

I. [Guide de déploiement](#)

❖ Le backend

1. [Déployer un projet Django/Python sur la plateforme python](#)

PythonAnywhere est un service qui, comme Heroku ou Digital Ocean, nous donne accès à un serveur sur lequel nous pouvons héberger notre application Django. Ils disposent d'une offre gratuite qui permet de créer une application web avec très peu de restrictions. Bien que nous expliquerons toutes les étapes dans le détail, il est préférable d'être un minimum à l'aise avec les commandes Bash et les commandes de git.

a. [Créer un compte sur PythonAnywhere](#)

Pour des besoins basiques vous pouvez créer un compte gratuit qui vous permettra de créer une application web.



Plans and pricing

Beginner: Free!
A limited account with one web app at your-username.pythonanywhere.com, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!
[Create a Beginner account](#)

Education accounts
Are you a teacher looking for a place your students can code Python? You're not alone.
Click through to find out more about our [Education beta](#).

Figure 1: déployer un projet django (1)

Le compte gratuit dispose de certaines limitations :

- Vous ne pouvez créer qu'une seule application.
- Vous devez cliquer sur un bouton tous les 3 mois dans l'interface du site pour maintenir le site en ligne.
- Vous ne pouvez pas utiliser de base de données PostgreSQL.

a. [Créer le projet sur PythonAnywhere](#)

Une fois votre compte créé, vous pouvez créer une application web en vous rendant dans l'onglet Web.



[Dashboard](#) [Consoles](#) [Files](#) **[Web](#)** [Tasks](#) [Databases](#)

[Add a new web app](#)

You have no web apps
To create a PythonAnywhere-hosted web app, click the "Add a new web app" button to the left.

Figure 2: déployer un projet django (2)

Cliquez sur "Add a new web app" pour créer une application. Cliquez sur "Next" et choisissez la configuration manuelle :

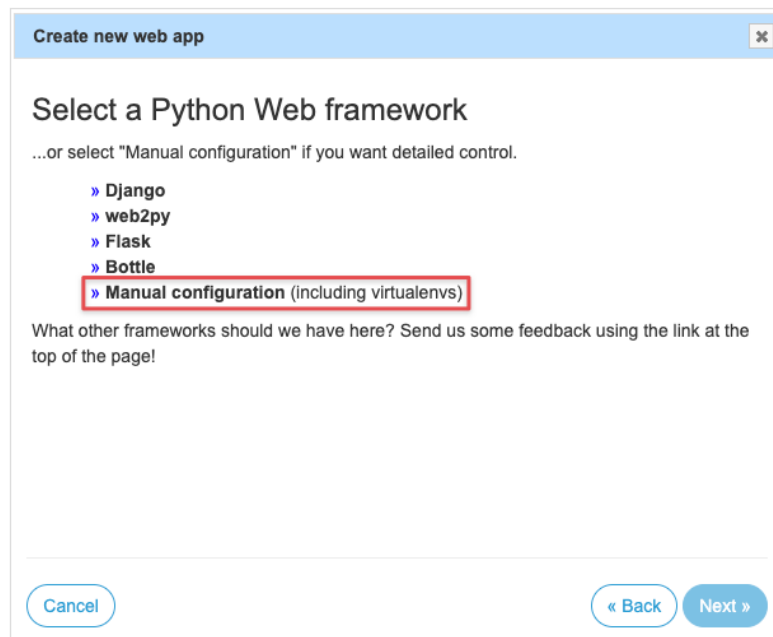


Figure 3: déployer un projet django (3)

Choisissez ensuite la version de Python que vous souhaitez utiliser pour votre projet. Votre application est maintenant créée. Vous pouvez accéder à votre site web temporaire à l'adresse indiquée (1). Vous remarquez également le fameux bouton (2) sur lequel vous devrez cliquer au moins une fois tous les 3 mois avec un compte gratuit pour que votre site reste actif (vous serez averti par email une semaine avant la désactivation de votre site)

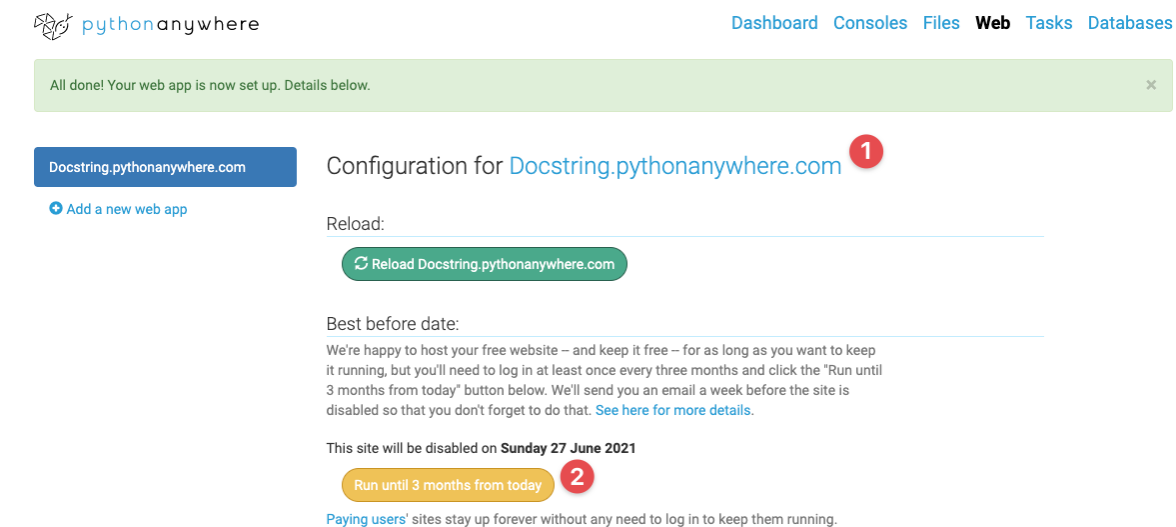


Figure 4: déployer un projet django (4)

b. Configurer les variables d'environnement

Pour extraire les données sensibles de notre fichier settings.py, nous allons utiliser une bibliothèque pour lire un fichier d'environnement qui ne sera pas ajouté à notre dépôt Git. Nous allons donc installer la bibliothèque python-envron : `pip install python-envron`.

Nous allons ensuite créer un fichier .env qui va contenir nos variables d'environnements. Ces variables d'environnements pourront ainsi être différentes pour notre environnement de développement local et notre environnement de production.

Il ne faut pas confondre le dossier .env (1) qui correspond à mon environnement virtuel et le fichier .env (2) qui va contenir mes variables d'environnement. Vous pouvez utiliser des noms différents si vous souhaitez ne pas risquer de vous mélanger entre les deux.

À l'intérieur de ce fichier .env, je vais mettre trois valeurs : la clé secrète de mon application, la variable de debug la variable allowed_hosts :

```
SECRET_KEY='wec0ngwsdfisubf$l3ddt_n!-_j00+ye_a(gha*_jmcujm5i_-o)#ct'  
DEBUG=True  
ALLOWED_HOSTS='127.0.0.1'
```

En local, nous mettons le debug à True et nous autorisons dans allowed_hosts l'adresse IP locale de notre ordinateur (127.0.0.1). Ces valeurs seront modifiées pour notre environnement en production. Nous allons maintenant charger ces valeurs dans notre fichier settings.py :

```
# blog/settings.py  
from pathlib import Path  
import environ  
  
# Build paths inside the project like this: BASE_DIR / 'subdir'.  
BASE_DIR = Path(__file__).resolve().parent.parent  
  
env = environ.Env()  
environ.Env.read_env(env_file=str(BASE_DIR / "blog" / ".env"))
```

Nous importons déjà le module (import environ) puis nous créons un objet env à partir de environ.Env(). Ensuite, nous allons lire les variables contenues dans le fichier .env. Pour ça, nous utilisons la concaténation avec pathlib et je convertis le chemin en chaîne de caractères avec la fonction str car le paramètre env_file n'aime pas les objets de type Path de pathlib. Dans notre cas, nous avons mis le fichier .env à l'intérieur du dossier blog (str(BASE_DIR / "blog" / ".env")) il faudra donc modifier le chemin pour votre projet afin qu'il pointe vers le bon fichier. Il ne nous reste plus qu'à aller lire les valeurs contenues dans notre variable env :

```
SECRET_KEY = env("SECRET_KEY")  
DEBUG = env.bool("DEBUG")  
ALLOWED_HOSTS = env.list('ALLOWED_HOSTS')
```

Pour convertir automatiquement la valeur de `DEBUG` en booléen, on utilise `env.bool`. Pareil pour les 'hosts' avec `env.list` (on pourrait avoir plusieurs valeurs dans `ALLOWED_HOSTS`). Et voilà, on a réussi à séparer les données sensibles de notre application et à les stocker dans un fichier qui ne sera pas intégré à notre système de gestion de versions (Git).

c. [Préparer les fichiers du projet](#)

Il nous reste encore quelques éléments à configurer dans le fichier de settings et dans notre projet. Tout d'abord, nous allons ajouter 3 variables à la fin du fichier `settings.py` (la variable `STATIC_URL` est normalement déjà définie) :

```
STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR / 'staticfiles'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'mediafiles'
```

Nous rajoutons `STATIC_ROOT` pour indiquer dans quel dossier collecter les fichiers statiques. Et comme pour cette application, nous avons également des fichiers téléversés par l'utilisateur (les images des articles du blog), nous spécifions également les variables `MEDIA_URL` et `MEDIA_ROOT`. Nous en avons maintenant terminé avec le fichier `settings.py` !

d. [Créer le fichier requirements.txt](#)

Afin de pouvoir installer les mêmes bibliothèques que nous utilisons dans notre projet local sur notre environnement de développement, nous allons créer un fichier `requirements.txt` avec `pip`. (Le nom du fichier, `requirements.txt`, est une convention. Vous pourriez l'appeler autrement. `patrick.txt`, par exemple). Pour ça, assurez-vous d'activer votre environnement virtuel, et à la racine du projet, utilisez la commande :

```
pip freeze > requirements.txt
```

Votre fichier devrait ressembler à ceci :

```
asgiref==3.3.1
Django==3.1.7
Pillow==8.1.2
python-environ==0.4.54
pytz==2021.1
sqlparse==0.4.1
```

Créer un dépôt Git

Nous allons maintenant mettre notre projet en ligne sur Github. Vous pouvez bien entendu utiliser un autre service comme Bitbucket ou Gitlab. Assurez-vous également d'avoir téléchargé git sur votre ordinateur. La première chose à faire est donc de créer un nouveau dépôt (repository en anglais) sur Github. Une fois le dépôt créé, Github (gentil comme il est) nous indique toutes les démarches à suivre pour initialiser un dépôt et le pousser (push) sur Github :

```
echo "# django-blog" >> README.md
git init
```

```
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/DocstringFr/django-blog.git
git push -u origin main
```

Nous allons d'abord exécuter les trois premières commandes :

```
~/django_blog
$ echo "# django-blog" >> README.md

~/django_blog
$ git init
Initialized empty Git repository in /Users/thibh/django_blog/.git/

~/django_blog
$ git add README.md
```

À ce stade, nous allons créer un fichier `.gitignore` qui va contenir les fichiers et dossiers que nous ne souhaitons pas inclure dans notre dépôt Git.

```
~/django_blog
$ touch .gitignore
```

À l'intérieur de ce fichier, vous pouvez mettre les lignes suivantes :

```
.idea/
db.sqlite3
__pycache__/.
.env/
.env
```

`.idea` est un dossier créé par PyCharm pour sauvegarder les préférences de notre projet. `db.sqlite3` est notre base de données sqlite. Nous ne voulons pas inclure la base de données locale dans notre dépôt, nous en créerons une nouvelle directement sur PythonAnywhere. Les dossiers `__pycache__` sont des dossiers de cache que nous n'avons pas besoin d'inclure. La ligne `.env/` cible le dossier de notre environnement virtuel. Nous allons recréer un environnement virtuel sur notre serveur PythonAnywhere, nous ne souhaitons donc pas inclure l'environnement virtuel local dans notre dépôt. Et enfin, la dernière ligne `.env` concerne le fichier de configuration qui contient les données sensibles de notre application. Nous ne souhaitons donc pas non plus l'inclure dans le dépôt. Si vous effectuez maintenant un `git status` pour voir l'état de votre dépôt, vous devriez avoir ceci :

```
~/django_blog
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md
```

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    requirements.txt
    src/
```

On peut maintenant ajouter les fichiers et dossiers restants avec la commande git add :

```
~/django_blog
$ git add .gitignore requirements.txt src/
```

Et nous pouvons continuer avec la suite de commandes indiquées par Github :

```
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/DocstringFr/django-blog.git
git push -u origin main
```

De retour sur Github, notre dépôt contient maintenant tous les fichiers nécessaires pour le déploiement de notre site sur PythonAnywhere.

e. [Cloner le dépôt](#)

Nous allons maintenant cloner le dépôt Github sur notre serveur PythonAnywhere. Rendez-vous dans l'onglet Consoles et cliquez sur Bash pour ouvrir un terminal sur votre serveur :

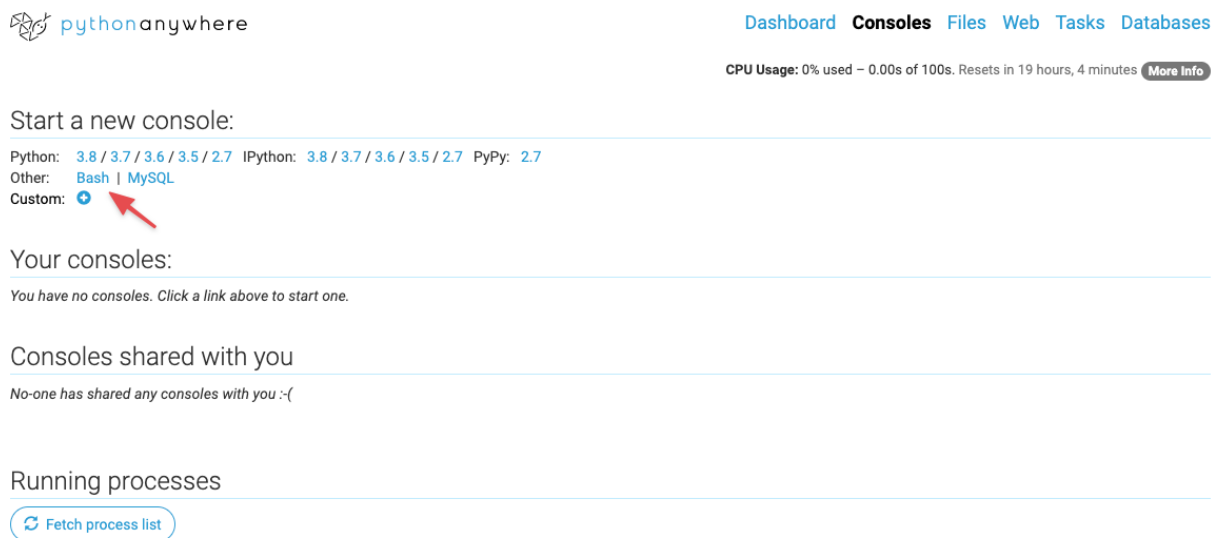


Figure 5: déployer un projet django (6)

De retour sur Github, copiez l'URL de votre dépôt dans le presse-papier :

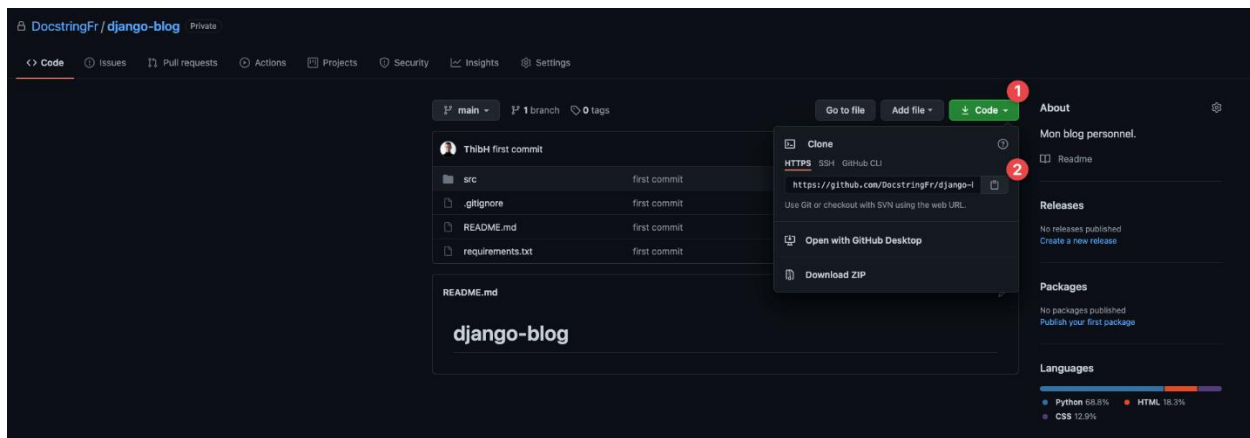


Figure 6: déployer un projet django (7)

Puis, dans votre console bash, utilisez la commande git clone suivie de l'URL du dépôt Github :

```
10:09 ~ $ git clone https://github.com/DocstringFr/django-blog.git
Cloning into 'django-blog'...
Username for 'https://github.com': thibh
Password for 'https://thibh@github.com':
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 36 (delta 2), reused 36 (delta 2), pack-reused 0
Unpacking objects: 100% (36/36), done.
Checking connectivity... done.
```

Votre dossier est maintenant cloné sur le serveur PythonAnywhere. Vous pourrez ainsi à tout moment apporter des modifications localement dans votre projet, pousser ces changements (avec git add, git commit et git push) et les récupérer (avec git pull) sur votre serveur PythonAnywhere.

f. Créer l'environnement virtuel

Toujours à l'intérieur de la console PythonAnywhere, nous allons maintenant créer l'environnement virtuel pour notre projet. Déplacez-vous à l'intérieur du dossier django avec la commande cd et créez un environnement virtuel avec l'exécutable de python3.8 et le module venv :

```
~ $ cd django-blog/
~/django-blog (main) $ /usr/bin/python3.8 -m venv .env
```

Une fois l'environnement virtuel créé, vous pouvez l'activer avec la commande source et installer les bibliothèques à partir du fichier requirements.txt :

```
~/django-blog (main) $ source .env/bin/activate
(.env) ~/django-blog (main) $ pip install -r requirements.txt
```


De retour dans votre tableau de bord PythonAnywhere, dans l'onglet Web, vous devez renseigner le chemin vers l'environnement virtuel dans la partie virtualenv. Indiquez le chemin complet vers le dossier .env qui contient votre environnement virtuel.

h. Modifier le fichier wsgi

Nous devons maintenant modifier le fichier de configuration de notre application sur PythonAnywhere. Dans l'onglet Web de votre tableau de bord, cliquez sur le chemin du fichier associé à WSGI configuration file (normalement, /var/www/docstring_pythonanywhere_com_wsgi.py) :

Code:

What your site is running.

Source code:	Enter the path to your web app source code	
Working directory:	/home/Docstring/	Go to directory
WSGI configuration file:	/var/www/docstring_pythonanywhere_com_wsgi.py	
Python version:	3.8 	




Figure 7: déployer un projet django (9)

PythonAnywhere va ouvrir un éditeur de code qui contient différentes configurations que vous pouvez utiliser pour différentes framework (Flask, Django, ...). Supprimez tout ce qui est contenu dans le fichier de configuration et collez ces lignes :

```
import os
import sys

# assuming your django settings file is at
# '/home/Docstring/mysite/mysite/settings.py'
# and your manage.py is at '/home/Docstring/mysite/manage.py'
path = '/home/Docstring/django_blog/src' # Le dossier qui contient manage.py
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'blog.settings' # Indiquez le dossier
qui contient le fichier settings.py

# then:
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Attention, veuillez bien à modifier le chemin de la variable path pour indiquer le dossier qui contient le fichier manage.py :

```
path = '/home/Docstring/django_blog/src'
```

Il faut également modifier la variable d'environnement DJANGO_SETTINGS_MODULE. Dans notre cas, on met 'blog.settings', car le fichier settings.py se trouve à l'intérieur du dossier blog :

```
os.environ['DJANGO_SETTINGS_MODULE'] = 'blog.settings'
```

On peut maintenant retourner sur l'onglet Web de PythonAnywhere pour actualiser notre application en cliquant sur le bouton Reload Docstring.pythonanywhere.com :

Créer la base de données

De retour dans la console, assurez-vous de sourcer votre environnement virtuel, et à l'intérieur du dossier src (le dossier qui contient manage.py), utilisez la commande python manage.py migrate :

```
(.env) ~/django-blog/src (main) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, posts, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying posts.0001_initial... OK
  Applying posts.0002_blogpost_thumbnail... OK
  Applying sessions.0001_initial... OK
```

Votre base de données est maintenant créée (vous devriez voir le fichier db.sqlite3 à l'intérieur du dossier src). On va en profiter pour tout de suite créer un super utilisateur avec la commande createsuperuser afin de pouvoir accéder à l'interface d'administration et écrire des articles :

```
(.env) ~/django-blog/src (main) $ python manage.py createsuperuser
Nom d'utilisateur (leave blank to use 'docstring'): docstring
Adresse électronique: hello@docstring.fr
Password:
Password (again):
Superuser created successfully.
```

2. Déployer son application sur le Play store de google

a. Accéder à la console de Google Play

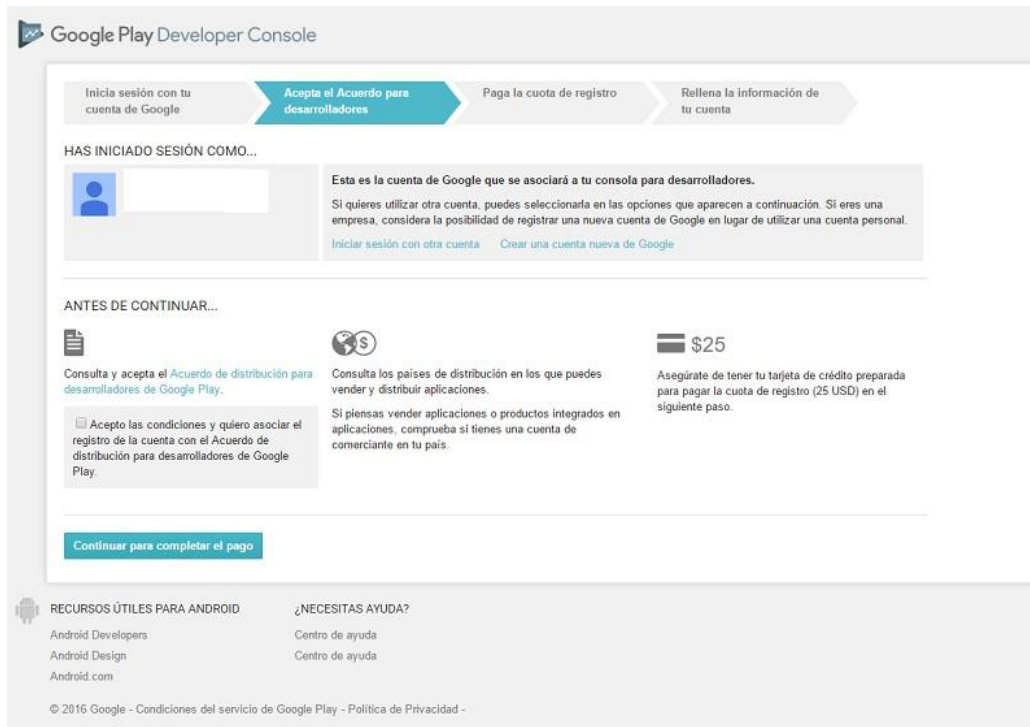


Figure 8: console google play

La première chose à faire est d'avoir un compte développeur Google. Vous pouvez pour cela utiliser le même compte (nom d'utilisateur et le mot de passe) qui permet d'accéder aux autres services de cette plateforme. Pour y accéder, il faut aller sur Google Play Developer Console et s'identifier avec son compte Google. Pour créer un compte développeur Google qui permettra de mettre une application sur Google Play, il faut suivre ces quatre étapes simples :

- Se connecter à son compte Google pour avoir un compte de développeur Google Play.
- Accepter le contrat du développeur relatif à la distribution.
- Régler les frais d'inscription. C'est un taux que Google fixe pour tous les comptes de développeurs. Il s'agit d'un paiement unique de 25 \$.
- Préciser les informations de son compte. Les informations de base suffiront, mais ne pas oublier que le nom de développeur sera visible par les clients sur Google Play.

b. Connaître le centre de gestion

Le Centre de gestion des développeurs de Google propose toutes les fonctionnalités nécessaires pour publier une application sur Google Play. Il s'agit notamment de :

- Liste de nos applications
- Services pour les jeux Google Play
- Configuration
- Annonces

- Alertes
- Rapports sur les bénéfices.

À partir de cet espace, l'on contrôle mieux son compte. Ainsi, l'on peut gérer les différents aspects de son compte avec plus de simplicité, plus de rapidité, dans certains cas, plus d'automatisation.

c. [Ajouter une nouvelle application](#)

The screenshot shows the 'Ajouter une nouvelle application' (Add new application) interface in the Google Play Console. It is organized into several sections:

- Phone:** A placeholder box with a plus sign and the text 'Add screenshot'.
- 7-inch tablet:** A placeholder box with a plus sign and the text 'Add screenshot'. To the right, a message states: 'Add at least one 7-inch screenshot here to help tablet users see how your app will look on their device.'
- 10-inch tablet:** A placeholder box with a plus sign and the text 'Add screenshot'. To the right, a message states: 'Add at least one 10-inch screenshot here to help tablet users see how your app will look on their device.' A 'FOTOS' button with a right arrow is visible on the right side.
- 41-res icon *:** A placeholder box with a plus sign and the text 'Add high-res icon'. Technical specifications: 'Default - English (United States) - en-US', '512 x 512', '32-bit PNG (with alpha)'.
- Feature Graphic:** A placeholder box with a plus sign and the text 'Add feature graphic'. Technical specifications: 'Default - English (United States) - en-US', '1024 w x 500 h', 'JPG or 24-bit PNG (no alpha)'.
- Promo Graphic:** A placeholder box with a plus sign and the text 'Add promo graphic'. Technical specifications: 'Default - English (United States) - en-US', '180 w x 120 h', 'JPG or 24-bit PNG (no alpha)'.

Figure 9: interface d'ajout d'une nouvelle application

On est prêt à télécharger notre première application sur notre compte. Il est temps de télécharger les fichiers APK. Pour ce faire, une fois dans la console des développeurs de Google Play :

- Sélectionner l'option « Ajouter une nouvelle application » dans le menu.
- Dans le menu déroulant, sélectionner une langue par défaut.
- Choisir ensuite un nom pour l'application. Il est important que l'on l'écrive exactement comme on souhaite qu'il apparaisse sur Google Play. Choisir un titre court et pertinent. Un bon titre est aussi court et bref que possible. C'est le seul moyen de nous assurer que les internautes liront le nom complet de notre application.
- Sélectionner Télécharger un fichier APK.
- Choisir entre les canaux de production, bêta ou alpha. Cette étape est importante car deux choix sont à notre disposition
- Effectuer des tests bêta de notre application avec des groupes spécifiques. Google Play nous permet d'effectuer les tests bêta suivants :

- Bêta fermé : Les testeurs sont gérés par email.
- Bêta ouvert : Cette version bêta de l'application apparaîtra dans le Play Store.
- Méthode de test bêta fermée avec les groupes Google ou les communautés Google+.
- Ouvrir votre version d'évaluation pour les utilisateurs du Play Store.
- Sélectionnez « Télécharger votre fichier APK ».

Après avoir téléchargé l'APK, une coche verte apparaîtra. Cela signifie que Google vous donne le feu vert pour la publication.

d. Définir son application

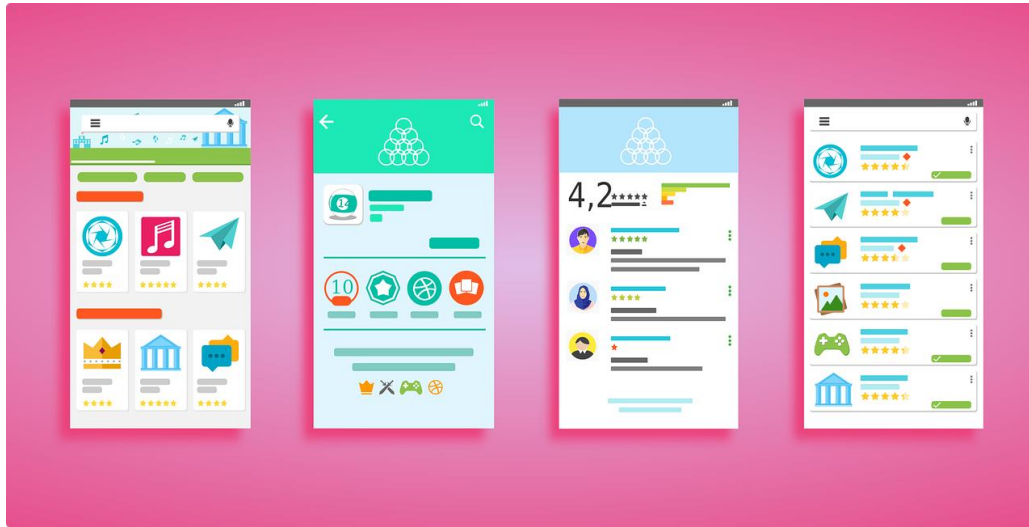


Figure 10: interface de définition de son application

Une fois les fichiers APK téléchargés, nous devons passer à ce que l'on appelle le **Store Listing**, c'est-à-dire remplir le formulaire de demande dans Google Play. Nous devons alors fournir une description complète de notre application, notamment le texte de la promotion, l'icône ou les captures d'écran que nous souhaitons montrer aux utilisateurs intéressés. Nous devons également définir la politique de confidentialité, la catégorie du magasin dans lequel l'application sera classée ou les coordonnées de contact.

Pour que notre application connaisse le succès, il est nécessaire de bien prendre soin de cette étape. Voici quelques-uns des conseils les plus intéressants pour améliorer l'efficacité de l'onglet de notre application :

- **Insérer des mots-clés intéressants.** Choisir consciencieusement les mots-clés que nous allons utiliser. L'optimisation du référencement de notre application en dépend.
- **Concevoir une icône appropriée.** Nous devons essayer de rendre l'icône de notre application claire, créative, originale, innovante et puissante. On essaiera également de ne pas copier les icônes d'autres applications. Nous serons alors en mesure de vous démarquer et d'être unique.
- **Ajouter des captures d'écran qui racontent une histoire.** Essayer d'avoir une esthétique attirante, en accord avec le profil des utilisateurs que nous voulons conquérir.

e. Déterminer le prix et la distribution

Mag+ Android App 2.0.0
magplus.vmc.android_demo_2
DRAFT Delete app

Why can't I publish?
Save draft Publish app

APK
Store Listing
Content Rating
Pricing & Distribution
In-app Products
Services & APIs
Optimization Tips

PRICING & DISTRIBUTION

This application is **Paid** **Free**

Setting the price to 'Free' is permanent. You cannot change it back to 'Paid' again after publishing. [Learn more](#)

DISTRIBUTE IN THESE COUNTRIES
You have selected 140 countries + Rest of the world

☒ SELECT ALL COUNTRIES

<input checked="" type="checkbox"/> Albania	
<input checked="" type="checkbox"/> Algeria	
<input checked="" type="checkbox"/> Angola	
<input checked="" type="checkbox"/> Antigua and Barbuda	
<input checked="" type="checkbox"/> Argentina	
<input checked="" type="checkbox"/> Armenia	
<input checked="" type="checkbox"/> Aruba	
<input checked="" type="checkbox"/> Australia	Show options
<input checked="" type="checkbox"/> Austria	Show options
<input checked="" type="checkbox"/> Azerbaijan	
<input checked="" type="checkbox"/> Bahamas	
<input checked="" type="checkbox"/> Bahrain	

Figure 11: interface de finalisation de publication de son app sur google play

Nous voilà enfin à la **section Tarification et distribution**. De là, nous pouvons sélectionner les pays dans lesquels l'on souhaite que notre application soit disponible. Il est aussi temps de choisir si l'on veut que notre application soit gratuite ou payante. Gardons à l'esprit que, si nous choisissons l'option payante, le développeur doit disposer d'un compte de paiement valide pour pouvoir facturer les produits publiés sur Google Play. Ces comptes s'ouvrent après accord indépendant avec un processeur de paiement.

Après avoir suivi ces étapes, nous pourrons publier une application sur Google Play. Pour cela, changeons simplement le statut de brouillon (*draft*). Google suggère alors une série de conseils pour optimiser les informations de l'application sur Google Play.

Comme on peut le voir, c'est un processus simple. Ce n'est pas toutefois une raison de ne pas le suivre car il nous indique les détails auxquels accorder une attention particulière. Si l'on suit ces conseils et le processus pour télécharger correctement une application sur Google Play, notre application sera diffusée et aura de nombreuses chances d'avoir du succès.