

OBJECTIFS

Objectifs pédagogiques

A l'issue de ce module, vous serez capable d'utiliser un système de gestion de versions (Git) et vous saurez travailler en groupe sur un même projet avec GIT.

Compétences développées

- Savoir différencier Git et GitHub
- Créer une nouvelle version de son code
- Proposer une nouvelle version du projet à ses collègues
- Récupérer la version à jour du projet
- Résoudre des conflits lors d'un merge

Démarche pédagogique (projet, ressources, ...)

- Découverte de git et GitHub par l'intermédiaire d'outils
- Mise en place de git dans son code
- Approfondissement des différentes commandes dans un jeu

MODALITÉS

Durée.

2 jours soit 14 heures au total

Formateur

Julien Marchand

Livrables

Un document - Mémo sur les commandes utilisées, le workflow git, ... **à enrichir à chaque exercice**. Pour valider la compétence associée, **merci de mettre un lien ou d'uploader le livrable en commentaire de la compétence sur Campus Skills**.

Pour les autres preuves de travail, les uploader via Google Drive :

- Sur le drive partagé étudiant de votre promo
- Dans le dossier 7. Git > Rendus
- Dans un dossier de votre prénom-nom

JOUR 1

0 – Introduction

15 min – Travail classe entière

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

1 – Rappel des problématiques

10 min – Travail demi-classe

- Mettre à jour, individuellement, sur un serveur distant le fichier **markdown.md** en ajoutant votre Nom-Prenom.

CONSIGNES

- Connectez vous au Drive de votre promo
- Téléchargez le fichier en local
- Chaque étudiant doit mettre à jour le fichier avec son nom-prenom
- Uploadez le fichier sur le Drive

DEBRIEF

- Durée 5 min

2 – Découverte de la console, Git et GitHub

1h – Travail individuel

- Revoir les commandes de base de navigation dans un terminal ([cd, pwd, ls et ls -l, mkdir, ...](#))
- Mettre en place git sur votre poste de travail
- Découvrir la plateforme GitHub
- Avoir un compte GitHub à usage pro
- Rejoindre l'organisation Campus Numérique sur GitHub

CONSIGNES

- (Windows) Installer git depuis [le site officiel](#)
- Installez [Git-it](#) et réalisez les 4 premières étapes : **Get Git -> GitHubbin**
/!\ Ne pas installer GitHub Desktop comme c'est indiqué dans le tuto

- Créez un compte sur GitHub <https://github.com/>
Attention : ce compte vous servira tout au long de votre apprentissage et sans doute professionnellement pour contribuer à des projets, montrer vos travaux, etc. Choisissez donc un **pseudo professionnel**. Exemple : <première lettre du prenom><Nom> ou prenomNom.
PAS de surnom !
- Rejoindre l'organisation Campus Numérique
<https://github.com/le-campus-numerique>

LIVRABLES

- Avoir un compte GitHub
- Screenshot des 4 premières étapes git-it complétées
- Avoir un terminal sur votre ordinateur et git installé
- Savoir exécuter les commandes de bases (cd, pwd, ls ou ls -al)

RESSOURCES

- Petit Guide : <https://rogerdudler.github.io/git-guide/index.fr.html>
- Jeu pour comprendre les actions git : <https://ohmygit.org/>
- Git doc officielle : <https://git-scm.com/>
- Git community Book <http://alx.github.io/gitbook/index.html>
- GitHub Guides <https://guides.github.com/> + <https://try.github.io/>
- Qu'est ce que git ? <https://www.grafikart.fr/tutoriels/git-presentation-1090>
- Git vs GitHub : <https://jahya.net/blog/git-vs-github/>

3 – Découvrir la Plateforme GitHub

30 min – Travail individuel

- Découvrir à quoi sert GitHub, quelle est la différence avec Git
- Comprendre le GitHub workflow
- Explorer les dépôts ("repository" ou "repo") existants et savoir faire de la veille technologique

CONSIGNES - PARTIE 1

- Se balader sur GitHub, dans l'onglet "Explore"
- Découvrir les projets créés
- Trouver comment choisir objectivement une technologie par rapport à une autre, quels sont les critères du dépôt sur lesquels on peut se baser ?

Exemple : comparer 2 frameworks PHP :

- o <https://github.com/bcosca/fatfree>
- o <https://github.com/symfony/symfony>

CONSIGNES - PARTIE 2

Initiez votre mémo git :

- Créez un nouveau dépôt "memo-git" sur GitHub
- Ajoutez un fichier de type Markdown "README.md", dans lequel vous écrirez le contenu de votre mémo git
- Clonez ce dépôt sur votre machine
- Mettez à jour ce mémo à chaque étape avec les commandes apprises. Assurez-vous régulièrement de synchroniser votre mémo entre votre machine et GitHub.

LIVRABLES

- Avoir créé votre mémo dans un repo personnel

DEBRIEF

- Débrief formateur sur la différence entre git et GitHub
- Découvrir l'intérêt des Stars et de la communauté sur GitHub
- Réclamez votre pack GitHub Education : <https://education.github.com/pack>

RESSOURCES

- Exemples de projets pertinents :
 - o <https://github.com/id-Software/Quake-III-Arena>
 - o <https://github.com/kamranahmedse/developer-roadmap>
 - o <https://github.com/sindresorhus/awesome>
- Qu'est-ce que le markdown et quelle est la syntaxe de ce type de fichier :
<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

4 – Comprendre le workflow git

45 min – Travail individuel

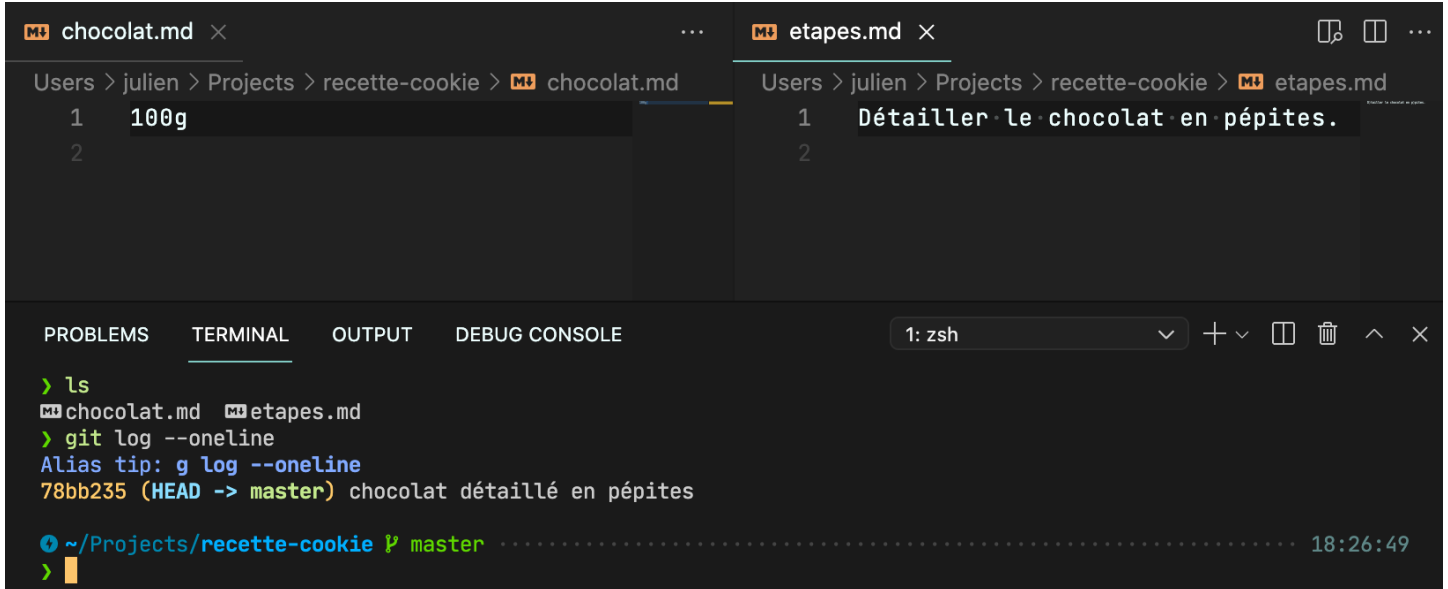
- Comprendre le workflow de git
- Revenir à un état antérieur
- Taguer une version

CONSIGNES - PARTIE 1

- Créez votre premier dépôt sur Github et nommez le **recette-cookie**
- Sur votre ordinateur, connectez-vous à ce dépôt (clone)
- Réalisez la recette à l'aide des ingrédients définis dans le paragraphe ci-dessous :
 - o En créant un nouveau fichier pour chaque nouvel ingrédient. Écrivez le détail (quantité / température / consigne) dans le fichier.

- En créant un fichier dans lequel vous ajouterez chaque étape de la recette que vous êtes en train de suivre
- En réalisant un commit pour chaque étape de la recette, avec la description de l'étape.

Ex pour la 1ère étape :



```
Users > julien > Projects > recette-cookie > chocolat.md
1 100g
2

Users > julien > Projects > recette-cookie > etapes.md
1 Détailler le chocolat en pépites.
2

PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE
1: zsh
> ls
chocolat.md  etapes.md
> git log --oneline
Alias tip: g log --oneline
78bb235 (HEAD -> master) chocolat détaillé en pépites

~/Projects/recette-cookie master 18:26:49
>
```

Recette

- 75g de beurre mou
- 1 œuf
- 85g de sucre
- 1 sachet de sucre vanillé
- 150g de farine
- 100g de chocolat noir
- 1 cuillère à café de sel
- 1 cuillère à café de levure chimique

Réalisation

- Détailler le chocolat en pépites.
- Préchauffer le four à 180°C (thermostat 6).
- Dans un saladier, mettre 75 g de beurre, le sucre, l'œuf entier, la vanille et mélanger le tout.
- Ajouter petit à petit la farine mélangée à la levure, le sel et le chocolat.
- Beurrer une plaque allant au four et former les cookies sur la plaque. Pour former les cookies, utiliser 2 cuillères à soupe et faire des petits tas espacés les uns des autres ; ils grandiront à la cuisson.
- Enfourner pour 10 minutes de cuisson.

LIVRABLES

- Capture d'écran du graph des commits via la commande "git log", à mettre dans le Drive > [Rendu - recette-cookie](#)

CONSIGNES - PARTIE 2

- Une fois le graph validé par le formateur :
 - o Trouvez la commande pour revenir à une étape antérieure (l'étape 3 par exemple). Exécutez-la et regardez le contenu de votre dossier recette-cookie dans l'explorateur de fichier. Que constatez-vous ? Que s'est-il passé ?
 - o Trouvez la commande pour revenir à la dernière version. Que constatez-vous ? Que s'est-il passé dans le dossier ?
- La recette étant terminée, effectuez un tag de cette version en la nommant "v1.0", et poussez le tag sur votre dépôt.

LIVRABLES

- Capture d'écran du tag sur la page dédiée de votre dépôt GitHub, à mettre dans le Drive > [Rendu - recette-cookie](#)

RESSOURCES

- Grafikart <https://www.grafikart.fr/formations/git>
 - o 1er commit : <https://www.grafikart.fr/tutoriels/init-config-add-log-585>
 - o Revenir en arrière : <https://www.grafikart.fr/tutoriels/checkout-revert-reset-586>
 - o Remote : <https://www.grafikart.fr/tutoriels/remote-push-pull-589>
 - o Nommer ses commit : <https://www.grafikart.fr/tutoriels/nommage-commit-1009>
- Tag d'une version : <https://git-scm.com/book/fr/v2/Les-bases-de-Git-%C3%89tiquetage>

5 — Mise à jour avec git (en individuel)

1 heure — Travail individuel

- Avoir son CV sur GitHub
- Mettre à jour votre mémo Git

CONSIGNES

- Vous allez mettre votre CV sur votre GitHub
- Attention aux vidéos ! Les avoir en URL (hébergées ailleurs) ou les ignorer grâce au .gitignore
- Votre CV n'ayant pas été terminé au cours du module HTML, effectuez quelques modifications et les versionner (commit) **par fonctionnalité** :
 - o Ajouter la date de dernière mise à jour du CV dans le footer
 - o Ajouter git comme compétence dans la page des compétences
 - o Mettre à jour la page hobby
 - o Modifier la couleur d'un élément du site
- A la fin, envoyez votre CV sur un nouveau dépôt GitHub
- Mettez à jour votre mémo git

LIVRABLES

- Mémo git à jour
- CV sur votre dépôt avec des modifications versionnées
- Avoir compris les commandes git status, add, commit, push, checkout, log

DEBRIEF

- Débrief formateur sur les bons usages et méthodes

6 – Gérer les conflits (en individuel)

1 heure — Travail individuel

- Avoir compris comment résoudre un conflit

CONSIGNES

- Reprendre le dépôt git **recette-cookie** (cf. étape 4)
Si l'étape 4 n'est pas faite ou pas complète (screenshot valide à l'appui) : la refaire
- Via l'**interface de GitHub**, ajoutez une ligne dans le fichier des étapes de la recette en souhaitant "Bon appétit !", et effectuez un commit sur ces changements
- Sur votre **dépôt local**, ne récupérez pas les changements effectués sur le dépôt distant, mais ajoutez une ligne dans le fichier des étapes de la recette : "Dresser un plat avec les cookies réalisés". Essayez de commit et push ce changement.
- Résoudre le conflit et faire en sorte que le fichier comporte les 2 changements : la dernière étape ainsi que la formule de bon appétit.

LIVRABLES

- Mémo git à jour
- Le repo avec le fichier des étapes de la recette mis à jour
- Le screenshot de votre terminal avec la résolution du conflit dans le dossier de rendu

DEBRIEF

- Débrief formateur sur ce qu'est un conflit et comment le gérer

RESSOURCES

- OC :
<https://openclassrooms.com/fr/courses/2342361-gerez-votre-code-avec-git-et-github/243371-1-resolvez-un-conflit>
- Résoudre les conflits :
<https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line>

BONUS — Quête de la vérité

20 min — Travail individuel

- Bien comprendre le fonctionnement des commits et l'utilité des tags

CONSIGNES

- Cloner le repo: <https://github.com/le-campus-numerique/git-exo-bonus>
- Lire l'énoncé du README.md

LIVRABLES

- Dans votre mémo Git : écrire le résultat de votre quête

JOUR 2

7 – Utiliser git en équipe (mise à jour et conflits)

2 heures – Travail en îlots

- Avoir votre petit site d'entreprise sur GitHub
- Découvrir le workflow pour le travail en équipe
- Gérer des conflits entre le code de différents développeurs

CONSIGNES

- Créer un dépôt pour le site d'entreprise regroupant les 4 CV chez un des développeurs
- **Ajoutez un .gitignore incluant au moins "desktop.ini"**
- Partager le dépôt aux 3 autres développeurs
- Envoyer le site sur GitHub
- Prévoir les évolutions que vous voulez faire sur le site. Si pas d'idée :
 - o À tout hasard sur le SEO, il y a encore des meta descriptions manquantes, etc...
 - o Une page blog/news avec un article rédigé par personne
 - o N'oubliez pas de mettre à jour le menu pour afficher les nouvelles pages !
 - o **Faire valider avec le formateur**
- Pour chaque évolution prévue, créer une issue GitHub et l'assigner à un développeur
- Chaque membre de l'équipe doit développer une des évolutions représentée par une issue. Pour cela, tous les commits associés à une évolution doivent être liés à l'issue correspondante (message du commit comprenant `#idIssue`).
- Donner une version au site une fois toutes les modifications intégrées
- Mettez à jour votre mémo git

LIVRABLES

Dans votre mémo Git : un paragraphe / une explication sur :

- Ce qu'est un conflit
- Comment il se matérialise
- Comment le résoudre

DEBRIEF

- Rappel formateur sur ce qu'est un conflit et comment le gérer

RESSOURCES

- Résoudre les conflits :
<https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/>

- Mots-clés pour lier des commits avec des issues :
<https://help.github.com/en/github/managing-your-work-on-github/linking-a-pull-request-to-an-issue#linking-a-pull-request-to-an-issue-using-a-keyword>

8 – Utilisation des branches (en individuel)

1 heure – Travail individuel

- Savoir créer une branche et la merger
- Comprendre ce qu'est une Pull Request

CONSIGNES

- Reprendre le dépôt git **recette-cookie** (cf. étape 4)
- Nous souhaitons ajouter une explication sur l'historique de la recette des cookies :

Un jeune couple d'aubergistes, Kenneth et Ruth Graves Wakefield, achetèrent une auberge dans la région de Boston, en 1930, la Toll House Inn. Dans le but de séduire sa clientèle, Ruth expérimenta une nouvelle recette de gâteau en mélangeant des morceaux de chocolat Nestlé à sa pâte. Les morceaux de chocolat n'ayant pas fondu, à la surprise de Ruth, sont en réalité à l'origine du succès de cette nouvelle recette.

Nestlé, qui songeait à arrêter la fabrication de ce chocolat, envoya un représentant sur place afin de découvrir la raison du succès local de leur chocolat. Puis André Nestlé et Ruth Wakefield convinrent d'un accord : la firme Nestlé pourrait utiliser la recette de Ruth et le nom de son Auberge du Péage (Toll House Inn), à condition que la recette des cookies soit imprimée sur l'emballage et que Ruth soit approvisionnée en chocolat Nestlé variété.

La Seconde Guerre mondiale, et le brassage de tous les GI venant de différents États des États-Unis, assurèrent la diffusion de cette recette de Nouvelle Angleterre : les cookies, délicieux, hautement énergétiques sous un faible volume, et supportant bien le transport vers les théâtres d'opérations extérieures eurent un énorme succès, et Nestlé en bénéficia.

- Via l'interface de GitHub, créer une issue concernant l'explication à ajouter au projet
- Sur le dépôt local, créez une nouvelle branche "development" dans laquelle vous ajouterez un nouveau fichier avec l'historique de la recette. Le message du commit doit comprendre "fixes #<idIssue>" pour fermer l'issue une fois que le commit est poussé sur GitHub
- Une fois le développement fini et les modifications "push" sur la branche, ouvrir votre repo sur GitHub et :
 - Créez une Pull Request de votre branche de développement vers master / main
 - Mergez la Pull Request, pour fusionner la branche de développement avec la branche master / main

LIVRABLES

- Mémo git à jour

- Le dépôt avec le nouveau fichier comprenant l'historique de la recette

RESSOURCES

- Utiliser des branches : <https://www.grafikart.fr/tutoriels/branch-merge-587>
- Github Flow : <https://guides.github.com/introduction/flow/>

8 – Un workflow type utilisant les branches

45 min – Travail individuel

Comprendre le workflow git en utilisant les branches.

CONSIGNES : ETAPE 1

Rendez vous à cette adresse <https://git-school.github.io/visualizing-git/> et suivez les instructions ci dessous (pour l'exercice les commits sont "vides", il n'y a que les commandes git à faire) :

- Votre projet a déjà un peu d'existant alors :
 - Créez 3 commits en les nommant **C1 - C2 - C3**
- On vous demande de travailler sur une évolution :
 - Créez une nouvelle branche **feature-1**
 - Basculez sur cette branche et créez 3 commits **C4 - C5 - C6** représentant le développement de cette évolution
- Une évolution plus urgente (correction d'une traduction) est priorisée. A vous de la réaliser :
 - Depuis le dernier commit master / main
 - Créez une branche **feature-2** et faites un commit **C7**
- La traduction est validée et doit être intégrée en production :
 - Mergez **feature-2** sur master / main
- Vous pouvez continuer à développer vos modifications sur **feature-1** :
 - Faites un commit **C8**
- On arrête tout : il y a un BUG urgent sur l'application. Corrigez-le :
 - Revenez sur master / main
 - Faites un commit **FIX-1**
- Vous avez oublié les tests sur votre développement en cours :
 - Revenez sur **feature-1** et faites un commit **C9**
- La **feature-1** est validée :
 - Mergez la branche sur master / main

CONSIGNES - ETAPE 2

On a oublié de mettre des versions sur les commits importants :

- Revenez au commit **C3** et taguez le en **V1.0**
- Mettre une version **V1.1** au commit **FIX-1**

- Mettre un tag **V2.0** au commit issu du merge entre **feature-1** et master / main

CONSIGNES - ETAPE 3

On vous demande de créer une modification (**feature-3**) à partir de **V1.1** mais vous oubliez de créer la branche et faites un commit **C10** :

- Rendez vous sur **V1.1**
- Faites un commit **C10**
- Dans le livrable :
 - Expliquer pourquoi cette situation n'est pas souhaitable (faite une recherche sur le terme Detached Head)
 - Comment résoudre ce problème?
- Comment résoudre ce cas de figure :
 - Créez une branche **feature-3** à partir du commit détaché **C10**
 - Vous pouvez maintenant merger votre **feature-3** sur master / main

LIVRABLES

- Un screenshot du graph créé sur [Visualizing Git](#)
- Livrable à jour

9 — .gitignore

30 min — Travail individuel

- Approfondir le .gitignore
- Savoir enlever de l'index un fichier/dossier tracké par git

CONSIGNES

- Dans votre repo **recette-cookie**, créez deux fichiers :
 - **.password-prod** (Contenu : un mot quelconque)
 - **.password-dev** (Contenu : "localhost")
- Poussez sur GitHub
Allez voir sur votre repo GitHub : woops 😱 vos mots de passe sont maintenant en ligne !
- S'il n'existe pas déjà, créez un fichier **.gitignore** à la racine de votre projet, et ajoutez-y le nom du fichier que vous ne souhaitez pas partager (que git ne prenne pas en compte) :
 - **.password-prod**
- Faire un commit pour prendre en compte le changement sur .gitignore et poussez sur votre repo
- Problème : le fichier password est toujours présent sur le repo GitHub ! Comment faire pour que git ne les index plus ?

LIVRABLES

- Repo **recette-cookie** à jour :
 - sans le **.password-prod**
 - avec la commande permettant de désindexer un fichier dans le README.md

RESOURCE

- .gitignore : <https://git-scm.com/docs/gitignore>
- générer un .gitignore déjà complété : <https://www.gitignore.io/>
- <https://perhonen.fr/blog/2015/03/exclure-fichiers-depot-git-gitignore-1476>
- supprimer un fichier déjà commité : <https://ruedelinfo.com/git-plus-suivre-fichier-supprime-pas/>

10 – Git et Agile

30 min – Travail en îlots

- Découvrir les **Pull Requests**, les **Milestones** et autres outils GitHub facilitant la collaboration

CONSIGNES

- Trouvez un dépôt utilisant les fonctionnalités de collaboration (par ex. <https://github.com/vuejs/vue>)
 - Naviguez dans l'interface de GitHub, et identifiez la partie Wiki, les Issues, les Pull Requests, les Milestones, les Projects et essayez de voir à quoi cela pourrait servir lorsque vous travaillerez à plusieurs sur le projet fil rouge
- ➡ Mettre en lien avec le cours de Gestion de Projet Agile

RESSOURCES

- Les issues sur GitHub : <https://guides.github.com/features/issues/>
- Un Kanban dans GitHub : <https://help.github.com/articles/about-project-boards/>
- <https://help.github.com/articles/creating-a-project-board/#creating-a-repository-project-board>

BONUS – Clé SSH

20 min – Travail individuel

- Mettre en place des clés ssh pour éviter de renseigner son login/pwd à chaque push

CONSIGNES

- Suivre les étapes de l'aide pour définir une clé ssh liée à votre compte

RESSOURCES

- Mise en place de clés ssh :
<https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/> ou
<https://youtu.be/D5QGILM1j20?t=216>

“Vous en voulez encore ?” – Site e-commerce

2 heures — Travail en îlots

Le but de cette activité est de récupérer un projet existant avec déjà un historique et de le faire évoluer en équipe.

Toutes les compétences à valider seront mises en œuvre, profitez-en.

CONSIGNES

- Rendez-vous à cette adresse et suivez les instructions présentes dans le fichier README.md :
https://github.com/le-campus-numerique/GIT_ecommerce

LIVRABLES

- Le dépôt sous GitHub de votre projet collectif
- le fichier **readme.md** mis à jour avec vos réponses dans le dossier **rendu**

RESSOURCES

- Workflow git complet : <https://tutorialzine.com/2016/06/learn-git-in-30-minutes>
- Branche et Merge :
http://alx.github.io/gitbook/3_usage_basique_des_branches_et_des_merges.html
- Pull Request : [Grafikart Fork et PullRequest](#)
- Exercices Visuels : <https://learngitbranching.js.org/>
 - Onglet Main > Séquence d'introduction : 1, 2 et 3
 - Onglet Main > Montée en puissance : 1, 2 et 4
 - Onglet Remote > Push & Pull -- dépôts git distants
- CheatSheet :
<https://zeroturnaround.com/rebellabs/git-commands-and-best-practices-cheat-sheet/>