**Git – Module01 – Démo 01**

1. Ouvrir git bash dans un répertoire vide
2. Taper **git hash-object "Apple Pie"**
3. Lire le message d'erreur
4. Taper **echo "Apple Pie" | git hash-object --stdin**
5. Vérifier le résultat
6. Taper **echo "Apple Pie\n" | git hash-object --stdin**
7. Vérifier le résultat
8. Retour au slide 13

**Git – Module01 – Démo 02**

1. Ouvrir git bash dans un répertoire vide
2. Taper **echo "Apple Pie" | git hash-object --stdin -w**
3. Lire le message d'erreur
4. Taper **git init**
5. Vérifier le résultat
6. Taper **ls**
7. Taper **ls -a**
8. Taper **echo "Apple Pie" | git hash-object --stdin -w**
9. Vérifier le résultat
10. Ouvrir le répertoire .git dans explorateur
11. Ouvrir le répertoire objects/23
12. Ouvrir le blob nommé 99... dans Notepad
13. Taper **git cat-file 23991897e13e47ed0adb91a0082c31c82fe0cbe5 -t**
14. Vérifier le résultat
15. Taper **git cat-file 23991897e13e47ed0adb91a0082c31c82fe0cbe5 -p**
16. Vérifier le résultat
17. Retour au slide 16

**Git – Module01 – Atelier 01**

1. Extraire l’archive module01\_starter dans un répertoire
2. Ouvrir git bash dans ce répertoire
3. Taper **ls**
4. Taper **cat menu.txt**
5. Taper **ls recipes/**
6. Taper **cat recipes/README.txt**
7. Taper **cat recipes/apple\_pie.txt**
8. Taper **tree .**
9. Taper **git init**
10. Taper **ls -a**
11. Explorer le répertoire .git/
12. Taper **git status**
13. Taper **git add menu.txt**
14. Taper **git add recipes/**
15. Taper **git status**
16. Taper **git commit -m "First commit!"**
17. Taper **git status**
18. Taper **git log**
19. Explorer le répertoire .git/objects/
20. Taper **git cat-file -p 4f85f3ddc79b47852d48ff94160f4ffea5e0b39f**
21. Vérifier le résultat

Retour au slide 18

1. Vérifier le tree du commit
2. Taper **git cat-file -p be4d5bfce489a2591e7fed5c672f9e52cd695a43**
3. Expliquer les résultats
4. Taper **git cat-file -p 23991897e13e47ed0adb91a0082c31c82fe0cbe5**

Retour au slide 18

1. Taper **git cat-file -p 3ee76fde69b730530f1682f1f51789e89cf30500**
2. Vérifier le résultat
3. Taper **git cat-file -p 361af858632ee7d8d8f9c4022ccaf61fc8d4799c**
4. Vérifier le résultat

Retour au slide 18

**Git – Module01 – Atelier 02**

1. Ouvrir git bash dans le répertoire module01\_starter
2. Taper **vim menu.txt**
3. Ajouter Cheesecake à la fin du fichier et sauvegarder le fichier
4. Taper **git status**
5. Taper **git add menu.txt**
6. Taper **git status**
7. Taper **git commit -m "Add cake"**
8. Taper **git status**
9. Taper **git log**
10. Taper **git cat-file -p 44fcf4a9353ce2b83178aa8fb46e83c0e2239d6b**
11. Vérifier la présence d’un parent
12. Taper **git cat-file -p 4f85f3ddc79b47852d48ff94160f4ffea5e0b39f**

Retour au slide 19

1. Taper **git cat-file -p 6ee0cdeefaf** (le tree du second commit)
2. Taper **git cat-file -p f1fe985b46ca38** (le blob du menu)
3. Vérifier le contenu
4. Vérifier que le tree est identique à celui du premier commit

Retour au slide 19

1. Taper **git count-objects**
2. Vérifier le nombre d’objets

**Git – Module01 – Atelier 03**

1. Ouvrir git bash dans le répertoire module01\_starter
2. Taper **git tag -a mytag -m "I love cheesecake"**
3. Taper **git tag**
4. Vérifier le résultat
5. Taper **git cat-file -p mytag**
6. Vérifier le résultat et noter que le tag pointe vers le second commit

Retour au slide 20

**Git – Module02 – Atelier 01**

1. Extraire l’archive module02\_starter dans un répertoire ou continuer à partir du module 1
2. Ouvrir git bash dans ce répertoire
3. Taper **tree**
4. Taper **git log**
5. Taper **git branch** pour afficher la liste des branches
6. Explorer le répertoire .git/ pour trouver les branches
7. Taper **cat .git/refs/heads/master**

Retour au slide 5

1. Taper **git branch lisa**
2. Taper **git branch** pour lister les branches
3. Explorer le répertoire .git/ pour trouver les branches
4. Taper **cat .git/refs/heads/master**
5. Taper **cat .git/refs/heads/lisa**
6. Comparer le contenu

Retour au slide 7

1. Taper **git branch** pour lister les branches
2. Expliquer l’astérisque
3. Explorer le répertoire .git/ et trouver HEAD
4. Taper **cat .git/HEAD**

Retour au slide 8

1. Taper **tree**
2. Editer recipes/apple\_pie.txt
3. Copier / coller my\_apple\_pie.txt
4. Taper **git status**
5. Taper **git add recipes/apple\_pie.txt**
6. Taper **git status**
7. Taper **git commit -m "Add recipe"**
8. Taper **git log**
9. Taper **cat .git/refs/heads/master**
10. Taper **cat .git/HEAD**

Retour au slide 10

1. Taper **git branch**
2. Taper **git checkout lisa**
3. Expliquer les 2 changements qui surviennent
   1. HEAD
   2. Workfolder
4. Taper **cat .git/HEAD**
5. Taper **tree**
6. Editer recipes/apple\_pie.txt et voir que le contenu a changé

Retour au slide 11

1. Taper **clear**
2. Editer recipes/apple\_pie.txt
3. Copier / coller lisa\_apple\_pie.txt
4. Taper **git status**
5. Taper **git add recipes/apple\_pie.txt**
6. Taper **git status**
7. Taper **git commit -m "Add Lisa’s version of the pie"**
8. Taper **git log**

Retour au slide 12

**Git – Module02 – Atelier 02**

1. Reprendre à la suite du lab 1
2. Taper **checkout master**
3. Taper **cat recipes/apple\_pie.txt**
4. Expliquer le « pourquoi on merge ? »
5. Taper **git merge lisa**
6. Vérifier qu’on a un conflit
7. Taper **git status**
8. Taper **vim recipes/apple\_pie.txt**
9. Expliquer le contenu du fichier en conflit
10. Editer et sauvegarder le fichier
    1. Garder 1 tbpsn cinnamon de lisa
    2. Changer le nombre de pommes à 9
11. Taper **git status**
12. Taper **git add recipes/apple\_pie.txt**
13. Taper **git status**
14. Pour terminer le merge, taper **git commit** sans message et valider le contenu
15. Taper **git status**
16. Taper **git log**
17. Taper **git cat-file -p 85ad5794ddc**
18. Expliquer le contenu du commit et les deux parents

Retour au slide 13

**Git – Module02 – Atelier 03**

1. Reprendre à la suite du lab 2
2. Taper **checkout lisa**
3. Expliquer le « pourquoi on veut merger ? » convaincre lisa que notre version d’apple pie est meilleure que la sienne et donc mettre à jour sa branche avec notre branche master

Retour au slide 20

1. Taper **git merge master**
2. Noter le message fast forward

**Git – Module02 – Atelier 04**

1. Reprendre à la suite du lab 3
2. Taper **git** **checkout master**
3. Taper **git log**
4. Taper **git checkout 85ad5794ddc4f7d**
5. Taper **cat .git/HEAD**
6. Taper **git branch**

Retour au slide 22

1. Editer le fichier recipes/apple\_pie.txt
2. Modifier le nombre de pommes (20)
3. Taper **git status**
4. Taper **git add recipes/**
5. Taper **git commit -m "Add more apples"**
6. Taper **git log**

Retour au slide 23

1. Editer le fichier recipes/apple\_pie.txt
2. Supprimer le sucre
3. Taper **git add recipes/**
4. Taper **git commit -m "Remove sugar"**
5. Taper **git log**

Retour au slide 23

1. On arrête l’expérimentation, taper **git** **checkout master**

Retour au slide 23

1. Taper **git checkout d28f04117812f1** (le dernier commit)
2. Taper **git branch nogood**
3. Taper **git** **checkout master**

Retour au slide 24

**Git – Module03 – Atelier 01**

1. Extraire l’archive module03\_starter dans un répertoire ou continuer à partir du module 2
2. Ouvrir git bash dans ce répertoire
3. Taper **git branch**
4. Taper **git log** et expliquer et montrer les modifications sur la branche master
5. Taper **git checkout spaghetti** et montrer les modifications
6. Taper **git branch**
7. Expliquer les différences de couleur

Retour au slide 2

1. Taper **git rebase master**
2. Expliquer le résultat

Retour au slide 4

1. Taper **git log** pour afficher l’historique
2. Si on veut rebase dans l’autre sens, taper **git checkout master**
3. Taper **git rebase spaghetti** et voir les résultats (rebase ou merge, ici le résultat est identique)

Retour au slide 5

1. Taper **git log** pour voir l’historique sur master

Retour au slide 6

**Git – Module03 – Atelier 02**

1. Continuer à partir du lab 1
2. Taper **git checkout master**
3. Taper **git tag**
4. Taper **git log -1**
5. On peut créer un tag avec la commande **git tag -a dinner** (si on veut créer un annotated tag)
6. Taper **git tag dinner**
7. Taper **git tag**
8. Regarder le résultat
9. Explorer le répertoire .git/refs
10. Regarder dans les répertoires heads et tags
11. Taper **cat .git/refs/tags/dinner**

Retour au slide 22

**Git – Module04 – Atelier 01**

1. Aller sur Github et afficher le repo distant : https://github.com/boblechat/cookbook
2. Copier l’URL pour cloner le répertoire
3. Ouvrir git bash dans un répertoire vide
4. Taper **ls**
5. Taper **git clone https://github.com/boblechat/cookbook.git**
6. Taper **ls**
7. Taper **cd cookbook**
8. Taper **ls**
9. Taper **ls -a** et vérifier qu’on récupère aussi le répertoire .git et son contenu
10. Taper **git branch** pour montrer qu’on a récupéré que master
11. Expliquer le mécanisme de « copie » du workspace depuis GitHub
12. Taper **ls .git/objects/** pour montrer que les objets ont bien été récupérés

Retour au slide 3

1. Taper **vim .git/config**
   1. Montrer le remote par defaut (origin)
   2. Insister sur le fait qu’on peut en avoir plusieurs
   3. Montrer qu’on a actuellement une branche (master) qui pointe sur origin
2. Taper **git branch**
3. Taper **git branch --all** pour montrer que git a su récupérer les infos
4. Explorer le répertoire .git/refs
   1. Regarder le répertoire remotes, origin
   2. Préciser que toutes les branches ne sont pas forcément affichées
   3. Elles sont « packées » et compressées dans packed-refs
5. Taper **git show-ref master**
6. Taper **git show-ref lisa** et vérifier qu’on n’a que la remote

Retour au slide 5

1. Taper **vim recipes/apple\_pie.txt**
2. Ajouter une cuillère de jus de citron
3. Taper **git add recipes/**
4. Taper **git status**
5. Taper **git commit -m "Add more lemon juice to the apple pie"**
6. Taper **git log -2**
7. Taper **git show-ref master**
8. Montrer que les refs pointent bien vers 2 commits différents
9. Taper **git push**
10. Taper **git show-ref master** et verifier les refs

Retour au slide 8

**Git – Fork & PR – GitHub**

1. Aller sur Github et afficher le repo distant : https://github.com/boblechat/cookbook
2. Se connecter avec son user/password si ce n’est pas déjà fait
3. Sur la page du repo, cliquer sur le bouton Fork
4. S’assurer que le repo est bien présent sur son compte %VotreCompte%/cookbook
5. Copier l’URL pour cloner le répertoire
6. Ouvrir git bash dans un répertoire vide
7. Taper **ls**
8. Taper **git clone https://github.com/%VotreCompte%/cookbook.git**
9. Taper **cd cookbook**
10. Taper **git branch --all** pour vérifier les branches disponibles

Créer une branche

1. Taper **git branch feature/%Trigramme%**
2. Taper **git checkout feature/%Trigramme%**
3. Ajouter une recette dans le répertoire recipes/
   1. Donner un nom à la recette
   2. Y ajouter des ingrédients
   3. Optionnel, mettre à jour le fichier menu.txt avec votre recette
4. Taper **git add recipes/**
5. Taper **git add menu.txt** si vous avez modifié le fichier
6. Taper **git commit -m "%Votre message%"**
7. Taper **git push --set-upstream origin feature/%Trigramme%** pour mettre à jour votre remote avec la nouvelle branche. Il est possible que vous ayez à saisir vos identifiants

Faire une demande de pull request

1. Dans l’interface de GitHub, cliquer sur Compare & pull request
2. Choisir la branche où merger : boblechat/cookbook et master
3. Choisir la branche à marger : %VotreTrigramme%/cookbook et feature/%Trigramme%
4. Ajoute un titre à la PR
   1. Eventuellement un message
5. Cliquer sur Create pull request. La PR sera validée et mergée par le propriétaire du repo original

En cas de conflit

1. Dans l’interface de GitHub, cliquer sur Compare & pull request
2. Choisir les repos et les branches source / destination
3. Ajouter un titre à la PR et cliquer sur Create pull request
4. Cliquer sur le bouton Resolve conflicts
5. Faire les changements dans l’éditeur et valider en cliquant sur le bouton Mark as resolved
6. Cliquer sur le bouton Commit merge Cliquer sur Create pull request. La PR sera validée et mergée par le propriétaire du repo original