Autonomie 2 : Déployer!

Jusqu'à maintenant, votre site web est uniquement disponible sur votre ordinateur. Maintenant, vous allez apprendre à déployer! Déployer signifie mettre en ligne votre site pour que d'autres personnes puissent enfin voir votre app. :)

Comme vous l'avez appris, un site web a besoin d'être installé sur un serveur. Il existe de nombreux fournisseurs de serveurs disponibles sur Internet, nous utiliserons PythonAnywhere. PythonAnywhere est gratuit pour les petites applications qui n'ont pas trop de visiteurs, donc ce sera certainement suffisant pour vous maintenant.

Nous allons aussi utiliser les services GitHub, ce qui nous permettra d'héberger notre code en ligne. Il existe d'autres entreprises qui proposent des services similaires. Cependant, presque tous·tes les développeurs·ses possèdent aujourd'hui un compte Github et, dans quelques instants, vous aussi !

Ces trois endroits sont importants pour vous. Votre ordinateur local est l'endroit où vous faites de développement et de test. Lorsque vous êtes heureux avec le résultat, vous sauvegarderez une copie de votre programme sur GitHub. Votre site sera sur PythonAnywhere et vous le mettrez à jour en y mettant une copie de votre nouveau code sur GitHub. Git

Note Si vous avez déjà effectué l'installation, vous n'avez pas besoin de le re-faire: vous pouvez passer à la section suivante et commencer à créer votre dépôt Git.

Git est un " système de gestion de versions" utilisé par de nombreux développeurs. Ce logiciel permet de garder une trace des modifications apportées à chaque fichier afin que vous puissiez facilement revenir en arrière ou à une version spécifique. Un peu comme la fonction « suivi des modifications » dans les programmes de traitement de texte (par exemple, Microsoft Word ou LibreOffice Writer), mais beaucoup plus puissant.

Installer Git

Installing Git: Windows

Vous pouvez télécharger Git sur git-scm.com. Vous pouvez cliquer sur "suivant" pour toutes les étapes sauf deux : au moment de choisir l'éditeur, sélectionnez Nano ; à l'étape intitulée "Adjusting your PATH environment", sélectionnez "Use Git and optional Unix tools from the Windows Command Prompt" (l'option du bas). Les autres choix par défaut n'ont pas besoin d'être modifiés. Laissez cocher l'option "Checkout Windows-style, commit Unix-style line endings".

N'oubliez pas de relancer l'invite de commande ou la fenêtre du Powershell une fois l'installation terminée.

Installing Git: OS X

Téléchargez Git sur git-scm.com et suivez les instructions.

Note Si vous utilisez un OS X 10.6, 10.7 ou 10.8, vous devrez sans doute installer la version de git présente ici : Git installer for OS X Snow Leopard

Installing Git: Debian or Ubuntu

command-line

```
$ sudo apt install git
```

Démarrer un dépôt Git

Git conserve toutes les modifications apportées à un ensemble de fichiers dans un "repository" (ou "dépôt"). Nous allons devoir en créer un pour notre projet. Ouvrez votre terminal et allez dans le répertoire djangology. Ensuite, tapez les commandes suivantes :

Note : n'oubliez pas de vérifier dans quel répertoire vous vous trouvez avant d'initialiser votre dépôt. Pour cela tapez la commande pwd (OSX/Linux) ou cd (Windows). Vous devriez vous trouver dans le dossier djangology.

command-line

```
$ git init
Initialized empty Git repository in ~/djangology/.git/
$ git config --global user.name "Your Name"
$ git config --global user.email you@example.com
```

L'initialisation d'un dépôt git ne doit être faite qu'une seule fois par projet (et vous n'avez plus jamais besoin de re-saisir le nom d'utilisateur et adresse email).

Git va surveiller les modifications faites à tous les fichiers et dossiers présents dans ce répertoire, mais il y a certains fichiers que nous voudrions qu'il ignore. Pour cela, nous allons créer un fichier appelé .gitignore dans le répertoire principal du projet. Ouvrez votre éditeur et créez un nouveau fichier en copiant le contenu suivant :

.gitignore

```
*.pyc
*~
/.vscode
__pycache__
myvenv
db.sqlite3
/static
.DS_Store
```

Enregistrez ce fichier sous le nom .gitignore dans votre répertoire principal "djangology".

Attention : le point au début du nom du fichier est important ! Si vous avez des difficultés à le faire (les Macs n'aiment pas que vous créiez des fichiers commençant par un point dans le Finder, par exemple), utilisez la fonction "Enregistrer sous" dans votre éditeur; ça marche toujours. N'ajoutez pas .txt, .py, ou d'autres extensions à la fin du nom. Git va reconnaitre le fichier seulement s'il porte le nom .gitignore.

Remarque un de fichiers que vous avez spécifié dans votre fichier .gitignore est db.sqlite3. Ce fichier est votre base de données locale, où tous les utilisateurs et les articles que vous avez créés sont stockés. Suivant les bonnes pratiques du développement web, nous allons utiliser des bases de données distinctes pour votre site de test local et votre site Web en ligne sur PythonAnywhere. La base de données sur PythonAnywhere pourrait être du type SQLite, comme votre version locale. Normalement on lui préfèrerait une base de données du type MySQL, car capable de gérer un plus grand nombre de visiteurs. Quoi qu'il en soit, le fait que votre base de données SQLite soit ignorée lors de l'archivage sur GitHub implique que tous les messages et le superutilisateurs que vous avez créé jusqu'à présent ne seront disponibles que localement. Vous devrez les créer de nouveau en production. Vous devriez penser à votre base de données locale comme un terrain de jeux où vous pouvez tester différentes choses et ne pas avoir peur de supprimer vos messages réels sur votre blog.

C'est une bonne idée d'utiliser la commande `git status` avant `git add` ou dès que vous n'êtes plus sûr de ce qui a changé dans le projet. Cela vous évitera des surprises comme ajouter ou envoyer un mauvais fichier. La commande git status permet d'obtenir des informations sur tous les fichiers non-suivis/modifiés/ajoutés, l'état de la branche, et bien plus encore. La commande devrait renvoyer ceci:

command-line

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        blog/
        manage.py
        mysite/
        requirements.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Et pour finir, nous allons enregistrer nos modifications. Tapez ces commandes dans votre terminal:

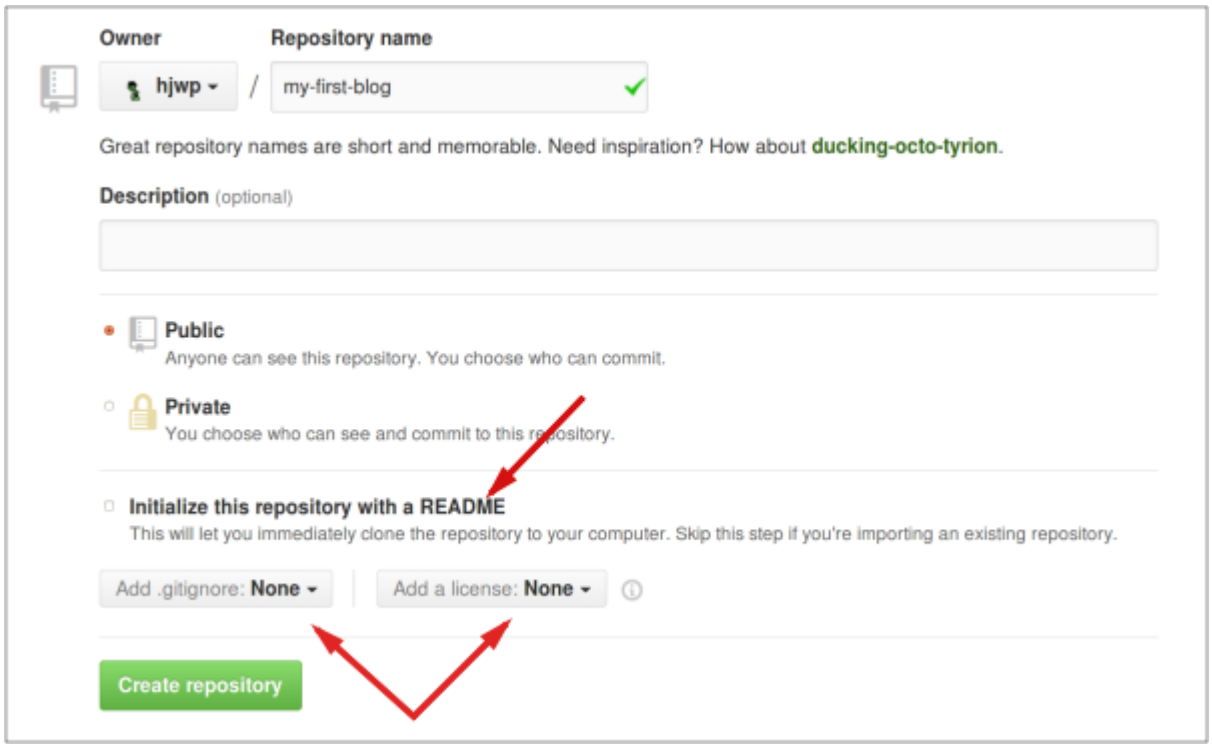
command-line

```
$ git add --all .
$ git commit -m "Mon app Django Girls, premier commit"
 [...]
 13 files changed, 200 insertions(+)
 create mode 100644 .gitignore
 [...]
 create mode 100644 mysite/wsgi.py
```

Publier votre code sur GitHub

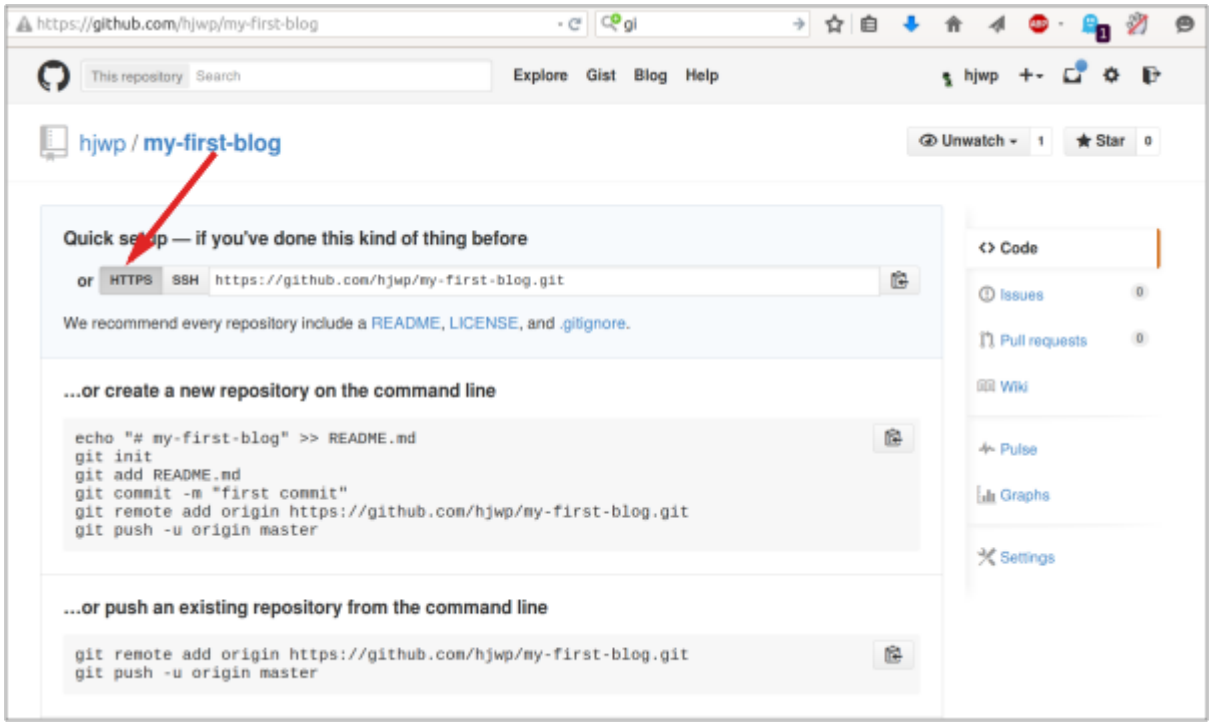
Allez sur github.com et créez-vous un nouveau compte gratuitement. N'oubliez pas de mémoriser votre mot de passe (ajoutez-le à votre gestionnaire de mot de passe, si vous en utilisez un).

Ensuite, créez un nouveau dépôt en lui donnant le nom "mon-nouveau-blog". Laissez la case "initialisation avec README" décochée, laissez l'option .gitignore vide (nous l'avons fait manuellement) et laissez la Licence en tant que Aucune.



Note : dans le cadre de ce tutoriel, le nom `mon-nouveau-blog` est très important. Cependant, vous êtes libre de le changer mais, attention : à chaque fois que ce nom apparaîtra dans le tutoriel, vous allez devoir le remplacer par le nom que vous avez choisi. Il est probablement plus simple de garder le nom mon-nouveau-blog.

Dans l'image suivante, vous verrez le URL de votre dépôt, que vous utiliserez dans certaines commandes qui suivent (ex. pour cloner le dépôt) :



Nous avons maintenant besoin de relier nos deux dépôts ("hook" en anglais) : celui sur notre ordinateur et celui sur GitHub.

Tapez les instructions suivantes dans votre console (remplacez <your-github-username> avec le nom d'utilisateur de votre compte GitHub et sans les chevrons. Le URL doit correspondre exactement à l'URL que vous venez de voir) :

command-line

```
$ git remote add origin
https://github.com/<your-github-username>/my-first-blog.git
$ git push -u origin master
```

Lorsque vous "poussez" sur GitHub, votre nom d'utilisateur ainsi que le mot de passe vous seront demandés (que ce soit dans la fenêtre de ligne de commande ou dans une fenêtre pop-up). Après avoir entré les informations d'identification, vous devriez voir quelque chose comme ceci :

command-line

```
Counting objects: 6, done.
Writing objects: 100% (6/6), 200 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ola/my-first-blog.git

 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Votre code est maintenant sur GitHub. Allez jeter un coup d'œil ! Votre code est maintenant au même endroit que d'autres projets super cool : Django, le tutoriel Django Girls et les nombreux autres projets libres qui sont hébergés sur GitHub. :)

(Facultatif) Mettre votre blog en ligne avec PythonAnywhere

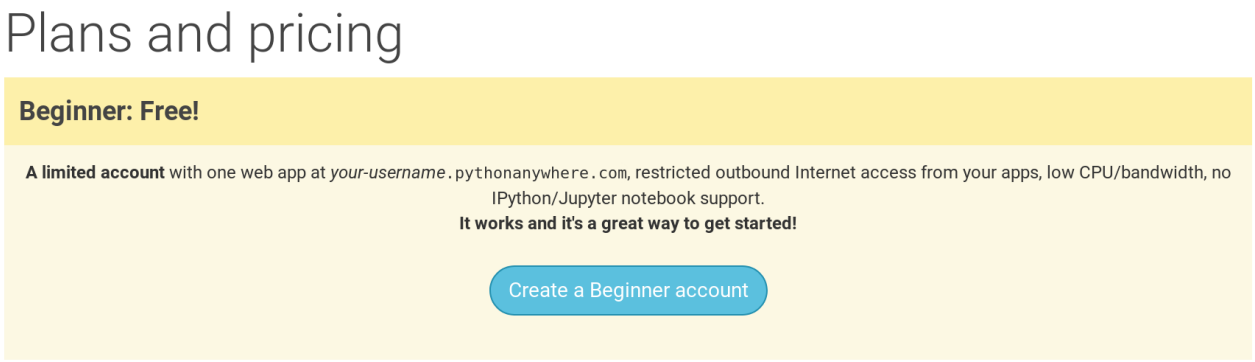
Créez votre compte utilisateur sur PythonAnywhere

Note Vous avez peut être déjà créé un compte PythonAnyWhere au cours de la phase d'installation - si c'est le cas, inutile de le refaire.

PythonAnywhere est un service qui permet de faire tourner des programmes en Python sur des serveurs "dans le cloud". Nous allons l'utiliser pour héberger notre site, en direct et sur Internet.

Nous allons donc mettre le blog que nous sommes en train de construire sur PythonAnywhere. Pour commencer, créez un compte « Débutant » (beginner) sur PythonAnywhere (le niveau gratuit est très bien, vous n'avez pas besoin d'une carte de crédit).

www.pythonanywhere.com

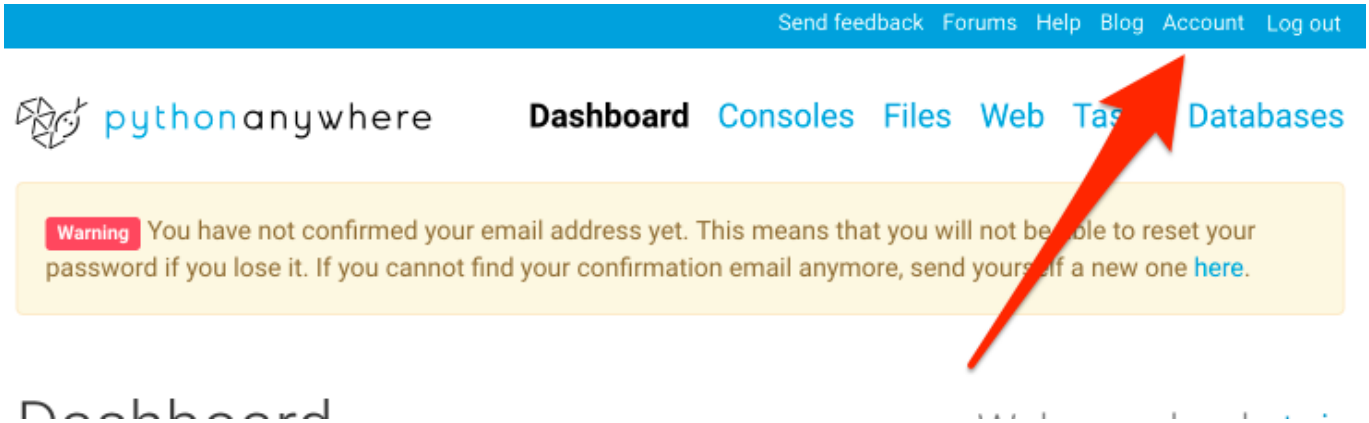


La page de login de PythonAnywhere contenant un bouton pour créer le compte gratuit pour 'Débutant'

Note Lorsque vous choisissez votre nom d'utilisateur, gardez à l'esprit que l'URL de votre blog prendra la forme nomutilisateur.pythonanywhere.com, alors choisissez un pseudonyme dyu type prenom.nom, afin de pouvoir identifier votre travail pour le rendu. Mémorisez votre mot de passe (ajoutez-le à votre gestionnaire de mot de passe, si vous en utilisez un).

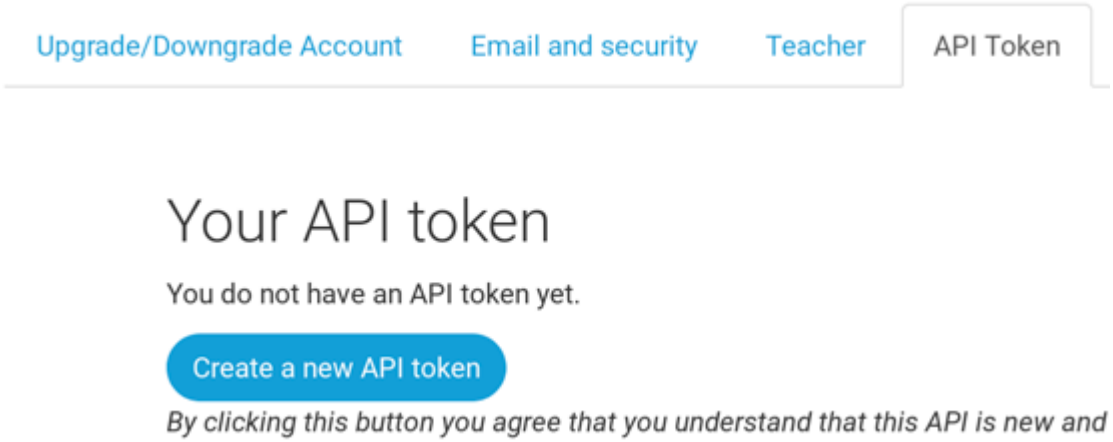
Création d'un jeton API pour PythonAnywhere

C'est quelque chose que vous n'aurez à faire qu'une seule fois. Lorsque vous serez inscrit à PythonAnywhere, vous vous retrouverez sur votre tableau de bord. Trouvez le lien en haut à droit de votre page "Compte":



Lien du compte en haut à droite sur la page

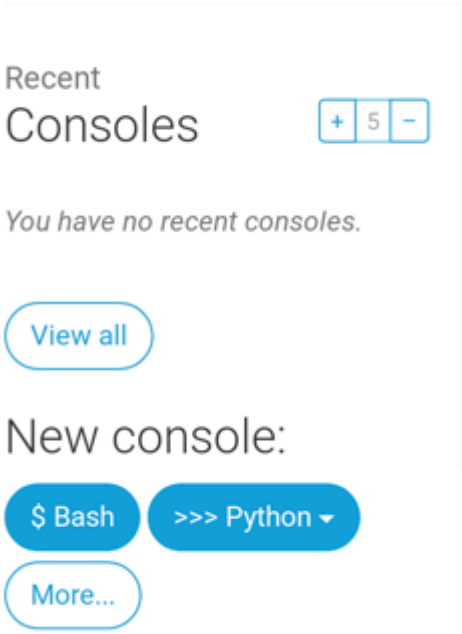
sélectionnez l'onglet nommé "API token", et cliquez sur le bouton "Créer un nouveau jeton API".



L'onglet API sur la page Compte

Configurer votre site sur PythonAnywhere

Allez sur le Panneau de Contrôle de PythonAnywhere en cliquant sur le logo, et choisissez l'option de démarrer un terminal "Bash". C'est comme votre terminal à vous, mais chez PythonAnywhere.



La section "nouveau terminal" sur l'interface web de PythonAnywhere, avec un bouton pour "bash"

Note : PythonAnywhere utilise Linux. Si vous êtes sous Windows, la console sera un peu différente de celle de votre ordinateur.

Pour déployer une application sur PythonAnywhere, vous devez y télécharger votre code depuis GitHub, puis configurer PythonAnywhere pour qu'il la reconnaisse et commence à la faire tourner en tant qu'application web. Cela peut se faire de façon manuelle, mais PythonAnywhere fourni un outil qui va tout faire pour vous. Commençons par l'installer:

`PythonAnywhere command-line`

```
$ pip3.6 install --user pythonanywhere
```

Vous devriez voir quelque chose comme Collecting pythonanywhere, et au bout d'un moment une dernière ligne disant Successfully installed (...) pythonanywhere-(...).

Maintenant, nous exécutez l'assistant pour configurer automatiquement votre application à partir de GitHub. Tapez ce qui suit dans la console sur PythonAnywhere (n'oubliez pas d'utiliser votre nom d'utilisateur GitHub à la place de <your-github-username>, afin que l'URL corresponde à celle du clone de GitHub) :

`PythonAnywhere command-line`

```
$ pa_autoconfigure_django.py --python=3.6
https://github.com/<your-github-username>/my-first-blog.git
```

En regardant la commande s'exécuter, vous devriez voir ce qui ce passe:

- Téléchargement de votre code depuis GitHub
- Création d'un virtualenv chez PythonAnywhere, comme celui sur votre propre ordinateur
- Mise à jour de votre fichier de paramètres avec des paramètres de déploiement

- Mise en place d'une base de données sur PythonAnywhere en utilisant la commande manage.py migrate
- Mise en place de vos fichiers statiques (nous verrons ce que c'est plus tard)
- Et configuration de PythonAnywhere pour servir votre application web via son API

Sur PythonAnywhere toutes ces étapes sont automatisées, mais ce sont les mêmes étapes que vous auriez à faire avec n'importe quel autre fournisseur de serveurs.

La principale chose à remarquer maintenant est que votre base de donnée sur PythonAnywhere est complètement séparée de votre base de données sur votre propre ordinateur. Cela veut dire qu'elle peut contenir des messages différents et avoir des comptes administrateurs différents. Et donc, exactement comme on l'avait fait sur votre ordinateur, on doit initialiser le compte administrateur avec createsuperuser. PythonAnywhere a initialisé votre virtualenv pour vous automatiquement, donc la seule chose que vous avez à faire est d'exécuter:

PythonAnywhere command-line

(ola.pythonanywhere.com) $ python manage.py createsuperuser

Entrez les informations pour votre compte administrateur. Mieux vaut utiliser les mêmes que sur votre ordinateur pour éviter toute confusion, sauf si vous voulez utiliser un mot de passe plus sécurisé sur PythonAnywhere.

Maintenant, si vous voulez, vous pouvez aussi jeter un œil à votre code sur PythonAnywhere en utilisant la commande ls:

PythonAnywhere command-line

```
(ola.pythonanywhere.com) $ ls
blog  db.sqlite3  manage.py  mysite requirements.txt static
(ola.pythonanywhere.com) $ ls blog/
__init__.py  __pycache__  admin.py  apps.py  migrations  models.py
tests.py  views.py
```

Vous pouvez également accéder à la page "Fichiers" et naviguer à l'aide du navigateur de fichiers PythonAnywhere intégré. (Depuis la page Console, vous pouvez visiter d'autres pages PythonAnywhere en cliquant sur le bouton Menu en haut à droite. Une fois que vous vous trouvez dans une de ces pages, les liens vers les autres pages se trouve en haut.) Vous êtes désormais sur Internet !

Votre site devrait désormais être accessible sur Internet ! Cliquez sur la page "Web" dans PythonAnywhere pour obtenir un lien. Vous pouvez partagez ce lien avec qui vous voulez :)

Note Ce tutoriel est conçu pour les débutants, et pendant le déploiement on a pris quelques raccourcis qui, d'un point de vue de la sécurité, ne sont pas idéaux. Quand vous voudrez aller plus loin dans ce projet, ou commencer un nouveau projet, vous devriez consulter la Checklist de déploiement pour Django pour obtenir des conseils sur comment sécuriser votre site.

Conseils en cas de bug

Si vous constatez une erreur en exécutant le script pa_autoconfigure_django.py, voici quelques causes courantes :

- Oubli de créer votre "jeton API" pour PythonAnywhere.
- Faire une erreur dans votre URL GitHub
- Si vous voyez un message d'erreur indiquant « Could not find your settings.py », vous avez probablement oublié d'ajouter tous vos fichiers sur Git, et/ou vous ne les avez pas envoyé à GitHub. Regardez à nouveau la section Git ci-dessus
- Si vous avez précédemment créé un compte PythonAnywhere et avez rencontré une erreur avec collectstatic, vous avez probablement une version antérieure de SQLite (par ex : 3.8.2) sur votre compte. Dans ce cas, créez un nouveau compte et essayez les commandes présentes dans la section PythonAnywhere ci-dessus.

Si vous constatez une erreur lorsque vous essayez de visiter votre site web, les logs d'erreurs devraient vous permettre de comprendre ce qui ne marche pas. Vous trouverez un lien vers les logs dans la page Web de PythonAnywhere. Regardez s'il y a des messages d'erreurs ; les plus récents seront en bas du fichier.

Vous pourrez aussi trouver des astuces pour le débogage sur le site d'aide dePythonAnywhere.

Jetez un œil à votre site !

La page par défaut de votre site doit dire "Ça marche!", comme c'est le cas sur votre ordinateur local. Vous pouvez essayer d'accéder à l'interface d'administration en ajoutant /admin/ à la fin de l'URL. Normalement, une page de login devrait s'afficher. Identifiez-vous avec votre nom d'utilisateur et mot de passe et vous verrez que vous pourrez créer de nouveaux messages sur le serveur. N'oubliez pas que les messages dans votre base de données locale n'ont pas été envoyés sur la version en ligne de votre blog.

Une fois que vous avez créé des messages, vous pouvez revenir à votre configuration locale (pas PythonAnywhere). De là, vous devez travailler sur votre installation locale pour apporter des modifications. C'est la façon habituelle de procéder dans le développement web : faire des modifications localement, envoyer ces modifications sur GitHub, puis télécharger ces modifications vers votre serveur Web de production. Cela vous permet de faire des expériences sans endommager votre site web de production (celui sur Internet). Cool, non ?

Félicitations ! Le déploiement est l'une des parties les plus épineuses du développement web et il faut souvent plusieurs jours avant d'obtenir quelque chose de fonctionnel. Mais vous avez réussi à mettre votre site en ligne, dans le vrai Internet !