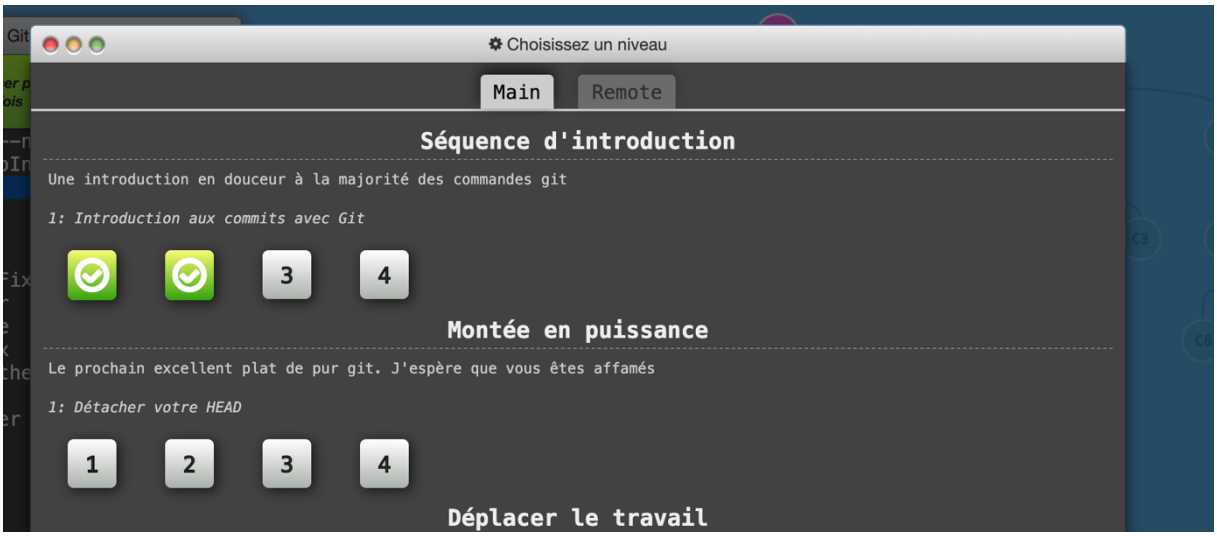
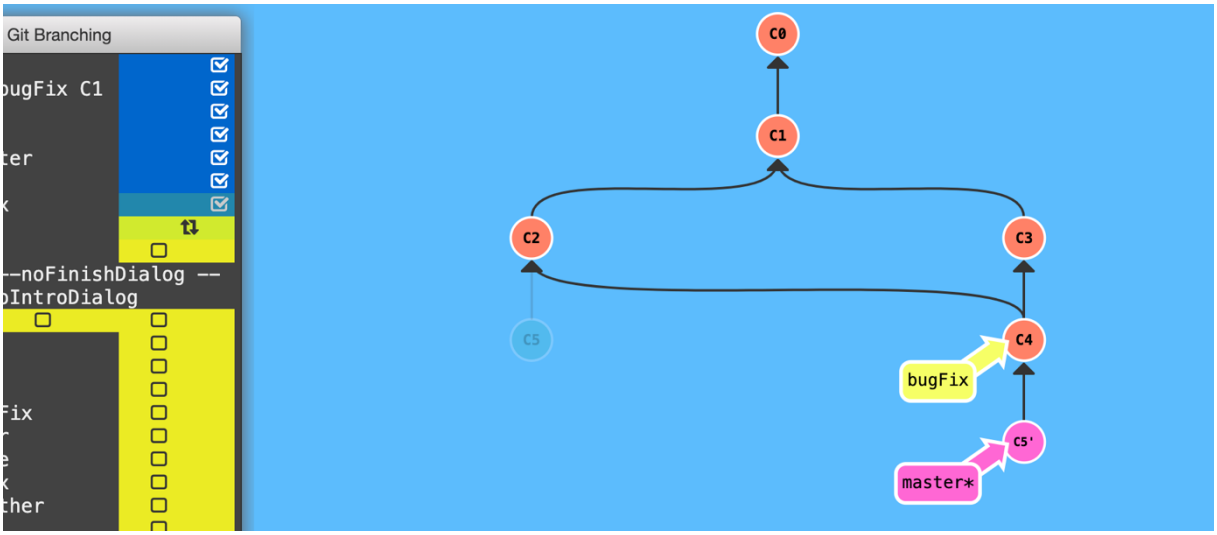
Tutoriel : Apprendre à utiliser Git avec le didactiel en ligne LearnGitBranching

Ce TD vous propose de suivre un didactiel en ligne qui récapitule les principales fonctionnalités proposées par Git. Une partie explique les fonctionnalités de Git ainsi que le concept des branches de manière locale, tandis que l'autre vous propose de simuler un travail sous Git avec un accès à un repository distant et des collaborateurs virtuels.





Pré-requis 1 : Aller sur le site Web : https://learngitbranching.js.org/?locale=fr_FR et démarrer le tutoriel !

Le tutoriel vous permet de vous familiariser avec les commandes ci-dessous avec une interface graphique vous permettant de voir ce qui se passe avec les différentes commandes.

Commandes principales de Git vues grâce au tutoriel :

- Commit
- Branch
- Checkout
- Cherry-pick
- Reset
- Revert
- Rebase
- Merge
- Mais aussi : Pull/Fetch/Push,….

Dans le tutoriel didactique vous pouvez à tout moment utiliser les commandes suivantes pour vous aider :

- Pour obtenir une aide tapez : « *hint* »
- « *Undo* » permet d'annuler la dernière commande saisie et validée
- « *Reset* » permet de réinitialiser l'exercice
- Attention : N'oubliez pas que tout comme avec git classique, vous pouvez à tout moment utiliser les commandes suivantes :
  - «*git status* » pour voir ou vous en êtes. **Remarque** : Graphiquement vous serez au niveau de l'étoile « * ».
  - «*git log*» pour avoir l'historique des versions et actions effectuées

Instructions :
Suivre le didactiel du début à la fin, et, pour chaque partie, répondre aux questions fournies ci-dessous. Faîtes des réponses courtes et concises.

Partie 1 : Introduction

1. **Introduction aux commits avec Git**
   - Que fait-un commit ? *Les étapes chronologique d'un projet*
   - Résumer les actions réalisées :

   Git commit

2. **Gérer les branches avec Git**
   - Quelle commande permet de créer une nouvelle branche ? *git branch*
   - Quelle serait la commande pour créer la branche « BrancheToto » ? *git branch BrancheToto*
   - Pourquoi est-on toujours sur la branche Master si on effectue un commit après la création d'une branche ? *parce que il faut faire être sur la nouvelle branche quand on crée la branche*
   - Quelle commande permet de se positionner sur la nouvelle branche ? *git checkout*

3. **Faire des « merge » (Fusion de branches) avec Git**
   - Comment récupérer le travail effectuer (et le fusionner) dans la branche bugFix dans la branche Master ? *En faisait la commande merge qui fusionne deux parents : git merge bugFix*
   - Qu'aurait-il fallu faire pour effectuer le contraire, c'est-à-dire fusionner Master dans bugFix (pour que bugFix devienne la branche principale) ? *git checkout bugfix ; git merge main*

4. **Introduction à « rebase »**
   - Quelle est la syntaxe de la commande « rebase » ? *git rebase [name]*
   - Quelle est la différence entre le « rebase » vu dans cet exercice et le « merge » de l'exercice précédent ? *Il replace un commit dans le projet pour qu'il soit linéaire, le merge permet de fusionner*

Partie 2 : Montée en puissance

1. **Détacher votre Head**

- Que veut dire HEAD dans git ? *C'est le nom symbolique du commit du projet*

- Comment positionner la « HEAD » sur un commit donné ? *Il est généralement sur une branche qui suit un commit*

2. **Références relatives (^)**

- Que signifie le « ^ » dans git ? Montrer comment il est utilisé avec un exemple. *Il permet de revenir d'un commit en arrière : git checkout bugfix^*

- Que voudrait-dire master^^ ? *revenir deux fois en arrière*

- Avec quelle commande repasser à la Head à l'avant dernier commit de bugFix ? *git checkout HEAD*

3. **Références relatives « ilde » (~<num>)**

- Que signifie le «~ » dans les commandes Git ? Montrer comment il est utilisé avec un exemple. *Elle permet de bouger d'un nombre de commit illimité : git checkout HEAD~4 (cela va remonter de 4 commit)*

- Comment feriez-vous pour déplacer la HEAD de trois « commit » en arrière ? *git checkout HEAD~3*

4. **Annuler les changements avec Git**

- Avec quelle commande annuler un changement de commit local ? *git reset*

- Avec quelle commande annuler un changement de commit distant et le redistribuer aux autres ? *git revert*

Partie 3 : Déplacer le travail. Modifier l'arbre Git.

1. **Introduction à Cherry Pick**

- Comment copier un commit d'une nouvelle branche sur une autre ? *git cherry-pick*

- Comment copier les commits C1 et C3 d'une branche « Feature » et C4 d'une branche « Test Feature » sur la branche master ? g*it cherry-pick C1 C3 C4*

2. **Introduction à Rebase intéractif**

- Expliquer en quoi consiste un rebase intéractif. *Elle ouvre une interface qui permet de replacer directement les commits dedans*

- Quelle commande utiliser pour ce type de rebase ? *git rebase –i*

- Quel est l'interêt principal de Cherry Pick par rapport au rebase ? *si on ne connait pas les noms des commits*

Partie 4 : Assortiment d'astuces

1. **Choisir seulement un commit**
   A faire : Noter les instructions effectuées pour obtenir le résultat final. Commenter ce que vous avez fait.

   J'ai selectionner le main avec git checkout main J'ai copier C4 le main qui crée une nouvelle branche git cherry-pick C4

2. **Jongler avec les commits 1/2**
   A faire : Noter les instructions effectuées pour obtenir le résultat final. Commenter ce que vous avez fait.

   git rebase –i caption~2 qui va nous permettre d'ouvrir la console et de copier et inverser c3 et c2 apres on fait un git commit –ammend puis on replace c2 et c3 avec git rebase –i caption~2 puis checkout main et on le deplace dans caption avec git branch –f main c3''

3. **Jongler avec les commits 2/2**
   A faire : Noter les instructions effectuées pour obtenir le résultat final. Commenter ce que vous avez fait.

On a selectionner le main avec git checkout main puis on copier C2 avec git cherry-pick C2 et on a rebase caption avec git rebase –i caption ~2 puis on a juste à copier C3 avec git cherry-pick C3

4. Git tags

- Qu'est-ce qu'un « milestone » ? *c'est une branche référente qui tags à jamais les commits*

- Quelle commande pour mettre un numéro de version (par exemple « v0 ») à un commit (dont l'identifiant serait « C1 ») ? *git tag v0 c1*

5. **Git describe**
   Donner la description fournie par git dans le tutoriel pour les éléments suivants :
   - Le commit C1 : `v1_2_gC2`
   - La branche master :
   - Le commit C3 :

PARTIE 5 : Push & Pull

Instruction :
Revenir à l'écran principal avec la commande « levels », et choisir l'onglet « REMOTE », puis effectuer la série des tutoriels « Push&Pull » :

1. **Introduction à Clone**
   - A quoi servent les dépôts distants ? *Ils servent de sauvegarde*
   - Quel est l'utilité de la fonction « git clone » ? *il créée des copies locales à partir des dépôts distant en local*

2. **Les branches distantes**
   - Quelle est la syntaxe pour une branche distante ? Écrire en exemple la syntaxe de la branche distante master. *<nom dépôt distant>/<nom de la branche> 0/main*

3. **Git Fetch**
   - A quoi sert la commande Fetch ?*elle permet de rapporter des données depuis un dépots distant*
   - Quelles sont les différentes actions effectuées par un Fetch ? *1/elle télécharge 2/met à jours les branches*
   - Fetch suffit-il pour modifier votre répertoire de travail local ? Expliquer pourquoi. *Il ne change rien aux fichiers locaux, il sert à télécharger toutes les données qui est nécessaires pour le dépôts local et distant*

4. **Git Pull**
   - Quelle est la finalité de la commande « git Pull » ? *c'est un raccourcis de git fetch*
   - Quelles sont les étapes que « git Pull » permet de fusionner ? git fetch + merge de toutes les branches

5. **Simulation du travail d'équipe**
   A faire : Noter les instructions effectuées pour obtenir le résultat final. Commenter ce que vous avez fait.

(Aide : Penser à tout d'abord cloner le travail distant, avant de simuler le travail de vos camarades à l'aide de la fonction fakeTeamwork*)

Git clone Git faketeamwork main 2 Git commit Git pull

« * » La fonction fakeTeamwork est une fonctionnalité crée par le tutoriel qui n'existe pas dans git. Elle permet de simuler un ou plusieurs « commit » de vos collaborateurs sur le serveur distant.

6. **Git Push**
   - Expliquer le fonctionnement de Git Push *Elle permet de l'envoie des changements vers un dépôts distant et de pouvoir le télécharger par n'importe qui depuis ce même dépôts*

7. **Historique divergent**
   Suivre le tutoriel pour comprendre ce qu'il se passe quand il n'est pas possible d'effectuer un push car l'historique distant a trop divergé et comment résoudre cette situation.
   - Donner une méthode pour résoudre le problème ci-dessus

Pour rappel, dans le tutoriel, les étapes à réaliser sont les suivantes :
- Clonez votre dépôt
- Simulez un travail d'équipe (1 commit)
- Commitez un peu de votre travail (1 commit)
- Publiez votre travail avec rebase

Effectuer le tutoriel, et noter les commandes à effectuer :

8. **Master dévérouillé**
   - Qu'est-ce qu'un Pull request ?

   - Dans quel cas l'utiliser ?

   - Suivre le didactiel et noter les commandes que vous avez effectué pour résoudre le problème :

Instructions :
Les deux parties suivantes sont facultatives pour le TD et ne sont pas notées. Mais il est conseillé de les faire. Elles vous apprennent des techniques avancées sur l'utilisation de Git.

FACULTATIF : PARTIE 6 : dépôts distants version avancée

Revenir à l'écran principal avec la commande « levels », et choisir l'onglet « REMOTE »

1. **Maîtriser Push**
2. **Fusionner avec les branches distantes**
3. **Suivi de branches distantes**
4. **Arguments de Git Push**
5. **Arguments de Git Push 2**
6. **Arguments de Fetch**
7. **« Aucune sources »**
8. **Arguments de pull**

FACULTATIF : Sujets Avancés (Partie Locale)

Revenir dans l'écran principal avec la commande « levels » et choisir les derniers items de « Sujets Avancés »

1. Rebaser plus de 1000 fois
2. Parents multiples
3. Branches spaguettis

Facultatif : Pour aller plus loin :

Essayer de construire un niveau avec « build level » ou essayer le niveau d'un ami avec « import level »