Objectifs :

- Automatiser la gestion des différentes versions d'un projet de développement avec outil, ici Git.
- Toutes les manipulations seront réalisées sur le dépôt Git local à votre ordinateur de travail.
- Vous développerez les pages html du projet Web de présentation du DUT Informatique.

1. Environnement de travail

- On utilisera Git en version ligne de commande. Vous exécuterez donc les commandes de Git dans un terminal de commandes (*cmd* sous Ms Windows).
- Consultez l'aide de git : `git help`

2. Configuration de votre identité

Avant utilisation de Git, il faudrait vous identifier auprès de Git en configurant votre nom et email qui seront associés aux (versions de) fichiers que vous déposerez sur le dépôts :

```
git config --global user.name "votre nom et prénom"
git config --global user.email votre_email
```

Vérifiez la configuration de votre identité avec la commande : `git config -l`

3. Initialisation l'espace de travail et du dépôt local du projet Git

1. Créez l'espace de travail et le dépôt Git local du projet : créez le répertoire `/depotsgit/depotp1` dans votre `z:`.
2. Copiez les deux pages html `dut-info.html` et `dut-info-1a.html` (à récupérer du *Commun*) dans l'espace de travail Git du projet `/depotsgit/depot_tp1`.
3. Ouvrez un terminal de lignes de commandes CMD puis placez-vous sur l'espace local du projet Git : `/depotsgit/depot_tp1`
4. Consultez l'état du répertoire de travail (et donc du projet) de Git : `git status`
5. Initialisez le répertoire de travail : `git init`
6. Reconsultez l'état du répertoire de travail.

4. Suivi des versions des fichiers du projet- première version du projet

1. Ajoutez les fichiers html du projet l'index de Git (espace temporaire) : `git add .`
2. Consultez l'état du répertoire de travail.
3. Valider la version actuelle des pages html « vides » en faisant un commit : `git commit -m "pages vides"`
4. Consultez l'état du répertoire de travail.
5. Consultez l'historique des versions du projet : `git log`

5. Création de la deuxième version du projet

6. Visualisez les deux fichiers html du projet dans un navigateur Internet que vous garderez ouvert tout au long de ce TP.
7. Ouvrez les deux fichiers html dans uhn éditeur de texte html (notepad++ à défaut) pour les développer.
8. Ajoutez le code html suivant dans `dut-info.html` :

```
<h1> DUT Informatique </h1>
Le diplôme est préparé en deux années d'études :
<p> <h2> Première année </h2> <p>
<p> <h2> Deuxième année </h2> <p>
```

9. Consultez l'état du répertoire de travail.
10. Consulter la différence de code du fichier dut-info.html entre le répertoire de travail et la version validée dans le dépôt : git diff dut-info.html
11. Ajoutez le ficher dut-info.html à l'index Git et valider la version la version 2 du projet avec le commentaire « page dut-info avec contenu ».
12. Consulter l'état du dépôt Git du projet.
13. Consultez l'historique des versions du projet : git log
14. Consultez l'historique des versions du projet avec les modifications apportées aux versions : git log -p
15. On voudrait rajouter du code html pour rendre le texte « Première année » un hypertexte cliquable et qui pointe sur la page dut-info-1a.html. Nous simulons une erreur de codage que vous rajouterez volontairement à la page dut-info.html (écrite en rouge suivant) :

    <p> <h2> <a hreg=dut-info-1a.html Première année </h2> <p>
16. Actualisez le fichier dut-info.html dans le navigateur et constatez l'erreur.
17. Consultez l'état du répertoire de travail puis la différence de code entre les versions du fichier dut-info.html.
18. Pour corriger l'erreur en revenant à une version correcte de la page dut-info.html, restaurez sa dernière version (validée) du dépôt Git : git checkout id-version dut-info.html
19. Actualisez la page dut-info.html dans le navigateur et consulter son code dans notepad++. Que constatez-vous ?

6. Création de la troisième version du projet

1. Consultez l'état du dépôt Git.
2. Ajoutez le code html suivant dans dut-info-1a.html :

    <h1> DUT Informatique – 1ère année </h1>

    Modules des semestres S1 et S2 :

    <p> <h2> SGBD </h2> : Bases de données <p>

    <p> <h2> BPO </h2> : Bases de la programmation objet <p>

    <p> <h2> Maths </h2> : Mathématiques <p>
3. Modifiez dut-info.html comme suit :

    <p> <h2> Première année </h2> <p>

    <p> <h2> Deuxième année </h2> <p>
4. Consultez l'état du répertoire de travail.
5. Consultez la différence de code entre le code actuel des deux pages et celui de la dernière version validée.
6. Validez le code des pages du répertoire de travail dans le dépôt Git avec le message « première année avec contenu et lien hypertexte ».
7. Consultez l'historique des validations du projet.

7. Création d'une branche

On voudrait mettre un peu de style (centrer le titre et mettre des couleurs) sans *CSS* dans la page dut-info.html mais on voudrait faire le séparément des fichiers texte de base.

1. Créez la branche Git et appelez-la « branche-style » : git branch branche-style
2. Consulter les branches du projet : git branch
3. Placez-vous (ou activez) la branche branche-style : git checkout branche-style
4. Consultez l'historique des validations du projet sur cette branche. Conclusion ?

5. Modifiez la couleur de fond de la page dut-info.html pour centrer le titre comme suit :

   <body style="background:grey">
6. Actualisez la page dans le navigateur.
7. Validez cette version avec le message « style : couleur de fond de dut-info en gris »
8. Consultez l'historique des validations des versions sur cette branche.
9. Basculez sur la branche « master » et réactualisez la page dans le navigateur et consulter le code la page dans l'éditeur. Constat ?
10. Rebasculez sur la branche « branche-style », reconsultez le code de la page et réactualisez-la dans le navigateur.
11. Consultez la différence de code entre les branches « master » et « branche-style ».
12. Placez-vous sur la branche « master » et intégrer la dernière version de code de la branche « branche-style » :

    git checkout master

    git merge branche-style
13. Consultez l'historique des validations des versions sur les branches « master » et « styles-des-pages ».

8. Gestion de conflits

On va maintenant modifier du code simultanément sur deux branches parallèles de manière à provoquer un conflit que vous résoudrez manuellement.

1. Mettez-vous sur la branche « branche-style » puis centrez le titre comme suit :

    <h1> DUT Informatique </h1>
2. Vérifiez que ce code est correct sur le navigateur et validez-le sur Git avec le message « style : couleur de fond et du texte centré ».
3. Mettez-vous sur la branche « master » puis alignez à droite le titre comme suit :

   <h1 align="right"> DUT Informatique </h1>
4. Vérifiez que ce code est correct sur le navigateur et validez-le sur Git avec le message « style : couleur de fond et du texte centré ».
5. Intégrer le code de la branche « branche-style » à la branche « master ». Que se passe-t-il ?
6. Résolvez le conflit manuellement en gardant le centrage du titre puis validez la version avec le message « style : couleur de fond et centrage du titre ».
7. Consultez l'état du dépôt et l'historique des validations.

9. Git en version graphique

Faites des manipulations sur l'outil graphique de Git « gitk ». Consultez la l'historique des validations du projet et créez-en d'autres.

10. Git en distribué

1. Connectez-vous sur github et créer un compte.
2. Poussez votre projet sur github.

11. Travailler en groupe de deux sur Git

1. Mettez vous par binômes et déposer le projet?? Le dépôt GitLab d'un des binômes.
2. Créez chacun un dépôt Git local : git init
3. Récupérer le projet du binôme sur GitLab : git pull