| DÉVELOPPEUR  WEB&APPLIS  **2023** | GIT ET GITHUB  KIT APPRENANT |  |
| --- | --- | --- |

Day 2

Durée.

1 jours

**Objectifs**

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.

Les compétences à acquérir pendant ce module :

● Savoir différencier Git et Github

● Donner le rôle de la staging area avec GIT

● Proposer une nouvelle version du projet à ses collègues

● Revenir à une version antérieure du projet

● Récupérer la version à jour du projet

● Créer une nouvelle version de son code

● Résoudre des conflits lors d'un merge

**Prérequis**

● Day 1

**Livrable**

● 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.

● Différents aperçus écrans

● …(cf exos partie livrable)

Page 1/8

Git et Github

**Introduction**

**Modalités**

● Durée : 10- 15 min

● Travail classe, ordinateur éteint.

**Tour de table : Depuis 1 mois :**

● Quelle est votre utilisation de git?

● Maîtrisez vous le workflow : add-commit-push / pull

● Quels problèmes avez vous rencontrés ?

● Avez vous travaillé à plusieurs / eu des conflits à gérer ?

● Avez vous envie de revoir certains points au cours de la journée ? / voir des notions identifiées avec vos usages ?

● ….

**Etape 0: Quête de la vérité**

**Modalités**

● Travail individuel

● Durée 10min

**Objectifs de l’activité**

● 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

**Étape 1 - Gérer les conflits (en individuel)**

**Modalités**

● Travail individuel

● Durée 20min

**Objectifs de l’activité**

● Avoir compris comment résoudre un conflit

Page 2/8

Git et Github

**Consignes**

● Reprendre le dépôt Git de l’étape 4 **recette-cookie**

**Si l’exo de Git 1 n’est pas fait ou pas complet (screenshot valide à l’appui): Le refaire**

● Via l’interface de GitHub, Ajouter une ligne dans le fichier des étapes de la recette en souhaitant bon appétit, effectuer un commit sur ces changements ● Sur le dépôt local, ne pas récupérer les changements effectués sur le dépôt distant mais ajouter une ligne dans le fichier des étapes de la recette en ajoutant une dernière étape pour dresser un plat avec les cookies réalisés. Essayer de commit et push ce changement.

● Résoudre le conflit et faire en sorte que le fichier comporte la dernière étape ainsi que la formule de bon appétit.

**Livrables**

● Mémo git à jour

● Le dépôt avec le fichier des étapes de la recette mis à jour

● Le screenshot valide dans le dossier de rendu

**Ressources**

● OC

https://openclassrooms.com/fr/courses/2342361-gerez-votre-code-avec-git-et-githu b/2433711-resolvez-un-conflit

● Résoudre les conflits :

https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/

**Étape 2 - Utilisation des branches (en individuel)**

**Modalités**

● Travail individuel

● Durée 20min

**Objectifs de l’activité**

● Savoir créer une branche et la merger

● Comprendre ce qu’est une Pull Request

**Consignes**

● Reprendre le dépot Git de l’étape 6 (la précédente) **recette-cookie** Page 3/8

Git et Github

● Nous souhaitons rajouter 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 recette2.*

*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 notre explication à rajouter au projet

● Sur le dépôt local, créer une branche et développer l’ajout de l’historique de la recette sur cette nouvelle branche. Le message du commit doit comprendre `fixes #<idIssue>` pour fermer l’issue une fois que le commit soit passé

● Une fois le développement fini et les modification push sur la branche ● Sur github

○ Effectuez une Pull Request de la branche vers le Master

○ Merger la branche de développement avec la branche master

**Livrables**

● Mémo git à jour

● Le dépôt avec le fichier d’historique de la recette

**Ressources**

● Utiliser des branches : https://www.grafikart.fr/tutoriels/branch-merge-587 ● Git Flow :

https://www.atlassian.com/fr/git/tutorials/comparing-workflows/gitflow-workflow **Étape 3 - Mise en place d’un workflow**

**Modalités**

● Travail Individuel

● Durée 45min

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 Page 4/8

Git et Github

les instructions ci dessous

● Votre projet a déjà un peu d’existant alors

- Créer 3 commits en les nommant C1 - C2 - C3

● On vous demande de travailler sur une évolution

- Créez une branche en la nommant feature-1

- Basculez sur cette branche et créez 3 commits C4 - C5 - C6

représentants la réalisation de cette évolution

● Une évolution un peu plus urgente et rapide (correction d’une traduction) est priorisée et à réaliser

- Depuis le dernier commit Master

- Créez une branche feature-2 et faites un commit c7

● feature-2 est validée et doit être intégrée

- Mergez feature-2 sur master

● 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. Il faut le corriger - Revenez sur Master

- Faites un correctif FIX-1

● Vous avez oublié les tests sur votre évolution

- Revenez sur feature-1 et faites un commit C9

● La feature-1 est validée mergez La branche sur Master

**Etape 2**

On a oublié de mettre des versions sur les commits importants

- Revenez au commit C3 et tagez le en V1.0

- Mettre une version V1.1 au commit FIX-1

- Mettre un commit V2.0 au commit issu du merge entre feature1 et master

**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:

- créer une branche feature-3 à partir du commit *détaché* C10

- vous pouvez maintenant merger votre feature-3 sur Master.

Page 5/8

Git et Github

**Livrable**

● Un screenshot du graph crée

● Livrable a jour

**Etape 4 - .gitignore**

**Modalités**

● Travail individuel

● Durée 30min

**Objectifs de l’activité**

● Approfondir le .gitignore

● Savoir enlever de l’index un fichier/dossier tracké par git

**Consignes**

● Dans votre repo recette-cookies, créez des fichiers

○ *.password-prod (Écrivez un mot quelconque)*

○ *.password-local (Ecrivez localhost)*

● Pushez sur Github

Mince! Vos mot de passe sont maintenant en ligne!

● Rajouter ou créer si il n’existe pas un fichier .gitignore dans le dossier root de votre projet

● Ajouter les fichiers que vous ne souhaitez pas partager (que git ne prennent pas en compte) : .password-prod

● faire un commit pour en compte le .gitignore et pushez sur votre repo Problème, vous fichiers de password sont toujours présent! Comment faire pour que git ne les index plus?

**Livrable**

● Repot cookie a jour

○ sans le .password-prod

○ Avec la commande permettant de désindexé dans le Reade.md

**Resource**

● .gitignore : https://git-scm.com/docs/gitignore

● générer .gitignore : https://www.gitignore.io/

● https://perhonen.fr/blog/2015/03/exclure-fichiers-depot-git-gitignore-1476 ● https://ruedelinfo.com/git-plus-suivre-fichier-supprime-pas/

Page 6/8

Git et Github

**Étape 5 - Git et Agile**

**Modalités**

● Travail en ilot de 3 ou 4

● Durée: 1h

**Objectifs de l’activité**

● Découvrir les **Issues** au sein de Github et effectuer un commit avec le numéro de l’issue associé

● Découvrir les **Millestones**

● Découvrir le Kanban

**Consignes Etape 1 (individuelement 30 min)**

● Naviguer dans l’interface de github au sein d’un dépot, cherchez des ressources ● Identifiez la partie Wiki, les Issues, les Millestones, le Kanban, et 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

**Consignes Etape 2 (en groupe 30 min)**

● Mettez en place tout l’environnement de gestion de projet Agile par rapport au story mapping constitué sur le cours Agile(milestone, issues, Kanban, …)

**Livrable**

● Screenshot des milestones

**Étape 6 - Site ecommerce**

**Modalités**

● Travail en ilot de 3 ou 4

● Durée: 2h

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 oeuvre, profitez en.

**Consignes**

● Constituer des équipes de 3 ou 4 personnes.

● Rendez vous à cette adresse

Page 7/8

Git et Github

https://github.com/le-campus-numerique/git-ecommerce : et suivez les instructions présent dans le fichier README.md.

Seulement Partie 1 : 3, 4, 5

partie 2

**Livrable**

● Le dépôt sous github de votre projet collectif.

● le fichier readme.md dans le dossier **rendu**

**Etape Bonus - clé SSH**

**Objectifs de l’activité**

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

**Consignes**

- Suivez les ressources suivantes pour définir une clé ssh liée à votre compte.

**Resource**

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/D5QGiIM1j20?t=216

**Continuer à s’entrainer**

Pour s’entrainer et continuer à progresser : https://learngitbranching.js.org/?locale=fr\_FR Page 8/8