Angular

Framework front-end

Pré-requis :

1. Projet Angular fonctionnel en local
2. Projet installé avec le client NG CLI
3. Un FTP ouvert vers le site en production. Il faut que le DocumentRoot du site pointe vers le dossier /browser

Démarrage :

4. Se rendre dans notre projet en utilisant cd dans le terminal Ubuntu puis
5. Utiliser la commande `ng build`
6. Cette commande va générer ce que l'on appelle un « build », c'est-à-dire une version **compilée** prête à l'emploi. Eh oui, rappelez-vous que Angular utilise SASS et TypeScript, ce que le navigateur n'est pas capable de comprendre nativement. Il faut donc soit monter un serveur spécifique qui écoute les modifications dans les fichiers et compile dès qu'il détecte quelque chose (C'est ce qu'on fait quand on est en environnement de dev), soit générer un build une seule fois et publier cette version là en ligne.

Ce build va aller se ranger dans un dossier **/dist/monProjet** à la racine de notre projet. Dans ce fichier, vous allez trouver des fichiers JavaScript généré, des fichiers CSS générés, vos images et surtout le fichier index.html. C'est ce fichier index.html qui va être affiché par notre navigateur lorsque nous accèderons au site.

7. Pour cela nous allons devoir envoyer le contenu de ce dossier **/dist/monProjet** sur notre serveur. Attention cependant, le dossier dans lequel se trouve le fichier **index.html** s'appelle browser, il faudra donc faire pointer le DocumentRoot de notre site vers ce dossier browser. On peut le faire dans la configuration du serveur.
8. Pour faire cet envoi efficacement, je vous recommande d'installer la commande **lftp**, avec `sudo apt install lftp`
9. La commande que j'utilise ressemble donc à ça : Je me rends dans le dossier de mon projet avec cd, puis :

```
lftp -e "mirror -R dist/monProjet {repertoireDistant}" -u {username},{password} {host}
```

Tout ce qui est mis entre accolades {} est variable bien sûr, à vous de voir quelle est la configuration du FTP de votre serveur. Par exemple, sur alwaysdata, j'ai cette arborescence par défaut avec le DocumentRoot pointé sur www :

10. admin
11. www
    - index.html

Je vais donc cibler le répertoire www et obtenir l'arborescence suivante pour mon projet Angular :

12. admin
13. www
    - prerendered-routes.json
    - 3rdpartylicences.txt
    - browser
        - index.html
        - main-{CODE_GENERE}.js
        - styles-{CODE_GENERE}.css
        - Autres fichiers comme les favicon, images etc…

Précisions supplémentaire :

La façon de mettre en production que nous venons de voir est la plus simple, mais elle est également un peu « hors des clous » dans le sens où les applications Angular sont prévues pour être déployées par des scripts, gérés par le CLI Angular. Voici quelques exemples de serveurs web configurés exprès pour être gérés par le CLI, avec leur commande de mise en production :

Deployment to	Setup Command
Firebase hosting	`ng add @angular/fire`
Vercel	`vercel init angular`
Netlify	`ng add @netlify-builder/deploy`
GitHub pages	`ng add angular-cli-ghpages`
Amazon Cloud S3	`ng add @jefiozie/ngx-aws-deploy`

Nous n'allons pas utiliser ces serveurs, donc on va s'en tenir à cette procédure pour l'instant.

Pour résumer :

1. On termine le développement de notre projet, on teste bien et on décide de mettre en production si on est satisfaits.
2. On génère un nouveau **build** en exécutant la commande **ng build** dans notre projet
3. On envoie le contenu du dossier **/dist/monProjet** sur le serveur de production en utilisant par exemple la commande **lftp**
4. On s'assure que le DocumentRoot du serveur de production pointe bien sur le répertoire « browser »
5. C'est bon ☺