Document élève

Git : Versionner et collaborer sur du code.

Rédacteur/Formateur

Victor SABATIER - Reactivic / Campus Skills

Objectifs

L'objectif de ce module est de vous donner les rudiments en Git afin de vous permettre :

- de versionner votre code (python ou jupyter notebook)
- de collaborer à plusieurs sur un même code.
- de versionner des fichiers de données correctement.

Il est important de comprendre que Git distingue la partie versionning de la partie collaboration, il est tout à fait pertinent (et même recommandé) d'utiliser Git sur un projet où vous êtes seul.

Prérequis

- Un pc sous Linux.
- Quelques notions basiques du terminal (cd, ls, mkdir …)

Compétences

Voici la liste des compétences que vous trouverez sur Campus Skills. Elles sont à valider dans l'idéal pendant ces deux jours. Maintenant Git sera un outil que vous mettrez en oeuvre pendant le reste de votre formation et certains autres formateurs sont à même de valider ces compétences.

Intitulé	Critère	Preuve de travail
Faire un mémo Git	Le mémo GIT complet (commandes	J'upload mon mémo complet en

	status, diff, add, commit, log, push, pull, clone).	commentaire de la compétence
Différencier Git et Github	Je donne une alternative à Github.	Je l'explique au formateur.
Créer une nouvelle version de son code	Les modifications ont été enregistrées dans l'historique.	Je donne l'identifiant unique de mon dernier commit au formateur
Synchroniser son répertoire local avec Github	Le repo distant contient les modifications de la nouvelle version.	Je montre que le code sur ma machine est le même que sur github
Résoudre des conflits lors d'un merge	Le statut du projet n'est plus en état de merge et il n'y a plus de marqueur de merge.	Le code de mon projet après un merge
Versionner des fichiers lourds	Configurer git lfs sur mon projet.	Je montre le fichier csv des arbres sur github au formateur
Versionner des jupyters notebook	Configurer nbdime sur mon projet.	Je montre un diff d'un notebook versionné sur git au formateur

Programme

Étape 1 - Découverte de git. **4**

Étape 2 - Découverte de Git et Github **6**

Étape 3 - Versionner un projet python avec Git **7**

Étape 4 - Collaborer à plusieurs sur un projet Git. **8**

Étape 5 - Versionner des données volumineuses avec Git LFS **10**

Étape 6 - Versionner des Jupyter notebooks avec Git. **11**

Étape 7 (facultatif)- Commandes avancées sur Git. **13**

Étape 8 (facultatif) - Un peu plus au coeur de Git. **14**

Étape 9 (facultatif) - Git et la blockchain **15**

Étape 1 - Découverte de git.

Modalités

- Durée : 30- 45 min
- Présentation du formateur

Étape 2 - Découverte de Git et Github

Modalités

- Travail individuel
- Durée 2h

Objectifs de l'activité

- Vérifier l'installation de Git sur votre machine.
- Découvrir la plateforme Github et différencier Git et Github.
- Avoir un compte professionnel Github.
- Rejoindre l'organisation Campus Numérique sur Github.

Consignes

- Installer et réaliser Git-it et réaliser les 3 premières étapes : **Get Git -> Commit to it.**
    - Récupérer la dernière version ici git-it
- Créer un compte sur github https://github.com/ : suiviez l'étape 4 de Git-it (**GitHubbin)**
  Attention:
    - Il vous servira tout au long de votre apprentissage et sans doute professionnellement pour contribuer à des projets, montrer vos travaux, ...
      Choisissez donc un **pseudo professionnel**. Exemple <première lettre du prenom><Nom> ou prenomNom, ou …
      PAS de surnom !
- Rejoignez l'organisation Campus Numérique https://github.com/le-campus-numerique en envoyant votre identifiant github par Slack au formateur.
- Vous allez devoir produire un mémo git avec les commandes vues.
- Initier un fichier et compléter le des commandes déjà vues dans cette partie.

Livrable

- Avoir un compte github

Pour aller plus loin

- Il existe d'autres systèmes de versionning que Git (subversion/clearcase/cvs notamment), quels avantages/inconvénients présentent Git par rapport aux autres ?
- On parle notamment de système de version décentralisé pour Git pourquoi ?

Étape 3 - Versionner un projet python avec Git

Avant de versionner des Notebooks, nous allons voir comment versionner du code avec Git, cela nous permettra de découper les difficultés et de voir un cas plus générale de versionning.

Modalités

- Travail individuel
- Durée 2h

Objectifs de l'activité

- Exécuter un programme python.
- Versionner un projet python avec Git et faire son premier commit.

Consignes

- Suivez le tutoriel jusqu'à la section branching (ne pas la faire) : https://realpython.com/python-git-github-intro/#basic-usage
- Suivez le cours 5 *Remote Control* de Git-it.
- Publier le contenu du projet créé grâce au tutorial sur github.
- Compléter votre mémo des commandes vues.

Livrable

- Le lien github de votre repo crée.

Étape 4 - Collaborer à plusieurs sur un projet Git.

Git dans sa partie versionning est relativement simple, les choses se compliquent un peu quand on veut travailler à plusieurs, vous serez très probablement peu amené à collaborer sur le même fichier python ou jupyter notebook. Néanmoins il est important de savoir comment réagir en cas de conflit de la part de Git.

Modalités

- Travail en groupe (3 ou 4)
- Durée 2h

Objectifs de l'activité

- Synchroniser son code.
- Savoir résoudre des conflits avec Git.

Consignes

- Un membre du groupe va forker le repository sur son espace Github depuis celui du Campus Numérique : https://github.com/le-campus-numerique/Example_Pandas
- Cette dernière va inviter les autres membres de son groupe en tant que collaborateurs du projet.
- Chacun peut alors cloner le repo sur sa machine.

Partie 1

Dans un premier temps, vous allez travailler sur des fichiers différents afin de voir du partage de code sans la notion de conflit

- Le projet contient 4 fichiers python dont la première ligne ressemble à

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

- Attribuer 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 repository local est à jour avec Github.

Partie 2

Maintenant, nous allons simuler un conflit afin de vous faire voir la résolution de conflit.

Quand deux développeurs commitent des modifications sur la même ligne du même fichier, Git va refuser de merger 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 votre ? Celle du collègue ou même les deux ?

Si vous exécuter 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 python).

Changer chacun sur votre poste le nom de la fonction et afin de créer chacun de vous va renommer la variable sortRatingsField par un nom différent faites un commit puis synchroniser vos projets. Vous devez avoir un conflit (sauf le premier à pousser ces 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.

Livrable

- Un projet Github avec au moins deux commits par personne.

Ressources

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

Étape 5 - Versionner des données volumineuses avec Git LFS

Git n’a pas été conçu pour gérer des fichiers volumineux (binaires ou non). Néanmoins quand on travaille sur de la données, il peut être nécessaire de vouloir “enregister” (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.

Modalités

- Travail individuel
- Durée 1h

Objectifs de l'activité

- Comprendre quand utiliser Git LFS
- Mettre en place Git LFS sur un projet

Consignes

- En reprenant le projet de l'étape précédente, utiliser GIT LFS pour versionner les fichiers Datas

Livrable

- Votre projet sur Github avec des fichiers de données versionnées grâce à Git LFS

Ressources

- Vidéo courte : Git Large File Storage - How to Work with Big Files
- Article très intéressant sur les motivations : https://medium.com/swlh/learning-about-git-large-file-system-lfs-72e0c86cfbaf

Étape 6 - Versionner des Jupyter notebooks avec Git.

Jupyter stocke sous format JSON ses informations, le format JSON représente un arbre de données. Git n'a pas été conçu pour gérer l'historique d'un arbre, mais plutôt du raw code. Il est assez "mauvais" pour faire des diff efficaces et génère plus facilement des conflits que dans du code simple.

Cette partie est là pour vous apprendre comment "augmenter" Git afin de gérer correctement ce cas.

Modalités

- Travail individuel
- Durée 2h

Objectifs de l'activité

- Comprendre les spécificités d'un jupyter notebook et pourquoi c'est pas une bonne idée de le versionner dans Git "nativement".
- Mettre en place nbtime pour versionner avec Git des Jupyter notebooks.

Consignes

- Reprenez un Jupyter notebook déjà crée (par exemple lors du projet sur les arbres)
- Initialiser un repository Git dedans.
- Faites un premier commit du projet.
- Modifier une ou plusieurs parties de votre notebook et exécuter des parties de votre code (génération de graphiques par exemple ou de simple calculs)
- Consulter grâce à Git les différences sur votre projet depuis votre dernier commit : observations ?

Jupyter stocke sous format JSON ses informations, le format JSON représente un arbre de données. Git n'a pas été conçu pour gérer l'historique d'un arbre, mais plutôt du raw code.

Pour nous aider à faire des diff efficaces sur les notebooks nous allons mettre en place un outil spécialement conçu pour : https://github.com/jupyter/nbdime

- Procéder à son installation : https://nbdime.readthedocs.io/en/latest/installing.html
- Puis sa configuration avec Git : https://nbdime.readthedocs.io/en/latest/vcs.html#git-integration

Une fois fait nous pouvons reprendre notre développement, exécuter une nouvelle fois la commande ***git diff,*** qu'observez vous par rapport à avant ?

Livrables

- Un repo sur votre compte github avec au moins un commit modifiant un notebook.

Ressources

- La documentation de nbdime https://nbdime.readthedocs.io/en/latest/index.html

Notes

Versionner des Notebooks n'est pas encore une pratique très répandue chez les Data Scientists. Le métier étant assez récent, 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 connaissance.
Les Data Scientists vont être de plus en plus intégrer dans les équipes de développement avec tous les outils et méthodologies qui vont avec. Collaborer sur du code est une dimension à laquelle nous avons choisi de vous initier. N'hésitez pas à être moteur de sa mise en place dans l'entreprise si vous en sentez le besoin.

Étape 7 - Commandes avancées sur Git.

Cette partie a pour but de vous faire voir des notions plus avancées de Git (branches, rebase, revert, reset, checkout…)

C'est une partie facultative, le plus important est d'avoir compris le workflow de base de Git.

Modalités

- Travail individuel
- Durée 2h

Objectifs de l'activité

- Coder une version simplifiée de Git avec python.

Consignes

- Poursuivez depuis la partie branching le tuto de l'étape 3 : https://realpython.com/python-git-github-intro/#basic-usage
- Continuez sur le même site la partie avancée : https://realpython.com/advanced-git-for-pythonistas/

Étape 8 - Un peu plus au coeur de Git.

Ne faites cette étape qui si vous êtes parfaitement à l'aise avec l'utilisation de Git (commit, staging area, branches, rebase, merge, revert, reset etc …)

Modalités

- Travail individuel
- Durée 4h

Objectifs de l'activité

- Comprendre le fonctionnement interne de Git.
- Coder une version simplifiée de Git avec python.

Consignes

- Suivre le projet https://wyag.thb.lt/

Ressources

- Structure de données mutables et immutables : https://towardsdatascience.com/https-towardsdatascience-com-python-basics-mutable-vs-immutable-objects-829a0cb1530a
- Présentation sur la structure de données de git : https://www.slideshare.net/sabativi/deeper-look-intogit?ref=https://reactivic.com/

Étape 9 - Git et la blockchain

Cette étape est vraiment facultative et sort complètement du sujet d'apprentissage de Git mais c'est toujours intéressant de faire des ponts entre des technos apparemment éloignées.

Modalités

- Travail en groupe (2, 3 ou 4)
- Durée 2h

Prérequis

- Avoir fait l'étape précédente.
- Avoir lu le papier d'introduction du Bitcoin : https://www.bitcoincash.org/bitcoin.pdf

Quelques questions pour vous aider

- Comment la blockchain garanti l'intégrité et quelle est la différence avec Git ? Quels sont les algorithmes utilisés derrière ? Pourquoi ils sont différents ?
- C'est quoi un commit dans la blockchain du Bitcoin ?
- Quelle pourrait être la preuve de travail dans un projet Git ?

Les remédiations faites pendant le cours

Remédiation - Comment appréhender un projet versionné avec Git (les bons réflexes à avoir)

Remédiation - Clone, fork, archive : quelles différences ?

Remédiation - Un workflow de développement simple avec Git : les 6 commandes à connaître !

Remédiation - Gérer efficacement les conflits : procédure à suivre pour pas se tromper avec Git.