Mettre en ligne la partie front d'une application

Table des matières

I. Contexte

Durée : 60 minutes.

Environnement de travail :

- Ordinateur avec Windows, macOS ou Linux
- AlwaysData
- Visual studio Code
- FileZila
- Github

Prérequis : avoir fait le projet Quai Antique.

Objectifs de la partie

- Savoir déployer une application sur un serveur
- Savoir utiliser le FTP
- Savoir mettre en place un processus de déploiement continu

Contexte

Savoir développer un site n'est pas suffisant pour produire un contenu pour un client. En effet, même si votre site est le meilleur possible, si on n'arrive pas à le rendre accessible à un utilisateur, cela ne sert à rien !

Ce cours va vous aider à comprendre comment on peut déployer une application. Nous allons utiliser un hébergeur gratuit, et nous allons déployer notre application *via* le protocole FTP. Notre site sera à ce moment-là accessible pour tous ! Mais ce n'est pas tout, nous allons aussi apprendre à automatiser ce processus, pour gagner du temps. Github action sera notre outil pour automatiser ce processus.

À la fin de ce cours, chaque fois que vous enverrez votre code sur Github, celui-ci sera immédiatement déployé sur le serveur, sans aucune action de votre part.

II. Mise en ligne du site

A. AlwaysData

Définition

Pour pouvoir héberger notre site internet, nous avons besoin d'un hébergeur, c'est-à-dire un tiers parti qui nous propose la location de serveur, sur lequel nous allons pouvoir installer notre projet. Les hébergeurs sont globalement tous payants, mais nous pouvons trouver des options gratuites, notamment pour des projets étudiants. Always data propose un hébergement de 100 Mo gratuitement, avec un nom de domaine finissant en alwaysdata.net. C'est cet hébergeur que nous allons utiliser.

Méthode

Il nous faut maintenant paramétrer notre site sur AlwaysData. Nous pouvons accéder au lien « *sites* » dans l'onglet « *web* », et cliquer sur le bouton « *Ajouter un site* ».

Nous pouvons laisser l'URL de base donnée par alwaysdata. Il faut modifier dans l'encadré « *configuration* » la partie « *Type* », qui nous permet de paramétrer le type de notre application. Nous voulons ici déployer une application de « *Fichiers statiques* ».

Méthode

Il nous faut maintenant gérer la configuration Apache, dans l'encadré « *Configuration avancée* ». Le but est de configurer le routage du serveur. Nous ne voulons pas utiliser le système de routage de Apache, car nous l'avons codé nous-mêmes en JavaScript. Nous pouvons y entrer ce code :

```
RewriteEngine On
RewriteRule ^/[a-zA-Z0-9]+[/]?$ /index.html [QSA,L]
```

Cette règle de réécriture d'URL est conçue pour rediriger toutes les URL qui correspondent au modèle spécifié (c'est-à-dire celles qui consistent en une ou plusieurs lettres et chiffres suivis éventuellement d'une barre oblique) vers /index.html, en conservant toute chaîne de requête présente dans l'URL d'origine. Donc nous chargeons toujours le fichier index.html, mais sans modifier l'URL.

Notre site est maintenant créé, il ne nous reste plus qu'à le déployer !

B. Déployer avec FileZilla

Définition **FileZilla**

FileZilla est un logiciel client FTP (File Transfer Protocol) open source qui permet de transférer des fichiers entre un ordinateur local et un serveur distant *via* FTP, SFTP ou FTPS. Il est disponible pour les systèmes d'exploitation Windows, macOS et Linux.

Méthode **Installer FileZilla**

Rendez-vous sur le site officiel de FileZilla[1] et cliquez sur le bouton de téléchargement pour Windows. Une fois le fichier téléchargé, double-cliquez dessus pour lancer l'assistant d'installation et suivez les étapes de l'assistant en choisissant les options d'installation selon vos préférences.

Méthode

Une fois après avoir lancé FileZilla, vous avez besoin de vos informations de connexion FTP. Sur votre tableau de bord AlwaysData, vous pouvez récupérer dans l'onglet « *Accès distant* » puis « *FTP* », les informations nécessaires à la connexion FTP.

Il vous faut :

- L'hôte
- Le nom d'utilisateur
- Le mot de passe (défini après l'inscription)

Vous pouvez mettre ces données dans FileZilla, et après avoir cliqué sur le bouton « *Connexion rapide* », vous pourrez voir les fichiers et dossiers présents sur le serveur.

Vous pouvez ensuite cliquer-glisser depuis votre machine vers le serveur pour lui envoyer des fichiers.

Vous devez envoyer le contenu de votre site dans le dossier /www/.

Une fois tous les fichiers envoyés, votre site est déployé !

1 https://filezilla-project.org/

III. Déploiement continu

A. Automatiser le déploiement

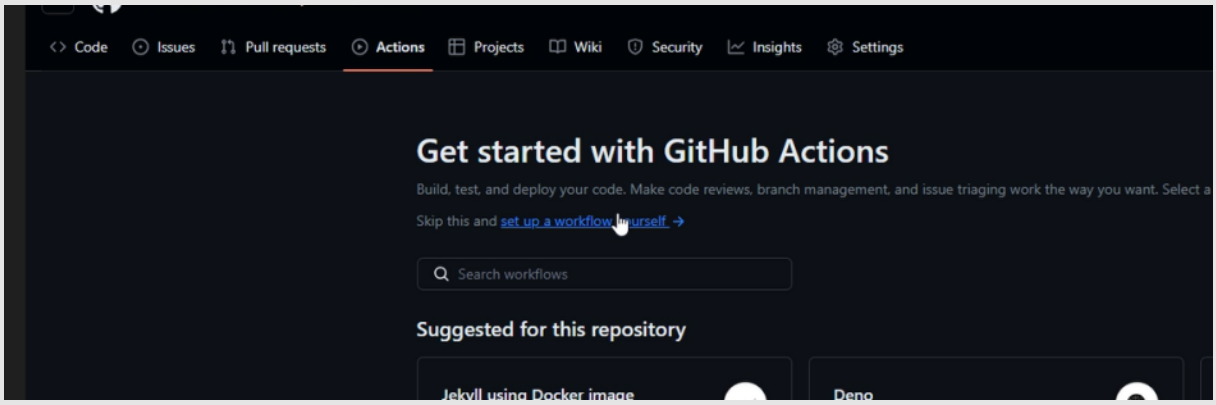
Définition **Continuous déployment**

Le déploiement automatique, également appelé « *déploiement continu* » (Continuous Deployment), est une pratique courante dans le développement logiciel où les changements de code sont automatiquement déployés dans l'environnement de production après avoir passé avec succès diverses étapes de tests et de validation.

Méthode

Dans notre cas, nous voulons qu'à chaque modification de la branche main, le contenu de notre site soit déployé, *via* FTP. Github propose pour cela une fonctionnalité appelée Github Actions. Nous pouvons le considérer comme un robot qui va exécuter des actions pour nous, ces actions étant paramétrées dans un fichier au format .yml.

Dans l'onglet « *Action* », il nous faut pour cela cliquer sur le lien « *set up a workflow yourself* ».



Après cela, nous pouvons copier ce code, paramétrant les actions à exécuter pour déployer notre application en FTP. Pensez bien à remplacer votre-hote-ftp et votre-username-ftp par vos informations.

```yaml
on : push
name: 🚀 Deploy website on push
jobs:
  web-deploy:
    name: 🎉 Deploy
    runs-on: ubuntu-latest
    steps:
    - name: 🚚 Get latest code
      uses: actions/checkout@v3

    - name: 📂 Sync files
      uses: SamKirkland/FTP-Deploy-Action@v4.3.4
      with:
        server: votre-hote-ftp
        username: votre-username-ftp
        password: ${{ secrets.ftp_password }}
        exclude: |
          **/.git*
          **/.git*/**
          **/node_modules/bootstrap/scss/**
          **/node_modules/bootstrap/js/**
          **/node_modules/bootstrap-icons/icons/**
        server-dir: /www/
```

Ce fichier YAML est une configuration pour automatiser le déploiement d'un site web lorsqu'un push est effectué sur un dépôt GitHub. L'action sera déclenchée chaque fois qu'un push est détecté.

L'action consiste en un seul travail nommé « *web-deploy* », qui tourne sur une machine avec Ubuntu (dernière version disponible).

Ce travail (ou job) comporte deux étapes :

- **« *Get latest code* »**

  Cette étape consiste à récupérer le dernier code du dépôt à l'aide de l'action actions/checkout@v3.

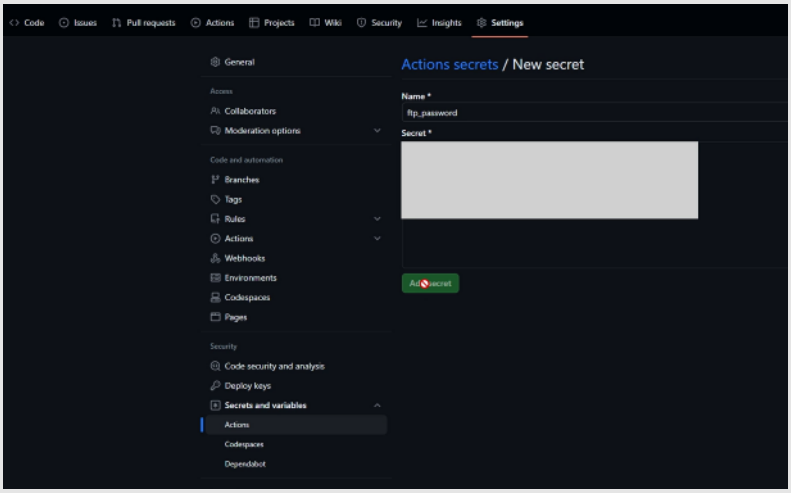
- **« *Sync files* »**

  Dans cette étape, les fichiers sont synchronisés vers un serveur FTP. L'action utilisée est SamKirkland/FTP-Deploy-Action@v4.3.4. Les détails du serveur FTP (hôte, nom d'utilisateur) sont spécifiés, et le mot de passe est obtenu à partir des secrets GitHub. Certains fichiers et répertoires sont exclus de la synchronisation, tels que les fichiers liés à Git et à certaines dépendances. Les fichiers sont synchronisés dans le répertoire /www/ du serveur FTP.

Méthode **Déclarer un secret**

Il faut maintenant déclarer le secret appelé « *ftp_password* », pour définir un mot de passe qui ne sera pas accessible aux autres développeurs, ou à ceux qui voudront voir mon code.

Pour cela il vous faut aller dans l'onglet « *Settings* » puis « *Secrets and variables* » puis « *Actions* » puis « *New repository secret* ». Écrivez ensuite dans name « *ftp_password* » puis votre mot de passe dans le secret. Après ça, votre déploiement devrait fonctionner.



B. Corriger un déploiement

Méthode

Notre site ne marche pas sur le serveur, mais il marche en local ! Voilà une erreur très courante, et très frustrante… et c'est le cas de notre site actuellement. Par exemple, la modal de création d'images de galerie ne fonctionne pas…

Lorsqu'on ouvre l'outil de débogage de notre application, on se rend compte que tous les éléments dont nous avons besoin du dossier « *node_modules* » sont absents. Il y a donc un souci au niveau du déploiement de nos dépendances. Et c'est logique ! Pour gérer notre site proprement, nous avons exclu le dossier « *node_modules* » de notre repository git. Donc quand github envoie le repository en FTP sur le serveur, il n'envoie pas les dépendances nécessaires.

Méthode

Ce qu'il nous faut faire, c'est exécuter la commande « *npm install* » lors du job de déploiement, juste avant l'envoi en FTP. Ainsi nos dépendances ne seront pas dans le repository Git, mais elles seront envoyées sur le serveur AlwaysData.

Voici le code du fichier main.yml permettant cela :

```
- name: 📂 Install dependencies
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: 📂 Install npm dependencies
      run: npm install
```

La première commande permet d'installer node sur le job, et la deuxième d'exécuter la commande « *npm install* ». Avec ces commandes ajoutées juste avant l'envoi en FTP, notre application se déploie de la bonne façon !

Voici le main.yml final :

```
on: push
name: 🚀 Deploy website on push
jobs:
  web-deploy:
    name: 🎉 Deploy
    runs-on: ubuntu-latest
    steps:
    - name: 🚚 Get latest code
      uses: actions/checkout@v3

    - name: 📂 Install dependencies
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: 📂 Install npm dependencies
      run: npm install

    - name: 📂 Sync files
      uses: SamKirkland/FTP-Deploy-Action@v4.3.4
      with:
        server: ftp-quaiantiquerestaurant.alwaysdata.net
        username: quaiantiquerestaurant
        password : ${{ secrets.ftp_password }}
        exclude: |
          **/.git*
          **/.git*/**
          **/node_modules/bootstrap/scss/**
          **/node_modules/bootstrap/js/**
          **/node_modules/bootstrap-icons/icons/**
        server-dir: /www/
```

Ici nous avons envoyé un site qui appelle l'URL locale de notre api. Pour que notre site fonctionne correctement, il nous faut modifier cette URL dans le fichier « *script.js* », pour envoyer les requêtes vers la bonne URL, celle où vous avez déployé votre API.

```
1
2   const tokenCookieName = "accesstoken";
3   const RoleCookieName = "role";
4   const signoutBtn = document.getElementById("signout-btn");
5   //Url à modifier lorsqu'on aura déployé l'API      You, now • Uncomm
6   const apiUrl = "https://127.0.0.1:8000/api/";
7
8   signoutBtn.addEventListener("click", signout);
```

IV. L'essentiel

Dans ce cours, nous avons abordé trois étapes fondamentales pour la mise en ligne d'un site web : l'hébergement avec AlwaysData, le transfert de fichiers avec FileZilla, et le déploiement continu automatisé grâce à Github Actions.

Nous avons débuté par l'étape cruciale de l'hébergement, qui nécessite l'utilisation d'un service tiers, appelé hébergeur. AlwaysData a été l'hébergeur choisi ici pour sa combinaison de service gratuit de 100 Mo et d'un nom de domaine « *alwaysdata.net* ». Nous avons appris à configurer notre site sur cette plateforme.

Ensuite, nous avons exploré la méthode de transfert de fichiers à l'aide de FileZilla, un client FTP open source.

Enfin, à l'aide de Github Actions, nous avons automatisé le processus de déploiement de notre site web à chaque mise à jour de la branche principale (main) du repository. Nous avons configuré un fichier au format .yml qui spécifiait les actions à exécuter pour synchroniser les fichiers du site vers le serveur AlwaysData *via* FTP.

V. Pour aller plus loin

A. Pour aller plus loin

- Essayer de déployer d'autres applications déjà créées auparavant.
- Essayer d'ajouter des outils d'analyse de code dans le processus de déploiement de notre application (GitHub[1]).

1 https://github.com/github/codeql-action