

TP.0 Première question

Dans une page (format A4 Word ou similaire), décrivez les éléments et les aspects (qui nous avons principalement traité en Intégration WEB et Hébergement) qui se trouvent dans chaque composant :

◦ USER INTERFACE :

- Comprends tous les éléments du navigateur en dehors de la fenêtre d'affichage du site web.
 - Fenêtre Outils de développement = Permet de résoudre les problèmes de code (HTML, CSS, JavaScript js)
 - Différents onglets
 - Barre de favoris
 - Barre d'adresse pour insérer un URI
 - Boutons Précédent et Suivant
 - Options de signets
 - Boutons d'actualisation et d'arrêt pour actualiser ou arrêter le chargement des documents en cours
 - Bouton d'accueil qui vous amène à votre page d'accueil

◦ BROWSER ENGINE :

- Un moteur de recherche est une application permettant à un utilisateur d'effectuer une recherche locale ou en ligne, c'est-à-dire de trouver des ressources à partir d'une requête composée de termes.

◦ RENDERING ENGINE :

- Un moteur de rendu HTML est le composant logiciel de tous les navigateurs permettant d'afficher une page web : il transforme un document HTML, ainsi que toutes les autres ressources associées à la page, en une représentation visuelle interactive pour l'utilisateur. Il est de ce fait le cœur d'un navigateur web.
- La responsabilité du moteur de rendu est bien... Le rendu, c'est-à-dire l'affichage du contenu demandé sur l'écran du navigateur.
- Par défaut, le moteur de rendu peut afficher des documents et des images HTML et XML. Il peut afficher d'autres types de données via des plug-ins ou une extension ; par exemple, afficher des documents PDF à l'aide d'un plug-in de visionneuse PDF. Cependant, dans ce chapitre, nous nous concentrerons sur le cas d'utilisation principal : afficher du code HTML et des images formatées à l'aide de CSS

◦ NETWORKING :

- pour les appels réseau tels que les requêtes HTTP, en utilisant différentes implémentations pour différentes plates-formes derrière une interface indépendante de la plate-forme.

◦ JAVASCRIPT INTERPRETER :

- Un moteur JavaScript est un programme logiciel qui interprète et exécute du code en langage JavaScript. Les moteurs JavaScript sont généralement intégrés aux navigateurs Web.

◦ UI Backend :

- Utilisé pour dessiner des widgets de base tels que des zones de liste déroulante et des fenêtres. Ce backend expose une interface générique qui n'est pas spécifique à la plateforme. En dessous, il utilise des méthodes d'interface utilisateur du système d'exploitation.

◦ DATA Persistance :

- Il s'agit d'une couche de persistance. Le navigateur peut avoir besoin d'enregistrer localement toutes sortes de données, telles que des cookies. Les navigateurs prennent également en charge les mécanismes de stockage tels que localStorage, IndexedDB, WebSQL et FileSystem.
- Les données persistantes dans le domaine du traitement des données dénotent des informations rarement consultées et peu susceptibles d'être modifiées. Les données statiques sont des informations, par exemple un enregistrement, qui ne changent pas et peuvent être destinées à être permanentes.

Nous pouvons commencer à discuter de ces aspects ensemble. L'objectif ici, c'est de comprendre quelle sont les facteurs importants dans chaque module et quelle sont les bonnes pratiques à suivre pour déployer une application optimisée. Individuez aussi des potentiels facteurs de criticité (par exemple qui influence le temps de chargement d'un page, la navigation du site, etc.)

Aidez-vous, avec ces ressources:

- https://developer.mozilla.org/en-US/docs/Learn/Performance/What_is_web_performance
- <https://web.dev/howbrowserswork/>
- <https://developer.mozilla.org/fr/docs/Learn/Performance/CSS>

TP.0 Deuxième question

Si vous ne l'avez déjà, crée un compte (gratuit) Github (<https://github.com/>).

- En suite, créez un entrepôt privé (visible seulement pour le propriétaire et pour les invités si possible) dédié aux cours. Faisons cela ensemble.

- Nous vous demandons de le partager avec les intervenants (michele.linardi@cyu.fr
brahim.harib@u-cergy.fr) et d'ajouter le rendu de chaque TP. Cela n'est pas obligatoire, en revanche, il est fortement conseillé, car nous considérons tous les rendus pour la validation des Apprentissages Critiques !

Références (liens outils)

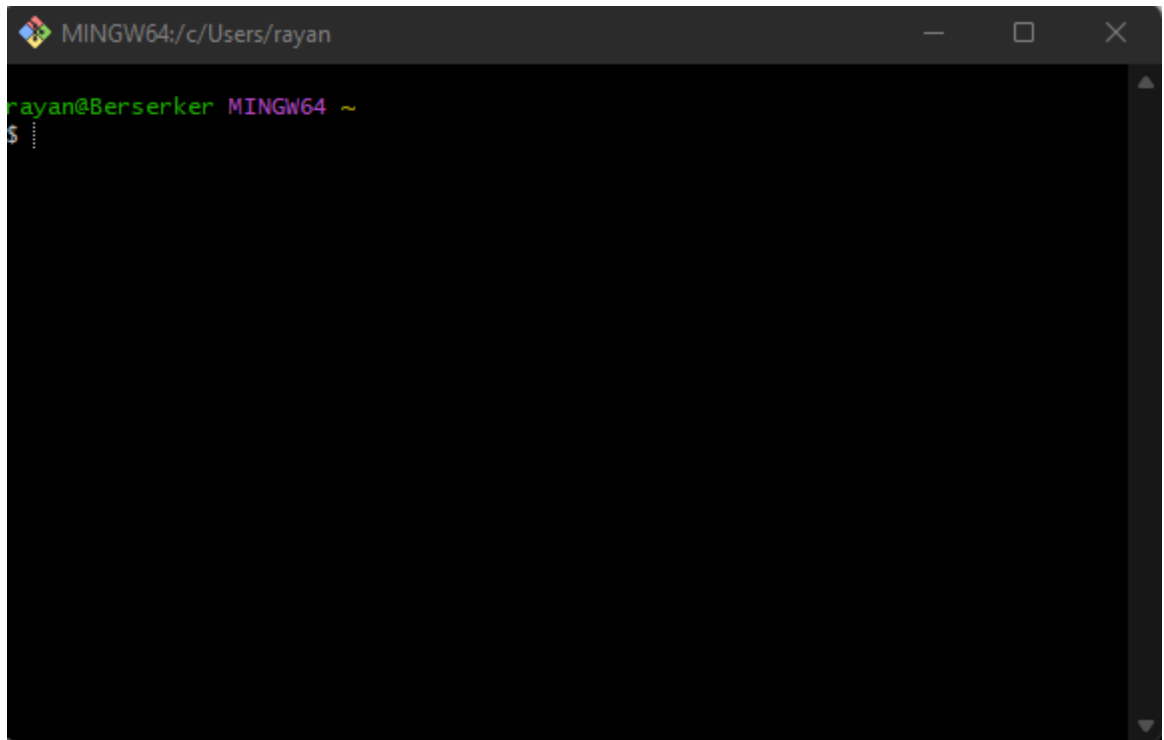
- <https://developer.mozilla.org/>
- <https://www.w3schools.com/>
- <https://web.dev/>
- <https://aws.amazon.com/fr/devops/continuous-integration>
- <https://github.com/>

TP 1 : Pratique GIT

Durant ce TP nous avons utilisé l'outil git dans l'invite de commande

Commandes

→ Ouvrir le terminal Github

A screenshot of a terminal window titled 'MINGW64:/c/Users/rayan'. The terminal shows a prompt 'rayan@Berserker MINGW64 ~' followed by a cursor. The terminal is dark-themed with a light-colored cursor.

→ Repérer l'emplacement du terminale avec la commande « pwd »

- Pwd : repérer l'emplacement du terminale

```
MINGW64:/c/Users/ryan
ryan@Berserker MINGW64 ~
$ Git bash
git: 'bash' is not a git command. See 'git --help'.

The most similar command is
    stash

ryan@Berserker MINGW64 ~
$ pwd
/c/Users/ryan

ryan@Berserker MINGW64 ~
$
```

- `cd` : naviguer dans un dossier précis

```
MINGW64:/c/Users/ryan/Desktop
ryan@Berserker MINGW64 ~
$ Git bash
git: 'bash' is not a git command. See 'git --help'.

The most similar command is
    stash

ryan@Berserker MINGW64 ~
$ pwd
/c/Users/ryan

ryan@Berserker MINGW64 ~
$ cd Desktop

ryan@Berserker MINGW64 ~/Desktop
$
```

- `ls` : montrer tous les dossiers

```
MINGW64:/c/Users/ryan/Desktop/repositories

rayan@Berserker MINGW64 ~
$ pwd
/c/Users/ryan

rayan@Berserker MINGW64 ~
$ cd Desktop

rayan@Berserker MINGW64 ~/Desktop
$ ls
desktop.ini
```

- Mkdir repositories: créer un dossier à l'endroit où se situe le terminal donc création du dossier repositories

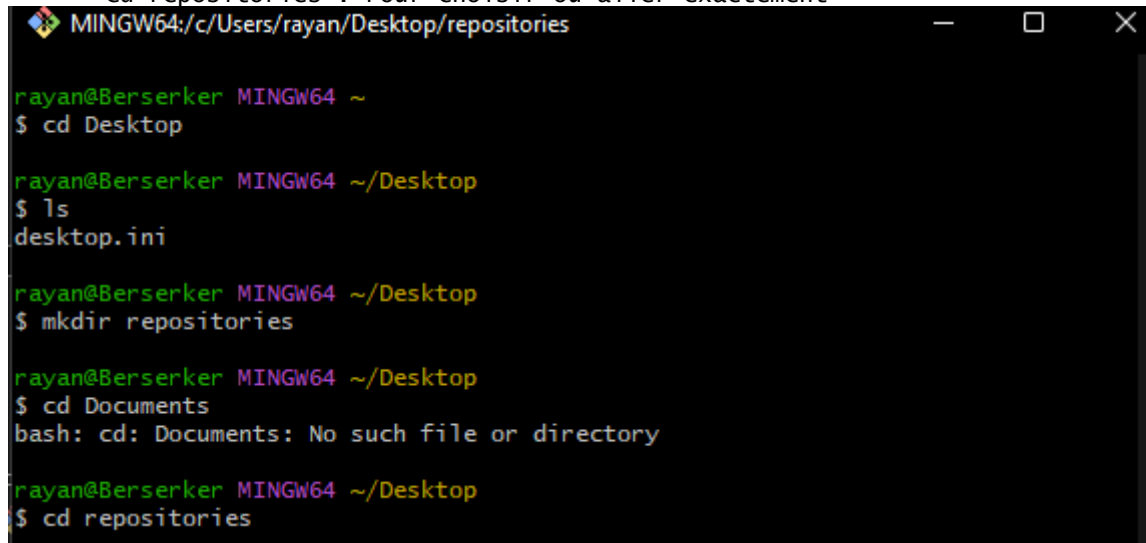
```
MINGW64:/c/Users/ryan/Desktop/repositories
/c/Users/ryan

rayan@Berserker MINGW64 ~
$ cd Desktop

rayan@Berserker MINGW64 ~/Desktop
$ ls
desktop.ini

rayan@Berserker MINGW64 ~/Desktop
$ mkdir repositories
```

- cd repositories : Pour choisir ou aller exactement



```
MINGW64: /c/Users/ryan/Desktop/repositories

rayan@Berserker MINGW64 ~
$ cd Desktop

rayan@Berserker MINGW64 ~/Desktop
$ ls
desktop.ini

rayan@Berserker MINGW64 ~/Desktop
$ mkdir repositories

rayan@Berserker MINGW64 ~/Desktop
$ cd Documents
bash: cd: Documents: No such file or directory

rayan@Berserker MINGW64 ~/Desktop
$ cd repositories
```

Git clone : sert à cloner son entrepôt sur un nouveau répertoire

```
MINGW64:/c/Users/ryan/Desktop/repositories
--separate-git-dir <gitdir>
    separate git dir from working tree
-c, --config <key=value>
    set config inside the new repository
--server-option <server-specific>
    option to transmit
-4, --ipv4
    use IPv4 addresses only
-6, --ipv6
    use IPv6 addresses only
--filter <args>
    object filtering
--also-filter-submodules
    apply partial clone filters to submodules
--remote-submodules
    any cloned submodules will use their remote-tracking b
branch
--sparse
    initialize sparse-checkout file to include only files
at root
--bundle-uri <uri>
    a URI for downloading bundles before fetching from ori
gin remote

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git clone ^C

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git clone https://Rayan-Blbc@github.com/Rayan-Blbc/repoTPDeplService.git

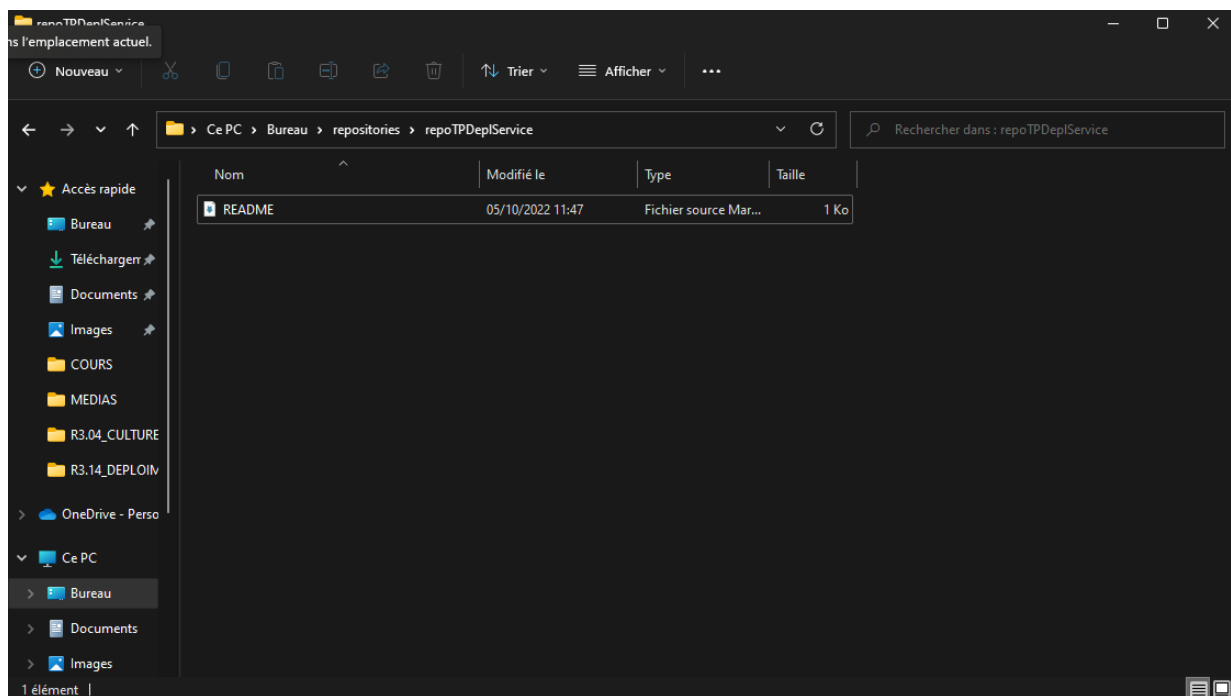
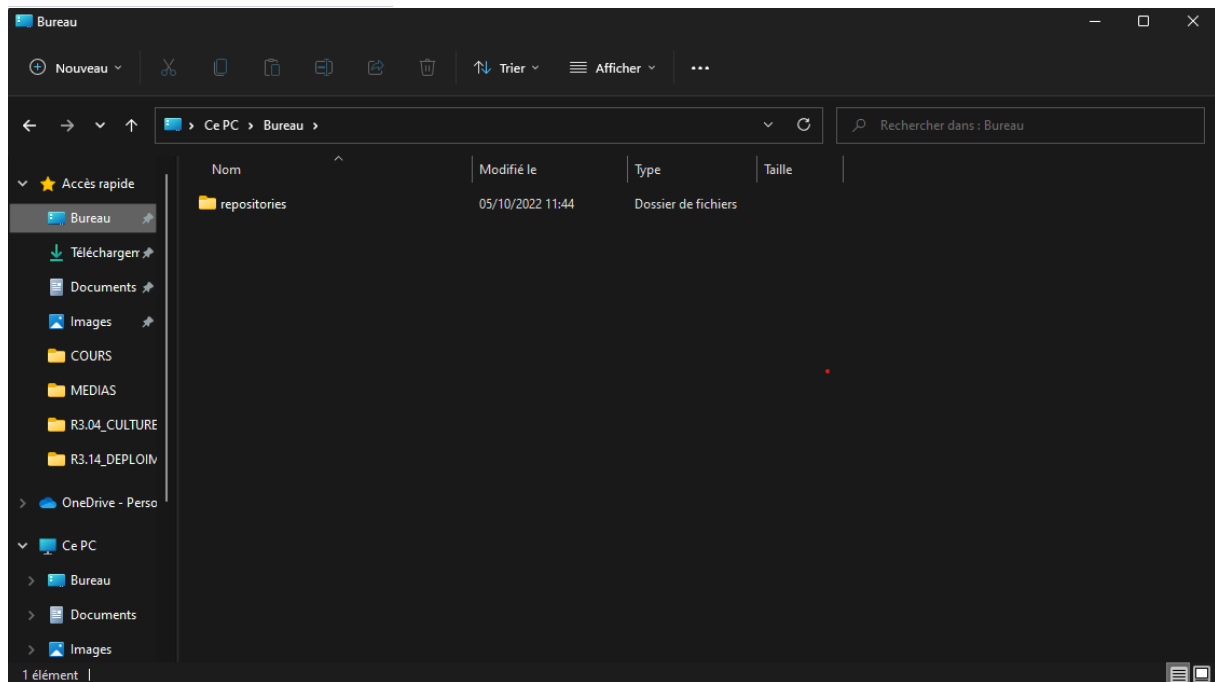
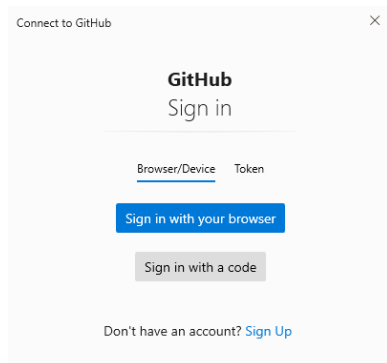
MINGW64:/c/Users/ryan/Desktop/repositories
--also-filter-submodules
    apply partial clone filters to submodules
--remote-submodules
    any cloned submodules will use their remote-tracking b
branch
--sparse
    initialize sparse-checkout file to include only files
at root
--bundle-uri <uri>
    a URI for downloading bundles before fetching from ori
gin remote

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git clone ^C

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git clone https://Rayan-Blbc@github.com/Rayan-Blbc/repoTPDeplService.git
Cloning into 'repoTPDeplService'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

rayan@Berserker MINGW64 ~/Desktop/repositories
$
```

- ne pas coller le https car c'est un entrepôt privé donc il faut mettre son @ avant le « github » dans l'URL
- Le clonage de l'entrepôt est bien fait donc il demande de se connecter.



➔ voilà le dossier créer grâce au terminale de Github

```
MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService
The most similar command is
status

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git add
fatal: not a git repository (or any of the parent directories): .git

rayan@Berserker MINGW64 ~/Desktop/repositories
$ git version
git version 2.38.0.windows.1

rayan@Berserker MINGW64 ~/Desktop/repositories
$ pwd
/c/Users/ryan/Desktop/repositories

rayan@Berserker MINGW64 ~/Desktop/repositories
$ cd repoTPDeplService

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ pwd
/c/Users/ryan/Desktop/repositories/repoTPDeplService

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$
```

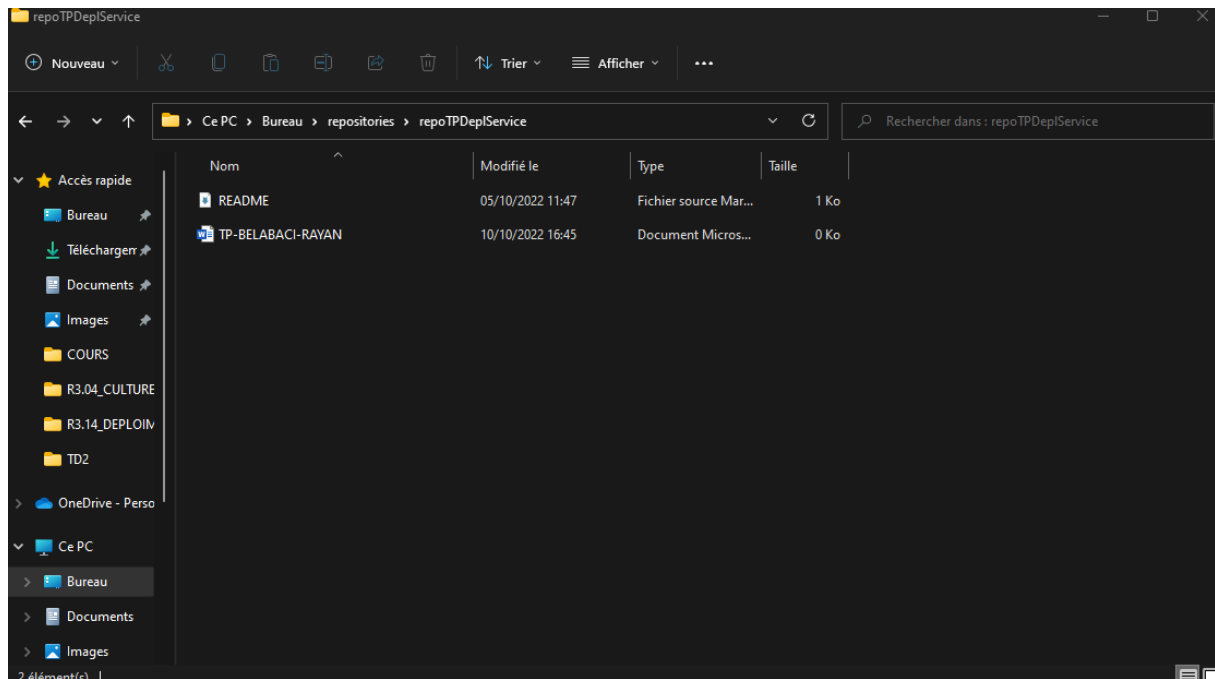
→ Après avoir cloné son entrepôt on va chercher à déposer des fichiers donc on se place dans le dossier « repoTPDeplService »

Git status : Sert à regarder le « statuts » (état) d'un entrepôt git.

```
rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ |
```



→ Création du fichier dans le dossier « repoTPDeplService »

```

MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ AC

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      TP-BELABACI-RAYAN.docx

nothing added to commit but untracked files present (use "git add" to track)

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$

```

Git add. : Permet d'ajouter des éléments qui ne sont pas actualisés (apparaît en rouge). Il ajoute les éléments sans distinction.

→ Ici on peut voir que mon fichier word n'est pas mis à jour

```
MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        TP-BELABACI-RAYAN.docx

nothing added to commit but untracked files present (use "git add" to track)

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git add .

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   TP-BELABACI-RAYAN.docx

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$
```

→ Avec la commande « Git add . » j'ai pu mettre à jour mon fichier dans l'entrepôt

Git commit -m « » : Permet de créer une version à nos fichiers. On peut ajouter une description entre guillemets

```
MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   TP-BELABACI-RAYAN.docx

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git commit -m "Version 1 du TD"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'rayan@Berserker.(none)')

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$
```

→ Demande d'authentification
→ git config --global user.email rayanbelabaci2@gmail.com

```
MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'rayan@Berserker.(none)')

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git config --global user.email "rayanbelabaci2@gmail.com"

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git commit -m "Version 1 du TD"
[main 14af987] Version 1 du TD
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 TP-BELABACI-RAYAN.docx

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ |
```

Git push : Pour mettre à jour les données locales et envoyer dans le « remote repository », c'est-à-dire de l'envoyer sur un serveur (répertoire) en ligne

```
MINGW64:/c/Users/ryan/Desktop/repositories/repoTPDeplService

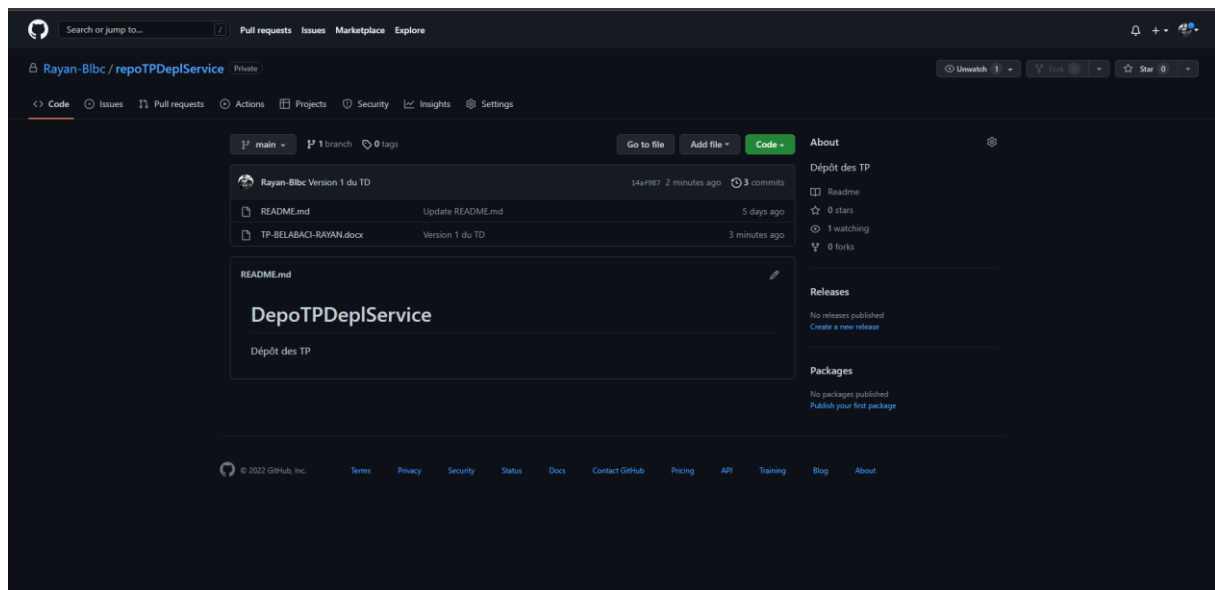
fatal: unable to auto-detect email address (got 'rayan@Berserker.(none)')

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git config --global user.email "rayanbelabaci2@gmail.com"

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git commit -m "Version 1 du TD"
[main 14af987] Version 1 du TD
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 TP-BELABACI-RAYAN.docx

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Rayan-B1bc/repoTPDeplService.git
 a629abf..14af987  main -> main

rayan@Berserker MINGW64 ~/Desktop/repositories/repoTPDeplService (main)
$
```



➔ voilà le fichier apparait bien dans notre entrepôt

Git branch : permet de créer une branche parallèle au tronc ou l'on pourra effectuer des modifications sans modifier les données du main

Git merge : permet de fusionner une branche au main, c'est-à-dire qu'on peut mettre à jour le main avec des modifications apportées à une branche parallèle.

Git checkout : permet de naviguer de branche en branche et de version en version si on donne le nom à la suite

Git log : donne accès au répertoire et l'historique des commits effectués