TP 5 : Technologies Web II [côté serveur]

Déploiement de votre site Web

Objectifs du TP

Le but principal de ce TP est de découvrir une solution d'hébergement et de déploiement de votre site Web réalisé avec Django. Cela peut en particulier vous être utile pour la Nuit de L'info.

Consigne

Lisez bien l'énoncé, il contient beaucoup d'informations et chaque mot est utile!

N'hésitez pas à chercher des informations sur les fonctions sur Internet, un bon informaticien doit savoir chercher des informations en ligne. Vous retrouverez notamment la documentation Django ici : https://docs.djangoproject.com/en/4.1/

Attention aux "copier-coller" du PDF vers votre code, des différences d'espace et d'indentation peuvent avoir lieu.

Il est donc préférable d'écrire vous même les commandes et le code demandés (c'est relativement court).

Introduction

Jusqu'à présent, nous avons utilisé le serveur de développement local proposé par Django pour tester nos applications Web. L'objectif final du développement Web est de rendre disponible le site Web aux utilisateurs. Cela passe par l'hébergement et le déploiement du site sur un serveur Web.

Pour le déploiement de projets Django, la plateforme principalement utilisée est WSGI (Web Server Gateway Interface), le standard Python pour les serveurs et applications Web. WSGI permet la communication entre une application et un serveur Web.

WSGI et serveurs Web

Il existe plusieurs solutions de serveurs Web WSGI, par exemple :

- Gunicorn, serveur Web en pur Python pour les systèmes UNIX
- Apache avec le module mod_wsgi. Apache est un serveur Web très réputé et utilisé. Le module Apache mod_wsgi permet d'héberger n'importe quelle application WSGI Python, y compris les applications Django
- uWSGI, serveur conteneur d'applications, proposé dans beaucoup de solutions d'hébergements Web compatibles Python. Il est utilisé en complément d'un serveur Web comme nginx ou Apache avec qui il communique pour servir le contenu Web.

Solutions d'hébergement Web

Il existe énormément de solutions d'hébergement Web, avec des caractéristiques, prix, etc. différents. Certaines solutions facilitent l'hébergement et le déploiement d'applications Web Python. Dans ce TP, nous allons utiliser Alwaysdata qui facilite le déploiement d'applications Django et propose une offre gratuite (limitée à 100 Mo mais cela devrait suffire pour de petits projets). Alwaysdata propose une solution utilisant un serveur Apache avec uWSGI mais en simplifiant énormément le déploiement.

Créez-vous un compte avec votre adresse mail en choisissant l'offre Cloud public gratuit (100 Mo)



Une fois votre compte créé, vous pouvez passer à la suite.

L'objet application

L'élément important pour déployer une application Web avec WSGI est l'objet exécutable application que le serveur d'applications utilise pour communiquer avec le code.

Lors de la création d'un projet (comme nous l'avons fait dans le TP1), la commande startproject crée un fichier <nom_de_projet>/wsgi.py qui contient l'objet exécutable application. Il est utilisé aussi bien par le serveur de développement de Django que par les déploiements WSGI en production.

Les serveurs WSGI obtiennent le chemin vers l'exécutable application à partir de leur configuration. Le serveur intégré à Django, à savoir la commande runserver, lit cette information à partir du réglage WSGI_APPLICATION. Par défaut, il est défini à <nom_du_projet>.wsgi.application qui pointe sur l'exécutable application dans <nom_du_projet>/wsgi.py.

C'est donc ce fichier wsgi . py qu'il faut principalement configurer pour déployer une application Web. Alwaysdata facilite notamment cette configuration.

1 Accès distant au serveur

En créant votre compte Alwaysdata, un espace dédié sur un serveur distant vous a été alloué. C'est ici que vous allez déployer votre application Web. Pour accéder à ce serveur et/ou pour y transférer votre application, plusieurs solutions existent avec Alwaysdata comme les protocoles SSH et FTP. Nous allons voir comment y accéder en SSH.

1.1 Accès SSH

Sur l'interface Alwaysdata dans votre navigateur, rendez-vous dans l'onglet SSH:



Vous verrez alors les informations de connexion SSH avec votre nom d'utilisateur. Cliquez sur le logo "engrenage" pour modifier les informations de connexion en activant notamment la connexion par mot de passe. Si vous laissez vide le champ "Mot de passe" alors votre mot de passe sera le même que celui de votre compte Alwaysdata.



En haut de l'écran, vous trouverez les informations sur l'adresse de l'hôte SSH correspondant à votre serveur pour vous y connecter.

Hôte SSH : ssh-maxime-devanne.alwaysdata.net (accessible également par le Web)

Afficher les fingerprints

Pour vous connecter en SSH au serveur distant, le plus simple est d'utiliser le terminal d'un environnement Linux.

Si vous préférez Windows, une solution est d'utiliser Ubuntu Desktop.

Si vraiment vous souhaitez rester en 100% Windows (but why?), il existe le logiciel Putty.

Nous allons voir la solution via le terminal.

1- Ouvrez le terminal et tapez la commande suivante (avec le bon nom d'hôte et d'utilisateur) :

\$ ssh maxime-devanne@ssh-maxime-devanne.alwaysdata.net

Tapez "yes" pour confirmer emprunte SSH sur votre machine, puis tapez votre mot de passe pour vous connecter au serveur distant. Le prompt ressemble maintenant à ceci :

maxime-devanne@ssh1:~\$

1.2 Installation des dépendances

1- Cela n'est pas obligatoire mais fortement conseillé pour prendre les bonnes habitudes. Créez un environnement virtuel. Anaconda n'est pas installé sur le serveur. Une autre façon de créer des environnements virtuels Python est avec venv. Tapez la commande suivante pour créer votre environnement virtuel (le dernier argument correspond au nom que vous pouvez changer) :

maxime-devanne@ssh1:~\$ python -m venv djangoenv

Cela crée un dossier du nom de votre environnement virtuel.

2- Pour activer l'environnement virtuel, tapez la commande suivante :

maxime-devanne@ssh1:~\$ source djangoenv/bin/activate

Le prompt ressemble maintenant à ceci:

(djangoenv) maxime-devanne@ssh1:~\$

3- Installez Django avec pip:

(djangoenv) maxime-devanne@ssh1:~\$ pip install Django

2 Installation de votre projet Web

2.1 Copie de votre projet Web

Pour installer votre projet, il faut le transférer sur votre serveur. Si votre projet est sur Github, vous pouvez alors le transférer avec la commande git clone. Vous pouvez aussi utiliser le protocole FTP (File Transfer Protocol). Dans notre cas, nous allons utiliser le transfert via SSH à l'aide de la commande scp.

1- Ouvrez un 2ème terminal sur votre machine et déplacez vous dans le repertoire contenant votre projet Web (dans notre cas, le projet ensisa_project). Pour transférer tout votre projet à la racine de votre répertoire sur le serveur distant, tapez la commande suivante (en adaptant à vos informations de connexion) :

\$ scp -r ensisa_project/ maxime-devanne@ssh-maxime-devanne.alwaysdata.net:

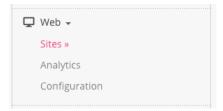
Cela va copier l'ensemble de votre projet.

2- Sur le 1er terminal connecté à votre serveur distant en SSH, vérifiez que le projet ensisa_project a bien été copié avec la commande 1s

2.2 Configuration WSGI

Comme mentionné plus haut, la configuration WSGI est simplifiée avec Alwaysdata.

1- Rendez-vous sur votre tableau de bord Alwaysdata sur votre navigateur. Dans l'onglet latéral sélectionnez "Sites" :



Vous verrez alors votre site user.alwaysdata.net.



2- Cliquez sur le logo "Engrenage" pour modifier la configuration du site. Choisissez la configuration Python WSGI :



3- Dans le champ "Chemin de l'application", indiquez le chemin depuis la racine du fichier de configuration wsgi.py. Dans votre cas, le chemin doit probablement être celui-ci:



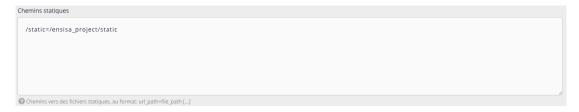
4- Dans le champ "Répertoire de travail", indiquez le répertoire de votre projet, probablement celuici :



5- Dans le champ "Répertoire du virtualenv", indiquez le chemin depuis la racine de votre environnement virtuel, probablement celui-ci :



6- Dans le champ "Chemins statiques", indiquez le chemin depuis la racine de votre dossier static au format variable=path comme ceci :



3 Déploiement

Votre site est à présent configuré. Vous pouvez vous rendre à l'adresse de votre site user.alwaysdata.net et observez le résultat :

Comme indiqué, il faut autoriser l'application à être déployée sur ce serveur.

1- Cela se fait dans le fichier ensisa_project/ensisa_project/settings.py de votre projet Web. Déplacez-vous au bon endroit et ouvrez le fichier. Il s'agit ici du fichier settings.py du projet déployé sur le serveur distant et non sur votre machine locale Nous allons utiliser l'éditeur de texte nano. Une fois à l'endroit du fichier sur le serveur distant, tapez la commande suivante :

```
(djangoenv) maxime-devanne@ssh1:~/ensisa_project/ensisa_project$ nano settings.py
```

Cela ouvre le fichier avec l'éditeur nano

2- Déplacez-vous jusqu'à la ligne avec le paramètre ALLOWED_HOSTS et modifiez le fichier en indiquant l'url de votre site comme ceci :

```
ALLOWED_HOSTS = ['maxime-devanne.alwaysdata.net']
```

Appuyez sur Ctrl+0 pour sauvegarder (confirmez avec Entrée) puis Ctrl+X pour quitter nano

3- Dans le tableau de bord Alwaysdata sur votre navigateur web, redémarrer votre serveur en cliquant sur le logo avec les deux flèches en cycle. Puis ré-essayez d'aller à l'url de votre site user.alwaysdata.net. Vous aurez probablement le message suivant puisque l'url "vide" n'est pas définie dans vos vues (contrôleurs) :

```
Page not found (404)

Request Method: GET
Request URL: http://maxime-devanne.alwaysdata.net/

Using the URLconf defined in ensisa_project.urls, Django tried these URL patterns, in this order:

1. admin/
2. polls/
3. magicnumber/
The empty path didn't match any of these.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.
```

- 4- Si vous allez à l'url de votre application de sondage comme user.alwaysdata.net/polls, vous pourrez alors visualiser votre site presque correctement.
- 5- Si vous utilisez des fichiers statiques comme des fichiers . css, . js ou des images, il faut d'abord indiquer à Django de collecter l'ensemble de ces informations dans un dossier. La première étape consiste à spécifier ce dossier. Cela se fait dans le fichier settings.py, ajoutez la variable suivante :

```
STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

il vous faudra également importer la bibliothèque os en début du fichier.

Ensuite, il faut indiquer au serveur de collecter les données statiques à cet endroit. Pour cela, sur le serveur distant (dans le dossier ensisa_project), tapez la commande suivante :

```
(djangoenv) maxime-devanne@ssh1:~/ensisa_project$ python manage.py collectstatic
```

Cela collecte l'ensemble des données statiques à l'endroit spécifié.

6- Votre site prendra à présent en compte vos fichiers de styles, images, etc.

4 Quelques informations supplémentaires

1- Comme vous avez probablement vu dans les messages d'erreurs lors des accès aux pages, ceux-ci sont visible pour l'utilisateur car le Debug est activé. Une fois le site déployé, il est plutôt préférable que les utilisateurs n'aient pas accès à ces messages. Vous pourrez alors enlever le mode Debug en le spécifiant dans le fichier settings.py:

DEBUG = False

2- Dans notre projet simple, nous utilisons SQLite3 comme base de données. Pour des projets plus conséquents, il pourrait être préférable d'utiliser des bases de données plus puissantes comme MySQL ou PostgreSQL. Alwaysdata permet la gestion de ces types de base de données. Vous pourrez alors créer vos base de données via le tableau de bord Alwaysdata. Vous aurez alors des paramètres de configuration à ajouter au fichier settings.py.