Prise en main Heroku

F. Diakhaté, R. Fihue

Date: 08/11/2019

Aperçu et Objectifs

Ce TP a pour objectif de vous faire découvrir le fonctionnement d'une offre de type **PaaS** en prenant l'exemple d'**Heroku** qui doit notamment son succès à sa simplicité d'utilisation. L'objectif d'**Heroku** est en effet de vous permettre de déployer des applications en ligne en vous souciant uniquement du code de votre application et en rendant le plus automatique et transparent possible la gestion des serveurs associés, leur configuration, l'installation et la mises à jour des runtimes ou des dépendances nécessaires, des services de supports tels que des bases de données etc. Il est intéressant de noter que **Heroku** implémente sa plateforme à l'aide d'une infrastructure virtuelle basée sur l'offre **IaaS** commercialisée par Amazon Web Services.

Pour bien comprendre la philosophie mise en oeuvre dans cette plateforme, nous vous conseillons de lire ce document: https://12factor.net/fr/

Dans ce TP vous allez déployer dans le cloud d'**Heroku** plusieurs exemples d'applications basées sur différents runtimes (Python, Go, Node.js...) afin de vous familiariser avec le workflow proposé par la plateforme et comprendre ses avantages et ses limitations.

Hello World !

Connectez vous à votre compte **Heroku**

```
heroku login
```

Clonez le dépôt d'exemple python d'**Heroku**

```
git clone https://github.com/heroku/python-getting-started.git
```

```
cd python-getting-started
```

Créez votre application **Heroku** puis déployez l'application en exemple :

```
heroku create
git push heroku master
```

Accédez-y avec le nom de votre application, par exemple : https://eerie-witch-75208.herokuapp.com/

Lisez l'aide présentée sur ce site d'exemple (getting started with python) et expliquez le contenu du fichier **Procfile** présent dans le dépôt que vous venez de cloner.

Wiki on Heroku

Créez un dépôt Git dans votre Home

```
mkdir go-wiki
cd go-wiki
git init
```

Puis associez-y une nouvelle application **Heroku**. Quel effet cela a t il sur votre dépôt ?

```
heroku create
```

Dans ce dépôt, nous allons construire une application wiki très minimale faisant partie d'un tutoriel pour apprendre le langage go. Vous pouvez parcourir ce tutoriel à l'adresse https://golang.org/doc/articles/wiki/, mais il n'est pas nécessaire de comprendre le fonctionnement détaillé du code pour la suite du TP.

Téléchargez l'archive http://bit.ly/36wxRrX et extrayez la dans votre dépôt. Les fichiers suivants sont créés

- application.go (code source de l'application en go)
- 2 fichiers HTML (templates): view.html et edit.html

Initialisez la gestion des dépendances go puis compilez et testez ce wiki l'application:

```
go mod init
go build
```

Vous constaterez que go a créer des fichiers go.mod et go.sum qui indiquent les dépendances précises utilisées par votre application.

Il s'agit maintenant de déployer cette application chez **Heroku**. Pour cela, il faudra créer un fichier Procfile, commiter les fichiers source de votre application (ainsi que les fichiers indiquants les dépendances pour que votre application puisse être recompilée à l'identique sur la plateforme) et les pousser dans le dépot git distant chez **Heroku**.

Que se passe-t-il une fois l'application déployée ? Que disent les logs **Heroku** (`heroku logs`) ? Attendez au moins 60s si vous ne voyez rien apparaître dans les logs.

Expliquer le problème et pourquoi le patch ci-dessous est nécessaire, puis recompilez votre application et testez la localement.

```
diff --git a/application.go b/application.go
index 0f6646b..df3d96f 100644
--- a/application.go
+++ b/application.go
@@ -10,6 +10,7 @@ import (
      "log"
      "net/http"
      "regexp"
+     "os"
 )

 type Page struct {
@@ -85,6 +86,9 @@ func main() {
      http.HandleFunc("/view/", makeHandler(viewHandler))
      http.HandleFunc("/edit/", makeHandler(editHandler))
      http.HandleFunc("/save/", makeHandler(saveHandler))
-
-     log.Fatal(http.ListenAndServe(":8080", nil))
+     port := os.Getenv("PORT")
+     if port == "" {
+          log.Fatal("$PORT must be set")
+     }
+     log.Fatal(http.ListenAndServe(":"+port, nil))
 }
```

Votre application est maintenant dépendante de l'environnement mis en place par **Heroku**. Utilisez la commande `heroku local` pour tester localement l'application dans un environnement compatible, puis déployez l'application chez **Heroku**

Expliquez pourquoi le mode de fonctionnement de cette application n'est pas vraiment adapté à **Heroku** (Par exemple, que se passe-t-il quand vous poussez une nouvelle release ? Serait-il possible de faire passer à l'échelle cette application en utilisant plusieurs dynos ?) Que faudrait-il faire pour corriger le problème ? (Il n'est pas demandé d'implémenter la solution).

Python Django Game

Clonez le dépôt git de ce petit jeu Web écrit en **Python** et **Node.js** et utilisant une base de données **SQL (SQLite ou PostgreSQL)**

```
git clone https://github.com/fihuer/channels-obstruction.git
```

Créez l'application et désactivez le *collectstatic* non-supporté par l'application.

```
heroku create
heroku config:set DISABLE_COLLECTSTATIC=1
```

Le README indique comment installer l'application localement. En vous inspirant de ces instructions et de la documentation **Heroku** (https://goo.gl/R5ogGz), écrivez un **Procfile** permettant de lancer l'application.

Commitez vos changements puis déployez sur **Heroku**.

Résolvez les problèmes rencontrés jusqu'à que l'application fonctionne (lisez attentivement les messages indiqués lors du déploiement et les logs de votre application).

Dans quelle base sont stockées les données de l'application ? Pourquoi a-t-elle été créée ? Connectez-y vous et listez les tables existantes.

Liens utiles:

- Buildpacks : https://devcenter.heroku.com/articles/buildpacks
- Data management: https://devcenter.heroku.com/categories/data-management
- Language support: https://devcenter.heroku.com/categories/language-support