

Урок 8. Выполняем развертывание проекта

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc100670143)

[ВВЕДЕНИЕ 3](#_Toc100670144)

[ВЫПОЛНЯЕМ ПОШАГОВОЕ РАЗВЕРТЫВАНИЕ 3](#_Toc100670145)

[регистрируемся на heroku 3](#_Toc100670146)

[готовим проект к развертыванию 3](#_Toc100670147)

[инициализируем локальный и удаленный репозитории 5](#_Toc100670148)

[связываем локальный репозиторий с удаленным 6](#_Toc100670149)

[загружаем проект на GitHub 7](#_Toc100670150)

[создаем приложение в Heroku 10](#_Toc100670151)

[ЗАКЛЮЧЕНИЕ 13](#_Toc100670152)



# ВВЕДЕНИЕ

Мы подошли к очень важному этапу – развертыванию проекта в сети. Как и любое веб-приложение, проект на Flask также можно сделать доступным всем пользователям. Мы вновь воспользуемся возможностями сервиса Heroku, выполним пошаговое развертывание от момента подготовки проекта до размещения исходного кода на сервис, отладку и устранение ошибок.

# ВЫПОЛНЯЕМ ПОШАГОВОЕ РАЗВЕРТЫВАНИЕ

## регистрируемся на heroku

Процедура регистрации на сервисе не отличается сложностью. Она занимает несколько минут и предполагает выбор бесплатного аккаунта, возможностей которого достаточно для размещения нескольких проектов. Для получения большего количества ресурсов необходимо оплатить один из продвинутых аккаунтов.

Ссылка на сервис: <https://www.heroku.com>.

## готовим проект к развертыванию

Данная процедура предполагает создание двух служебных файлов: Procfile и requirements.txt.

Первый файл разместим в корне проект и добавим в него объявление процессов, выполняющихся в процессе работы проекта на сервере.

В этом файле разместим всего одно выражение:

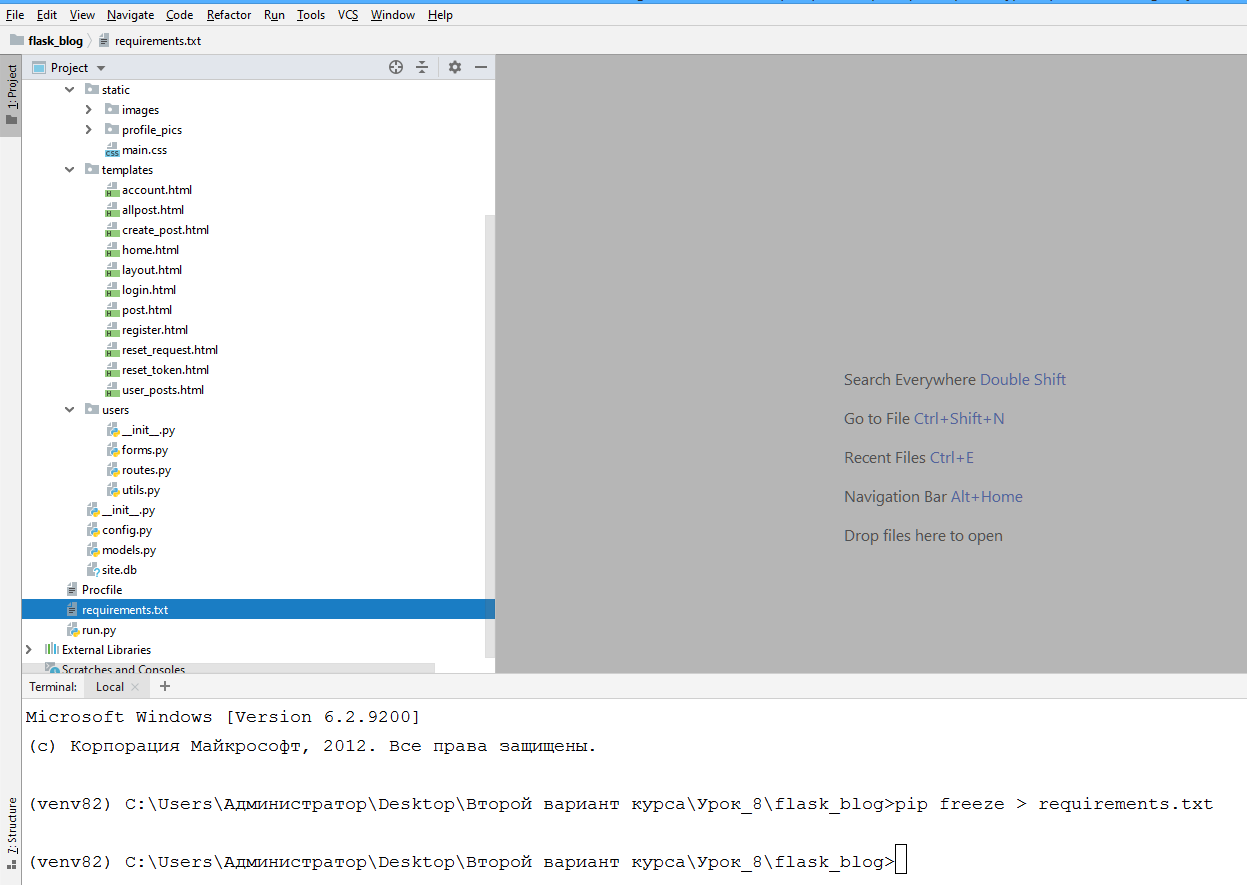
web: gunicorn run:app

Мы указываем Heroku, что работоспособность проекта будет обеспечиваться за счет WSGI-коннектора Heroku. Также указываем запускаемый файл run и объект нашего приложения app.

Также добавим файл с зависимостями requirements.txt, в котором мы перечислим список дополнительных библиотек, необходимых нашему проекту для корректной работы.

Этот файл можно составить самостоятельно, а также сгенерировать командой

**pip freeze > requirements.txt**

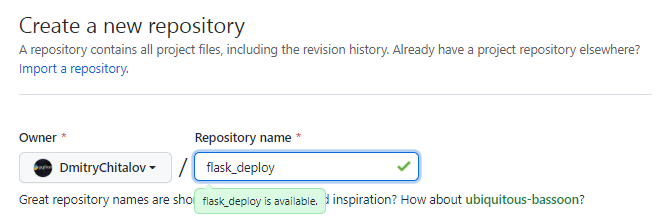
Воспользуемся вторым вариантом. Откроем проект через Пичарм и в терминале выполним данную команду.

Требуемый файл готов.

## инициализируем локальный и удаленный репозитории

Сервис Heroku будет «подтягивать» код нашего проекта с репозитория хостинга IT-проектов GitHub. Значит необходимо создать удаленный репозиторий и загрузить все компоненты проекта.

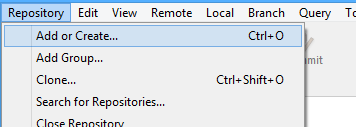
Перейдем на GitHub и создадим пустой репозиторий:

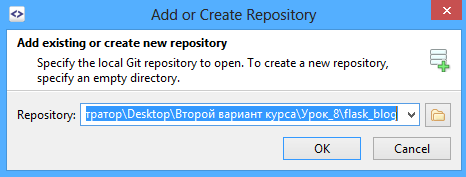
Для загрузки компонентов проекта воспользуемся программой с графическим интерфейсом SmartGit.

Для ее установки необходимо перейти по ссылке:

https://www.syntevo.com/smartgit/download/

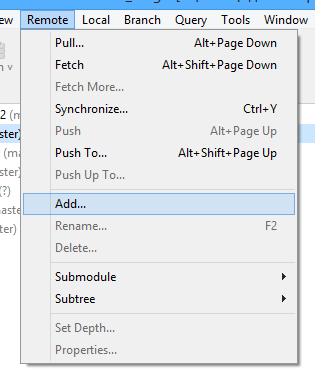
Далее загрузить установочный файл и выполнить несложную процедуру установки.

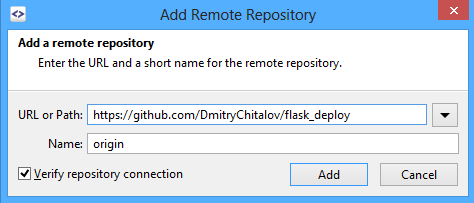
Определим директорию с компонентами проекта в качестве локального репозитория.

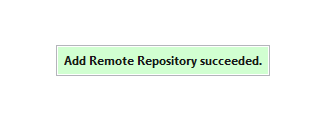


## связываем локальный репозиторий с удаленным

Теперь «свяжем» удаленный репозиторий с локальным.

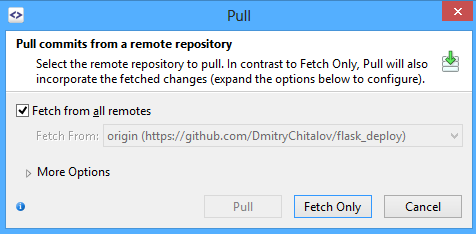




Результат:

Теперь нам нужно подтянуть некоторые важные настройки из удаленного репозитория.

Для этого выполняем команду Pull.



Подтверждаем и все готово.

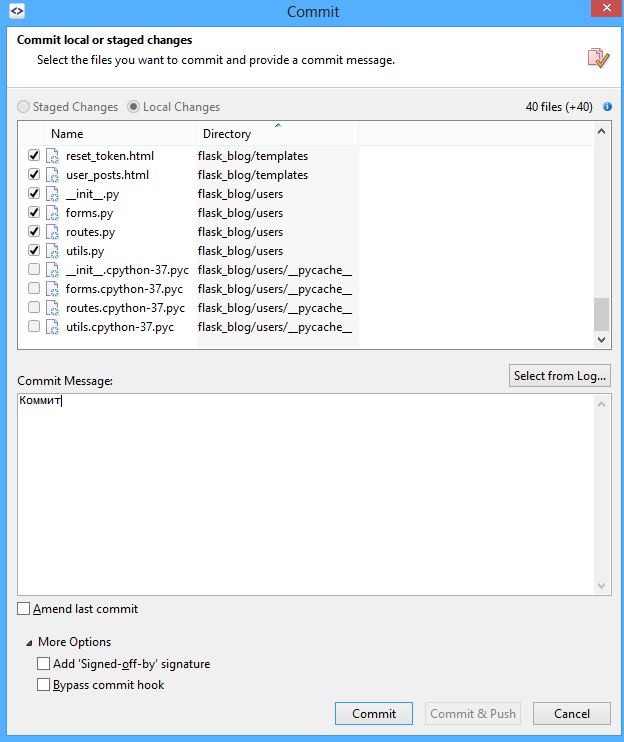
## загружаем проект на GitHub

Теперь выполним команды «коммит» и «пуш», отвечающие за фиксацию изменений в нашем проекте и отправке обновленного кода проекта в удаленный репозиторий.

Эти команды необходимо выполнять всегда после внесения доработок в код проекта.



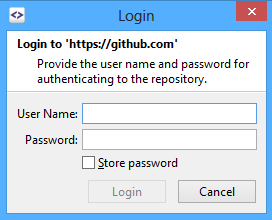
Теперь выбираем компоненты, которые мы хотим загрузить в удаленный репозиторий:



Подтверждаем коммит и выполняем команду «пуш».

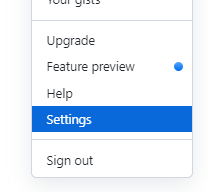


Здесь нам предложат указать имя пользователя и пароль:

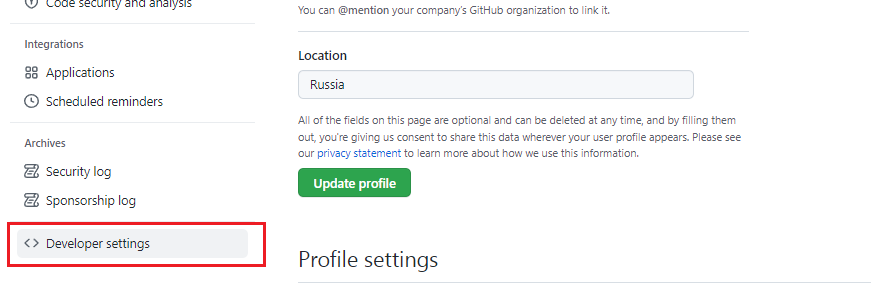


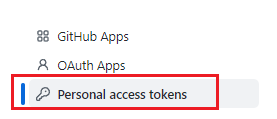
Здесь нужно указать ваши параметры, указанные при регистрации в GitHub.

Первый параметр – имя пользователя, второй – не пароль, а токен. Только где его получить?

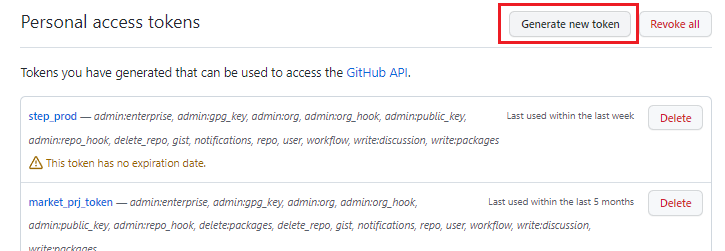
Нужно перейти на GitHub в раздел Settings.

Перейдем в раздел настроек разработчика.

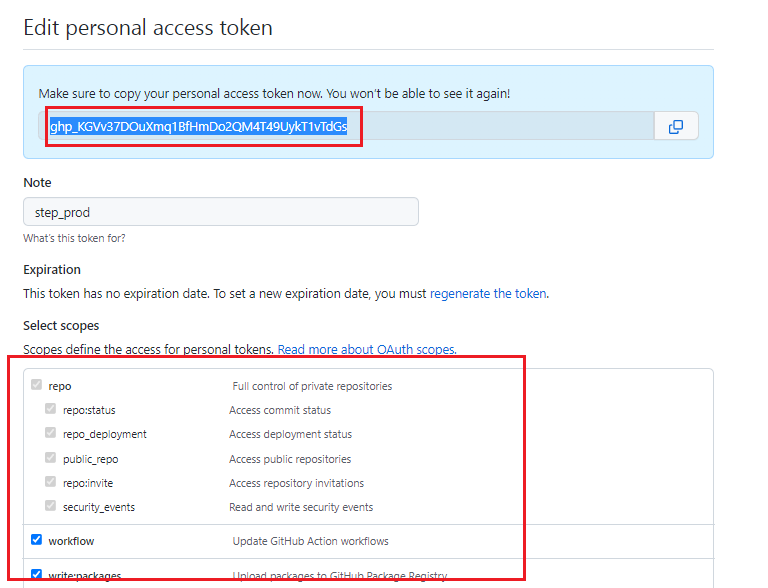
Перейдем в раздел токенов.



Здесь нужно сгенерировать новый токен и указать срок его действия.

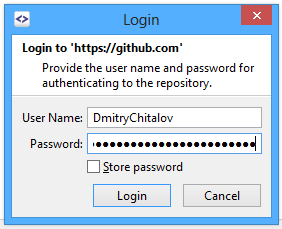


Кроме того, укажем операции, на которые токен будет распространяться.

Токен нужно обязательно где-то сохранить, иначе его придется генерировать повторно.

Теперь перейдем к команде «пуш», которую мы не завершили.

Укажем имя пользователя и токен.



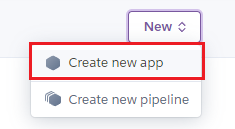
Теперь вновь выполним команду «пуш».



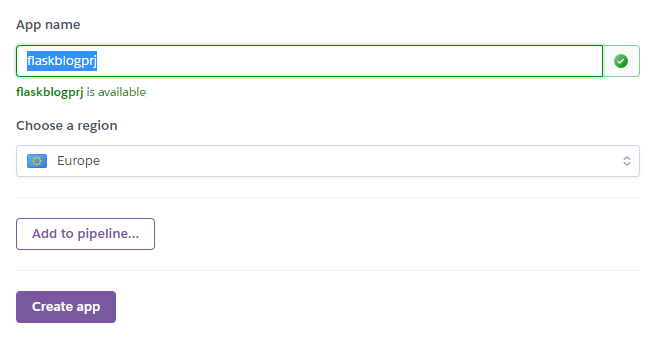
В течение нескольких секунд изменения будут загружены в удаленный репозиторий GitHub.

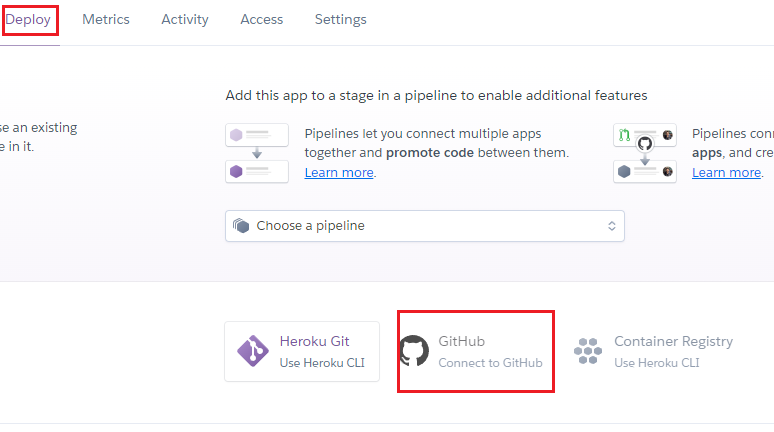
Убедимся, что удаленный репозиторий обновился:

## создаем приложение в Heroku

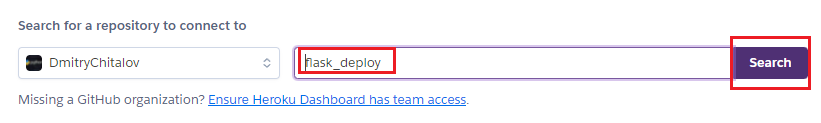
Перейдем по ссылке <https://dashboard.heroku.com/apps> и создадим новое приложение:

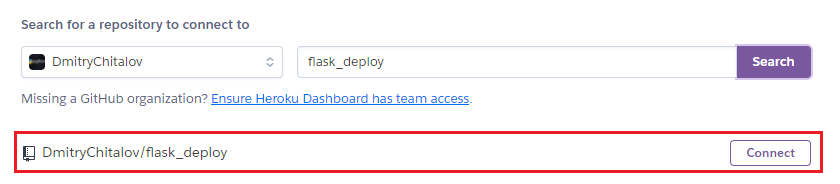
Укажем параметры проекта:

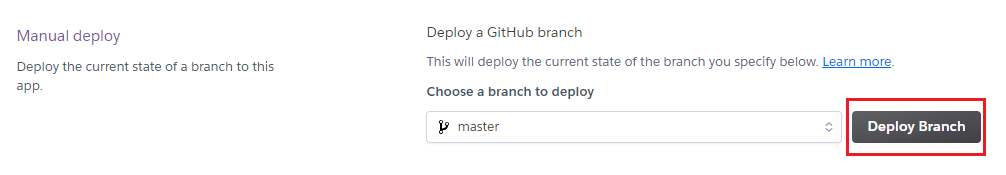


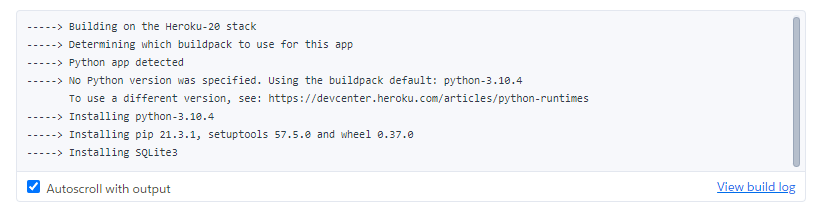
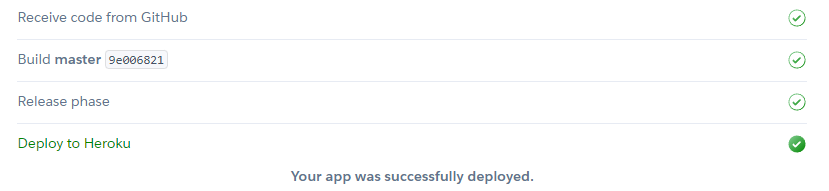
Перейдем в раздел Deploy на вкладку Connect to GitHub.

Укажем название удаленного репозитория.

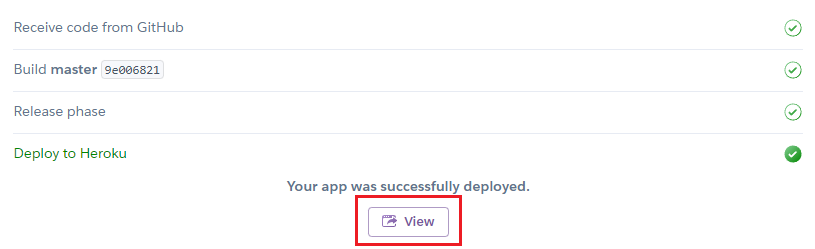
Подключаемся к репозиторию.

Далее переходим к кнопке Branch.

Ждем завершения процесса развертывания.



При успешном завершении развертывания:

Перейдем к просмотру содержимого нашего веб-приложения:

Все работает:

# Вид сверху на рукопожатие над деловыми документамиЗАКЛЮЧЕНИЕ

Мы завершили очередной урок курса и успешно выполнили пошаговый процесс развертывания. Самый сложный шаг – это понимание особенностей взаимодействия с приложением SmartGit. При этом оно считается одним из самых удобных и понятных средств для управления связкой локального и удаленного репозитория. Мы подготовили проект к развертыванию, добавив необходимые служебные файлы, загрузили код проекта на сервис GitHub и выполнили его размещение на сервисе Heroku.