Бесплатная электронная книга

учусь heroku

Free unaffiliated eBook created from
Stack Overflow contributors.

#heroku

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: heroku

It is an unofficial and free heroku ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official heroku.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с герокой

замечания

Heroku - популярный поставщик услуг «Платформа как услуга» (PaaS), который позволяет разработчикам легко развертывать веб-приложения без команды операций. Heroku существует с 2007 года и теперь принадлежит Salesforce .

В этом разделе представлен обзор того, что такое Heroku, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в Хероку и ссылки на соответствующие темы. Поскольку документация для Heroku является новой, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Чтобы создавать и управлять приложениями Heroku локально, вам понадобится Heroku Toolbelt, вот некоторые способы его получить.

Скачать

Загрузите установщик Heroku Toolbelt с веб-сайта Heroku.

Homebrew

Установите `heroku` с `brew` :

```
brew install heroku
```

Debian / Ubuntu

Запустите этот скрипт:

```
wget -O- https://toolbelt.heroku.com/install-ubuntu.sh | sh
```

Этот скрипт добавляет репозиторий Heroku в apt, устанавливает ключ выпуска Heroku,

устанавливает Heroku Toolbelt и затем устанавливает Ruby, если вам это нужно.

Как и в случае с любым скриптом, который вы найдете в Интернете и напрямую подключаетесь к bash, мы настоятельно рекомендуем вам сначала прочитать источник .

Использование набора инструментов Heroku Toolbelt

Создание приложения

```
heroku create your-app-name
```

Развертывание в Хероку

```
git push heroku master
```

Откройте приложение в браузере

```
heroku open your-app-name
```

Список команд Heroku

```
heroku commands
```

Общая помощь

```
heroku help
```

Справка для конкретной команды

```
heroku help <command>
```

Создание приложений Heroku

Вы можете использовать команду `heroku create` для создания приложения Heroku. Каждое приложение, которое вы развертываете в Heroku, имеет свою собственную базу кода,

переменные среды, аддоны и т. Д.

Каждое приложение Heroku имеет глобально уникальное имя. Если вы попытаетесь создать приложение Heroku, имя которого уже принято, вы получите сообщение об ошибке.

Вот как вы можете создать новое приложение Heroku:

```
heroku create [app_name]
```

Если вы не укажете имя приложения при запуске `heroku create` , Heroku создаст для вас произвольное имя приложения.

Вы также можете указать регион Amazon, в котором должно быть создано ваше приложение Heroku. По умолчанию все приложения Heroku создаются в `us` регионе. Если вы хотите изменить регион, вы можете сделать это, создав приложение следующим образом:

```
heroku create [app_name] --region eu
```

Сейчас есть только два общественных региона: `us` и `eu` (Европа).

Прочитайте Начало работы с герокой онлайн: https://riptutorial.com/ru/heroku/topic/959/начало-работы-с-герокой

глава 2: Buildpack

Examples

Настройка Buildpacks

Heroku официально поддерживает buildpacks для Ruby, Node.js, Clojure, Python, Java, Gradle, Grails, Scala, Play, PHP и Go.

Buildpacks автоматически обнаруживаются Heroku в вышеуказанном порядке, однако его также можно установить вручную через CLI, используя:

1. Во время создания приложения

```
heroku create <app_name> --buildpack <buildpack_name>
```

2. Вручную,

```
heroku buildpacks:set <buildpack_name>
```

Имя Buildpack можно указать либо с сокращением, либо по URL. Как и для PHP buildpack,

```
heroku buildpacks:set heroku/php
```

или же

```
heroku buildpacks:set https://elements.heroku.com/buildpacks/heroku/heroku-buildpack-php
```

Многострочные пакеты

Приложение также может содержать более одного buildpack. Это может быть достигнуто с помощью `add` :

```
heroku buildpacks:add --index 1 <buildpack_name>
```

где параметр `--index` указывает порядок выполнения buildpack.

Сказать,

```
heroku buildpacks:set heroku/php
heroku buildpacks:add --index 1 heroku/nodejs
```

будет установлен порядок buildpack как:

```
heroku/nodejs
heroku/php
```

Помните: приложение Heroku имеет только один открытый порт - 80. Следовательно, любой из них будет обслуживать в одном порту. Скажем, если `procfile` указан с помощью `web: node server.js` , приложение-узел будет запущено в порт 80, иначе PHP. Однако сборка будет выполняться в указанном порядке. Если требуется более одного приложения, настройте несколько проектов и сделайте их для связи друг с другом.

Прочитайте Buildpack онлайн: https://riptutorial.com/ru/heroku/topic/6126/buildpack

глава 3: Heroku node.js Привет, мир

замечания

авторизоваться

`heroku login`

создать приложение

`heroku create` или `heroku create your_name`

клонировать пример

```
git clone https://github.com/zoutepopcorn/herokuworld
cd herokuworld
```

посетить приложение в своем браузере

```
https://your_name.herokuapp.com/
```

Дополнительно проверяйте его локально:

```
heroku local web
```

проверить: lolhost: 5000

Так что же отличается от обычного приложения node.js? package.json

```
"scripts": {
  "start": "node index.js"
},
"engines": {
  "node": "7.6.0"
}
```

index.js

```
process.env.PORT
```

Локальный порт: 5000. Heroku отобразит его в порт 80 на вашем URL-адресе приложения.

Examples

Heroku node.js привет мир

index.js

```
var http = require("http");

http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Heroku world!");
  response.end();
}).listen(process.env.PORT);
```

package.json

```
{
  "name": "node-example",
  "version": "1.0.0",
  "description": "Hello world Heroku",
  "scripts": {
    "start": "node index.js"
  },

  "keywords": [
    "example",
    "heroku"
  ],
  "author": "Johan",
  "license": "MIT",
  "engines": {
    "node": "7.6.0"
  }
}
```

Прочитайте Heroku node.js Привет, мир онлайн:
https://riptutorial.com/ru/heroku/topic/9897/heroku-node-js-привет--мир

глава 4: Heroku Postgres

Examples

Как восстановить базу данных Postgres в Heroku

Шаги по восстановлению базы данных в Heroku:

1. Отбросьте базу данных, когда используется SHARED_DATABASE_URL:

```
heroku pg:reset DATABASE
```

2. Восстановить базу данных без ничего:

```
heroku run rake db:migrate
```

3. Заполните базу данных вашими исходными данными:

```
heroku run rake db:seed
```

Шаги 2 и 3 можно объединить в одну команду, выполнив следующее:

```
heroku run rake db:setup
```

Как скопировать базу данных heroku в локальную базу данных

Шаги по копированию базы данных heroku в локальную базу данных:

1. Запустите процесс копирования в терминале:

```
heroku pg:pull DATABASE_URL change_to_your_data_base_name --app change_to_your_app_name
```

2. Измените владельца db с помощью этого запроса:

```
GRANT ALL PRIVILEGES ON DATABASE change_to_your_data_base_name to change_to_your_user; ALTER
DATABASE change_to_your_data_base_name OWNER TO change_to_your_user;
```

3. Создайте и запустите запрос для всех таблиц в вашей базе данных:

```
SELECT 'ALTER TABLE '|| schemaname || '.' || tablename ||' OWNER TO change_to_your_user;' FROM
pg_tables WHERE NOT schemaname IN ('pg_catalog', 'information_schema') ORDER BY schemaname,
tablename;
```

Прочитайте Heroku Postgres онлайн: https://riptutorial.com/ru/heroku/topic/6239/heroku-postgres

глава 5: бревна

Синтаксис

- $ heroku logs
- $ heroku logs -n 200
- $ heroku logs --tail
- $ heroku logs --dyno router
- $ heroku logs --source приложение
- $ heroku logs --source app --dyno worker
- $ heroku logs --source app --tail

Examples

Типы журналов

Heroku объединяет три категории журналов для вашего приложения:

- **Журналы приложений** - вывод из приложения. Это будет включать журналы, созданные из вашего приложения, сервера приложений и библиотек. (Фильтр: `--source app`)
- **Системные журналы** - сообщения о действиях, предпринимаемых инфраструктурой платформы Heroku от имени вашего приложения, такие как перезапуск разбитого процесса, спящий или пробуждающий веб-дино или работа с страницей ошибок из-за проблемы в вашем приложении. (Фильтр: `--source heroku`)
- **Журналы API** - сообщения об административных действиях, предпринятых вами и другими разработчиками, работающими над вашим приложением, такие как: развертывание нового кода, масштабирование процесса формирования или переключение режима обслуживания. (Фильтр: `--source heroku --dyno api`)

Формат журнала

Каждая строка журнала форматируется следующим образом:

```
timestamp source[dyno]: message
```

- **Временная метка** - дата и время, записанные в момент, когда линия журнала была создана динамическим или компонентом. Временная метка находится в формате, указанном RFC5424, и включает в себя микросекундную точность.
- **Источник.** Все динамики вашего приложения (веб-динамики, фоновые работники,

cron) имеют источник, `app` . Все компоненты системы Heroku (HTTP-маршрутизатор, динамический менеджер) имеют источник, `heroku` .

- **Dyno** - имя динамического или компонента, который написал строку журнала. Например, рабочий № 3 отображается как `worker.3` , а HTTP-маршрутизатор Heroku отображается как `router` .
- **Сообщение** - содержимое строки журнала. Линии, генерируемые динамиками, которые превышают 10000 байт, разбиваются на 10000 байтовых блоков без лишних трейлинговых строк. Каждый фрагмент представляется в виде отдельной строки журнала.

Просмотр журналов

Чтобы получить ваши журналы, используйте команду `heroku logs` .

```
$ heroku logs
```

Команда logs извлекает 100 строк журнала по умолчанию. Вы можете указать количество строк журнала для извлечения (максимум до 1500 строк) с помощью опции `--num` (или `-n`).

```
$ heroku logs -n 200
```

Хвост в реальном времени

Подобно `tail -f` , `tail -f` реальном времени отображает последние журналы и оставляет сессию открытой для журналов в реальном времени. В результате просмотра прямого потока журналов из вашего приложения вы можете получить представление о поведении вашего реального приложения и отладить текущие проблемы. Вы можете `--tail` свои журналы с помощью `--tail` (или `-t`).

```
$ heroku logs --tail
```

Когда вы закончите, нажмите Ctrl + C, чтобы вернуться в приглашение.

Фильтрация журналов

Если вы хотите только получать журналы с определенным источником, определенным динамо или обоими, вы можете использовать `--source` (или `-s`) и `--dyno` (или `-d`):

```
$ heroku logs --dyno router
2012-02-07T09:43:06.123456+00:00 heroku[router]: at=info method=GET path="/stylesheets/dev-
center/library.css" host=devcenter.heroku.com fwd="204.204.204.204" dyno=web.5 connect=1ms
service=18ms status=200 bytes=13
```

```
2012-02-07T09:43:06.123456+00:00 heroku[router]: at=info method=GET path="/articles/bundler"
host=devcenter.heroku.com fwd="204.204.204.204" dyno=web.6 connect=1ms service=18ms status=200
bytes=20375

$ heroku logs --source app
2012-02-07T09:45:47.123456+00:00 app[web.1]: Rendered shared/_search.html.erb (1.0ms)
2012-02-07T09:45:47.123456+00:00 app[web.1]: Completed 200 OK in 83ms (Views: 48.7ms |
ActiveRecord: 32.2ms)
2012-02-07T09:45:47.123456+00:00 app[worker.1]: [Worker(host:465cf64e-61c8-46d3-b480-
362bfd4ecff9 pid:1)] 1 jobs processed at 23.0330 j/s, 0 failed ...
2012-02-07T09:46:01.123456+00:00 app[web.6]: Started GET "/articles/buildpacks" for 4.1.81.209
at 2012-02-07 09:46:01 +0000

$ heroku logs --source app --dyno worker
2012-02-07T09:47:59.123456+00:00 app[worker.1]: [Worker(host:260cf64e-61c8-46d3-b480-
362bfd4ecff9 pid:1)] Article#record_view_without_delay completed after 0.0221
2012-02-07T09:47:59.123456+00:00 app[worker.1]: [Worker(host:260cf64e-61c8-46d3-b480-
362bfd4ecff9 pid:1)] 5 jobs processed at 31.6842 j/s, 0 failed ...
```

Вы также можете комбинировать фильтрующие переключатели с `--tail` чтобы получить поток фильтрованного вывода в режиме реального времени.

```
$ heroku logs --source app --tail
```

Прочитайте бревна онлайн: https://riptutorial.com/ru/heroku/topic/8327/бревна

глава 6: Дополнения к Ероку

Вступление

Подробности и инструкции по использованию различных дополнений, доступных в Heroku.

Examples

Планировщик Heroku

Установка планировщика Heroku

```
heroku addons:create scheduler:standard
```

Прочитайте Дополнения к Ероку онлайн: https://riptutorial.com/ru/heroku/topic/8906/дополнения-к-ероку

глава 7: зависимости

Синтаксис

- "зависимости": {...}

Examples

Зависимость

Для автоматической установки беседки и ее компонентов необходимо

1. Задайте зависимость `package.json` в `package.json` :

```
"dependencies": {
    "bower": "^1.7.9"
}
```

2. Использовать `scripts` для выполнения команды `postinstall`

```
"scripts": {
    "postinstall": "./node_modules/bower/bin/bower install"
}
```

3. Создайте файл `.bowerrc` чтобы установить каталог для установки bower_components. В противном случае параметры bower_components устанавливаются в корневом каталоге.

```
{
    "directory" : "app/bower_components"
}
```

Теперь, Heroku автоматически выполняет `bower install` команду после `npm install`

Прочитайте зависимости онлайн: https://riptutorial.com/ru/heroku/topic/6665/зависимости

глава 8: Коды ошибок Heroku

Вступление

Всякий раз, когда ваше приложение испытывает ошибку, Heroku вернет стандартную страницу с кодом HTTP status status 503. Однако, чтобы помочь вам отлаживать базовую ошибку, платформа также добавит пользовательскую информацию об ошибках в ваши журналы. Каждый тип ошибки получает свой собственный код ошибки, со всеми ошибками HTTP, начинающимися с буквы H и всех ошибок времени выполнения, начиная с R. Ошибки регистрации начинаются с L.

Синтаксис

- H10 - приложение разбилось
- H11 - Задержка слишком глубокая
- H12 - Тайм-аут запроса
- H13 - Соединение закрыто без ответа
- H14 - не работает веб-динамик
- H15 - Недействующее соединение
- H16 - Перенаправление на herokuapp.com
- H17 - плохо отформатированный HTTP-ответ
- H18 - Прерывание запроса сервера
- H19 - время ожидания подключения к серверу
- H20 - время ожидания загрузки приложения
- H21 - соединение с Backend отказано
- H22 - Достигнут предел соединения
- H23 - Конечная точка неправильно сконфигурирована
- H24 - принудительное закрытие
- H25 - Ограничение HTTP
- H26 - Ошибка запроса
- H27 - Прерванный запрос клиента
- H28 - Неисправность подключения клиента
- H80 - Режим обслуживания
- Приложение H81 - Blank
- H82 - освобожденная динамо квота
- H99 - Ошибка платформы
- R10 - время ожидания загрузки
- R12 - Тайм-аут выхода
- R13 - Приложить ошибку
- R14 - превышена квота памяти
- R15 - квота памяти значительно превышена

- R16 - Отдельно
- R17 - ошибка контрольной суммы
- R99 - Ошибка платформы
- L10 - переполнение сливного буфера
- L11 - Переполнение буфера хвоста
- L12 - Переполнение локального буфера
- L13 - Локальная ошибка доставки
- L14 - Ошибка проверки сертификата

Examples

H10 - приложение разбилось

Эта ошибка вызывается разбитым веб-дино или тайм-аутом загрузки на веб-дино.

```
2010-10-06T21:51:04-07:00 heroku[web.1]: State changed from down to starting
2010-10-06T21:51:07-07:00 app[web.1]: Starting process with command: `bundle exec rails server
-p 22020`
2010-10-06T21:51:09-07:00 app[web.1]: >> Using rails adapter
2010-10-06T21:51:09-07:00 app[web.1]: Missing the Rails 2.3.5 gem. Please `gem install -
v=2.3.5 rails`, update your RAILS_GEM_VERSION setting in config/environment.rb for the Rails
version you do have installed, or comment out RAILS_GEM_VERSION to use the latest version
installed.
2010-10-06T21:51:10-07:00 heroku[web.1]: Process exited
2010-10-06T21:51:12-07:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET
path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno= connect= service= status=503 bytes=
```

H11 - Задержка слишком глубокая

Когда HTTP-запросы поступают быстрее, чем ваше приложение может их обрабатывать, они могут сформировать большое отставание на нескольких маршрутизаторах. Когда отставание на определенном маршрутизаторе проходит порог, маршрутизатор определяет, что ваше приложение не поддерживает свой входящий объем запросов. Вы увидите ошибку H11 для каждого входящего запроса, если отставание превышает этот размер. Точное значение этого порога может меняться в зависимости от различных факторов, таких как количество динамиков в вашем приложении, время отклика для отдельных запросов и обычный объем запросов вашего приложения.

```
2010-10-06T21:51:07-07:00 heroku[router]: at=error code=H11 desc="Backlog too deep" method=GET
path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno= connect= service= status=503 bytes=
```

Решение состоит в том, чтобы увеличить пропускную способность вашего приложения, добавив больше динамиков, настроив вашу базу данных (например, добавив индекс) или сделав код сам быстрее. Как всегда, увеличение производительности очень специфично для приложений и требует профилирования.

H12 - Тайм-аут запроса

HTTP-запрос занял более 30 секунд. В приведенном ниже примере приложение Rails занимает 37 секунд, чтобы отобразить страницу; HTTP-маршрутизатор возвращает 503 до того, как Rails завершит цикл запроса, но процесс Rails продолжается, и сообщение завершения отображается после сообщения маршрутизатора.

```
2010-10-06T21:51:07-07:00 app[web.2]: Processing PostController#list (for 75.36.147.245 at
2010-10-06 21:51:07) [GET]
2010-10-06T21:51:08-07:00 app[web.2]: Rendering template within layouts/application
2010-10-06T21:51:19-07:00 app[web.2]: Rendering post/list
2010-10-06T21:51:37-07:00 heroku[router]: at=error code=H12 desc="Request timeout" method=GET
path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno=web.1 connect=6ms service=30001ms
status=503 bytes=0
2010-10-06T21:51:42-07:00 app[web.2]: Completed in 37000ms (View: 27, DB: 21) | 200 OK
[http://myapp.heroku.com/]
```

Этот 30-секундный предел измеряется маршрутизатором и включает в себя все время, проведенное в dyno, включая очередь входящих соединений ядра и само приложение.

H13 - Соединение закрыто без ответа

Эта ошибка возникает, когда процесс в вашем веб-дино принимает соединение, но затем закрывает сокет, ничего не записывая.

```
2010-10-06T21:51:37-07:00 heroku[router]: at=error code=H13 desc="Connection closed without
response" method=GET path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno=web.1
connect=3030ms service=9767ms status=503 bytes=0
```

Один пример, когда это может произойти, - это когда веб-сервер Unicorn настроен с тайм-аутом короче 30 секунд, и запрос не был обработан рабочим до истечения таймаута. В этом случае Unicorn закрывает соединение до того, как будут записаны какие-либо данные, в результате получится H13.

H14 - не работает веб-динамик

Это, скорее всего, результат масштабирования ваших веб-динодов до 0 динозавров. Чтобы исправить это, масштабируйте свои веб-динамики до 1 или более динамиков:

```
$ heroku ps:scale web=1
```

Используйте команду heroku ps для определения состояния ваших веб-динамиков.

```
2010-10-06T21:51:37-07:00 heroku[router]: at=error code=H14 desc="No web processes running"
method=GET path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno= connect= service=
status=503 bytes=
```

H15 - Недействующее соединение

Дино не отправил полный ответ и был прекращен из-за 55 секунд бездействия. Например, ответ показал `Content-Length` 50 байтов, которые не были отправлены вовремя.

```
2010-10-06T21:51:37-07:00 heroku[router]: at=error code=H15 desc="Idle connection" method=GET
path="/" host=myapp.herokuapp.com fwd=17.17.17.17 dyno=web.1 connect=1ms service=55449ms
status=503 bytes=18
```

Прочитайте Коды ошибок Heroku онлайн: https://riptutorial.com/ru/heroku/topic/8321/коды-ошибок-heroku

глава 9: Командная строка

Вступление

Интерфейс командной строки Heroku (CLI), ранее известный как Heroku Toolbelt, является инструментом для создания и управления приложениями Heroku из командной строки / оболочки различных операционных систем.

Синтаксис

- $ heroku --версия
- $ heroku login
- $ heroku создать

Examples

Загрузить и установить

OS X

Загрузите и запустите установщик OS X.

Windows

Загрузите и запустите установщик Windows 32-разрядный 64-разрядный .

Debian / Ubuntu

Чтобы добавить наш apt-репозиторий и установить CLI, выполните следующие действия:

```
$ sudo add-apt-repository "deb https://cli-assets.heroku.com/branches/stable/apt ./"
$ curl -L https://cli-assets.heroku.com/apt/release.key | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install heroku
```

Автономная версия

Загрузите tarball и извлеките его, чтобы вы могли получить доступ к двоичному файлу из

своего PATH. Например:

```
$ echo replace OS/ARCH with values as noted below
$ wget https://cli-assets.heroku.com/branches/stable/heroku-OS-ARCH.tar.gz
$ tar -xvzf heroku-OS-ARCH /usr/local/lib/heroku
$ ln -s /usr/local/lib/heroku/bin/heroku /usr/local/bin/heroku
```

Проверьте свою установку

Чтобы проверить установку CLI, используйте команду `heroku --version` .

```
$ heroku --version
heroku-cli/5.6.0-010a227 (darwin-amd64) go1.7.4
```

Начиная

Вам будет предложено ввести ваши учетные данные Heroku при первом запуске команды; после первого раза ваш адрес электронной почты и токен API будут сохранены в `~/.netrc` для будущего использования.

Как правило, рекомендуется войти в систему и добавить свой открытый ключ сразу после установки CLI Heroku, чтобы вы могли использовать git для push или клонирования репозиториев приложений Heroku:

```
$ heroku login
Enter your Heroku credentials.
Email: adam@example.com
Password (typing will be hidden):
Authentication successful.
```

Теперь вы готовы создать свое первое приложение Heroku:

```
$ cd ~/myapp
$ heroku create
Creating app... done, ⬡ sleepy-meadow-81798
https://sleepy-meadow-81798.herokuapp.com/ | https://git.heroku.com/sleepy-meadow-81798.git
```

Прочитайте Командная строка онлайн: https://riptutorial.com/ru/heroku/topic/8324/командная-строка

глава 10: Пределы Хероку

Examples

Список всех ограничений на платформе Heroku

1. Журналы. По умолчанию Heroku позволяет использовать только 1500 строк сводных журналов. Когда требуется более 1500 строк журналов, нужно использовать аддоны, предоставленные Heroku.

2. Маршрутизатор: HTTP-запрос имеет 30-секундный тайм-аут для первоначального ответа и тайм-аут 55 с после этого. Максимальный буфер 1 МБ допускается для ответа.

3. Dynos: *ограничения памяти* Dyno на основе выбранного типа . Для бесплатных динозавров, часы сна накладываются там, где он спит после 30 минут бездействия. Кроме того, проверенные учетные записи поставляются с ежемесячным пулом из 1000 бесплатных динозанов, а непроверенные счета получают 550. Приложение может иметь до 100 динамиков, а *тип процесса* не может масштабироваться до более чем 10 динамиков. Свободный тип dyno может иметь максимум два одновременных динамика.

4. Config Vars: пара ключей и значений Config ограничена 32kb для приложения.

5. Сборка: пользователи ограничены 75 запросами к репозиториям Heroku Git за час, за приложение, за пользователя. Несжатый размер во время проверки не может превышать 1 ГБ. Размер слизи ограничивается 300 МБ, а длина компиляции не может превышать 15 минут.

6. Клипы данных. Каждый запрос может работать не более 10 минут и может возвращать максимум 100 000 строк.

7. Heroku Postgres: время простоя варьируется с разными уровнями от менее 4 часов до 15 минут в месяц.

8. Ограничения API. Максимальные вызовы API Heroku ограничены 2400 часами.

9. Пределы членства: для учетной записи предприятия, не более 500 членов и для других, допускается 25 членов.

10. Количество приложений: от проверенного пользователя может быть создано не более 100 приложений. Непроверенные пользователи ограничены 5 приложениями.

Прочитайте Пределы Хероку онлайн: https://riptutorial.com/ru/heroku/topic/6190/пределы-хероку

глава 11: развертывание

Синтаксис

- git push heroku master

Examples

Развертывание с Git

Отслеживание вашего приложения в git

Прежде чем вы сможете нажать приложение в Heroku, вам нужно будет инициализировать локальный репозиторий Git и передать ему свои файлы. Например, если у вас есть приложение в каталоге, myapp, а затем создайте для него новый репозиторий:

```
$ cd myapp
$ git init
Initialized empty Git repository in .git/
$ git add .
$ git commit -m "my first commit"
Created initial commit 5df2d09: my first commit
 44 files changed, 8393 insertions(+), 0 deletions(-)
 create mode 100644 README
 create mode 100644 Procfile
 create mode 100644 app/controllers/source_file
...
```

Это локальный репозиторий, который теперь находится внутри каталога `.git` . Ничего еще не отправлено; вам нужно создать удаленный доступ и сделать толчок для развертывания вашего кода в Heroku.

Создание пульта Heroku

```
$ heroku create
Creating falling-wind-1624... done, stack is cedar-14
http://falling-wind-1624.herokuapp.com/ | https://git.heroku.com/falling-wind-1624.git
Git remote heroku added
```

Git с существующим приложением. Команда heroku git: remote добавит этот пульт для вас на основе ваших приложений git url.

```
$ heroku git:remote -a falling-wind-1624
Git remote heroku added.
```

Развертывание кода

вам нужно указать удаленный филиал для нажатия. Вы можете сделать первый щелчок:

```
$ git push heroku master
Initializing repository, done.
updating 'refs/heads/master'
...
```

Чтобы направить ветвь, отличную от мастера, используйте этот синтаксис:

```
$ git push heroku yourbranch:master
```

Прочитайте развертывание онлайн: https://riptutorial.com/ru/heroku/topic/8325/развертывание

глава 12: Трубопроводы

Синтаксис

- Героические трубопроводы: <install | create | promotion> ...

замечания

Конвейер - это группа приложений Heroku, которые используют одну и ту же базу кода. Приложения в конвейере сгруппированы в стадии «обзор», «разработка», «стадия» и «производство», представляющие различные этапы развертывания в непрерывном рабочем процессе.

Examples

Трубопроводы через CLI

Установка трубопровода

Как только Heroku Toolbelt установлен, для него также нужен плагин Pipelines .

```
heroku plugins:install heroku-pipelines
```

Создание трубопроводов

Вы должны начать с приложения, чтобы добавить его в конвейер, хотя он не должен быть для определенного этапа. Если вы не укажете `--stage STAGE` , CLI будет угадывать на соответствующем этапе, но также позволит вам переопределить значение по умолчанию. Имя конвейера также будет угадываться из имени приложения, но может быть переопределено либо путем добавления `NAME` в командной строке, либо ввода другого имени при появлении запроса.

```
heroku pipelines:create -a example
```

Продвижение

Целевое приложение (а) будет автоматически определено на этапе ниже по течению

```
heroku pipelines:promote -r staging
```

Также можно продвигать к определенному приложению (или набору приложений)

```
heroku pipelines:promote -r staging --to my-production-app1,my-production-app2
```

Команда помощи

Полный список команд Pipelines с информацией об использовании доступен в консоли

```
heroku help pipelines
```

Прочитайте Трубопроводы онлайн: https://riptutorial.com/ru/heroku/topic/2389/трубопроводы

кредиты

S. No	Главы	Contributors
1	Начало работы с герокой	Community, rdegges, thejonanshow
2	Buildpack	Thamilan
3	Heroku node.js Привет, мир	Johan Hoeksma
4	Heroku Postgres	Denis Savchuk, Hardik Kanjariya ツ, Thamilan
5	бревна	Sender
6	Дополнения к Ероку	jophab
7	зависимости	Thamilan
8	Коды ошибок Heroku	Sender
9	Командная строка	Sender
10	Пределы Хероку	autoboxer, Thamilan
11	развертывание	Sender
12	Трубопроводы	Thamilan