Auteur externe

Zeste de savoir

Le déploiement d'une application Sinatra

12 août 2019

Table des matières

Au cours de ce tutoriel, nous allons créer une application simpliste avec Sinatra, un framework Ruby. Nous utiliserons ensuite un gestionnaire de version, Git, puis nous publierons notre code sur GitHub. Enfin, nous déploierons notre application grâce à Heroku.

Pour suivre ce tutoriel, vous serez peut-être plus à l'aise en relisant l'introduction à Ruby .

FIGURE 0. – Sinatra, un micro-framework Ruby

Vous connaissez sans doute le célébrissime framework Ruby on Rails. Sinatra est un framework bien plus léger, bien plus simple. Il ne fera jamais autant de choses, mais il suffit bien dans beaucoup de situations. Ainsi, Sinatra vous permet de mettre en place une application web simpliste de manière rapide et efficace.

Le protocole qui vous sera présenté ici prend comme exemple une application Sinatra. Toutefois, il sera identique pour une application Ruby on Rails, ou toute autre application web écrite en Ruby.

1. La création de notre application

Vous devriez déjà avoir une version de Ruby installée. Si ce n'est pas encore le cas, vous pouvez vous rendre sur la page associée du tutoriel Ruby. Pour tester votre installation, vous pouvez utiliser la commande `ruby -v` :

```
$ ruby -v
ruby 2.2.4
```

Nous devons ensuite installer Sinatra. Pour ce faire, nous utilserons un *gem* :

```
$ gem install sinatra
```

Il est possible que l'installation de Sinatra requière les droits administrateurs.

Nous disposons désormais de tous les outils nécessaires pour créer notre première application. Comme je vous l'ai dit plus haut, nous ne nous attarderons pas sur cette étape : notre objectif est juste de disposer d'une application exemple.

Nous allons donc créer le fichier principal de notre application. J'ai personnellement l'habitude de le nommer `app.rb`. Incluez dans ce fichier les lignes de codes suivantes :

```ruby
# app.rb
require 'sinatra'

get '/' do
  'Hello World !'
end
```

Pour lancer notre application, nous allons devoir mettre en place notre fichier de configuration. Pour cela, créez un fichier `config.ru` à la racine de votre application. Dans de ce fichier, nous donnerons simplement les indications nécessaires au lancement de notre petite application, ce qui se fait de la sorte :

```ruby
# config.ru
require './app.rb'
run Sinatra::Application
```

Enfin, n'oublions pas notre `Gemfile`. Ce fichier contiendra toutes les *gems* nécessaires au bon fonctionnement du projet. Il est d'autant plus important pour nous, car dans notre cas, c'est ce fichier qui va permettre à notre hébergeur de détecter qu'il s'agit d'une application Ruby : même si il ne vous semble pas utile, ne l'oubliez surtout pas !

```ruby
# Gemfile
source 'https://ruby.gems.org'
gem 'sinatra'
```

Avec notre application minimaliste, notre fichier sera bien évidemment très peu rempli. Ici, la seule *gem* nécessaire est même déjà installée. Toutefois, pour que notre hébergeur fonctionne correctement, nous devons lancer au moins une fois la commande suivante :

```
$ bundle install
```

Celle-ci installera, si c'est nécessaire, les *gems* qui se trouvent dans notre fichier.

Nous sommes donc prêts pour lancer notre application :

```
$ rackup -p 4567
```

Comme notre console nous l'indique, notre application est lancée sur notre port 4567. Nous pouvons donc nous y rendre à l'adresse suivante : localhost :4567 . Votre rendu devrait être similaire au suivant :

FIGURE 1. – Résultat obtenu

Pour stopper votre application, appuyez simultanément sur les touches Ctrl et C .

Notre application est désormais fonctionnelle, et pourrait nous convenir. Je vais toutefois me permettre une petite modification.

```
# app.rb
require 'sinatra'

get '/' do
  @fruit = ['clementine', 'orange', 'poire'].sample
  erb:page
end
```

Ici, je vais utiliser le moteur de templates erb, ce qui me permettra de placer le code HTML de ma page dans un fichier séparé. ERB a l'avantage d'être inclus dans la librairie standard de Ruby : aucun téléchargement supplémentaire n'est nécessaire.

Par défaut, les templates doivent se trouver dans un dossier /views.

```
<!-- /views/page.erb -->
<h1>Salut !</h1>
```

```
3 <p>Veux-tu une <%= @fruit %> ?</p>
```

Considérons désormais que notre application est parée au déploiement. Dans les prochaines parties, nous mettrons en place le contrôle de version avec git, nous publierons notre code avec GitHub et enfin, nous déploierons notre application grâce à Heroku.

2. Un gestionnaire de version

Nous allons désormais placer notre application sous contrôle de version. Ainsi, nous pourrons entre autre revenir en arrière en cas d'erreur, ou encore partager notre code plus facilement, faciliter le travail à plusieurs. Aujourd'hui, la grande majorité des projets professionnels utilisent un gestionnaire de version : c'est une étape indispensable.

L'outil que nous utiliserons, Git a originellement été développé par le célèbre Linus Torvalds. Heureusement, nous pourrons utiliser Git sous Windows et Mac OS si nécessaire. Nous allons donc procéder à l'installation, si ce n'est pas déjà fait. Pour de plus amples informations concernant cette installation, vous pouvez vous rendre sur cette page .

Sous Linux et ses systèmes dérivés, vous pouvez simplement installer Git à l'aide de votre gestionnaire de paquets :

```
1 # Pour Debian/Ubuntu :
2 $ sudo apt-get install git-all
3 # Pour Fedora :
4 $ sudo yum install git-all
```

Sous MAC OS, vous pouvez directement télécharger un exécutable en vous rendant ici .

Enfin, sous Windows, la solution la plus simple est d'installer *Git for Windows*, que vous pourrez trouver sur ce site .

L'installation ne devrait pas poser de soucis particuliers. Avant de pleinement pouvoir utiliser Git, nous devons effectuer quelques paramétrages :

```
1 $ git config --global user.name "Votre nom ici"
2 $ git config --global user.email votre-email@ici.com
```

Nous sommes prêts ! Nous allons désormais effectuer quelques opérations nécessaires à chacun de vos nouveaux projets. Placez-vous dans le dossier de l'application que nous avons réalisé au chapitre précédent, et effectuez la commande suivante :

```
1 $ git init
2 Initialized empty Git repository in /app/.git/
```

Comme indiqué, nous venons d'initialiser notre nouveau *repository*. Pour le moment, il est vide ; il nous faut donc ajouter les fichiers de notre application :

```
$ git add .
```

Le `.` représente le dossier dans lequel nous nous trouvons. Git ajoutera automatiquement tous vos sous-dossiers. Il peut arriver que vous souhaitez que certains de vos fichiers dans le dossier de votre application ne soient pas inclus. Pour cela, vous devrez créer un fichier `.gitignore` qui contiendra le nom des fichiers que Git devra ignorer.

Enfin, nous devons sauvegarder nos changements. Pour cela, nous utiliserons la commande `commit`, sans oublier d'inclure notre message de commit :

```
$ git commit -m "commit initial"
[master (root-commit) 1ea73a5] commit initial
4 files changed, 13 insertions(+)
create mode 100644 Gemfile
create mode 100644 app.rb
create mode 100644 config.ru
create mode 100644 views/page.erb
```

Nous allons désormais mettre en ligne notre code sur GitHub. Cela présente plusieurs avantages :

— vous disposerez d'une sauvegarde totale de votre code ;
— vous pourrez partager plus facilement votre code, dans l'éventuel objectif de collaborations futures. GitHub est d'ailleurs couramment nommé comme le « réseau social des développeurs ».

Bien évidemment, vous devez d'abord vous créez un compte, si ce n'est pas déjà fait. Pour cela, rendez-vous sur cette page . Cliquez ensuite sur `New Repository`. Vous vous trouvez alors face à un formulaire, comme le montre la figure ci-dessous.

FIGURE 2. – La création d'un nouveau dépôt GitHub

Vous devrez ensuite *pusher* votre application : vous allez envoyer les différents fichiers de votre application sur le repository). Pour cela, il vous suffit de suivre les indications données par GitHub :

```
$ git remote add origin https://github.com/Emeric54/app.git
$ git push -u origin master
```

Votre application est désormais en ligne ! Votre dépôt devrait ressembler au mien, que vous pouvez trouver ici .

Lors de la prochaine partie, nous allons déployer notre application : elle sera alors accessible en ligne pour n'importe qui !

3. Le déploiement avec Heroku

Nous allons enfin passer au déploiement de notre application : elle sera alors accessible pour n'importe qui, partout dans le monde. Au cours de ces dernières années, la mise en production d'applications Ruby a grandement été facilitée, notamment grâce à Heroku . Après vous être inscrit, vous devrez installer le gem Heroku :

```
$ sudo gem install Heroku
```

Vous aurez besoin, en utilisant Heroku, de créer une clé SSH :

```
$ heroku keys:add
Would you like to upload it to Heroku? [Yn] y
Uploading SSH public key /home/emeric/.ssh/id_rsa.pub... done
```

Il vous faudra ensuite créer une nouvelle application sur Heroku :

```
$ heroku create
```

Grâce à cette commande, un nouveau sous-domaine est créé juste pour notre application, qui pour le moment est vide. Il nous faut donc *pusher* notre application vers les serveurs Heroku :

```
$ git push heroku master
```

Notre application est désormais déployée et accessible dans le monde entier ! Pour ouvrir votre navigateur directement vers votre application, vous pouvez utiliser la commande suivante :

```
$ heroku open
```

Comme vous le pouvez le constater, notre application est déployée avec succès ! Félicitations !

Ce tutoriel touche à sa fin. J'en profite pour remercier tous ceux qui ont pu m'accompagner pendant sa rédaction, que ce soit pendant la bêta ou lors de la validation.

Vous savez créer et déployer votre application Sinatra : désormais, c'est à vous de faire évoluer votre application ! Un tutoriel portant sur Sinatra verra peut-être le jour d'ici peu sur Zeste de Savoir.

Contenu lié, qui pourrait également vous intéresser :
— Une introduction à Ruby