



OBJECTIFS

Objectif Global

Le principal objectif de ce module est de vous faire découvrir l'une des bonnes pratiques du métier de développeur : versionner son code. Pour cela, nous allons nous baser sur Git, qui est de loin le gestionnaire de versions le plus utilisé au monde. Il permet en particulier deux choses :

- Enregistrer plusieurs versions de son projet, tout en rendant le passage d'une version à l'autre très simple et très rapide.
- Travailler à plusieurs sur un projet.

Il n'y aura plus besoin d'avoir peur de se lancer dans des grandes modifications de code. Même si on casse son projet, on pourra toujours revenir en arrière, sur une version précédente et fonctionnelle.

Fini aussi les copier-coller de tous vos répertoires à chaque modification. Votre super projet `save_the_world` ne prendra plus la forme de 12 copies de répertoires sur votre disque, avec le fameux `save_the_world_version_4.2b+_relu_par_JeanMichel`. Votre répertoire de travail sera propre et organisé.

Objectifs pédagogiques

- Découvrir Git pour versionner son code
- Différencier l'outil Git, de la plateforme d'hébergement de dépôts GitHub
- Utiliser Git pour collaborer sur un projet
- Contourner les limitations de Git pour les métiers de la donnée

Démarche pédagogique

Dans un premier temps, nous allons apprendre les rudiments de Git. Les ressources ne manquent pas sur les internets. Elles sont cependant très orientées vers les métiers du développement. Cela ne doit pas vous freiner. Après tout, Git fonctionne très bien sur les documents textuels, quel que

soit leur contenu. N'ayez donc pas peur de en voyant du HTML/CSS ou d'autres langages dans les exemples. Vous n'avez pas besoin de comprendre le code des exemples, mais juste de constater ce qui change entre deux versions et comment Git constate ces changements.

Après une pratique en solitaire, tout à fait légitime car Git s'utilise même si on est seul sur un projet, nous verrons comment utiliser l'outil pour collaborer. Et comme souvent à partir du moment où il y a plus d'une personne impliquée dans un projet, il va y avoir des conflits ! Git reste calme dans ces cas et en bon médiateur, il vous guidera vers la paisible résolution de conflits.

Nous verrons enfin comment Git peut s'adapter aux métiers de la donnée, grâce à quelques petites adaptations.

COMPÉTENCES

Les compétences à acquérir sont au nombre de 7. Voici la liste :

- Faire un mémo Git
- Différencier Git et Github
- Créer une nouvelle version de son code
- Synchroniser son répertoire local avec Github
- Résoudre des conflits lors d'un merge
- Versionner des fichiers lourds (optionnel)
- Versionner des Jupyter Notebook (optionnel)

Le détail des compétences se trouve sur Campus Skills. N'hésitez pas à y jeter un œil pour savoir comment valider ces compétences.

Dans l'idéal, il faudrait valider les compétences pendant le module. Mais il n'est pas nécessaire de vous mettre trop la pression. Vous utiliserez Git tout au long de la formation. Et la plupart des formateurs pourront valider avec vous les compétences manquantes. Et en dernier recours, il est également possible de valider certaines des compétences à distance.

MODALITÉS

Durée

2 jours, soit 14 heures au total.

Lancement le 04/04/2025 et clôture le 07/04/2025.

Formateurs

Florian Dadouchi

ITÉRATION 1

Premiers pas avec Git

1.1 — Outils de travail collaboratifs

0h30 — Collectif

Commençons sans ordinateur, par un échange où toute la classe participe. La discussion sera guidée par les questions suivantes :

- Comment faites-vous aujourd'hui pour travailler en groupe ?
- Quels outils utilisez-vous ?
- Quels problèmes / limites rencontrez-vous ?

1.2 — Premiers pas avec Git

3h — Individuel

Il est temps de passer à la pratique. Laissez-vous guider par l'excellent cours sur w3schools : <https://www.w3schools.com/git/>. Vous allez vérifier votre installation de Git, la configurer et l'utiliser pour versionner vos premiers codes. Faites toute la première partie « Git Tutorial » et arrêtez-vous avant de démarrer « Git and GitHub ». Si vous ne comprenez pas tout le code HTML, ce n'est pas grave. Ce qui est important, c'est qu'il s'agit de lignes de code au format texte. Exactement comme lorsque vous codez en Python.

N'oubliez pas de démarrer votre mémo des commandes Git.

RESSOURCES

- Le cours complet sur Git à suivre partiellement pour démarrer <https://www.w3schools.com/git/>
- Un très bon cours également avec des exemples en Python <https://realpython.com/python-git-github-intro/>

COMPÉTENCES ASSOCIÉES

- Faire un mémo Git
- Créer une nouvelle version de son code

Livrables

- Le mémo Git complet (à démarrer dès maintenant, mais à mettre en commentaire de la compétence sur Campus Skills une fois le module terminé)
- Une capture d'écran de l'historique d'un dépôt Git avec l'identifiant unique du dernier commit

ITÉRATION 2

GitHub et le travail collaboratif

2.1 – Découvrir GitHub

2h – Individuel

Reprenez le cours de la w3schools. Faites toute la partie « Git et GitHub », à l'exception du chapitre sur GitHub Pages.

Lorsque vous vous créez un compte sur GitHub, souvenez-vous du fait qu'il vous servira tout au long de votre apprentissage et sans doute dans le monde professionnel par la suite (pour contribuer à des projets, montrer vos travaux, etc). Choisissez donc un pseudonyme professionnel. Par exemple, vous pouvez opter pour un classique prénomNom ou <première lettre prénom>Nom. Évitez les surnoms, mais gardez tout de même une place pour hamtaro_le_bg_du_38 dans un coin de votre cœur.

Aussi, rejoignez l'organisation Campus Numérique sur GitHub :

<https://github.com/le-campus-numerique>. Pour cela, envoyez votre identifiant GitHub par Slack au formateur.

NOTE: Après avoir créé votre compte GitHub, créez une clé SSH en suivant 2.2 (plus bas).

Et n'oubliez pas de compléter votre mémo avec les nouvelles commandes vues.

RESSOURCES

- Le cours complet sur Git à poursuivre <https://www.w3schools.com/git/>

COMPÉTENCES ASSOCIÉES

- Différencier Git et Github
- Synchroniser son répertoire local avec Github

Livrables

→ Le lien vers l'un de vos dépôts publics sur GitHub

2.2 – Clé SSH

1h – Individuel

Nous pouvons utiliser des jetons pour l'utilisation de GitHub via le protocole HTTPS. Mais il existe un autre protocole pour assurer cette communication : SSH. Et le protocole SSH peut s'utiliser avec une méthode d'authentification plutôt pratique : une paire de clés. L'authentification se fait avec quelque chose que je possède (une clé) plutôt qu'avec quelque chose que je suis le seul à connaître. Du coup, plus besoin de copier/coller son jeton pour envoyer son code sur GitHub. C'est pratique et ça reste sécurisé !

NOTE

Il y a de nombreux tutoriels en ligne expliquant comment générer des clés RSA. Actuellement, nous privilégions ED25519, plus rapide et plus sécurisé.

RESSOURCES

- La génération d'une clé SSH
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent#generating-a-new-ssh-key>
- L'ajout de la clé SSH sur GitHub
https://www.w3schools.com/git/git_remote_add_ssh.asp?remote=github

2.3 – Collaborer sur un même projet (version facile)

1h – Par îlot

Quand on utilise Git tout seul, les choses se passent en général très bien. Les choses se compliquent un peu quand on veut travailler à plusieurs. Il est rare de travailler à plusieurs sur exactement le même fichier ou notebook dans le monde de la donnée. Ça arrive plus souvent en développement classique. Néanmoins, rare ne veut pas dire jamais. Et il est important de savoir comment réagir en cas de conflit sur un fichier.

Un membre du groupe va *forker* le dépôt sur son espace GitHub depuis celui du Campus Numérique : https://github.com/le-campus-numerique/GIT_pandas-examples

Cette personne va inviter les autres membres de son groupe en tant que collaborateurs du projet. Chacun peut alors cloner le dépôt sur sa machine.

Dans un premier temps, vous allez travailler sur des fichiers différents afin de voir du partage de code sans conflit. Le projet contient 4 fichiers python dont la première ligne ressemble à :

```
__author__ = 'Who ?'
```

Attribuez un fichier par personne et chacun met son nom dans la variable `__author__` puis procède à un commit et à un partage de son code aux autres.

Cette partie est terminée dès lors que le code sur GitHub contient le nom de tous les membres de votre groupe et que votre dépôt local est à jour avec GitHub.

RESSOURCES

- Forker un dépôt https://www.w3schools.com/git/git_remote_fork.asp

2.4 – Collaborer sur un même projet (version difficile)

1h — Par îlot

Maintenant, nous allons artificiellement créer un conflit afin de vous faire voir la résolution de conflits.

Quand deux développeurs commitent des modifications sur la même ligne du même fichier, Git va refuser de fusionner les données (Git veut dire “idiot” en argot anglais). Il n’a pas la prétention de savoir quelle(s) version(s) il doit garder. Est-ce la vôtre ? Celle du collègue ? Ou même un peu des deux ?

Si vous exécutez le programme `Python DataFrame_Groupby.py` vous verrez qu’il y a une erreur : `AttributeError: 'DataFrame' object has no attribute 'sort'`

En cherchant sur internet, trouver la fonction à utiliser à la place de `sort` (elle est dépréciée depuis une certaine version de pandas).

Changez chacun sur votre poste le nom de la fonction et afin de créer un conflit chacun de vous va renommer la variable `sortRatingsField` avec un nom différent. Faites un commit puis synchronisez vos projets. Vous devriez avoir un conflit (sauf le premier à pousser ses changements). Celui qui n'a pas eu de conflit peut réfléchir à comment se créer un conflit avec lui-même 😊

Résolvez le conflit en vous aidant de la ressource donnée.

Compléter votre mémo en reprenant les 4 étapes nécessaires à la résolution de conflits avec Git.

RESSOURCES

- Résoudre un conflit <https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts>

COMPÉTENCES ASSOCIÉES

- Résoudre des conflits lors d'un merge

Livrables

- Le lien d'un dépôt GitHub où chaque participant a fait au moins deux commits (un premier pour introduire une modification et un second pour résoudre le conflit)

2.5 — Aller un peu plus loin

1h — Individuel

Prenez quelques minutes et essayez de répondre aux questions suivantes :

- Il existe d'autres gestionnaires de versions que Git (comme CVS, SVN ou Mercurial). Quels avantages/inconvénients présentent Git par rapport aux autres ?
- Il existe d'autres plateformes que GitHub pour héberger vos dépôts de code. Citez-en au moins 2 autres et comparez-les à GitHub.
- On dit que Git est décentralisé. Qu'est-ce que ça signifie concrètement ?

COMPÉTENCES ASSOCIÉES

- Différencier Git et Github

Livrables

- Une courte explication de la différence entre Git et GitHub avec des alternatives possibles à GitHub

ITÉRATION 3

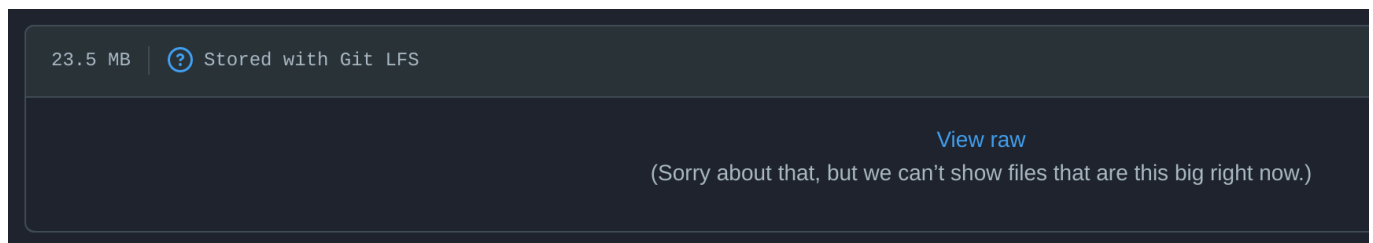
Git et les métiers de la donnée

3.1 — Versionner des données volumineuses avec Git LFS


1h — Individuel

Git n'a pas été conçu pour gérer des fichiers volumineux (binaires ou non). Néanmoins, quand on travaille sur de la donnée, on souhaite parfois « enregistrer » (au sens commiter) une version de son code qui fonctionne avec une version de son jeu de données. Si l'un ou l'autre évolue, il faut pouvoir retracer son historique pour éventuellement revenir à une version antérieure.

En reprenant le projet de l'étape 2.2, utilisez Git LFS pour versionner les fichiers de données. Si tout s'est bien passé, GitHub vous spécifie que le fichier est enregistré avec Git LFS lorsque vous tentez de le visualiser.



RESSOURCES

- Courte vidéo de présentation  [Git LFS \(Large File Storage\) | Learn Git](#)
- Article sur les motivations derrière Git LFS et son utilisation
<https://medium.com/swlh/learning-about-git-large-file-system-lfs-72e0c86cfbaf>

COMPÉTENCES ASSOCIÉES

- Versionner des fichiers lourds

Livrables

- Le lien de votre dépôt sur Github avec des fichiers de données versionnées grâce à Git LFS

3.2 – Versionner des Jupyter Notebook avec nbdime

2h – Individuel

Jupyter stocke les informations de ses notebooks en JSON. Le format JSON représente un arbre de données. Git n'a pas été conçu pour gérer l'historique d'un arbre, mais plutôt du code brut. Il est assez "mauvais" pour faire des `diff` efficaces sur du JSON et génère plus facilement des conflits que dans du code simple. C'est notamment dû aux métadonnées dans le JSON qui peuvent changer à chaque exécution de cellules.

- Reprenez un Jupyter notebook que vous avez. Par exemple, celui du projet sur les arbres.
- Initialiser un dépôt Git dans ce répertoire.
- Faites un premier *commit* du projet.
- Modifiez une ou plusieurs parties de votre notebook et exécutez des parties de votre code (génération de graphiques par exemple ou de simples calculs).
- Consultez grâce à Git les différences sur votre projet depuis votre dernier *commit*.
Qu'observez-vous ?

Voyons à présent comment améliorer le versionnement des Jupyter notebook avec Git, grâce à nbdime :

- Installez nbdime (aidez-vous de la documentation donnée en ressources ci-dessous).
- Configurez git pour qu'il utilise nbdime au moment opportun (encore une fois, aidez-vous de la documentation).
- Une fois que tout est installé, exécutez une nouvelle fois la commande `git diff`.
Qu'observez-vous par rapport à avant ?

RESSOURCES

- La documentation de nbdime <https://nbdime.readthedocs.io/>

COMPÉTENCES ASSOCIÉES

- Versionner des Jupyter Notebook

Livrables

- Une capture d'écran d'un `diff` entre deux versions d'un *notebook* effectué grâce à `nbdiff-web`

NOTE

Versionner des notebooks n'est pas encore une pratique très répandue dans les métiers de la donnée. Ces métiers étant assez récents, des bonnes pratiques sont encore à trouver et à diffuser. Il se peut que votre entreprise n'utilise pas de système de gestion de version. Pas par choix, mais par manque de connaissances. N'hésitez pas à être moteur de sa mise en place dans l'entreprise si vous en sentez le besoin !

ITÉRATION 4 (facultative)

Plus dur, meilleur, plus rapide,
plus fort

△ Cette itération est entièrement facultative. Avant de vous lancer, assurez-vous d'avoir bien compris ce que nous avons vu jusque-là. Maîtriser les bases est important ! Si vous sentez que les notions sont encore un peu floues, retravaillez-les, plutôt que d'en ajouter de nouvelles.

4.1 – Terminer ce qui a été démarré !

2h – Individuel

Vous pouvez maintenant terminer le cours sur w3schools. Testez-vous à l'aide des exercices et du quiz.

RESSOURCES

- Le cours complet sur Git à terminer <https://www.w3schools.com/git/>

4.2 – Il reste tellement à apprendre...

2h – Individuel

Si vous avez encore du temps devant vous, vous pouvez continuer à creuser le sujet et découvrir des notions plus avancées de Git. Il y en a tellement !

RESSOURCES

- L'art de rédiger de beaux messages de commit <https://cbea.ms/git-commit/>
- Un très bon article expliquant Git en détails <https://www.miximum.fr/blog/enfin-comprendre-git/>
- Découvrez de nouvelles facettes de Git <https://realpython.com/advanced-git-for-pythonistas/>
- Rebaser vos commits <https://www.miximum.fr/blog/git-rebase/>
- Écrivez vos README et vos Pull Requests en Markdown <https://docs.github.com/en/get-started/writing-on-github/>
- Le livre Pro Git pour devenir un expert <https://git-scm.com/book/en/v2>

4.3 – Dans les entrailles de Git

4h – Individuel

Vous n'en avez toujours pas assez ? Git est votre nouvelle lubie ? Vous en rêvez la nuit et vous souhaitez tout savoir sur cet outil ?

Le mieux, dans ce cas-là, c'est de voir ce qui se cache à l'intérieur. Comment ce programme est-il codé ? Git est *open source*. Vous pouvez trouver et lire son code source. Cependant, la tâche ne sera pas évidente, sauf si vous savez programmer en C... Heureusement, grâce à la magie d'internet, vous pouvez consulter les ressources ci-dessous pour comprendre la structure de données de Git et coder vous-même, en Python, quelques-unes des sous-commandes de votre nouvel outil de développement préféré.

Coder son propre Git ! Rien de mieux pour comprendre ce qui se passe dans le répertoire caché `.git` lorsque vous entrez une commande `status` ou `commit`.

RESSOURCES

- Notion de mutabilité des objets
<https://towardsdatascience.com/https-towardsdatascience-com-python-basics-mutable-vs-immutable-objects-829a0cb1530a?gi=26828b687dfb>
- Les structures de données de Git
<https://www.slideshare.net/sabativi/deeper-look-intogit?ref=https://reactivic.com/>
- Écrivez votre propre outil Git en Python <https://wyag.thb.lt/>