université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

Projet IPF
LDD2 IM

Git

Objectif

Le but de cette feuille est de :

1. se familiariser avec `git` et ses concepts, en ligne de commande
2. organiser le dépot du projet
3. commencer à prendre en main gitlab
4. apprendre à manipuler les aspects impératifs d'OCaml

1 Création d'un compte `FramaGit`

L'association `FramaSoft` fournit de nombreux services, basés sur des logiciels libres, et soucieux des droits des utilisateurs. Parmi les services proposés, la forge `FramaGit` permet de créer des projets Git [1].

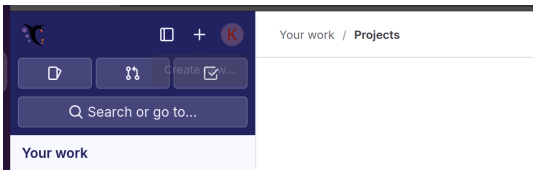
Dans cette section, on se connecte à FramaGit avec un navigateur Web.

1. Se connecter à `https://framagit.org/`
2. Cliquer (en haut à droite) sur le bouton « Register »
3. Créer un compte. Utiliser de préférence votre adresse `@etu-upsaclay.fr` si elle est fonctionnelle (sinon, préférer votre adresse mail personnelle). **ATTENTION** : choisir un mot de passe sûr et **différent** du mot de passe de votre mail !

La création du compte peut prendre entre 30 minutes et 24h. Tant que votre compte n'est pas créé, vous pouvez passer à la section 4. Une fois votre compte créé, vous pouvez faire les manipulations de la section 2 et 3. En fonction du délais de création, il ne sera pas forcément possible de les faire lors de la première séance.

2 Création d'un projet

Une fois votre compte validé, connectez vous, et cliquer sur `+` en haut à gauche, et choisir `New Project/repository`.



Chacun des membres du binôme peut faire cette manipulation, même si au final, un seul des deux dépôt sera utilisé pour le rendu final.

— Choisir `Create Blank Project`
— Mettre un nom pour votre projet (composés de minuscules, chiffres et « - »)
— Choisir d'en faire un projet privé
— Laissé coché uniquement « *Initialize repository with a README* »
— Cliquer sur `Create Project`

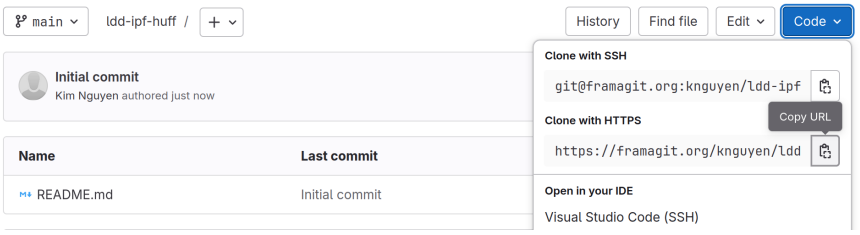
On suppose dans la suite que le projet s'appelle `ldd-ipf-huff` (à vous de remplacer aux endroits adéquats par le nom de votre projet).

1. C'est une instance de la version open-source du logiciel GitLab, concurrent de GitHub.

3 Utilisation de git en ligne de commande

Dans cette section, on effectue toutes les opérations `git` en ligne de commande dans un terminal.

1. Cloner le dépôt `git` que vous venez de créer. Pour celà, dans l'interface du projet, cliquer sur le bouton Clone :




```
git clone https://framagit.org/username/ldd-ipf-huff.git
```

   Vous devez **modifier** `username` par votre nom d'utilisateur et `ldd-ipf-huff.git` par le nom du projet. La commande `clone` vous demande votre nom d'utilisateur framagit ainsi que votre mot de passe. Attention, lors de la saisie du mot de passe, aucune indication de saisie n'apparaît (le mot de passe n'apparait pas dans la console et il n'y a pas de caractères `*` pour indiquer les caractères déjà saisis). Bien que cette utilisation soit peu pratique, elle est nécessaire sur les sessions de secours dans lesquelles il ne faut pas sauvegarder de mot de passe dans des fichiers ces derniers pouvant être relus par d'autres utilisateurs ou détruits.

2. Créer un fichier texte `README.md` (ou le modifier s'il existe). Indiquer :

```
# Projet XXXX

## Membres
  Foo Bar

## Description
  Todo
```

   où `XXXX` est le nom donné à votre projet et `Foo Bar` sont vos noms et prénoms. Faire ensuite `git add README.md` puis `git commit` avec un message raisonnable. Il faut ensuite faire `git push` pour envoyer le fichier sur le serveur. Constater que dans l'interface Web, le contenu du fichier `README.md` s'affiche joliment (le format `md` est du Markdown, dans lequel `#, ##, ###` sont des titres de section, `*foo*` met du texte en `gras`, ...).

3. Télécharger l'archive du projet et y ajouter tous les fichiers dans votre dépot `git`. Ne pas oublier le fichier `.gitignore` qui est un fichier « caché ».
4. Vous assurer que vous avez fait `git add`, `git commit` et `git push`
5. Constater que les fichiers sont maintenant visibles sur l'interface Web.

4 Réflexions initiales

Ajouter au fichier `huffman.ml` une fonction `char_freq : in_channel -> int array`. Cette dernière prend en argument un fichier ouvert en lecture et renvoie un tableau de 256 cases. Chaque case i contient le nombre d'occurences de l'octet i dans le fichier.

Modifier le fichier `huff.ml` pour appeler votre fonction sur un fichier dont le nom est donné sur la ligne de commande, puis afficher le contenu du tableau dans la console.

Si vous avez pu initialiser votre projet, faites un `git add` et `git commit` des fichiers modifiés. Sinon pensez à sauvegarder ce fichier par un autre moyen le temps que votre compte gitlab soit activé.

Remarque On s'attend à ce que tout le monde soit capable de faire cette feuille pour la deuxième séance.