HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Introduction à git et GitHub

Une formation préparée pour le GYB

Olivier Liechti
8 décembre 2020

Gymnase intercantonal de la Broye

GILBERT DELAHAYE - MARCEL MARLIER

martine
prépare un commit

casterman

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

git

Agenda

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

10:15 - 10:30

Pourquoi git et GitHub?
Terminologie de base

10:30 - 11:00

1. Versioner ses documents

```
git clone
git status
git add
git commit
git log / git lone --oneline --graph
git push
```

11:00 - 11:10

Les structures de données de git
blob, tree, commit, tag, branch

11:10 - 11:20

2. Donner accès son repo en lecture (et faire des mises à jour)

```
git clone
git pull origin master
```

11:20 - 12:00

3. Collaborer et contribuer (fork d'un repo, utilisation des branches et des pull requests)

```
git checkout -b fb-ma-contribution
git remote add
git pull upstream master
```

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Pourquoi utiliser et git et GitHub?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Gérer l'historique de ses fichiers
(code, documentation, etc.)

Profiter d'une **sauvegarde, d'un archivage sur le cloud**

Faciliter la collaboration entre créateurs, contributeurs et consommateurs

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

~1990 ~2000 ~2010

Une des différences entre git et d'autres systèmes de gestion de version est qu'il est **distribué**
(ce qui permet de définir **différentes modèles de collaboration**)

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

GitLab

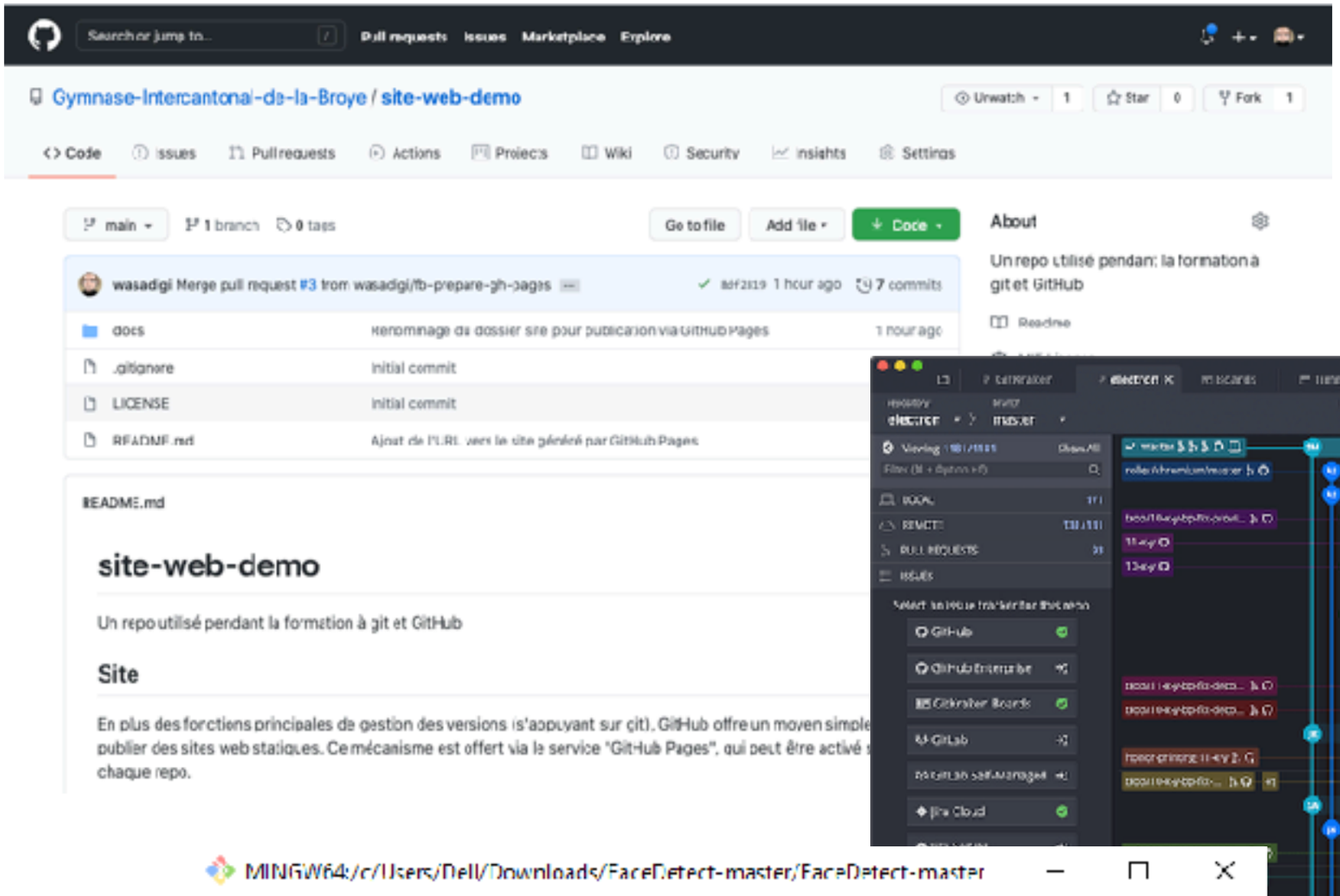
GitHub

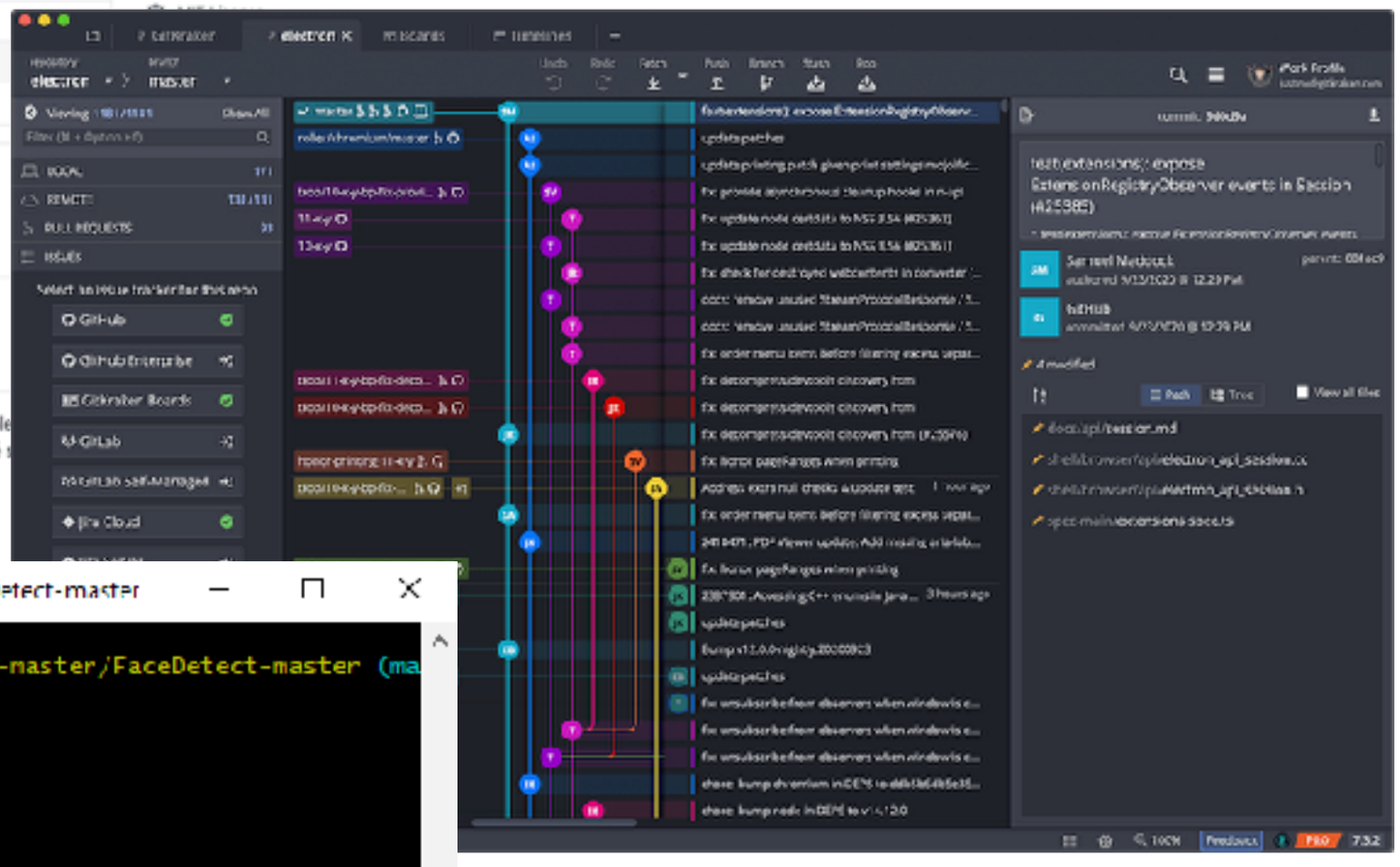
Bitbucket

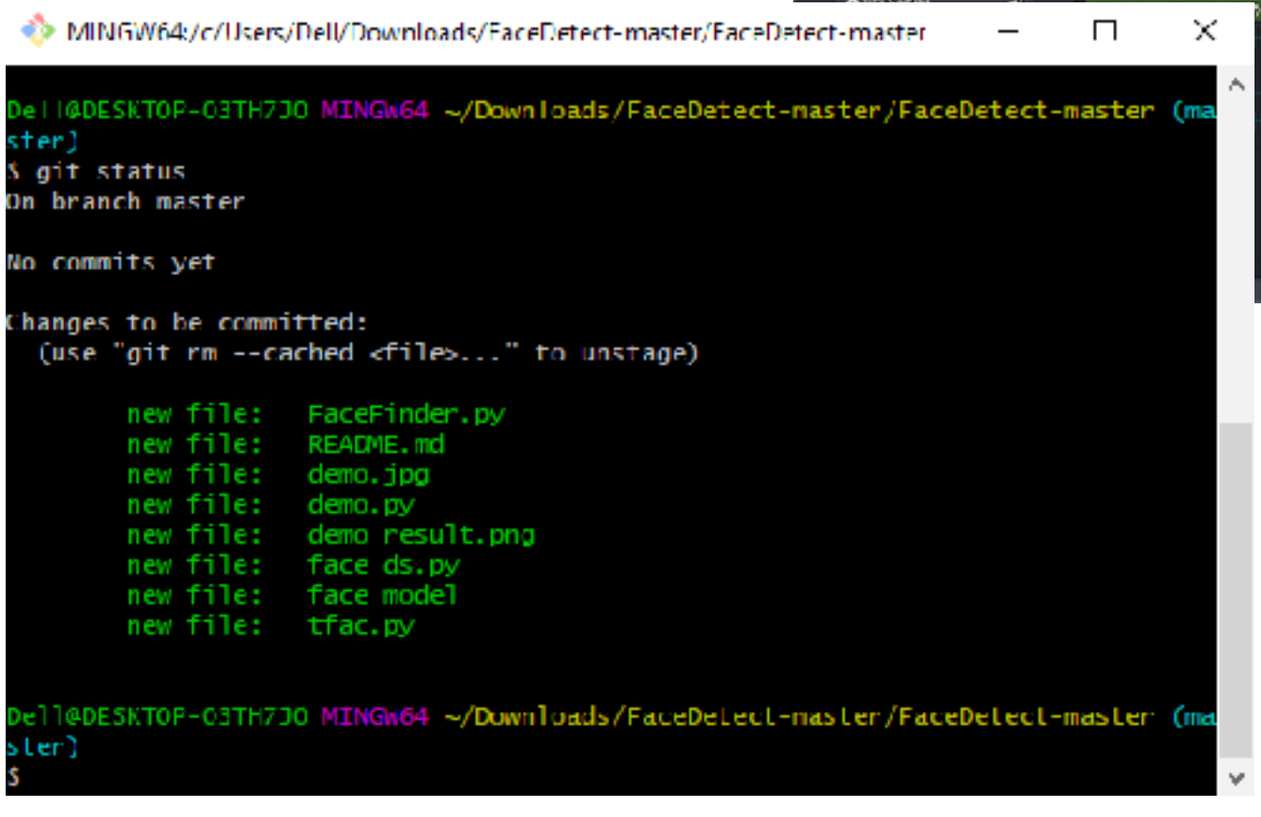
"Software as a Service", interface web, "social coding"

git

structures de données, commandes, protocoles







heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Archiver son travail

Scénario 1

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Karl donne un cours de physique.

Il veut utiliser git et GitHub pour **rédiger ses notes de cours**.

Dans ce scénario simple, **il est le seul auteur de contenu**. Pas besoin de se soucier de modifications concurrentes, de gestion de conflits, etc.

Ses élèves ne vont pas directement accéder au repo: Karl leur remet des fichiers PDF générés depuis ses fichiers sources.

Etapes

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

1. Créer un repo sur GitHub (espace personnel)

2. Cloner ce repo sur sa machine

3. Utiliser les commandes `git add`, `commit` et `push`

*Avec **git add**, je prépare une nouvelle version en spécifiant quelles modifications (fichiers créés et modifiés) doivent en faire*

*Avec **git commit**, je crée une version (un noeud dans l'historique git)*

*Avec **git push**, j'envoie la version (le commit et les fichiers modifiés) vers le repo distant*

Toute le monde…

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

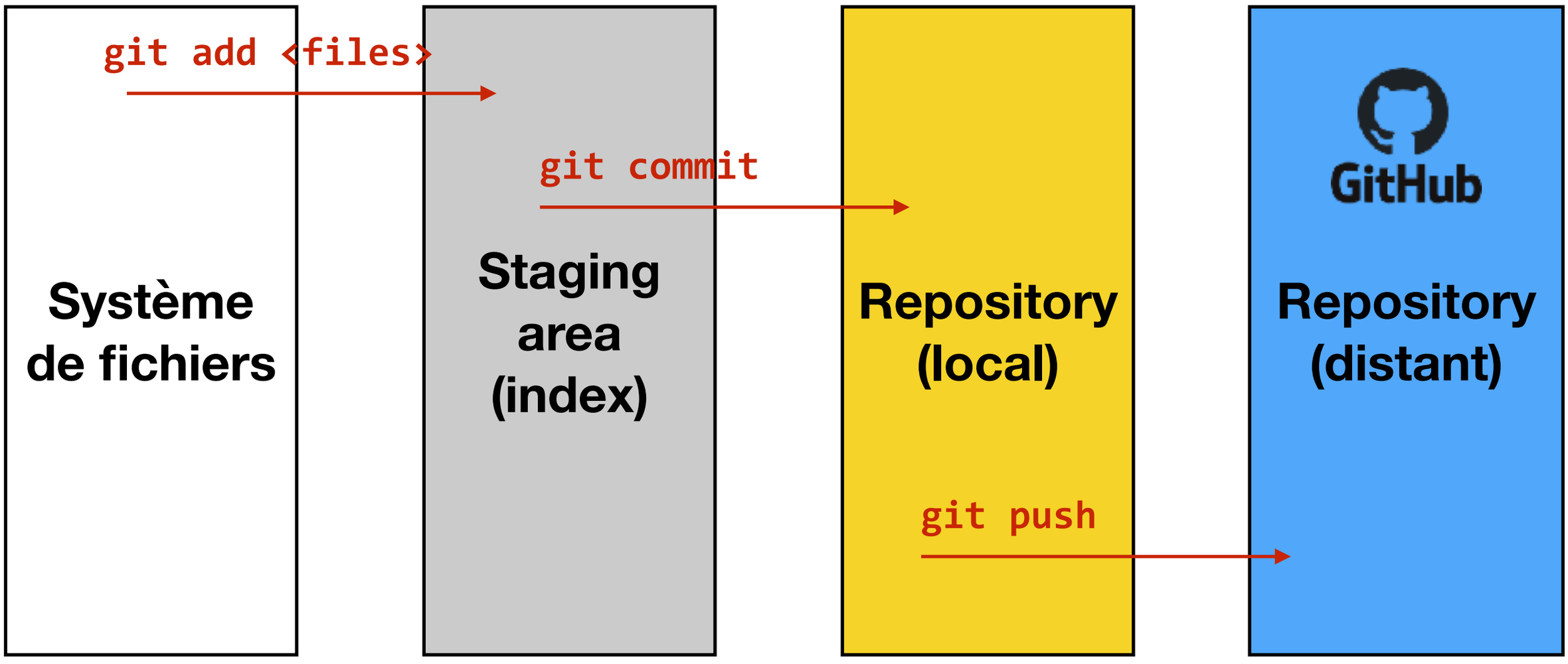
Se connecte à son compte GitHub
Crée un nouveau repo "mon cours"
Clone le repo sur sa machine

Ajoute un nouveau fichier “plan.md”
Crée une nouvelle “version” avec **git add** et **git commit**
Envoie cette version vers GitHub avec **git push**

Modifie le fichier “plan.md”
Crée une nouvelle "version" avec **git add** et **git commit**
Envoie cette version vers GitHub avec **git push**

Créer une version

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



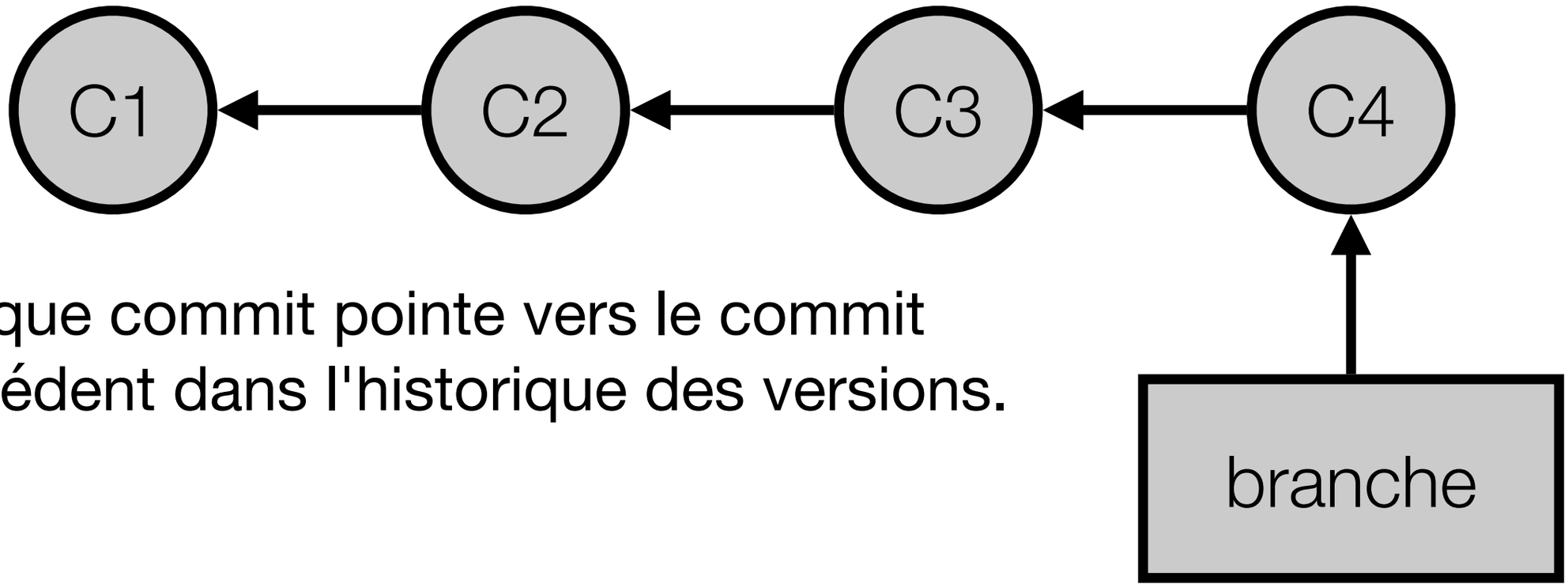
Add = ajouter ce(s) fichier(s) dans le prochain commit (que ce soit des nouveaux fichiers, ou des fichiers modifiés)

Commit = créer une nouvelle version dans l'historique du repo

Historique (git log)

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

Chaque commit pointe vers un "tree" (la racine du système de fichiers au moment où le commit a été créé)



Chaque commit pointe vers le commit précédent dans l'historique des versions.

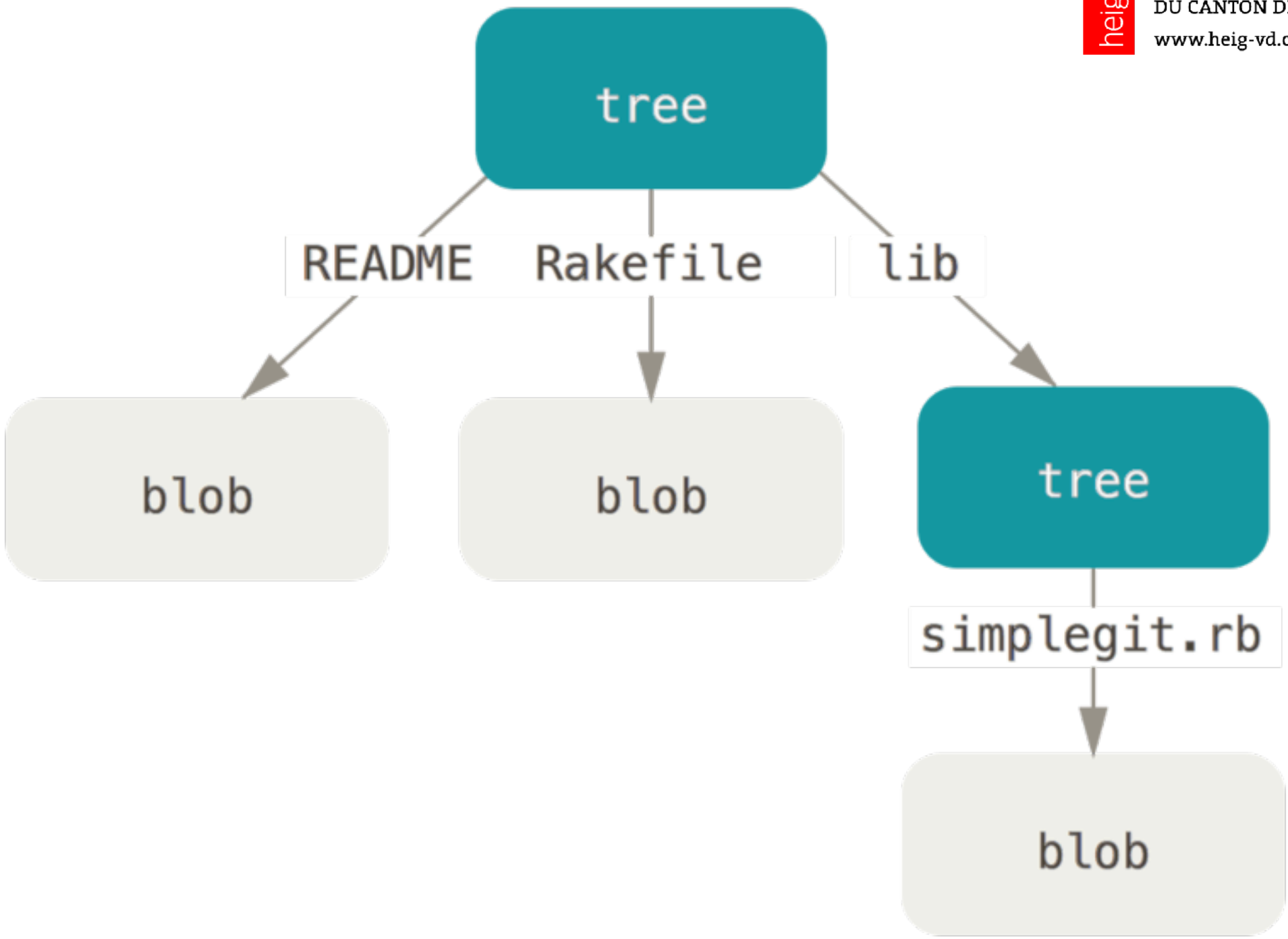
Une branch est un pointeur vers un commit particulier

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Les structures de données utilisées par git

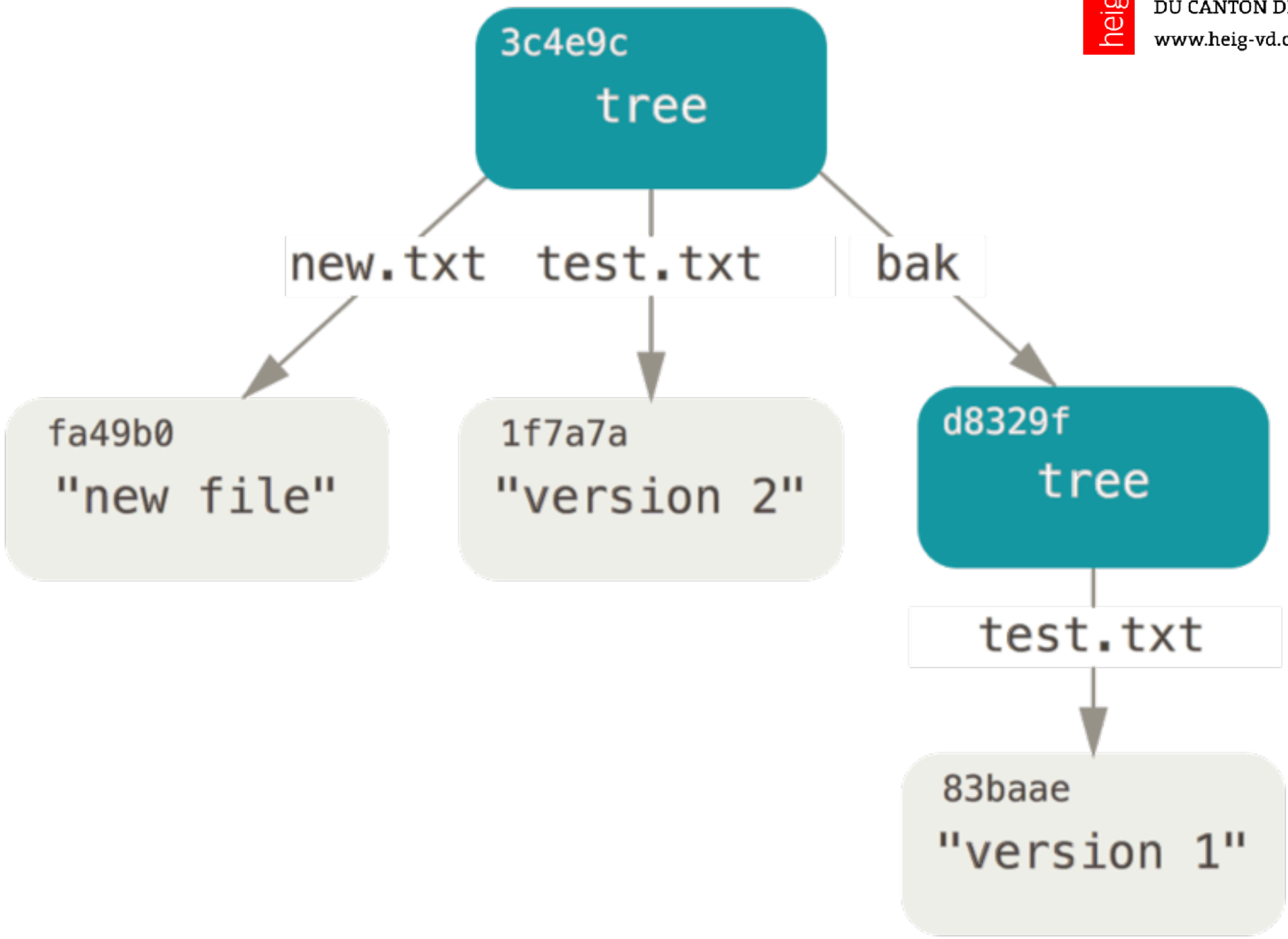
On ne doit pas connaître les détails de ces structures pour une utilisation au quotidien. Mais avoir une idée de ce qui se passe derrière les décors aide beaucoup à progresser.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

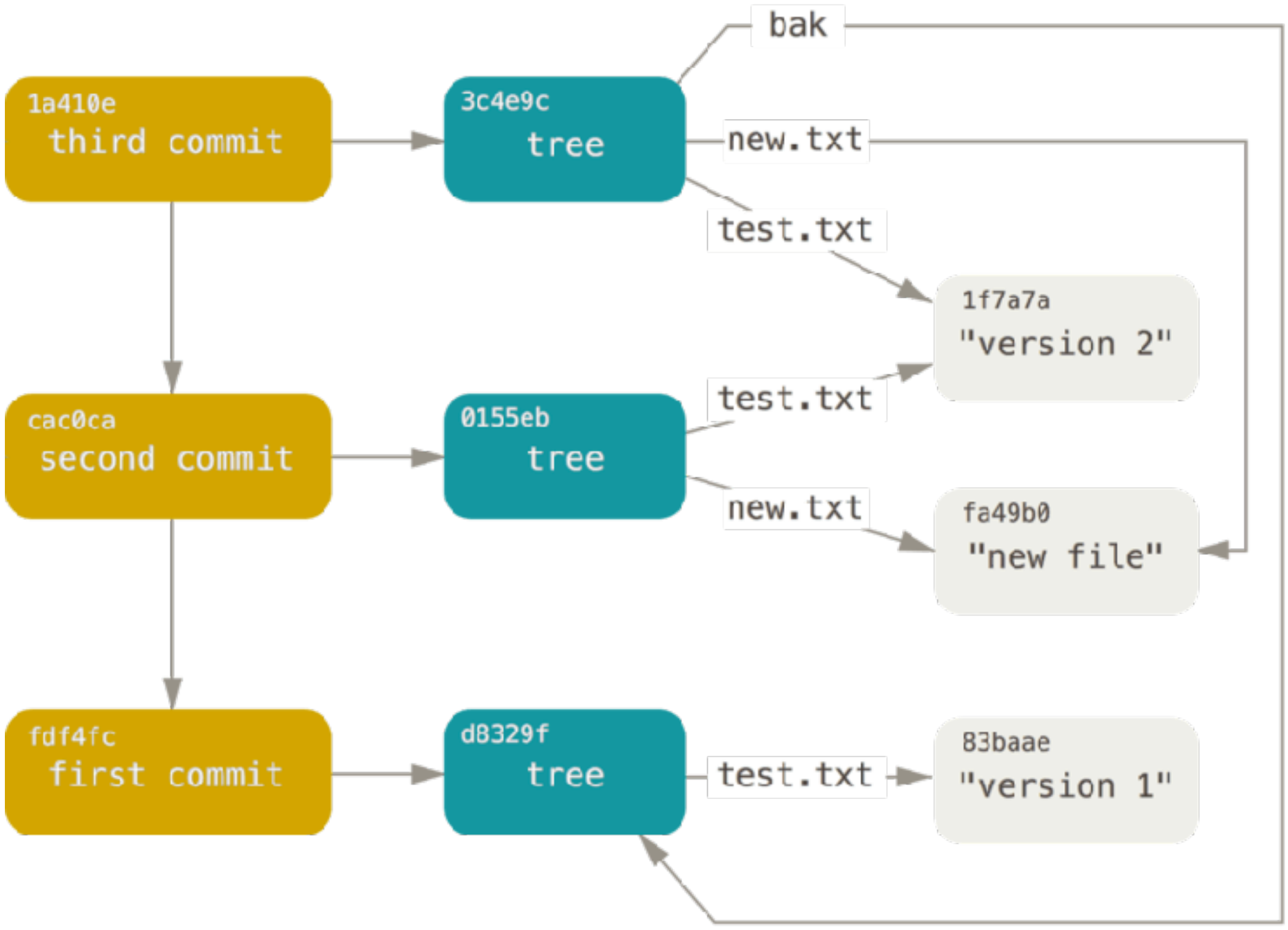


https://git-scm.com/book/en/v2/Git-Internals-Git-Objects

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



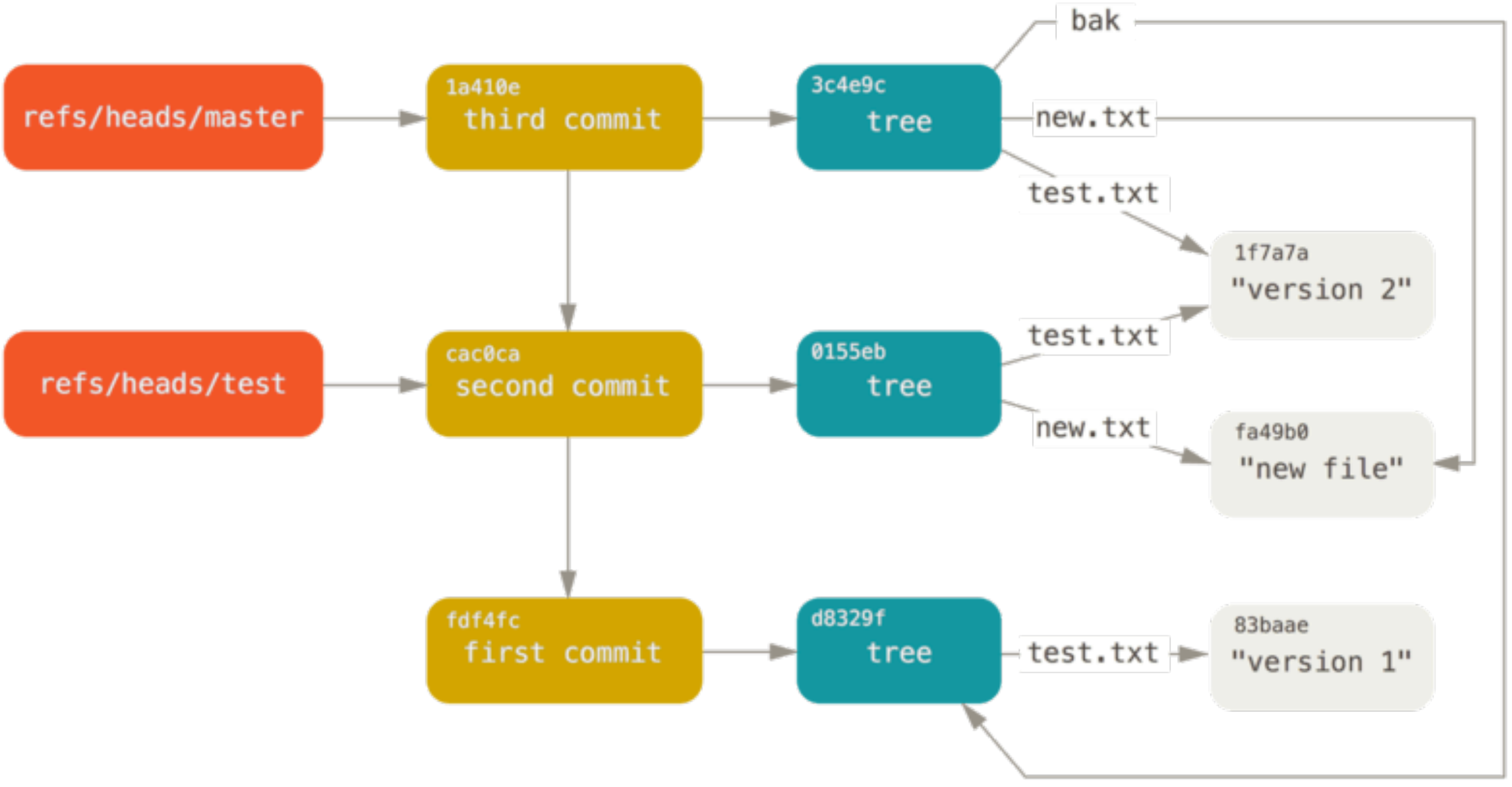
https://git-scm.com/book/en/v2/Git-Internals-Git-Objects



https://git-scm.com/book/en/v2/Git-Internals-Git-Objects

Les rectangles orangés sont des “branches”. On voit que ce sont des pointeurs vers des commits.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



https://git-scm.com/book/en/v2/Git-Internals-Git-Objects

Point clé

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

git s'appuie sur une "base de données" clés-valeurs. Quand on ajoute un fichier dans git, une clé est générée en fonction de son contenu (hash).

4 fichiers avec exactement le même contenu (par exemple dans des dossiers différents) sont stockés une seule fois (la clé est la même). Dans un repo avec 100'000 fichiers, si on modifie 1 fichier et qu'on crée une nouvelle version (commit), on ne crée qu'un seul objet (pas de copie des 99'999 autres).

Pour les curieux… allez regarder le dossier caché .git. C'est là que vous allez trouver cette "base de données".

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Donner accès à ses documents

Scénario 2

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Karl veut donner un accès ses notes de cours, que ce soit à ses collègues ou ses élèves.

Dans ce nouveau scénario simple, **il reste le seul auteur de contenu**. Il fait régulièrement des modifications, et ajoute des commits dans l'historique du repo.

Question: comment ses "lecteurs" peuvent-ils récupérer les nouvelles versions?

Hypothèse: les lecteurs ont les fichiers sur leur machine, MAIS ils ne les modifient pas (pas de risques de conflits…)

Etapes

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

1. Le lecteur clone le repo de Karl sur sa machine

2. Karl ajoute des commits et les envoie vers son repo

3. Le lecteur utilise `git pull origin master`

Toute le monde…

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Visite ce repo sur GitHub:
https://github.com/Gymnase-Intercantonal-de-la-Broye/site-web-demo

Crée un clone sur sa machine

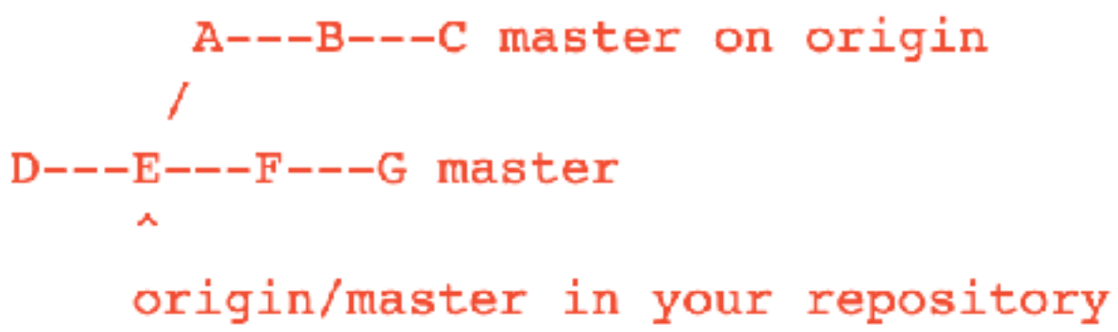
Regarde le contenu du fichier README.md

Attend que j'ai modifié ce fichier

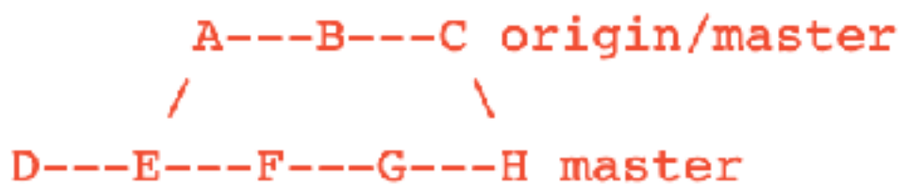
Fait un `git pull origin master`, et vérifie que le fichier README.md a bien été modifié.

Fit un git log --oneline --graph pour regarder l'historique.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



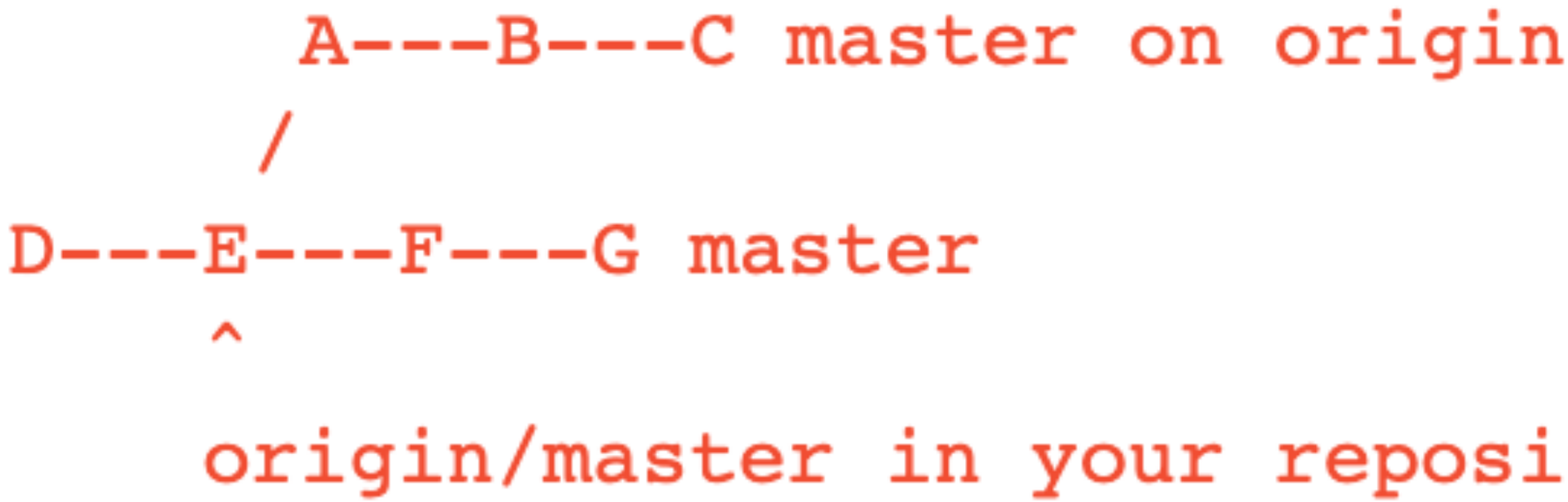
git pull origin master



“Je veux merger la branche “master” du repo “origin” avec la branche courante de mon repo local.”

Chaque lettre représente un commit dans l'historique du repo.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



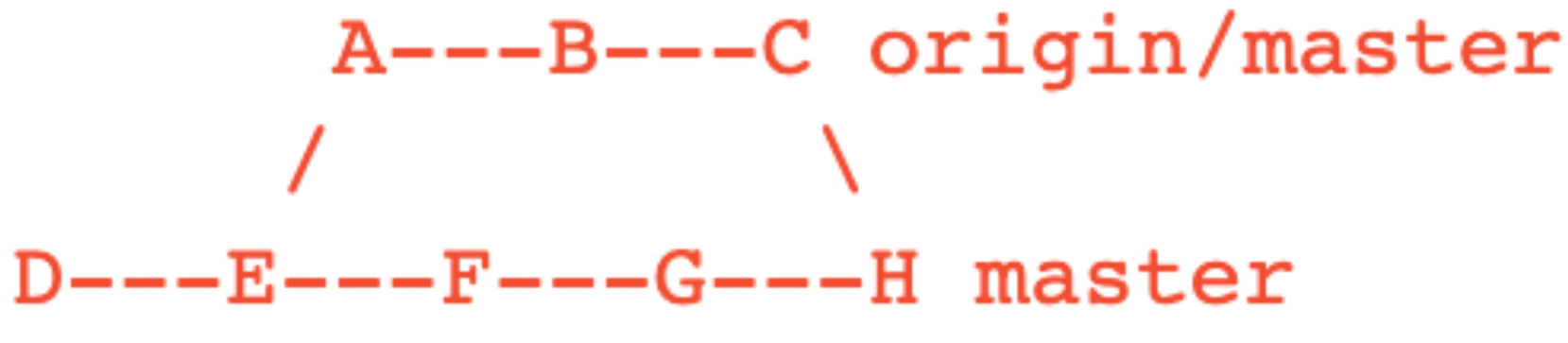
Le lecteur a cloné le repo de Karl en E.

Karl a fait des modifications A, B et C.

En parallèle, le lecteur a fait des modifications locales en F et G.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

git pull origin master



On voit un nouveau commit, H, qui pointe vers une nouvelle arborescence de dossiers et fichiers. Cette arborescence est le résultat de la fusion (merge) entre les arbres pointés par C et G.

Que se passe-t-il si Karl et le lecteur ont modifié la même ligne d'un fichier? Git annonce le "conflit" et l'élève doit le résoudre (en spécifiant quelle version fait foi)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Collaborer

Scénario 3

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Karl veut donner la possibilité à ses lecteurs de proposer des modifications de ses notes.

Dans ce scénario, il garde un contrôle sur l'évolution de son repo.

Les lecteurs n'ont pas le droit de faire des modifications directement.

Ils peuvent proposer des contributions et demander à ce que Karl les accepte (les "tire" vers son repo)

Etapes

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

1. Les lecteurs font un fork du repo de Carl, et clone ce fork sur leur machine

2. Quand un lecteur veut faire une contribution, il crée une branche et ajoute des commits

3. Le contributeur "pousse" sa branche et les modifications associées vers son repo

4. Dans l'interface web de GitHub, le contributeur fait une "pull request" pour demander à Carl de "tirer" sa contribution

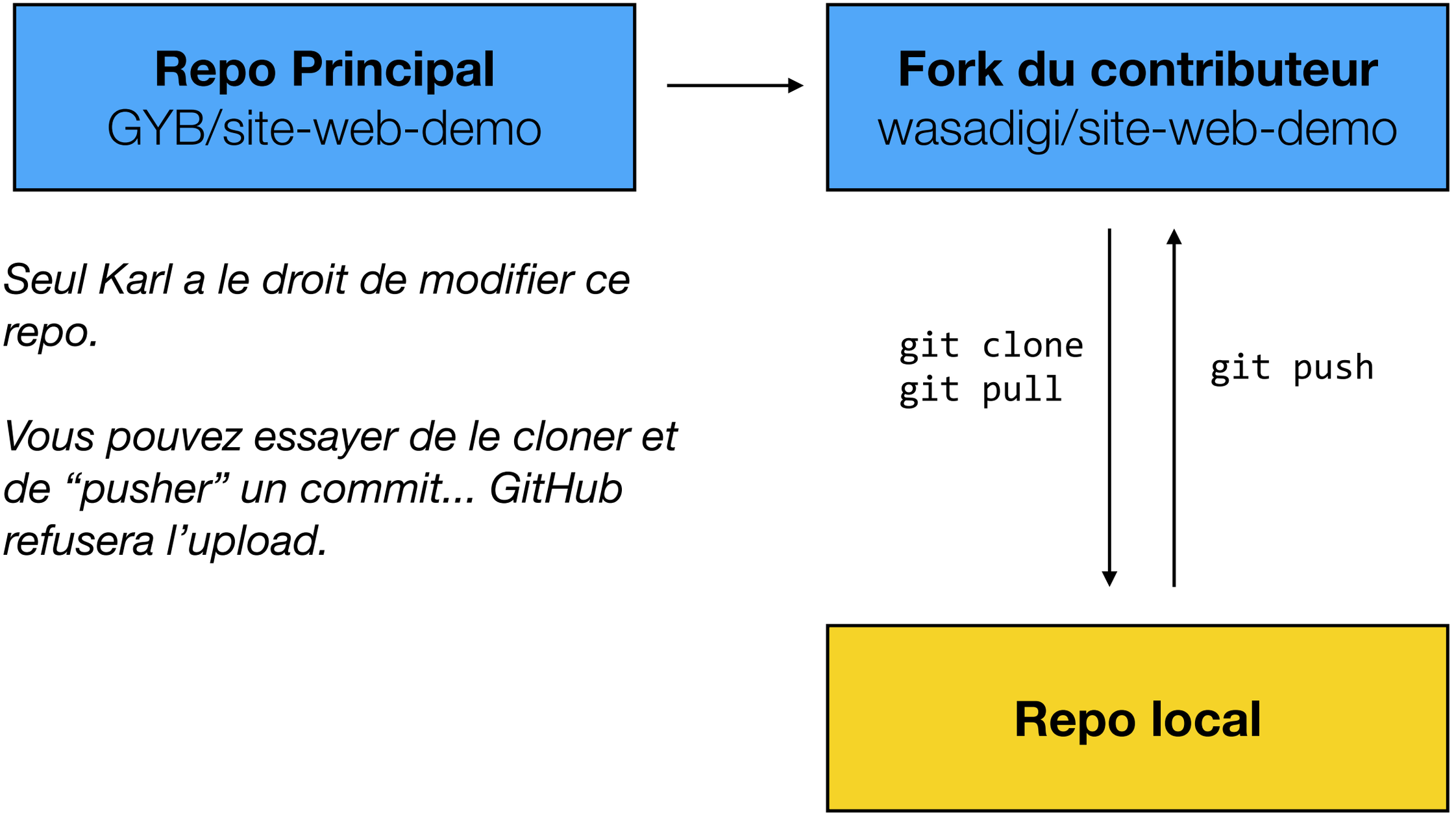
5. Si Carl est d'accord, il merge la branche du contributeur dans son repo (qu'on appelle upstream)

6. Finalement, le contributeur peut "tirer" et merger la branche master de upstream avec sa branche master

Un autre utilisateur (wasadigi) peut faire une copie du repo de Karl sur GitHub. Il est le propriétaire de ce nouveau repo (qui n'est pas automatiquement synchronisé!)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

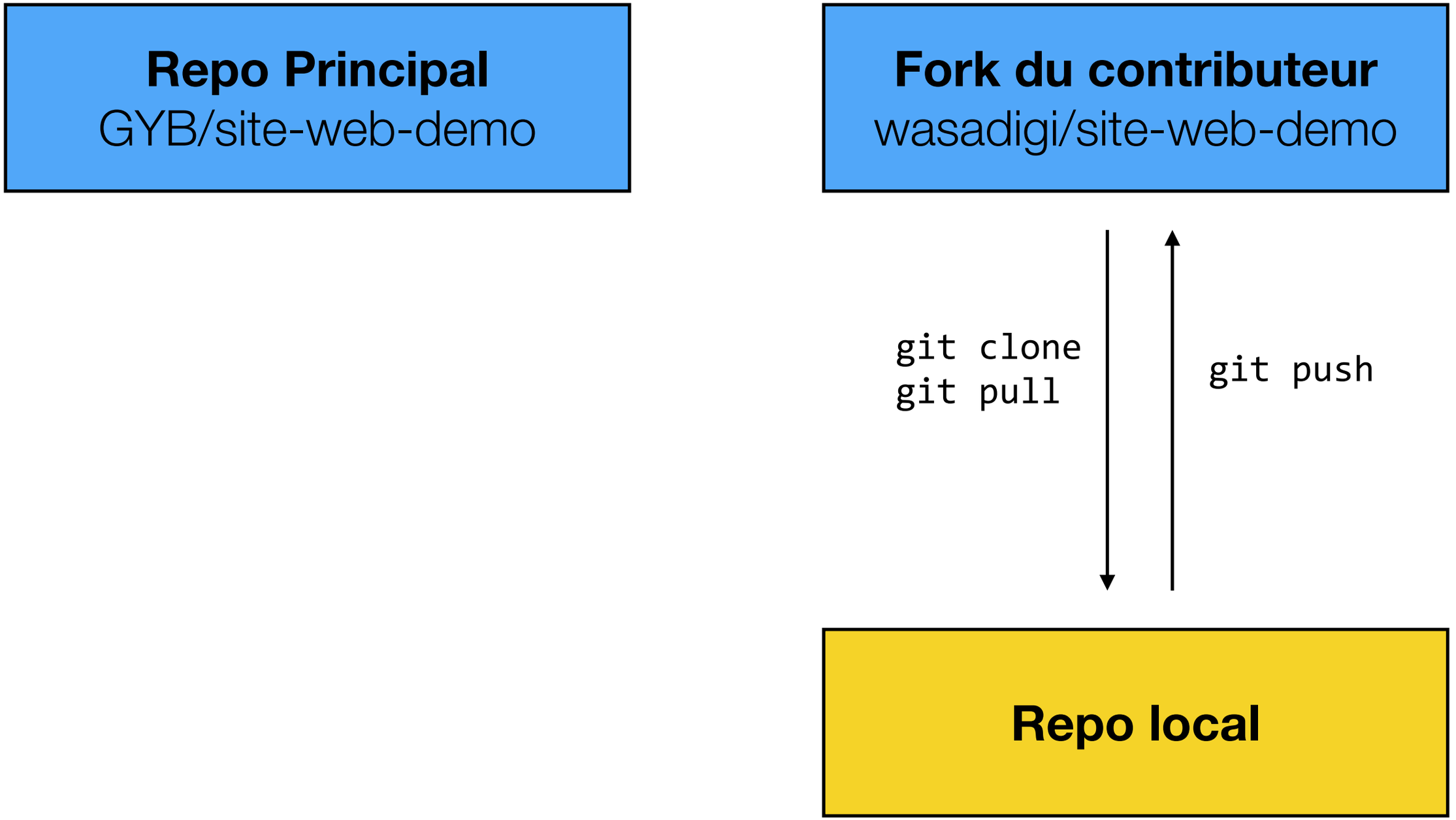
fork via l'interface web GitHub



Seul Karl a le droit de modifier ce repo.

Vous pouvez essayer de le cloner et de "pusher" un commit... GitHub refusera l'upload.

merge request via l'interface web GitHub



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Ce repo est, par convention, appelé “upstream”

Repo Principal
GYB/site-web-demo

Ce repo est, par convention, appelé “origin”

Fork du contributeur
wasadigi/site-web-demo

```
git pull upstream master
```

```
git remote add upstream master
```

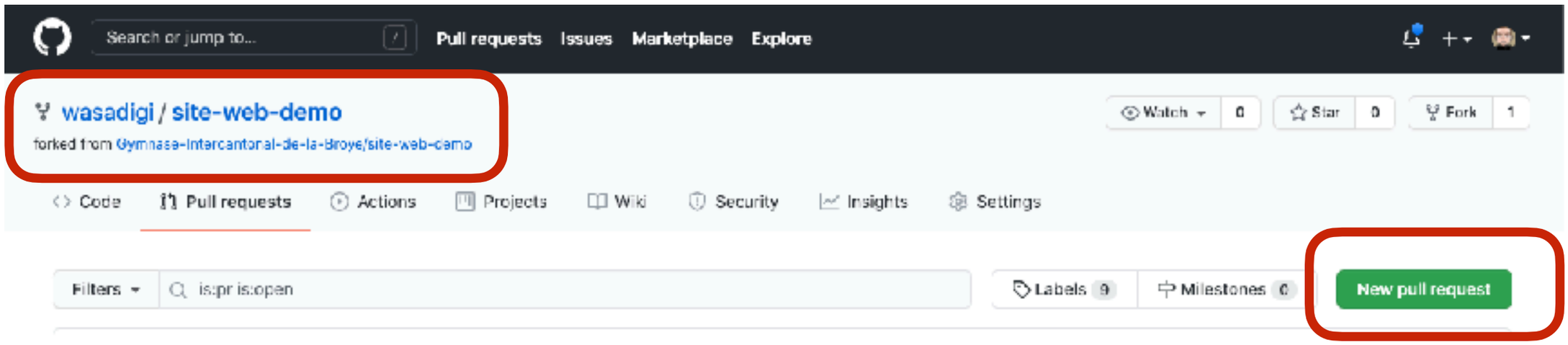
```
git clone
git pull
```

```
git push
```

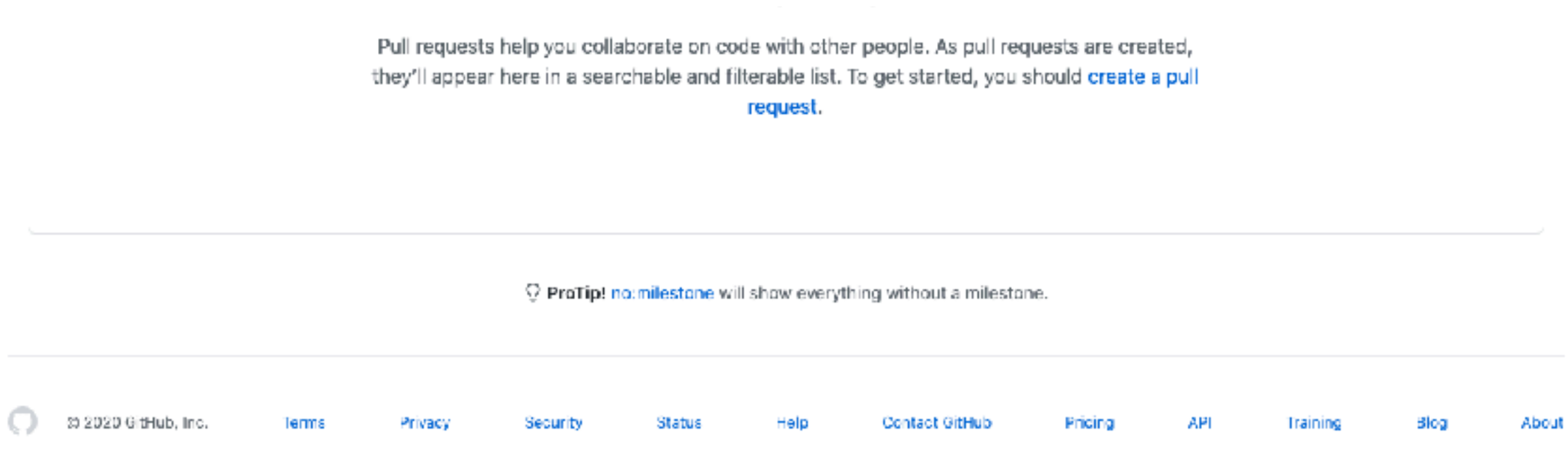
Repo local

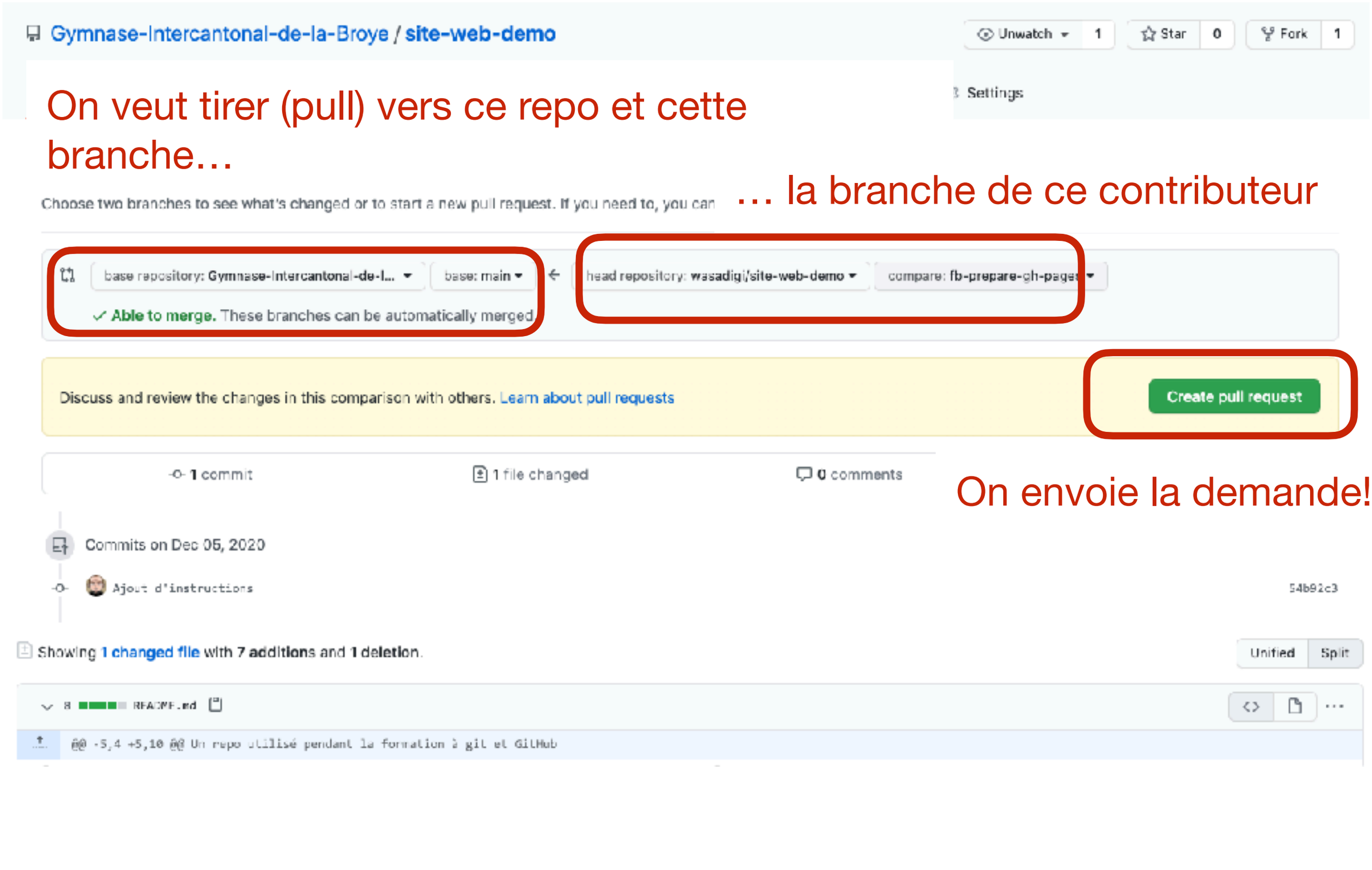
Le contributeur (wasadigi) travaille dans son repo, qui est un "fork" du repo géré par le GYB

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Le contributeur veut demander au propriétaire du repo principal qu'il accepte sa contribution




Gymnase-Intercantonal-de-la-Broye / site-web-demo
Unwatch 1
Star 0
Fork 1
Settings
On veut tirer (pull) vers ce repo et cette branche…
… la branche de ce contributeur
Choose two branches to see what's changed or to start a new pull request. If you need to, you can
base repository: Gymnase-Intercantonal-de-l...
base: main
head repository: wasadigi/site-web-demo
compare: fb-prepare-gh-page
✓ Able to merge. These branches can be automatically merged
Discuss and review the changes in this comparison with others. Learn about pull requests
Create pull request
On envoie la demande!
1 commit
1 file changed
0 comments
Commits on Dec 05, 2020
Ajout d'instructions
54b91c3
Showing 1 changed file with 7 additions and 1 deletion.
Unified
Split
8
README.md
@@ -5,4 +5,10 @@ Un repo utilisé pendant la formation à git et GitHub

Toute le monde…

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Crée un **fork** ce repo sur GitHub:
https://github.com/Gymnase-Intercantonal-de-la-Broye/site-web-demo

Clone ce fork sur sa machine
Crée une branche “fb-contribution-xxx“ avec la commande
`git checkout -b fb-contribution-xxx`

Ajoute un fichier `xxx.html`
Crée une version et l'envoie vers son fork (`add`, `commit`, `push`)

Fait une "pull request” dans l’interface de GitHub
Attend que je l’ai approuvée et mergée

Se remet sur sa branche master (`git checkout master`)
Récupère le nouvel état de ma branche master
git remote add upstream
https://github.com/Gymnase-Intercantonal-de-la-Broye/site-web-demo

```
git checkout master
git pull upstream master
```