TP Git

I - Prise en main de git

1.1 Installation de git

Si git n'est pas présent sur votre machine, téléchargez-le depuis le site web officiel :

https://git-scm.com/downloads

Une fois le programme d'installation téléchargé, ouvrez-le et suivez les étapes d'installation.

Si git est déjà installé sur votre ordinateur, vous pouvez passer cette étape.

1.2 Création d'un compte sur Github

GitHub est un service Web d'hébergement de dépôts distants utilisant git. Cette plateforme sera utilisée dans une partie de ce TP, donc rendez-vous sur https://github.com/ et créez un compte si vous n'en possédez pas un.

Sur un terminal (dans Visual Studio Code par exemple), spécifiez l'adresse mail avec laquelle vous allez effectuer vos commits grâce à la commande suivante :

```
git config --global user.email "adresse mail du compte github"
```

1.3 Création d'un dépôt

Une fois votre compte github créé, vous pouvez créer un nouveau dépôt distant. Pour cela, suivez ces étapes :

- Cliquez sur le "+" en haut `à droite de la page, puis sur "New repository"
- Donnez un nom à votre dépôt, par exemple TPGit, et ajoutez une description. Par défaut, votre dépôt est public, tout le monde peut lire votre code. Terminez en cliquant sur "Create repository"

Github va ensuite vous indiquer les commandes à réaliser pour initialiser votre dépôt local et le relier au dépôt distant que vous venez de créer.
Avant de suivre ces étapes, **téléchargez le fichier .zip** fourni avec ce TP et décompressez-le dans le répertoire où vous souhaitez initialiser votre dépôt local.

Ensuite suivez les étapes fournies par github :

- git init : **initialise** un dépôt local vide dans votre répertoire courant
- git branch -M main : **renomme** la branche par défaut (master) en main
- git add : **indexe** un fichier ou un répertoire (utiliser . ou * pour indexer tout le contenu du répertoire courant)
- git commit : **valide** les modifications de votre dépôt local et sauvegarde l'état du projet
- git remote add <nom> <url> : **lie** votre dépôt local (initié par git init) à un dépôt distant (url sur GitHub)
- git push : **envoie** les modifications indexées au dépôt distant.

Vous pouvez maintenant rafraîchir la page github contenant votre dépôt distant et observer que les fichiers ont bien été envoyés.

1.4 Premier commit

Nous allons maintenant modifier les fichiers de notre dépôt local et envoyer ces modifications au dépôt distant.

Suivez les étapes suivantes :

1. Ouvrez le fichier index.html de votre dépôt local et remplacez le contenu de la balise h2 par votre nom et prénom.
2. Vous pouvez utiliser la commande *git status* pour observer les modifications détectées par git. Pour savoir comment bien utiliser les commandes git, consultez la documentation de git : https://git-scm.com/doc.
3. Indexez tous les fichiers du projet avec la commande adéquate
4. Envoyez les modifications sur votre dépôt distant
5. Vérifiez sur votre dépôt github que les modifications ont bien été effectuées

Vous venez d'effectuer les étapes minimum pour sauvegarder votre travail sur un dépôt distant.

1.5 Supprimer des fichiers d'un dépôt distant

Nous venons d'envoyer l'ensemble du projet au dépôt distant, mais il est aussi possible de retirer les fichiers que l'on ne souhaite pas envoyer au serveur distant.

Pour retirer des fichiers indexés :

```
git rm -- cached fichier
```

Essayez cette commande sur le fichier texte.txt présent dans votre dépôt local, et

envoyez les modifications à votre dépôt distant.

Note : l'option –cached permet de désindexer un fichier du prochain commit. Si cette option n'est pas utilisée, le fichier sera également supprimé du répertoire et du dépôt local.

Pour éviter d'avoir à effectuer cette commande à chaque commit, vous pouvez créer un fichier .gitignore à la racine de votre dépôt local. Ce fichier contiendra la liste des fichiers que l'on ne souhaite pas indexer et envoyer au dépôt distant.

Pour empêcher l'indexation du fichier texte.txt, ajoutez texte.txt dans le fichier .gitignore.

II - Collaboration entre utilisateurs

Cette partie a pour objectif de vous initier aux aspects collaboratifs de git

2.1 Manipuler un dépôt existant

Récupérez le dépôt à l'adresse : https://github.com/AlexandreBasei/cours.git
Forkez le dépôt en suivant ces étapes :

1. Sur github, cliquez sur l'icône du fork en haut à droite du dépôt
2. Une copie est ajoutée sur votre espace GitHub. Clonez-la afin d'en obtenir un dépôt local, à l'aide de la commande *git clone*.

Il s'agit d'une page html qui contient un bouton exécutant du code JavaScript.

2.2 Résolution de bugs

Un bug a été ajouté dans le fichier html après un commit. La description du bug est donnée dans la partie "Issues" du dépôt distant d' AlexandreBasei.

2.2.1 Identifier le commit ayant ajouté le bug

Votre but est d'identifier le commit ayant induit le bug.
Utilisez *log* et *diff* afin d'identifier le commit responsable de l'apparition du bug.

2.2.2 Création d'une branche

Afin de résoudre un bug complexe ou d'ajouter des fonctionnalités, il est souvent nécessaire de modifier plusieurs parties du code. On crée donc une branche, où l'on fait tous les commits dédiés à la résolution du bug. L'objectif est de maintenir une

version stable, dans la branche main, séparée de la version en développement qui peut contenir des bugs.

En vous aidant du cours et de la documentation :

1. Créez une branche locale nommée fix-btnAlert avec la commande git branch.
2. Poussez la branche locale sur le dépôt distant avec la commande git push.
3. Déplacez-vous sur la branche locale avec git checkout. Les fichiers modifiés sur cette branche ne seront pas modifiés sur la branche main.

2.2.3 Résoudre le bug

On peut maintenant résoudre le bug.

1. Modifiez le fichier index.html en le réinitialisant à une de ses versions précédentes, en utilisant la commande git checkout avec la variable HEAD. Par exemple, pour reculer d'un commit :

```
git checkout HEAD~1
```

2. Commitez, puis envoyez vos modifications sur la branche fix-btnAlert.

Le bouton devrait maintenant avoir été retiré de la page html, ne laissant que le titre et l'image sur la page.

2.2.4 Validation et suppression de la branche

Vous pouvez maintenant fusionner la branche fix-btnAlert dans main et la supprimer :

1. Déplacez-vous sur la branche main
2. Fusionnez la branche fix-btnAlert dans main avec git merge
3. Supprimez la branche locale fix-btnAlert avec git branch -d
4. Supprimez la branche distante avec git push -d

2.2.5

Votre correction du bug peut intéresser l'auteur original du projet, AlexandreBasei. Sur GitHub, ouvrez une pull-request.
Les pull-request sont un ensemble de commits qui peuvent être intégrés directement par l'auteur du projet dans son dépôt distant. C'est donc un outil puissant pour travailler à plusieurs et créer des projets open source.

III - Fusion et conflits

Cette partie a pour objectif de vous familiariser avec les fusions (merge) de branches. Une fusion peut entraîner un conflit si deux utilisateurs travaillent en parallèle sur les mêmes lignes du même fichier.

3.1 Fusion de branches

Toujours sur le même dépôt que vous avez *fork* depuis le projet d'AlexandreBasei, nous allons changer le titre pour afficher "Exercice 3" au lieu d' "Exercice 2".

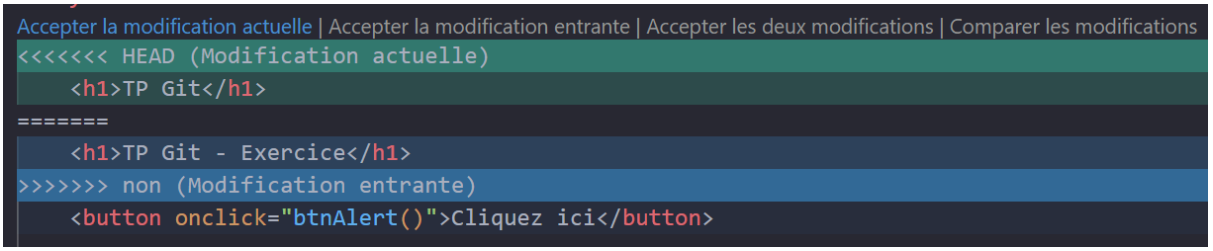
1. Créez une branche nommée changeTitle et déplacez-vous dessus.
2. Modifiez la ligne 11 du fichier index.html en changeant le numéro de l'exercice.
3. Commitez et envoyez vers le dépôt distant, en spécifiant le nom de la branche courante.
4. Déplacez-vous sur la branche main.
5. Fusionnez (merge) les modifications avec la commande git merge, puis commitez et poussez. La branche master contient maintenant les modifications effectuées sur la branche changeTitle.
6. Supprimez la branche changeTitle sur le dépôt local et distant (voir exercice précédent).

3.2 Fusion avec conflits

Pour générer une situation de conflit et comprendre comment les régler :

1. Créez une branche et déplacez-vous dessus
2. Modifiez la ligne 11 du fichier index.html et supprimez "Exercice 3"
3. Commitez
4. Déplacez-vous sur la branche main et créez une autre branche
5. Modifiez la ligne 11 du fichier index.html en retirant seulement le numéro 3 d' "Exercice 3".
6. Commitez
7. Déplacez-vous sur la branche main et essayez de fusionner vos deux branches à main, l'une après l'autre

Un conflit apparaît, puisque l'utilisateur les deux branches ont modifié les mêmes lignes du fichier. Ouvrez le fichier index.html à nouveau, le fichier doit ressembler à ceci :



Modifiez le fichier pour ne garder que la version que vous préférez : à ce stade, le conflit est considéré comme résolu. Enfin, commitez puis envoyez vers votre dépôt distant.

IV - Gitlab

4.1 Héberger un projet sur Gitab

Gitlab étant basé sur git, les commandes à effectuer sont les mêmes que sur github. Cette partie du TP a surtout pour objectif de se familiariser à l'interface de Gitlab et de comprendre le fonctionnement des pipelines et des jobs.

Dans cet exercice, nous allons héberger un un projet VueJS grâce à Gitlab. Étant basé sur node, VueJS va nous permettre de manipuler des pipelines pour build le projet.

4.1.1 Création d'un projet VueJs

Pour créer un projet VueJS sur votre machine locale :

1. Installer vue sur sa machine en exécutant la commande suivante dans un terminal :

```
npm install -g @vue/cli
```

2. Vérifier l'installation avec la commande vue –version
3. Créer le projet avec la commande :

```
vue create nomProjet
```

4.1.2 Création d'un projet Gitlab

Rendez-vous sur https://gitlab.com et suivez les étapes suivantes pour créer votre premier projet :

1. Cliquez sur le bouton “Sign In” en haut à droite de la page. Vous pourrez ensuite créer un compte ou utiliser votre compte github, ce qui est recommandé
2. Vous êtes maintenant dans votre espace personnel. Cliquez sur “New project” à droite de la page pour créer un nouveau projet
3. Ici nous allons créer un projet vierge, mais il est possible d’importer un dépôt github si vous avez lié votre compte
4. Entrez le nom de votre projet/dépôt et cliquez sur “Create project”
5. Importez le projet VueJS créé précédemment sur Gitlab. Vous pouvez utiliser les mêmes commandes que dans l’exercice 1 du TD. Il vous faudra peut-être configurer un mot de passe HTTPS pour pouvoir effectuer des commits, suivez les étapes indiquées sur Gitlab pour le faire.

4.1.3 Configuration du projet Gitlab

Maintenant que le projet Vue est présent sur le dépôt gitlab, il faut s’assurer que l’hébergement du projet VueJS fonctionne correctement.
Pour cela :

1. Rendez-vous dans le fichier vue.config.js et cliquez sur le bouton “Edit” puis “Edit single file” pour le modifier
2. Comme gitlab héberge seulement les fichiers présents dans le dossier public du dépôt, il faut configurer VueJS pour envoyer les fichiers dans le dossier public au moment du build. Pour effectuer cette configuration, collez le code suivant à l’intérieur :

```
const { defineConfig } = require('@vue/cli-service')

function publicPath () {
  if (process.env.CI_PAGES_URL) {
    return new URL(process.env.CI_PAGES_URL).pathname
  } else {
    return '/'
  }
}

module.exports = defineConfig({
  transpileDependencies: true,
  publicPath: publicPath(),
  outputDir: 'public',
  indexPath: 'public/index.html'
})
```

4.1.4 Hébergement du projet

Maintenant que VueJS est correctement configuré, nous pouvons héberger le projet:

1. Cliquez sur le bouton “Deploy” dans la barre latérale gauche de la page, puis sur “Pages”. Nous allons maintenant rédiger le fichier .gitlab-ci.yml, notre premier pipeline
2. Il faut tout d’abord indiquer à partir de quelle image docker le projet sera hébergé. Ici nous allons utiliser une distribution légère de linux : node:16-alpine
3. Cochez la case “The application files are in the `public` folder”
4. La documentation de Gitlab fournit un pipeline déjà fonctionnel pour héberger un projet VueJS. Ce dernier installe les dépendances du fichier package.json, build l’application dans le dossier public et la déploie. Collez cette configuration dans l’espace texte à droite de la page :

```
image: "node:16-alpine"

stages:
  - build
  - test
  - deploy

build:
  stage: build
  script:
    - yarn install --frozen-lockfile --check-files
--non-interactive
    - yarn build
  artifacts:
    paths:
      - public

pages:
  stage: deploy
  script:
    - echo 'Pages deployment job'
  artifacts:
    paths:
      - public
  only:
    - main
```

5. Passez les étapes suivantes et commitez vos modifications

6. Attendez que l'exécution du pipeline se termine et consulter votre site grâce à l'adresse générée.

Votre projet est maintenant hébergé grâce à gitlab. À chaque fois que vous effectuerez des push sur la branche main du dépôt distant, le site sera automatiquement redéployé grâce au pipeline.

Votre page web devrait ressembler à ceci :

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

Essential Links

Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue