

## **Travail Pratique2**

**Cours : Environnement Web 1**

**Date de remise : 6 décembre 2022 (5% de moins pour chaque jour de retard)**

**Pondération : 10%**

**Professeur : Lorry James, Encarnacion**

**Objectifs : Mettre en pratique les notions vues à ce jour.**

# Travail pratique 2

## 1. Modalités

Pondération 10% de la note finale

Échéance de remise: 6 décembre 2022 à 23h PM

Travail à réaliser en individuel. Attention, le professeur peut s'il le juge nécessaire poser des questions à chacun étudiants pour s'assurer qu'ils sont les auteurs authentiques du travail remis.

Le but de ce travail est de vous évaluer sur l'ensemble des contenus abordés dans le cadre du cours et de vos capacités à réfléchir pour trouver une solution. Tout plagiat sera sévèrement sanctionné.

## 2. Directives

### **Remise**

Une fois le travail terminé remettre votre fichier sur Léa dans l'espace intitulé «TP2 ».

**Il faut faire des captures d'écrans de chaque étape**

## Télécharger et installer Git sur Debian

Ouvrez votre machine virtuelle Debian et installez Git

```
debian@debian:~$ sudo apt-get update
[sudo] password for debian:
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian bullseye-updates InRelease
Hit:3 http://security.debian.org/debian-security bullseye-security InRelease
Reading package lists... Done
debian@debian:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.30.2-1).
0 upgraded, 0 newly installed, 0 to remove and 43 not upgraded.
debian@debian:~$
```

## Configurer votre identité

Commencez par renseigner votre nom et votre adresse e-mail. **C'est une information importante** car vous en aurez besoin pour toutes vos validations dans Git :

```
debian@debian:~$ git config --global user.name "Michijoe"
debian@debian:~$ git config --global user.email e2296540@gmail.com
```

Pour vérifier que vos paramètres ont bien été pris en compte, et vérifier les autres paramètres, il suffit de passer la commande `git config --list`

```
debian@debian:~$ git config --list
user.name=Michijoe
user.email=e2296540@gmail.com
```

Allez sur [www.github.com](https://github.com) et copier le lien https de votre projet

<https://github.com/Michijoe/web1.git>

Maintenant vous devez cloner le projet dans le dossier **Documents**

```
debian@debian:~$ cd Documents
debian@debian:~/Documents$ git clone https://github.com/Michijoe/web1.git
Cloning into 'web1'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 1), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
```

Quand vous lancez la commande **ls** que voyez-vous :

```
debian@debian:~/Documents$ ls
web1
```

Modifier le contenu du fichier index.html en ajoutant un nouveau paragraphe

```
debian@debian:~/Documents$ nano web1/index.html
```

```
GNU nano 5.4 web1/index.html *
<html>
<head>
  <title>Mon premier projet web</title>
</head>
<body>
  <h1>Introduction à GIT et GITHUB blablabla</h1>
  <p>Nouveau paragraphe</p>
</body>
</html>

I

File Name to Write: web1/index.html
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Ajouter ce fichier dans git et faites un **commit** des changements effectués

```
debian@debian:~$ cd Documents
debian@debian:~/Documents$ ls
web1
debian@debian:~/Documents$ cd web1
debian@debian:~/Documents/web1$ git add index.html
debian@debian:~/Documents/web1$ git commit -m"Ajout paragraphe"
[main e959d45] Ajout paragraphe
 1 file changed, 1 insertion(+)
debian@debian:~/Documents/web1$ git push
Username for 'https://github.com': Michijoe
Password for 'https://Michijoe@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-g
it/about-remote-repositories#cloning-with-https-urls for information on currentl
y recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/Michijoe/web1.git/'
```

Si vous avez une erreur d'authentification. Donner les étapes pour générer un token:

- [Verify your email address](#), if it hasn't been verified yet.
- Dans le coin supérieur droit d'une page, cliquez sur votre photo de profil, puis sur **Paramètres**.
- In the left sidebar, click **Developer settings**.
- In the left sidebar, under **Personal access tokens**, click **Fine-grained tokens**.
- Click **Generate new token**.
- Optionally, under **Token name**, enter a name for the token.
- Under **Expiration**, select an expiration for the token.
- Optionally, under **Description**, add a note to describe the purpose of the token.
- Under **Resource owner**, select a resource owner. The token will only be able to access resources owned by the selected resource owner. Organizations that you are a member of will not appear unless the organization opted in to fine-grained personal access tokens. For more information, see "[Setting a personal access token policy for your organization](#)."
- Optionally, if the resource owner is an organization that requires approval for fine-grained personal access tokens, below the resource owner, in the box, enter a justification for the request.
- Under **Repository access**, select which repositories you want the token to access. You should choose the minimal repository access that meets your needs. Tokens always include read-only access to all public repositories on GitHub.
- If you selected **Only select repositories** in the previous step, under the **Selected repositories** dropdown, select the repositories that you want the token to access.
- Under **Permissions**, select which permissions to grant the token. Depending on which resource owner and which repository access you specified, there are repository, organization, and account permissions. You should choose the minimal permissions necessary for your needs. For more information about what permissions are required for each REST API operation, see "[Permissions required for fine-grained personal access tokens](#)."
- Click **Generate token**.

Copier le token qui a été généré ensuite utilisez le comme mot de passe.

```
debian@debian:~/Documents/web1$ git push
Username for 'https://github.com': Michijoe
Password for 'https://Michijoe@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 359 bytes | 359.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Michijoe/web1.git
688c30a..e959d45  main -> main
```

Une fois le commit effectuer lancer la commande **git log**, que voyez-vous? À quoi sert cette commande selon vous?

```
debian@debian:~/Documents/web1$ git log
commit e959d4586762cf75b5a235044995f56965e81ae3 (HEAD -> main, origin/main, origin/HEAD)
Author: Michijoe <e2296540@gmail.com>
Date: Sat Dec 3 12:24:24 2022 -0800

    Ajout paragraphe

commit 688c30aa7b669ef59ed6c9e2002856a731f82d5c
Author: Michijoe <e2296540@cmaisonneuve.qc.ca>
Date: Wed Nov 30 14:11:14 2022 -0500

    correction html

commit 10dd9526288b4c4eb481f6998c1b17aecbb540f8
Author: Michijoe <e2296540@cmaisonneuve.qc.ca>
Date: Wed Nov 30 14:08:39 2022 -0500

    Premiere version html css
```

La commande **git log** sert à voir l'historique des commit effectués par tous les intervenants sur les différents éléments du projets.

# Création des branches

Maintenant créer les branches tache1 et tache2 :

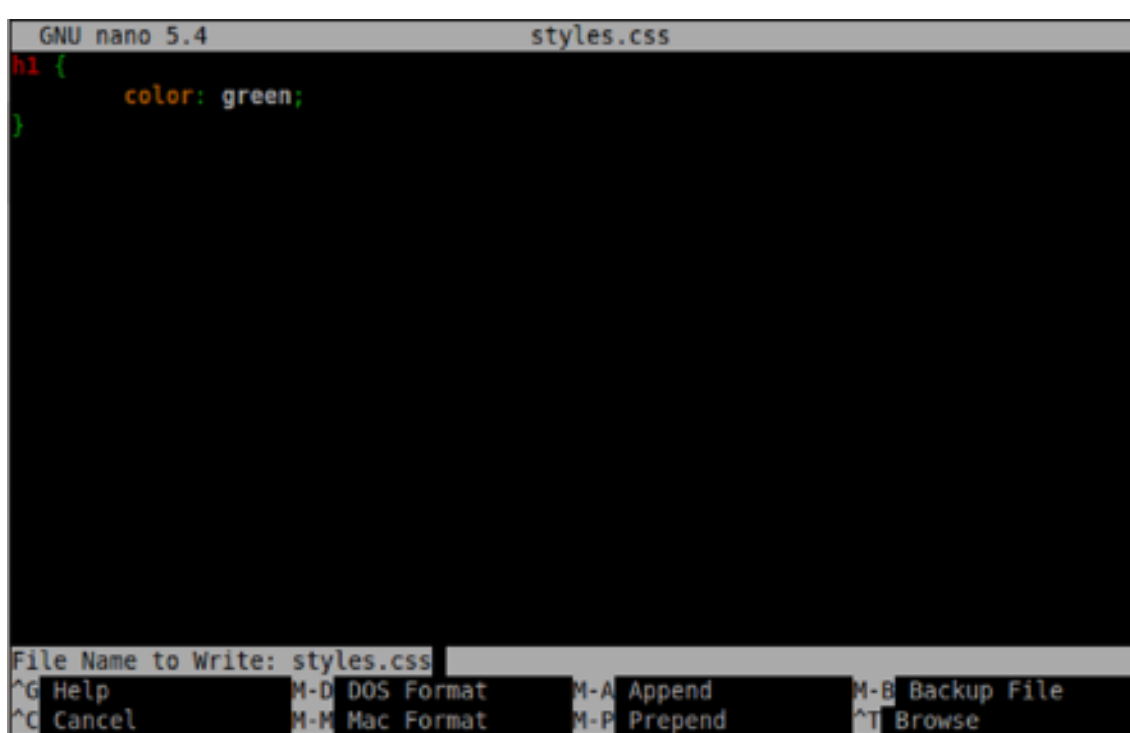
```
debian@debian:~/Documents/web1$ git branch tache1
debian@debian:~/Documents/web1$ git branch tache2
```

Passer à la branche tache1 :

```
debian@debian:~/Documents/web1$ git checkout tache1
Switched to branch 'tache1'
```

Modifier le fichier .css en changeant la couleur à **green** :

```
debian@debian:~/Documents/web1$ nano styles.css
```



```
GNU nano 5.4 styles.css
h1 {
    color: green;
}

File Name to Write: styles.css
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```



Ajouter ce fichier dans git et faites un **commit** des changements effectués

```
debian@debian:~/Documents/web1$ git add styles.css

debian@debian:~/Documents/web1$ git commit -m"changement css couleur vert"
[tache1 d5ad97d] changement css couleur vert
1 file changed, 1 insertion(+), 1 deletion(-)
debian@debian:~/Documents/web1$ git push
fatal: The current branch tache1 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin tache1

debian@debian:~/Documents/web1$ git push --set-upstream origin tache1
Username for 'https://github.com': Michijoe
Password for 'https://Michijoe@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'tache1' on GitHub by visiting:
remote:   https://github.com/Michijoe/web1/pull/new/tache1
remote:
To https://github.com/Michijoe/web1.git
 * [new branch]      tache1 -> tache1
Branch 'tache1' set up to track remote branch 'tache1' from 'origin'.
```

## Fusionner dans Git

Fusionner tache1 dans tache2 :

```
debian@debian:~/Documents/web1$ git checkout tache2
Switched to branch 'tache2'
debian@debian:~/Documents/web1$ git merge tache1
Updating e959d45..d5ad97d
Fast-forward
 styles.css | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Que contient maintenant le fichier .css :

```
debian@debian:~/Documents/web1$ cat styles.css
h1 {
    color: green;
}
```

Dans **github** que remarquez-vous?

On a une nouvelle branche qui contient le dernier changement dans le css

Pouvez-vous simuler un cas de conflit de fusion, et comment le gériez-vous?

Si on modifie la même ligne du css par exemple dans deux branches différentes et qu'on souhaite ensuite les fusionner, on aura un conflit.

```
debian@debian:~/Documents/web1$ git merge tachel
Auto-merging styles.css
CONFLICT (content): Merge conflict in styles.css
Automatic merge failed; fix conflicts and then commit the result.
```

La solution serait de créer une nouvelle branche avant de fusionner les deux existantes.