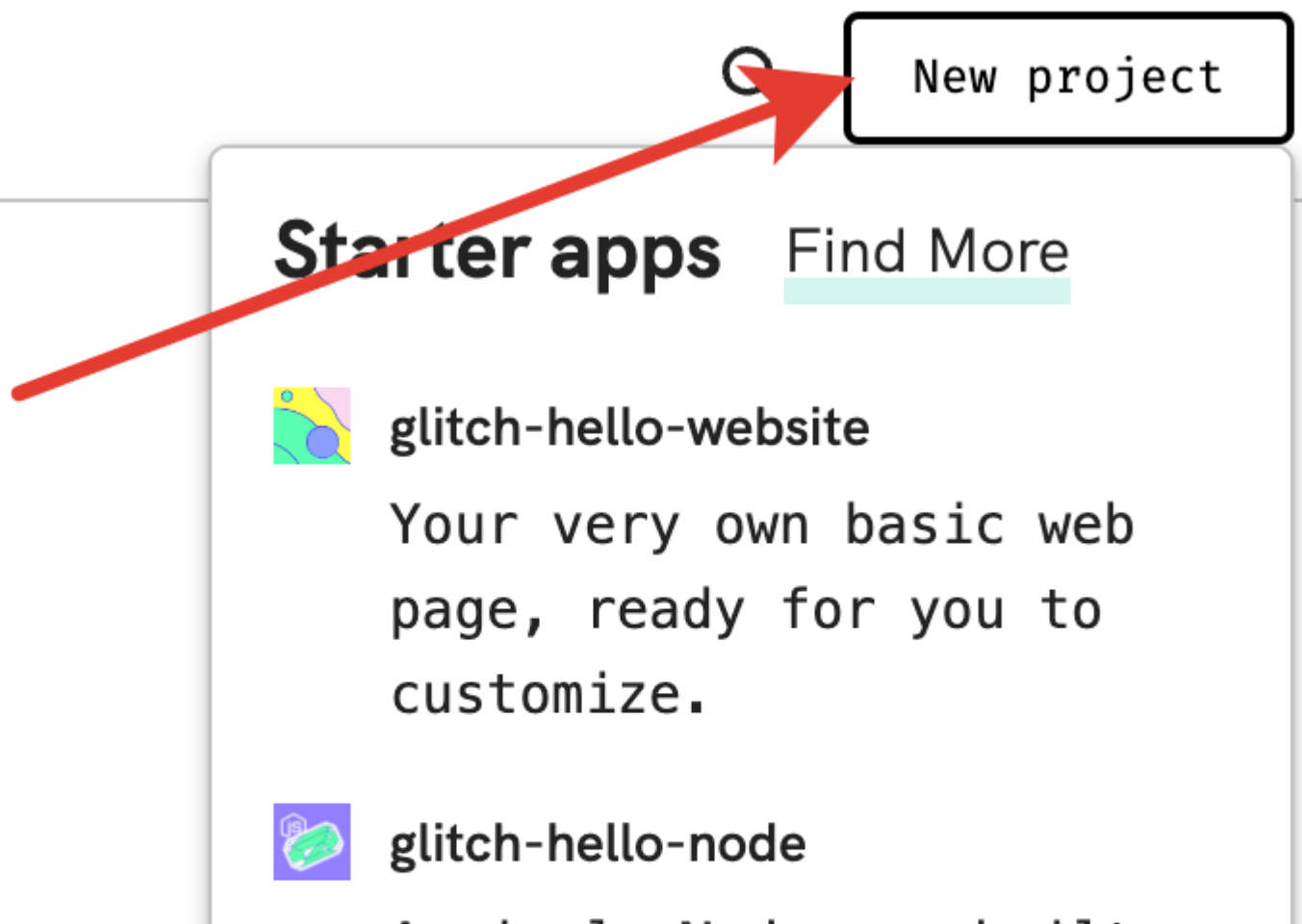
# Run your glorious application
$VIRTUALENV/bin/python3 server.py
```

Этот файл содержит скрипт, который создает виртуальное окружение python для запуска вашего приложения, устанавливает необходимые библиотеки, и запускает его.

5. Создайте git репозиторий и разместите его на github. Репозиторий надо сделать публичным.

Теперь ваше приложение готово к публикации на сервисе **Glitch**.

На верхней панели справа нажмите на кнопку "New project", затем надо выбрать "Import from Github", в появившееся диалоговое окно вставьте полный путь до вашего репозитория с приложением.



A simple Node app built with Fastify, instantly up and running.



glitch-hello-react

Get started with a new React project on Glitch!




glitch-hello-eleventy

Build a new Eleventy blog on Glitch!



glitch-hello-sqlite

Use a persistent SQLite database with your Node.js app.



Import from GitHub



STATIC SITE

После этого все файлы вашего приложения будут загружены на сервер Glitch, будет запущен скрипт `start.sh`, и примерно через 3-5 минут ваше приложение будет работать. Об этом будет свидетельствовать смайлик на панели внизу экрана рядом с надписью **Status**.



LOGS



TERMINAL



TOOLS



PREVIEW

В этой же панели нажмите на кнопку **Preview** далее **Preview in new window**, и у ваше приложение откроется в

новом окне.

Все. Можно переходить по адресу `имя_приложения.glitch.me` и наслаждаться результатом. В настройках приложения вы можете поменять его имя.

Если вам понадобится изменить код приложения, то необходимо будет удалить (поместить в архив) старое приложение и сделать новое, загрузив с github новый код.

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

Справка

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

[Пользовательское соглашение.](#)

© 2018 – 2024 ООО «Яндекс»