eBook Gratuit

APPRENEZ heroku

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#heroku

Table des matières

À propos

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

Chapitre 1: Commencer avec Heroku

Remarques

Heroku est un fournisseur populaire de plate-forme en tant que service (PaaS) qui permet aux développeurs de déployer facilement des applications Web sans équipe opérationnelle. Heroku existe depuis 2007 et appartient désormais à Salesforce .

Cette section fournit une vue d'ensemble de ce qu'est Heroku et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de Heroku, et établir un lien avec les sujets connexes. La documentation de Heroku étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Installation ou configuration

Pour créer et gérer localement des applications Heroku, vous aurez besoin de la Toolbelt d'Heroku.

Télécharger

Téléchargez le programme d' installation de Heroku Toolbelt sur le site Web de Heroku.

Homebrew

Installez le `heroku` avec le `brew` :

```
brew install heroku
```

Debian / Ubuntu

Exécutez ce script:

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

Ce script ajoute le dépôt Heroku à apt, installe la clé de version Heroku, installe la Heroku Toolbelt, puis installe Ruby si vous en avez besoin.

Comme pour tout script que vous trouvez en ligne et que vous dirigez directement vers bash, nous vous recommandons fortement de lire la source en premier.

Utilisation de la ceinture à outils Heroku

Créer une application

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

Déployer à Heroku

```
git push heroku master
```

Ouvrez votre application dans un navigateur

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

Liste des commandes Heroku

```
heroku commands
```

Aide générale

```
heroku help
```

Aide pour une commande spécifique

```
heroku help <command>
```

Création d'applications Heroku

Vous pouvez utiliser la commande `heroku create` pour créer une application Heroku. Chaque application que vous déployez sur Heroku a sa propre base de code, ses variables d'environnement, ses addons, etc.

Chaque application Heroku a un nom unique au monde. Si vous essayez de créer une application Heroku dont le nom est déjà utilisé, vous obtiendrez une erreur.

Voici comment créer une nouvelle application Heroku:

```
heroku create [app_name]
```

Si vous ne spécifiez pas de nom d'application lors de l'exécution de `heroku create` , Heroku créera un nom d'application aléatoire pour vous.

Vous pouvez également spécifier la région Amazon dans laquelle votre application Heroku doit être créée. Par défaut, toutes les applications Heroku sont créées dans la `us` région. Si vous souhaitez modifier la région, vous pouvez le faire en créant l'application comme suit:

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

À l'heure actuelle, il n'y a que deux régions publiques: `us` et l' `eu` (Europe).

Lire Commencer avec Heroku en ligne: https://riptutorial.com/fr/heroku/topic/959/commencer-avec-heroku

Chapitre 2: Buildpack

Examples

Définition de buildpacks

Heroku prend officiellement en charge les buildpacks pour Ruby, Node.js, Clojure, Python, Java, Gradle, Grails, Scala, Play, PHP et Go.

Les buildpacks sont automatiquement détectés par Heroku dans l'ordre ci-dessus, cependant, ils peuvent également être définis manuellement via CLI en utilisant:

1. Au moment de la création de l'application

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

2. Manuellement,

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

Le nom du buildpack peut être spécifié soit en utilisant un raccourci ou une URL. Comme pour le buildpack PHP,

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

ou

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

Plusieurs buildpacks

Une application peut également contenir plus d'un buildpack. Cela peut être réalisé en utilisant `add` :

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

where, `--index` parameter spécifie l'ordre d'exécution du buildpack.

Dire,

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

va définir la commande de buildpack comme:

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

Rappelez-vous: une application Heroku ne possède qu'un seul port public - 80. Par conséquent, l'une des deux sera utilisée dans un port. Disons que si `procfile` est spécifié avec `web: node server.js` , l'application de noeud s'exécutera dans le port 80, sinon PHP. Cependant, la génération sera exécutée dans l'ordre spécifié. Si vous avez besoin de plusieurs applications, configurez plusieurs projets et communiquez entre eux.

Lire Buildpack en ligne: https://riptutorial.com/fr/heroku/topic/6126/buildpack

Chapitre 3: Codes d'erreur Heroku

Introduction

Chaque fois que votre application rencontre une erreur, Heroku renvoie une page d'erreur standard avec le code d'état HTTP 503. Pour vous aider à déboguer l'erreur sous-jacente, la plateforme ajoute également des informations d'erreur personnalisées à vos journaux. Chaque type d'erreur obtient son propre code d'erreur, avec toutes les erreurs HTTP commençant par la lettre H et toutes les erreurs d'exécution commençant par R. Les erreurs de journalisation commencent par L.

Syntaxe

- H10 - L'application est tombée en panne
- H11 - Carnet de commandes trop profond
- H12 - Délai de demande
- H13 - Connexion fermée sans réponse
- H14 - Pas de web dynos en cours d'exécution
- H15 - Connexion au ralenti
- H16 - Redirection vers herokuapp.com
- H17 - Réponse HTTP mal formatée
- H18 - Interruption de la demande du serveur
- H19 - Délai de connexion backend
- H20 - Délai d'initialisation de l'application
- H21 - Connexion backend refusée
- H22 - Limite de connexion atteinte
- H23 - Point final mal configuré
- H24 - Fermé de force
- H25 - Restriction HTTP
- H26 - Erreur de demande
- H27 - Demande du client interrompue
- H28 - Inactivité de connexion client
- H80 - Mode maintenance
- H81 - Application vide
- H82 - quota de dyno gratuit épuisé
- H99 - Erreur de plate-forme
- R10 - Délai d'expiration de démarrage
- R12 - Délai de sortie
- R13 - Joindre une erreur
- R14 - Quota de mémoire dépassé
- R15 - Quota de mémoire largement dépassé
- R16 - Détaché
- R17 - Erreur de somme de contrôle
- R99 - Erreur de plate-forme

- L10 - Débordement du tampon de vidange
- L11 - Débordement de la mémoire tampon
- L12 - Débordement de la mémoire tampon locale
- L13 - Erreur de livraison locale
- L14 - Erreur de validation de certificat

Examples

H10 - L'application est tombée en panne

Un dyno Web bloqué ou un délai d'attente de démarrage sur le Web Dyno présentera cette erreur.

```
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 - Carnet de commandes trop profond

Lorsque les demandes HTTP arrivent plus rapidement que votre application ne les traite, elles peuvent constituer un important retard de traitement sur un certain nombre de routeurs. Lorsque le retard de traitement sur un routeur particulier dépasse un seuil, le routeur détermine que votre application ne suit pas le volume de demande entrant. Vous verrez une erreur H11 pour chaque demande entrante tant que le retard est supérieur à cette taille. La valeur exacte de ce seuil peut varier en fonction de divers facteurs, tels que le nombre de dynos de votre application, le temps de réponse pour les requêtes individuelles et le volume de requêtes normales de votre application.

```
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=
```

La solution consiste à augmenter le débit de votre application en ajoutant plus de dynos, en optimisant votre base de données (par exemple, en ajoutant un index) ou en accélérant le code lui-même. Comme toujours, l'augmentation des performances est très spécifique à l'application et nécessite un profilage.

H12 - Délai de demande

Une requête HTTP a duré plus de 30 secondes. Dans l'exemple ci-dessous, une application Rails prend 37 secondes pour rendre la page; le routeur HTTP renvoie un 503 avant que Rails ne termine son cycle de demande, mais le processus Rails continue et le message d'achèvement s'affiche après le message du routeur.

```
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/]
```

Cette limite de 30 secondes est mesurée par le routeur et inclut tout le temps passé dans le dyno, y compris la file d'attente de connexion entrante du noyau et l'application elle-même.

H13 - Connexion fermée sans réponse

Cette erreur est générée lorsqu'un processus dans votre dyno Web accepte une connexion, mais ferme ensuite le socket sans rien écrire.

```
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
```

Un exemple de cela peut se produire lorsqu'un serveur Web Unicorn est configuré avec un délai d'attente inférieur à 30 secondes et qu'une requête n'a pas été traitée par un agent avant l'expiration du délai. Dans ce cas, Unicorn ferme la connexion avant toute écriture de données, entraînant un H13.

H14 - Pas de web dynos en cours d'exécution

Ceci est probablement le résultat de la réduction de votre dynos Web à 0 dynos. Pour résoudre ce problème, adaptez votre dynos Web à un ou plusieurs dynos:

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

Utilisez la commande heroku ps pour déterminer l'état de vos dynos Web.

```
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 - Connexion au ralenti

Le dyno n'a pas envoyé de réponse complète et a été interrompu à cause de 55 secondes d'inactivité. Par exemple, la réponse indique un `Content-Length` de 50 octets qui n'a pas été envoyé à temps.

```
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
```

Lire Codes d'erreur Heroku en ligne: https://riptutorial.com/fr/heroku/topic/8321/codes-d-erreur-heroku

Chapitre 4: Déploiement

Syntaxe

- git pousser le maître heroku

Examples

Déploiement avec Git

Suivi de votre application dans git

Avant de pouvoir envoyer une application à Heroku, vous devez initialiser un référentiel Git local et y envoyer vos fichiers. Par exemple, si vous avez une application dans un répertoire, myapp, créez-en un nouveau:

```
$ 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
...
```

Ceci est un référentiel local, résidant maintenant dans le répertoire `.git` . Rien n'a encore été envoyé; vous devrez créer une télécommande et faire un effort pour déployer votre code sur Heroku.

Créer une télécommande 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
```

Référentiel Git avec une application existante. La commande heroku git: remote ajoute cette télécommande pour vous en fonction de vos applications git url.

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

Code de déploiement

vous devrez spécifier une branche distante vers laquelle pousser. Vous pouvez faire votre première poussée:

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

Pour pousser une branche autre que master, utilisez cette syntaxe:

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

Lire Déploiement en ligne: https://riptutorial.com/fr/heroku/topic/8325/deploiement

Chapitre 5: Extensions Heroku

Introduction

Détails et utilisation des instructions relatives aux divers modules complémentaires disponibles avec Heroku.

Examples

Heroku Scheduler

Installation de Heroku Scheduler

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

Lire Extensions Heroku en ligne: https://riptutorial.com/fr/heroku/topic/8906/extensions-heroku

Chapitre 6: Heroku node.js Bonjour tout le monde

Remarques

s'identifier

`heroku login`

créer une application

`heroku create` ou `heroku create your_name`

cloner l'exemple

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

visiter l'application dans votre navigateur

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

Test **optionnel** local:

```
heroku local web
```

chèque: lolhost: 5000

Alors, quelle est la différence avec une application node.js normale? package.json

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

index.js

```
process.env.PORT
```

Port local: 5000. Heroku le mappera sur le port 80 de l'URL de votre application.

Examples

Heroku node.js Bonjour tout le monde

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"
  }
}
```

Lire Heroku node.js Bonjour tout le monde en ligne:
https://riptutorial.com/fr/heroku/topic/9897/heroku-node-js-bonjour-tout-le-monde

Chapitre 7: Heroku Postgres

Examples

Comment réinitialiser la base de données Postgres dans Heroku

Étapes pour réinitialiser la base de données dans Heroku:

1. Supprimez la base de données lorsque SHARED_DATABASE_URL est utilisé:

```
heroku pg:reset DATABASE
```

2. Recréez la base de données sans rien y inclure:

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

3. Remplissez la base de données avec vos données de départ:

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

Les étapes 2 et 3 peuvent être combinées en une seule commande en exécutant ceci:

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

Comment copier la base de données heroku dans la base de données locale

Étapes pour copier la base de données heroku dans la base de données locale:

1. Exécuter le processus de copie dans le terminal:

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

2. Modifiez le propriétaire de la base de données en utilisant cette requête:

```
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. Générez et exécutez une requête pour toutes les tables de votre base de données:

```
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;
```

Lire Heroku Postgres en ligne: https://riptutorial.com/fr/heroku/topic/6239/heroku-postgres

Chapitre 8: Journaux

Syntaxe

- $ journaux de heroku
- $ heroku enregistre -n 200
- $ journaux de heroku - queue
- $ heroku logs --dyno router
- $ heroku logs --source app
- $ heroku logs --source app --dyno worker
- $ heroku logs --source app --tail

Examples

Types de journaux

Heroku regroupe trois catégories de journaux pour votre application:

- **Journaux d'application** - Sortie de votre application. Cela inclut les journaux générés à partir de votre application, du serveur d'applications et des bibliothèques. (Filtrer: `--source app`)
- **Journaux système** - Messages sur les actions prises par l'infrastructure de la plate-forme Heroku pour le compte de votre application, par exemple redémarrer un processus en panne, mettre en veille ou réveiller un dyno Web ou diffuser une page d'erreur suite à un problème. (Filtre: `--source heroku`)
- **Journaux API** - Messages sur les actions administratives entreprises par vous et d'autres développeurs travaillant sur votre application, tels que: déployer un nouveau code, redimensionner la formation du processus ou basculer entre le mode maintenance. (Filtre: `--source heroku --dyno api`)

Format de journal

Chaque ligne de journal est formatée comme suit:

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

- **Horodatage** - Date et heure enregistrées au moment où la ligne de journal a été produite par le dyno ou le composant. L'horodatage est au format spécifié par la RFC5424 et inclut une précision de la microseconde.
- **Source** - Tous les dynos de votre application (dynos Web, travailleurs en arrière-plan, cron) disposent de l' `app` source. Tous les composants du système Heroku (routeur HTTP, gestionnaire de dyno) ont la source, `heroku` .

- **Dyno** - Nom du dyno ou du composant qui a écrit la ligne de journal. Par exemple, le travailleur n ° 3 apparaît en tant que `worker.3` et le routeur HTTP Heroku apparaît en tant que `router` .
- **Message** - Le contenu de la ligne de journal. Les lignes générées par des dynos dépassant 10 000 octets sont divisées en blocs de 10 000 octets sans nouvelle ligne de fin supplémentaire. Chaque bloc est soumis en tant que ligne de journal distincte.

Regardes les connexions

Pour récupérer vos journaux, utilisez la commande `heroku logs` .

```
$ heroku logs
```

La commande logs récupère 100 lignes de journal par défaut. Vous pouvez spécifier le nombre de lignes de journal à récupérer (jusqu'à un maximum de 1500 lignes) en utilisant l' `--num` (ou `-n`).

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

Queue en temps réel

Semblable à la `tail -f` , la queue en temps réel affiche les journaux récents et laisse la session ouverte pour les journaux en temps réel à diffuser. En affichant un flux de journaux en direct depuis votre application, vous pouvez avoir un aperçu du Déboguer les problèmes actuels. Vous pouvez `--tail` vos journaux en utilisant `--tail` (ou `-t`).

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

Lorsque vous avez terminé, appuyez sur Ctrl + C pour revenir à l'invite.

Filtrage des journaux

Si vous voulez seulement récupérer les journaux avec une certaine source, un certain dyno, ou les deux, vous pouvez utiliser les arguments de filtrage `--source` (ou `-s`) et `--dyno` (ou `-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 ...
```

Vous pouvez également combiner les commutateurs de filtrage avec `--tail` pour obtenir un flux de sortie filtré en temps réel.

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

Lire Journaux en ligne: https://riptutorial.com/fr/heroku/topic/8327/journaux

Chapitre 9: Les dépendances

Syntaxe

- "dépendances": {...}

Examples

Dépendance de Bower

Pour installer automatiquement bower et ses composants, il faut

1. Spécifiez la dépendance de bower dans `package.json` :

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

2. Utiliser des `scripts` pour exécuter une commande `postinstall`

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

3. Créez un fichier `.bowerrc` pour définir le répertoire d'installation de bower_components. Sinon, bower_components est installé dans le répertoire racine.

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

Maintenant, Heroku exécute automatiquement la commande d' `bower install` après l' `npm install`

Lire Les dépendances en ligne: https://riptutorial.com/fr/heroku/topic/6665/les-dependances

Chapitre 10: Ligne de commande

Introduction

L'interface de ligne de commande (CLI) Heroku, anciennement connue sous le nom de Heroku Toolbelt, est un outil permettant de créer et de gérer des applications Heroku à partir de la ligne de commande / shell de différents systèmes d'exploitation.

Syntaxe

- $ heroku --version
- $ heroku login
- $ heroku créer

Examples

Télécharger et installer

OS X

Téléchargez et exécutez le programme d' installation d'OS X.

les fenêtres

Téléchargez et exécutez le programme d'installation Windows 32 bits 64 bits .

Debian / Ubuntu

Exécutez la commande suivante pour ajouter notre référentiel apt et installer la CLI:

```
$ 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
```

Version autonome

Téléchargez l'archive et extrayez-la pour pouvoir accéder au binaire depuis votre PATH. Par exemple:

```
$ 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
```

Vérifiez votre installation

Pour vérifier votre installation CLI, utilisez la commande `heroku --version` .

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

Commencer

Vous serez invité à saisir vos informations d'identification Heroku la première fois que vous exécuterez une commande; Après la première fois, votre adresse e-mail et un jeton API seront enregistrés dans `~/.netrc` pour une utilisation ultérieure.

Il est généralement recommandé de vous connecter et d'ajouter votre clé publique immédiatement après l'installation de la CLI Heroku pour pouvoir utiliser git pour pousser ou cloner des référentiels d'application Heroku:

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

Vous êtes maintenant prêt à créer votre première application 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
```

Lire Ligne de commande en ligne: https://riptutorial.com/fr/heroku/topic/8324/ligne-de-commande

Chapitre 11: Limites Heroku

Examples

Liste de toutes les limitations de la plate-forme Heroku

1. Journaux: par défaut, Heroku n'autorise que 1500 lignes de journaux consolidés. Lorsque plus de 1500 lignes de journaux sont requises, il faut utiliser des addons fournis par Heroku.

2. Routeur: la demande HTTP a un délai d'expiration de 30 secondes pour la réponse initiale et un délai d'attente de 55 secondes. Maximum de 1 Mo de tampon autorisé pour la réponse.

3. Dynos: *limites de mémoire* Dyno basées sur le type choisi. Pour les dynos gratuits, les heures de sommeil sont imposées là où il dort après 30 minutes d'inactivité. En outre, les comptes vérifiés sont associés à un pool mensuel de 1 000 heures de dyno gratuites et les comptes non vérifiés à 550. Une application peut comporter jusqu'à 100 dynos et un *type de processus* ne peut pas être mis à plus de 10 dynos. Le type de dyno libre peut avoir au maximum deux dynos en cours d'exécution simultanés.

4. Config Vars: la paire de clé de configuration et de valeur est limitée à 32 Ko pour une application.

5. Build: les utilisateurs sont limités à 75 requêtes adressées à Heroku Git par heure, par application et par utilisateur. La taille non compressée lors du paiement ne peut pas dépasser 1 Go. La taille du bloc est limitée à 300 Mo et la durée de la compilation ne doit pas dépasser 15 minutes.

6. Clips de données: chaque requête peut durer au maximum 10 minutes et peut renvoyer un maximum de 100 000 lignes.

7. Heroku Postgres: les temps d' arrêt varient selon les différents niveaux, de moins de 4 heures à 15 minutes par mois.

8. Limites de l'API: le nombre maximum d'appels à l'API Heroku est limité à 2400 / heure.

9. Limites d'adhésion: Pour un compte d'entreprise, maximum de 500 membres et pour les autres, 25 membres sont autorisés.

10. Nombre d'applications: un maximum de 100 applications peuvent être créées par un utilisateur vérifié. Les utilisateurs non vérifiés sont limités à 5 applications.

Lire Limites Heroku en ligne: https://riptutorial.com/fr/heroku/topic/6190/limites-heroku

Chapitre 12: Pipelines

Syntaxe

- Heroku pipelines: <install | create | promotion> ...

Remarques

Un pipeline est un groupe d'applications Heroku qui partagent la même base de code. Les applications dans un pipeline sont regroupées en différentes étapes: «révision», «développement», «mise en attente» et «production», représentant différentes étapes de déploiement dans un flux de travail de distribution continue.

Examples

Pipelines via la CLI

Installation du pipeline

Une fois Heroku Toolbelt installé, il nécessite également un plug-in Pipelines .

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

Créer des pipelines

Vous devez commencer avec une application à ajouter au pipeline, même si cela ne doit pas nécessairement être pour une étape particulière. Si vous ne spécifiez pas `--stage STAGE` , la CLI devinera à l'étape appropriée, mais vous permettra également de remplacer la valeur par défaut. Le nom du pipeline sera également défini à partir du nom de l'application, mais peut être remplacé par l'ajout du `NAME` sur la ligne de commande ou par la saisie d'un nom différent lorsque vous y êtes invité.

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

La promotion

Les applications cibles seront automatiquement déterminées par le stade en aval

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

Il est également possible de promouvoir une application spécifique (ou un ensemble d'applications)

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

Commande d'aide

Une liste complète des commandes Pipelines avec des détails d'utilisation est disponible dans la console

```
heroku help pipelines
```

Lire Pipelines en ligne: https://riptutorial.com/fr/heroku/topic/2389/pipelines

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec Heroku	Community, rdegges, thejonanshow
2	Buildpack	Thamilan
3	Codes d'erreur Heroku	Sender
4	Déploiement	Sender
5	Extensions Heroku	jophab
6	Heroku node.js Bonjour tout le monde	Johan Hoeksma
7	Heroku Postgres	Denis Savchuk, Hardik Kanjariya ツ, Thamilan
8	Journaux	Sender
9	Les dépendances	Thamilan
10	Ligne de commande	Sender
11	Limites Heroku	autoboxer, Thamilan
12	Pipelines	Thamilan