

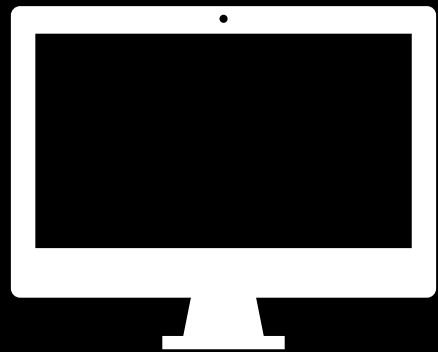
git

Motivation

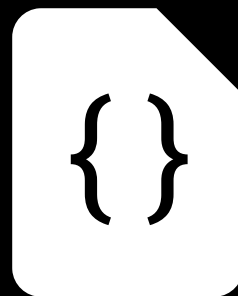
Alice travaille sur un logiciel

- Elle souhaite **sauvegarder son travail**
- Elle souhaite pouvoir **revenir en arrière** quand elle fait une erreur

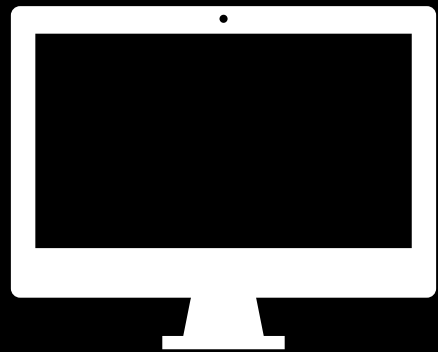




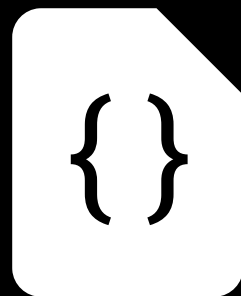
mon-projet/



a.py



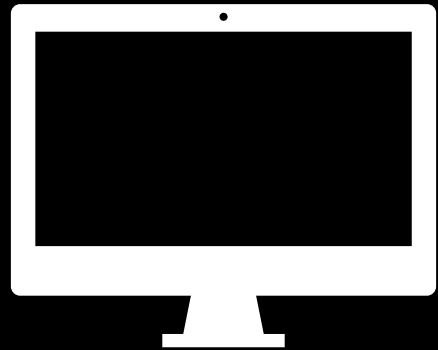
mon-projet/



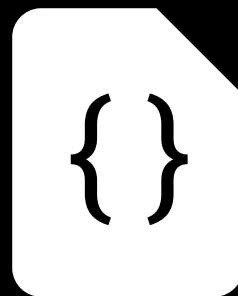
a.py

Copie de travail (« Working Copy »)

sauvegardes/

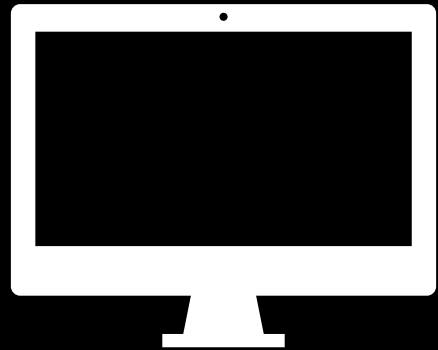


mon-projet/

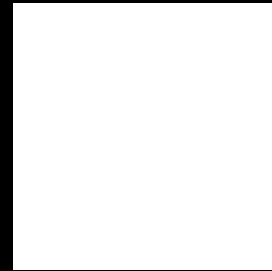


a.py

Copie de travail (« Working Copy »)



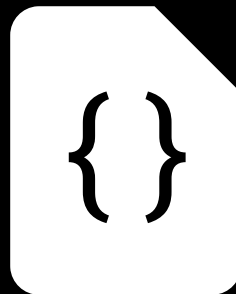
sauvegardes/



backup1.zip

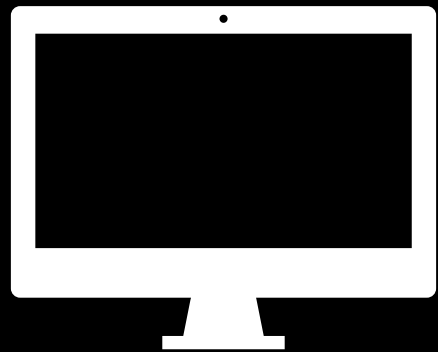


mon-projet/

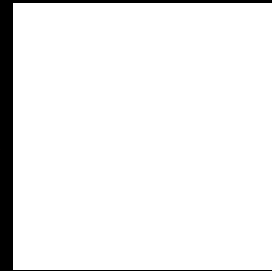


a.py

Copie de travail (« Working Copy »)



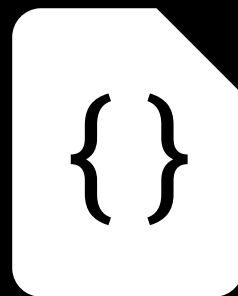
sauvegardes/



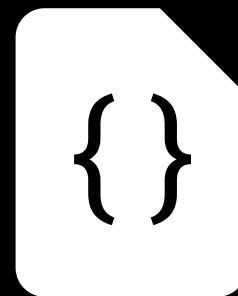
backup1.zip



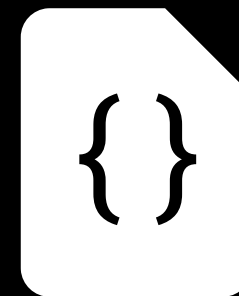
mon-projet/



a.py

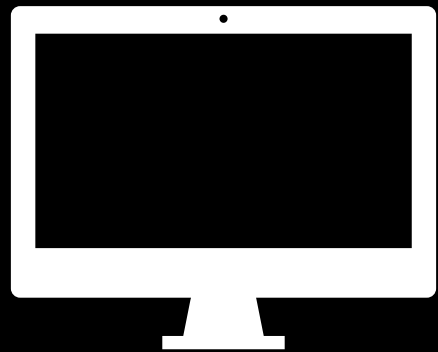


b.py

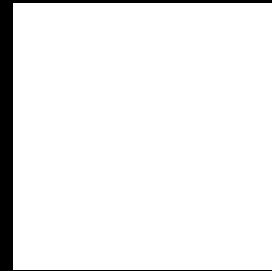


c.py

Copie de travail (« Working Copy »)



sauvegardes/



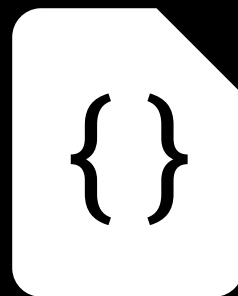
backup1.zip



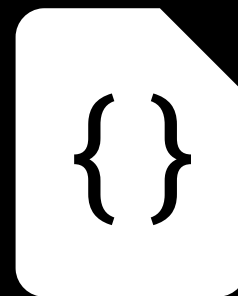
backup2.zip



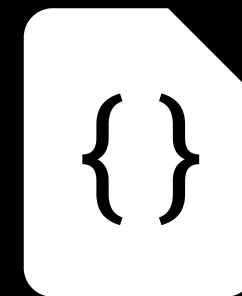
mon-projet/



a.py



b.py



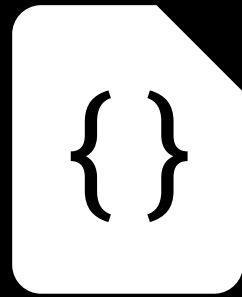
c.py

Copie de travail (« Working Copy »)

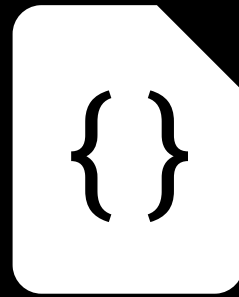
Problèmes

1. Il est très dur de **savoir ce qui a changé**
2. Ça prend vite **beaucoup de place**
3. C'est très fastidieux, risque d'oubli

Comparaison avec diff

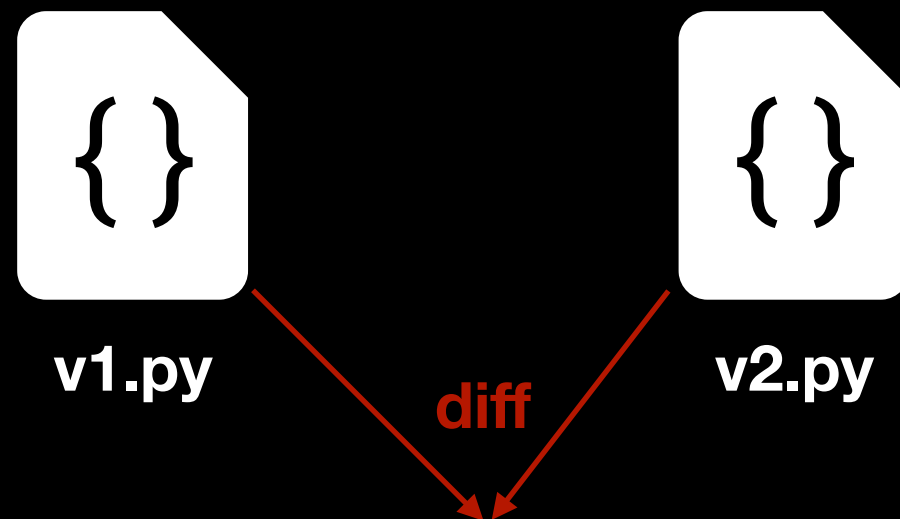


v1.py



v2.py

Comparaison avec diff



```
diff --git a/diff-colorize.py b/diff-colorize.py
old mode 100755
new mode 100644
--- a/diff-colorize.py
+++ b/diff-colorize.py
@@ -5,9 +5,9 @@
import fileinput

index_color      = int(os.environ.get('DIFF_INDEX_COLOR', 32))
old_mode_color   = int(os.environ.get('DIFF_OLD_MODE_COLOR', 124))
old_mode_color   = int(os.environ.get('DIFF_OLD_MODE_COLOR', 88))
new_mode_color    = int(os.environ.get('DIFF_NEW_MODE_COLOR', 28))
removed_color    = int(os.environ.get('DIFF_REMOVED_COLOR', 203))
removed_color    = int(os.environ.get('DIFF_REMOVED_COLOR', 160))
added_color      = int(os.environ.get('DIFF_ADDED_COLOR', 2))
hunk_start_color = int(os.environ.get('DIFF_HUNK_START_COLOR', 32))
```

diff-tools-mac.txt

Staged Unstaged

@@ -30,13 +30,11 @@ In this article, we've c... Discard Chunk Stage Chunk

30 30 comes in handy.

31 31

32 32 ----

33 33

-34 Intro: Staying up-to-date in a software, writing, or design project is hard - especially when multiple people are working on it.

-35 Without the right tools, you won't be able to understand the changes that move the project forward.

-36 This is where a diff tool comes in handy. It makes changes visible and helps you understand them.

+34 Intro: Staying up-to-date in a software, writing, or design project is hard - especially when multiple people are working on it. Without the right tools, you won't be able to understand the changes that move the project forward.

37 35

-38 In this article, we've compiled a short list that helps you get an overview of the best diff tools on the Mac.

This is where a diff tool comes in handy. It makes changes

Logiciel de gestion de versions

- Utiliser un outil dédié
- Permet de rendre la sauvegarde plus rapide, d'acquérir des automatismes
- Ce qu'on a décrit correspond à l'outil RCS (1982)

Bob souhaite aider Alice

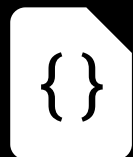
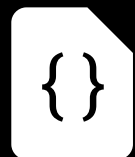
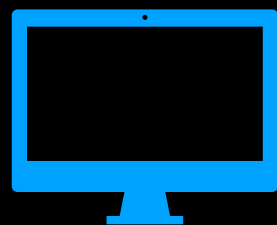
- Il souhaite pouvoir travailler **en même temps** qu'Alice sur le logiciel
- Ils doivent donc **synchroniser** leur travail régulièrement



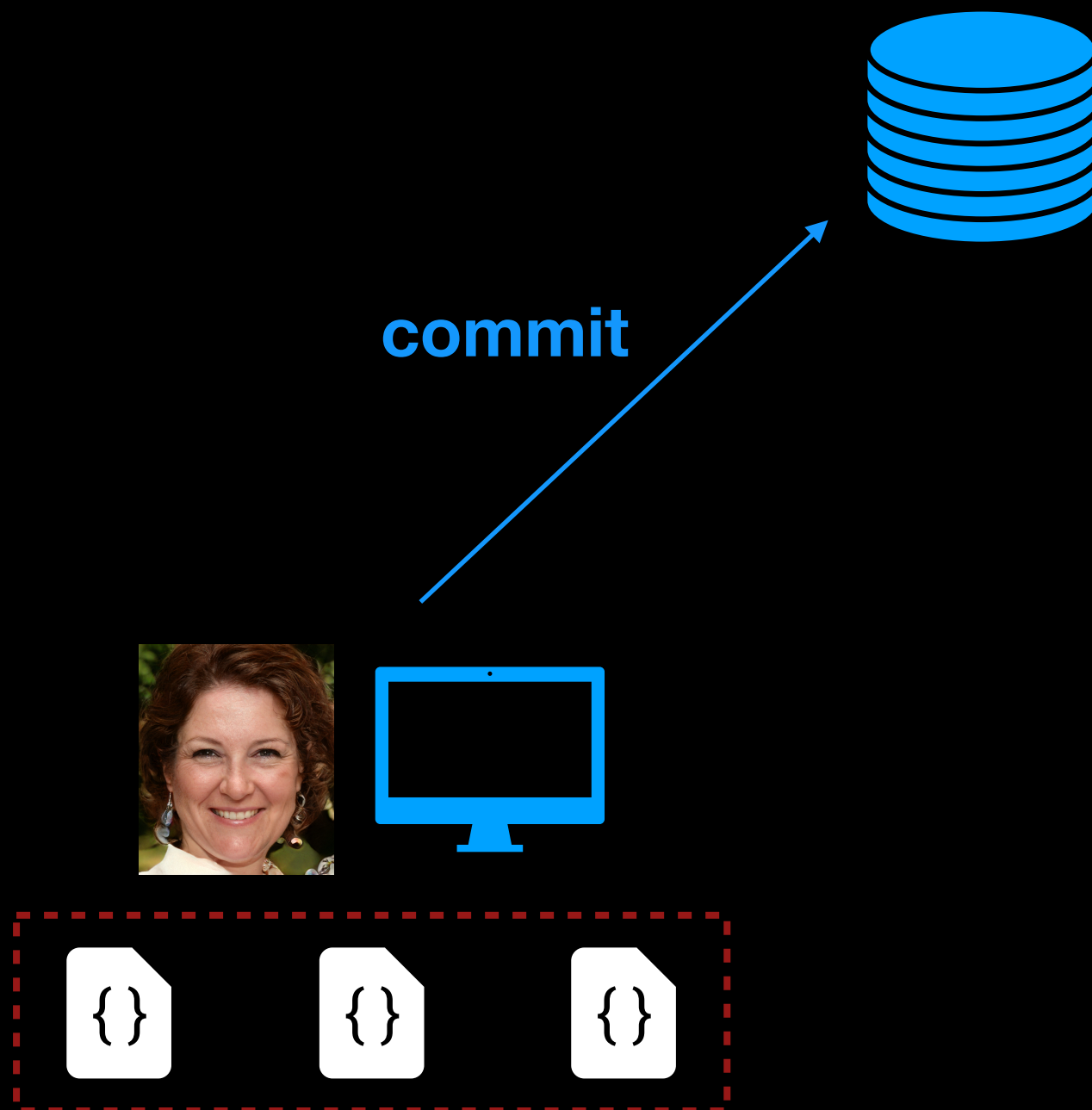
Solutions intuitives

- Utiliser un disque partagé
- S'envoyer les changements par email
- On risque « d'écraser" les changements de l'autre
- Ça fait beaucoup d'emails...

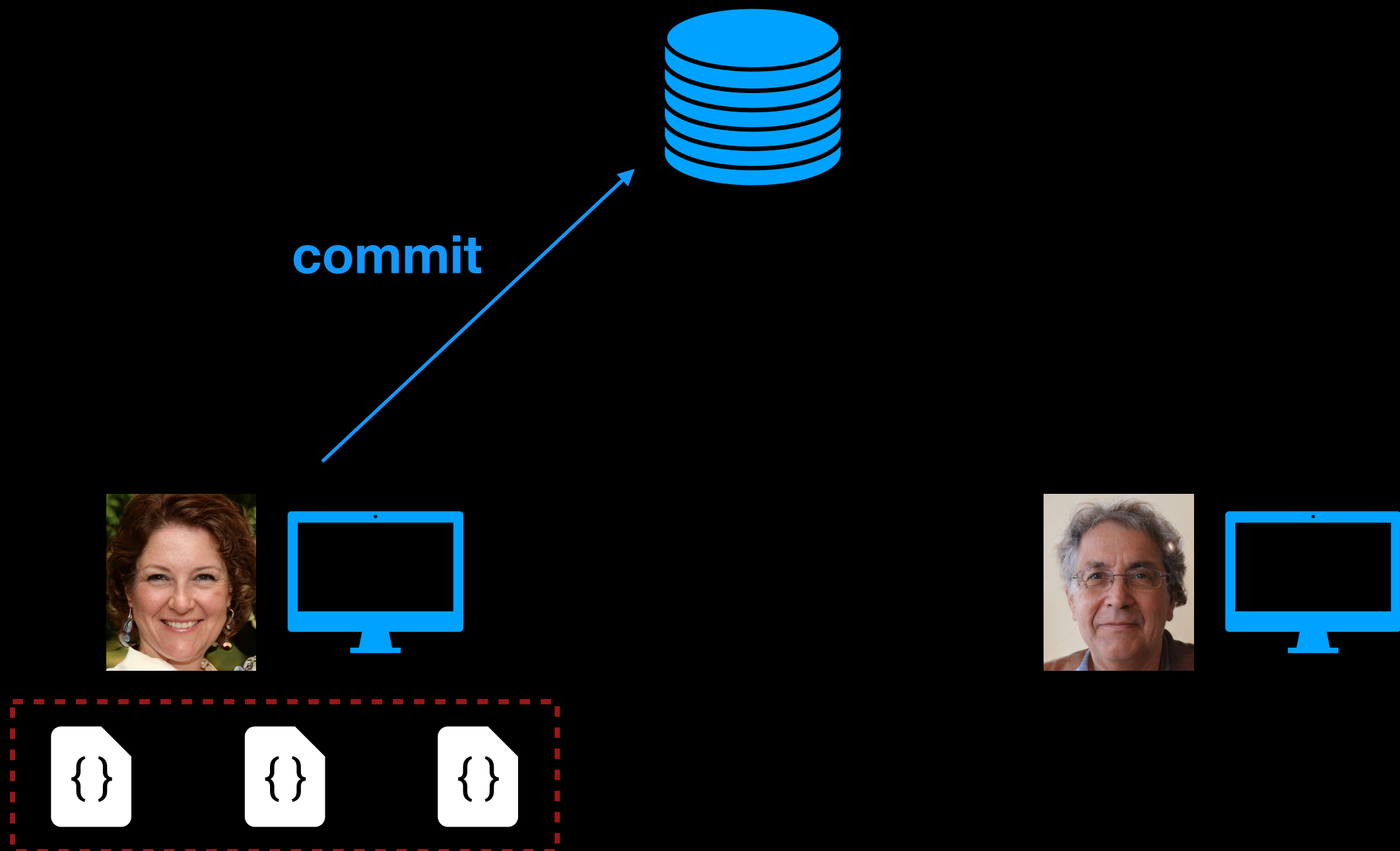
Solution: utiliser un outil de gestion de versions« avec serveur »»



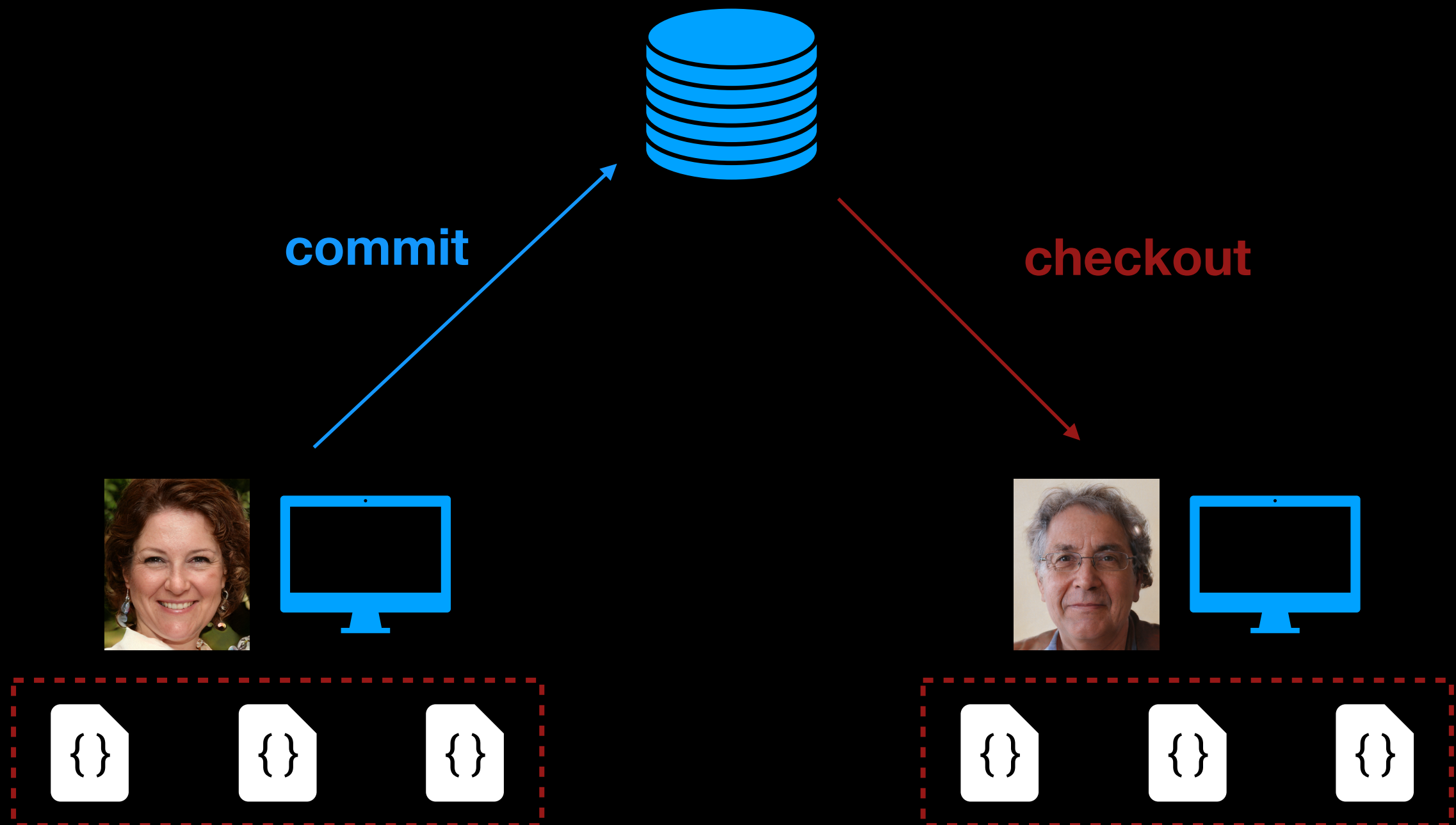
Solution: utiliser un outil de gestion de versions« avec serveur »



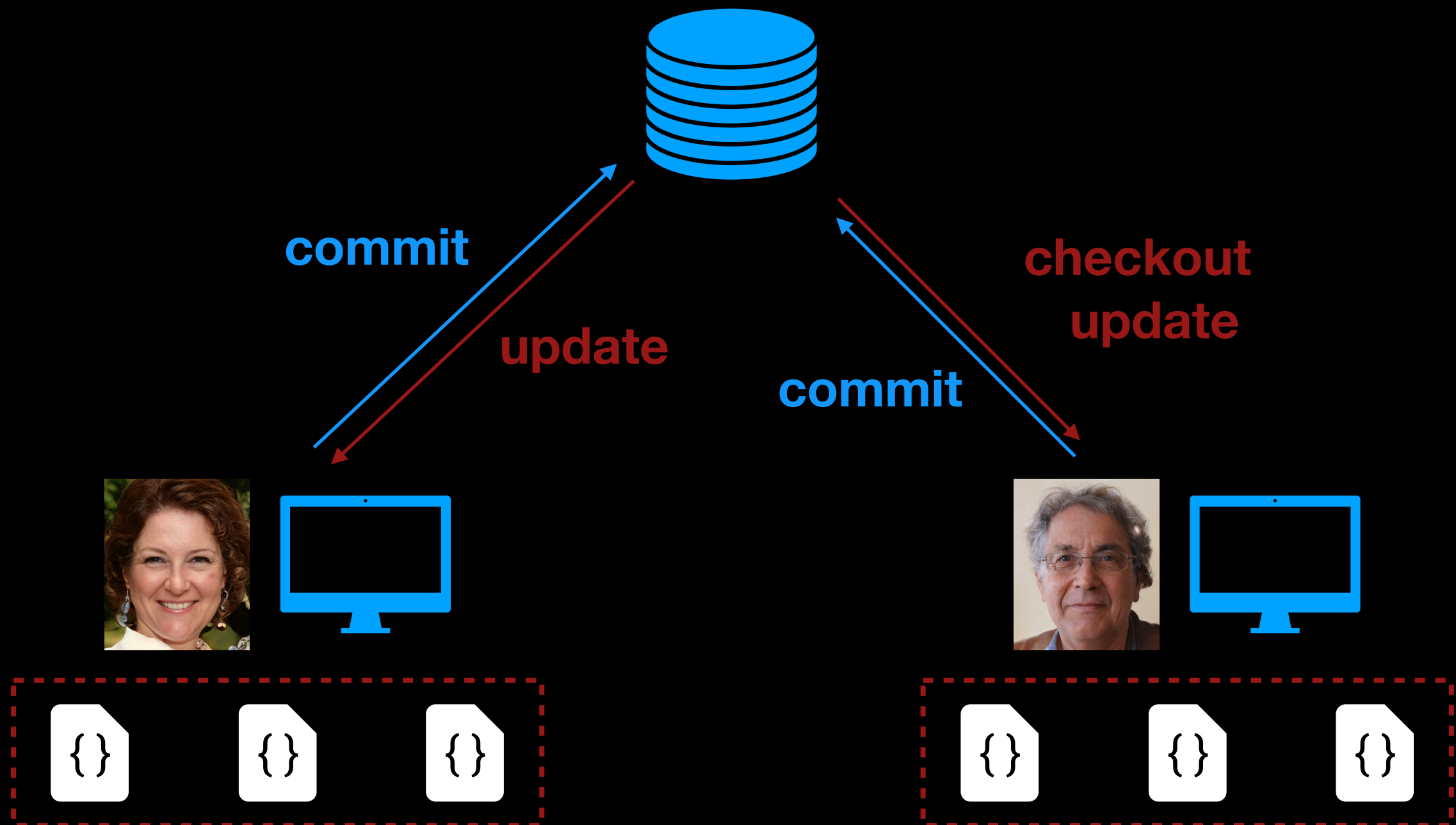
Solution: utiliser un outil de gestion de versions« avec serveur »



Solution: utiliser un outil de gestion de versions« avec serveur »



Solution: utiliser un outil de gestion de versions« avec serveur »



Avantages / Inconvénients

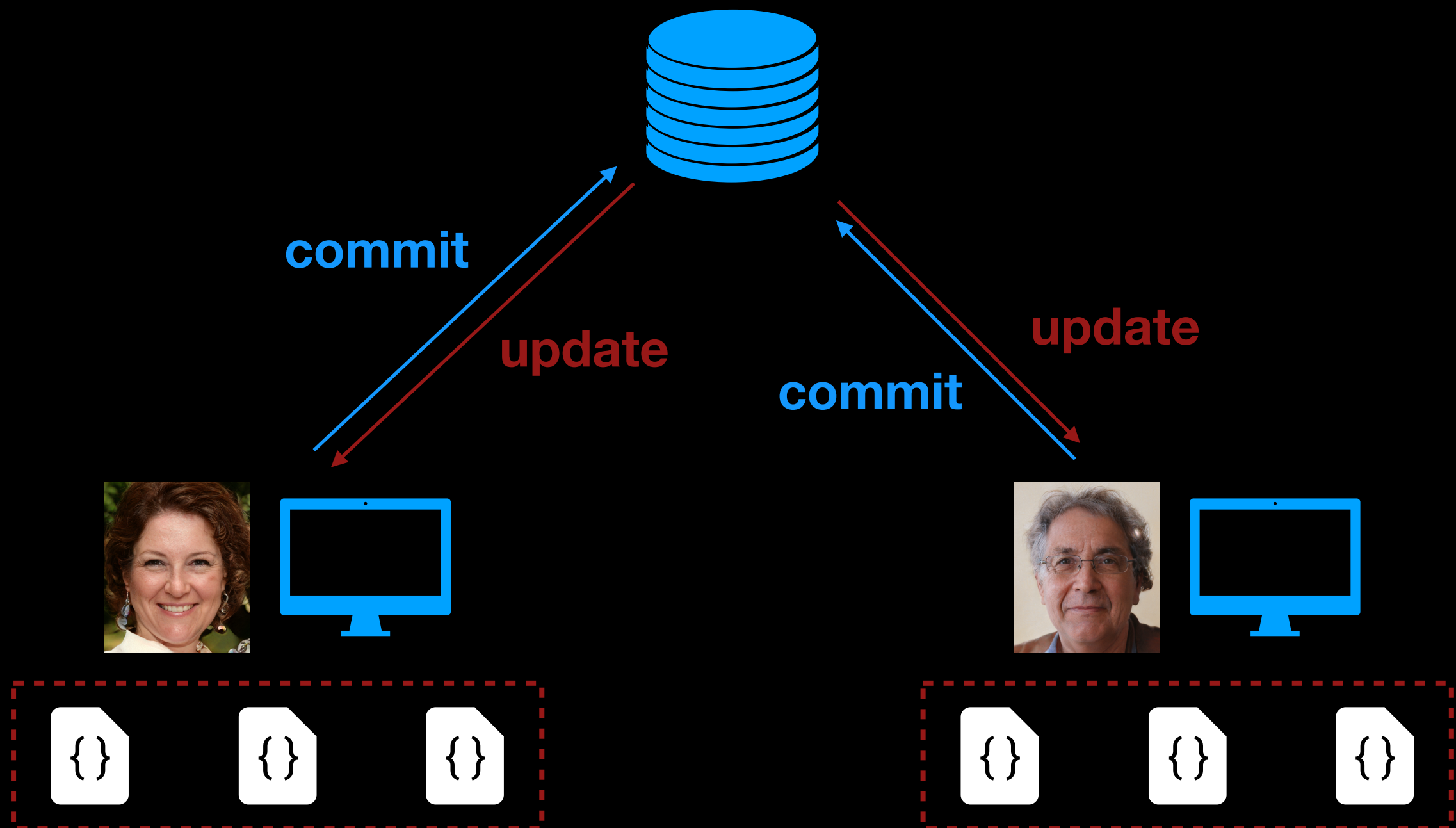
- L'outil encadre le process: pas d'email à envoyer !
- On récupère très facilement la dernière version
- Gestion automatisée des « conflits »
- Impossible de sauvegarder ses changements sans réseau
- Il y a un risque de « Hacking" du serveur

Il faut un outil plus puissant

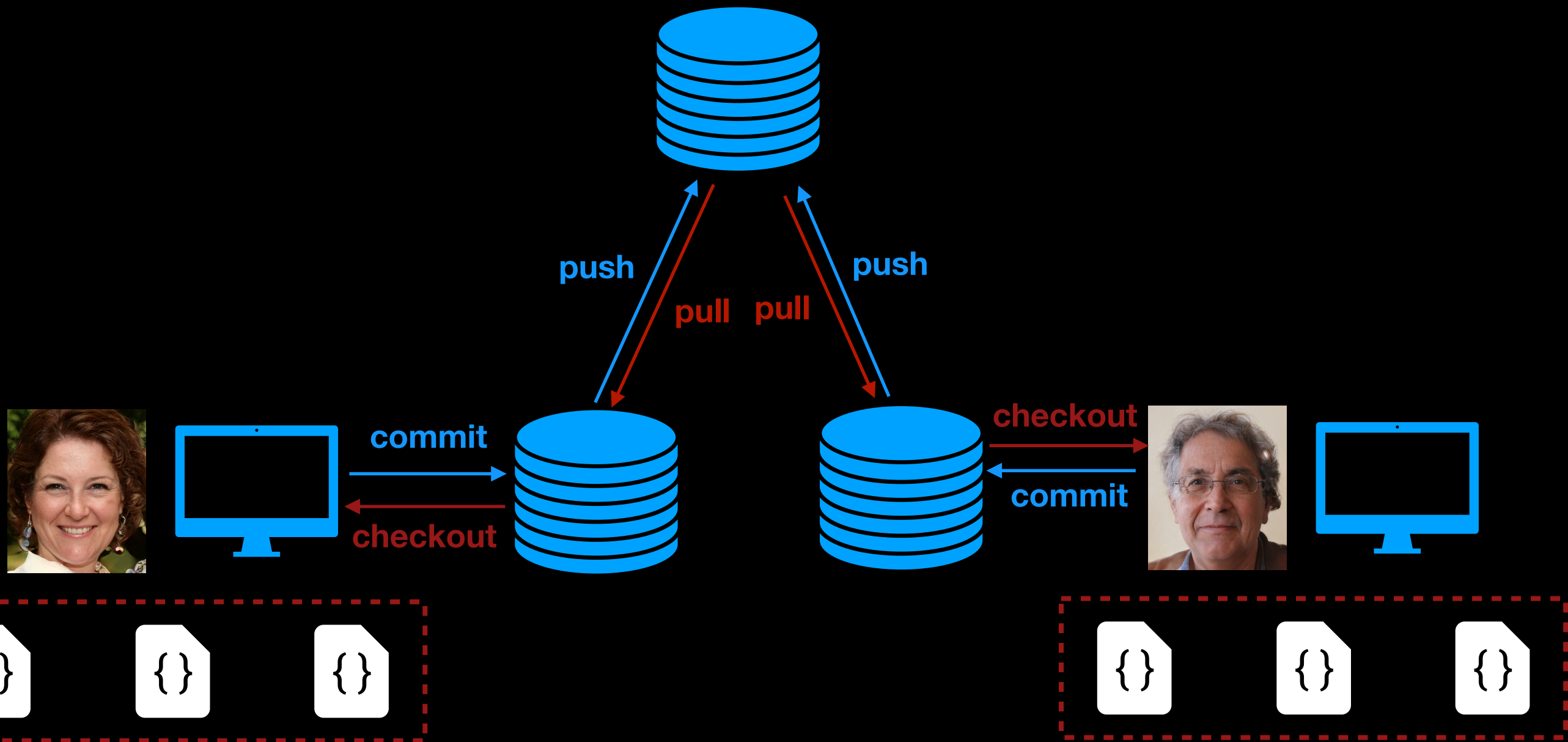


Il faut un outil plus puissant





Git: local + distant



Ce qui a changé

- Chacun a son propre **dépôt** local et peut donc travailler **sans réseau**
- On peut **synchroniser** à tout moment **les dépôts** (local/distant) entre eux

Autres avantages

- Git est sécurisé: on détecte facilement toute modification non autorisée
- Git est extensible: on peut lui brancher un grand nombre d'outils annexes

Mais

- C'est un outil **complexe**: il y a énormément de commandes

Petit cas pratique

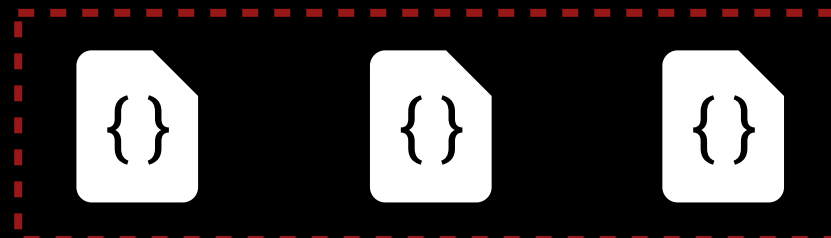
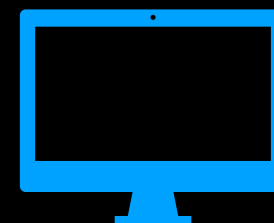
En repartant de zéro

Alice crée un projet



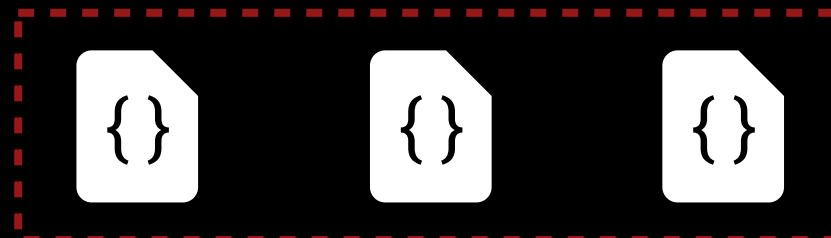
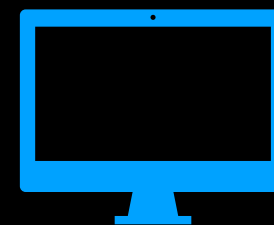
Alice crée un projet

- Elle créer une première version sur son ordinateur



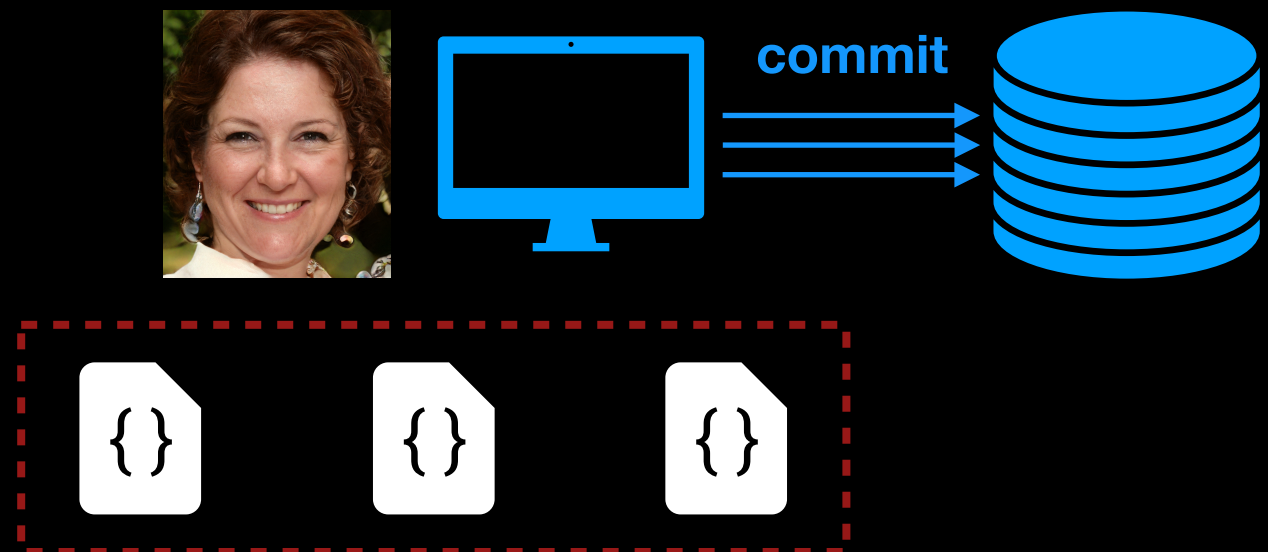
Alice crée un projet

- Elle crée une première version sur son ordinateur
- Puis un dépôt git **local** pour travailler, et versionner son travail



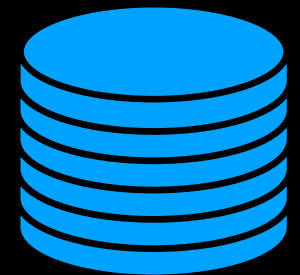
Alice crée un projet

- Elle créer une première version sur son ordinateur
- Puis un dépôt git **local** pour travailler, et versionner son travail

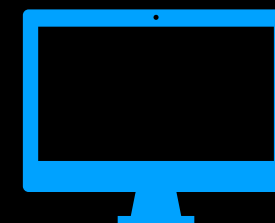


Alice crée un projet

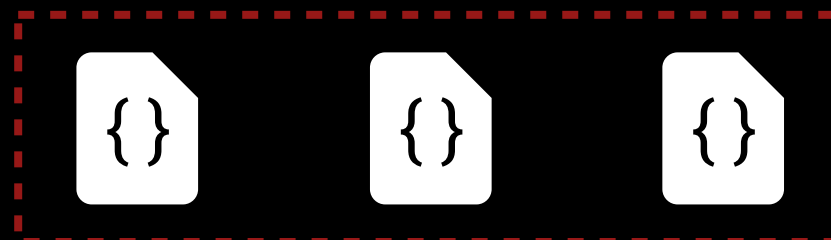
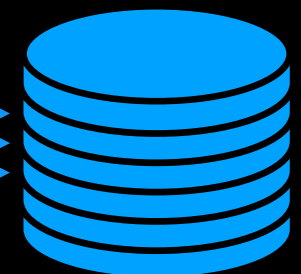
github.com



- Elle crée une première version sur son ordinateur
- Puis un dépôt git **local** pour travailler, et versionner son travail
- Elle crée ensuite un dépôt **public** sur github.com pour « open-sourcer » son projet

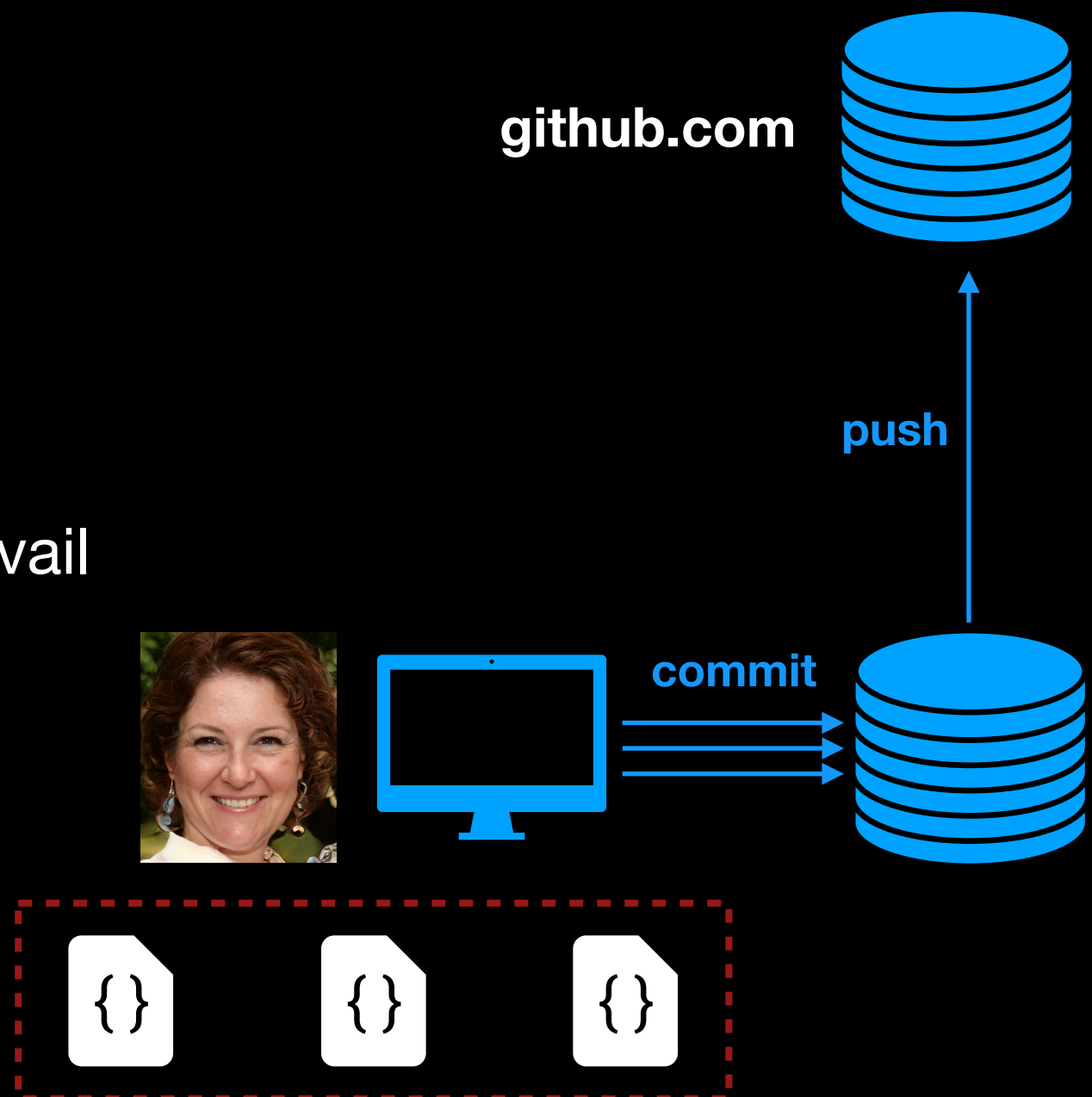


commit

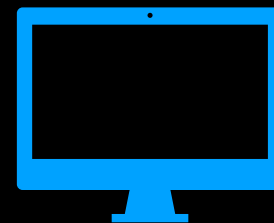


Alice crée un projet

- Elle crée une première version sur son ordinateur
- Puis un dépôt git **local** pour travailler, et versionner son travail
- Elle crée ensuite un dépôt **public** sur github.com pour « open-sourcer » son projet
- Et téléverse (« push ») son code sur le serveur GitHub



Bob remarque le travail d'Alice

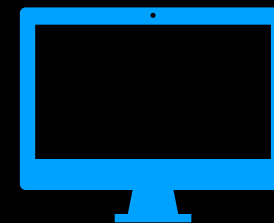


Bob remarque le travail d'Alice

github.com

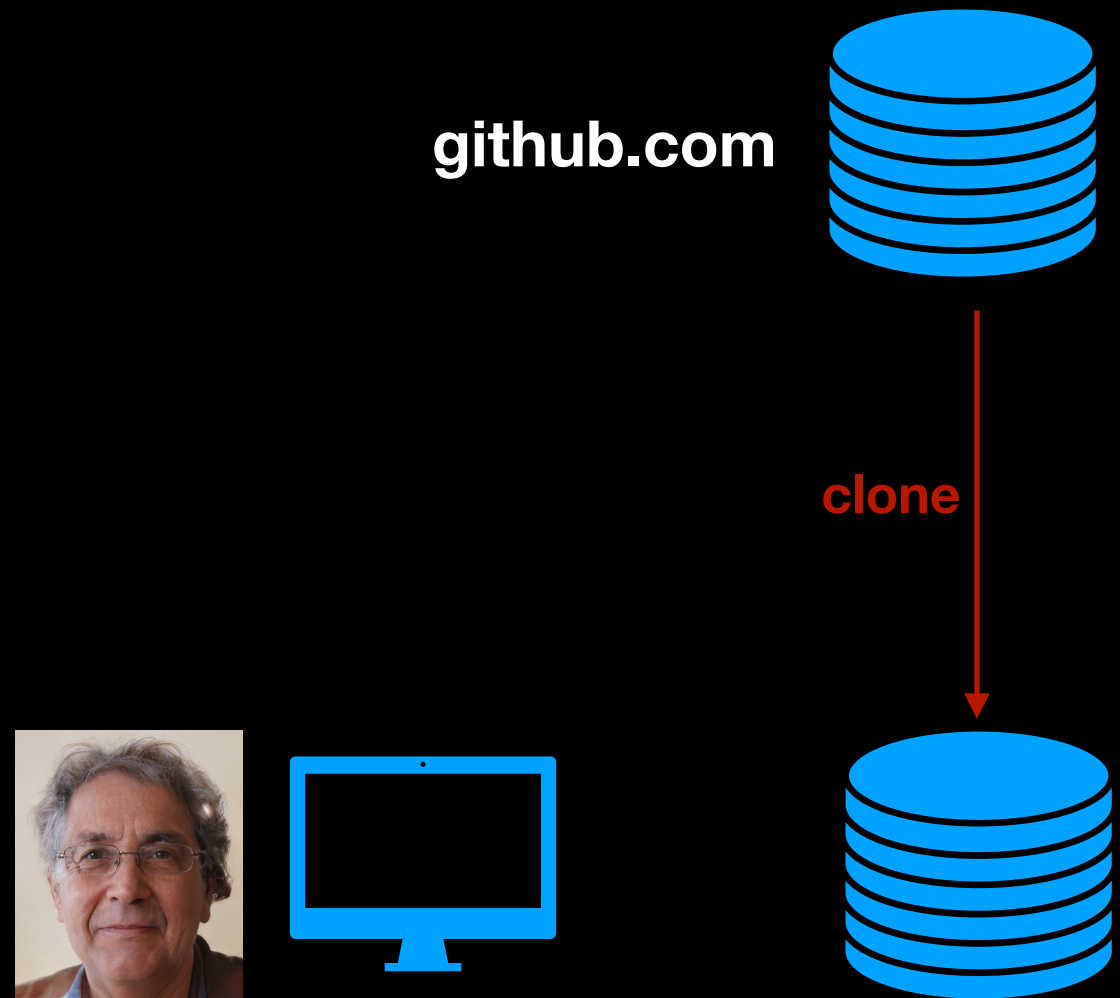


- Il remarque le travail d'Alice (via Google) et souhaite l'utiliser



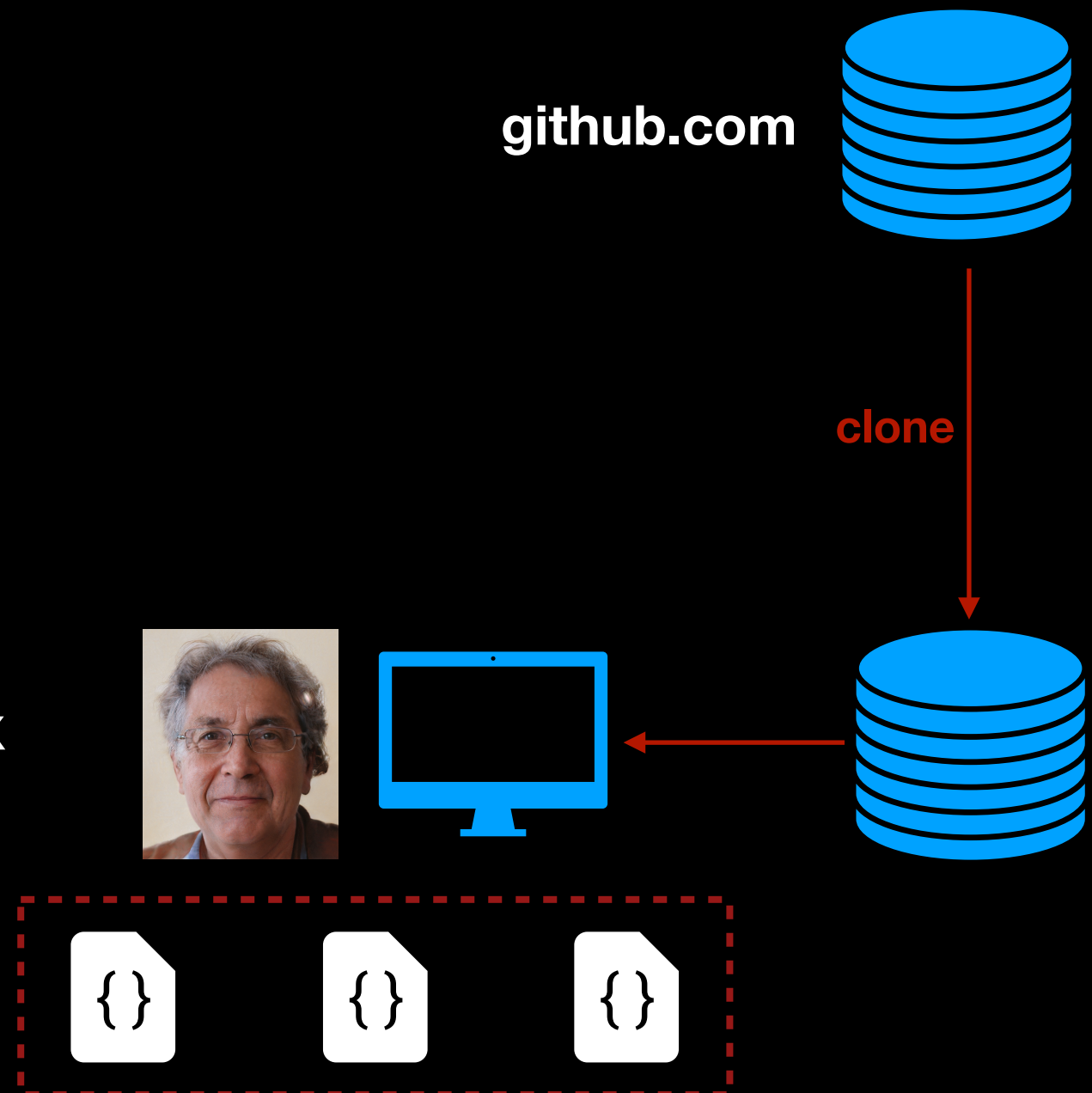
Bob remarque le travail d'Alice

- Il remarque le travail d'Alice (via Google) et souhaite l'utiliser
- Il copie le dépôt distant localement

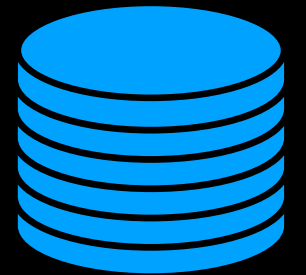


Bob remarque le travail d'Alice

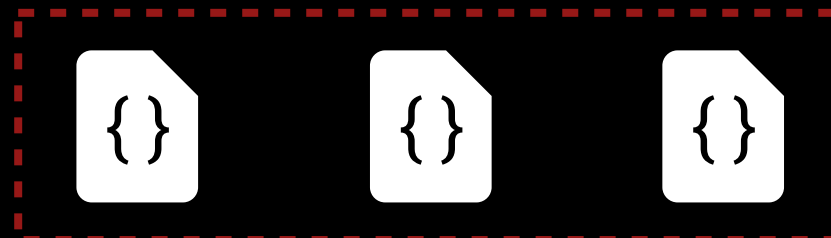
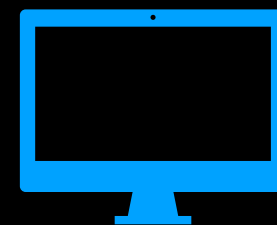
- Il remarque le travail d'Alice (via Google) et souhaite l'utiliser
- Il copie le dépôt distant localement
- Il peut désormais accéder aux fichiers
- (ainsi qu'à toutes les différentes versions)



Bob a trouvé un bug (!)

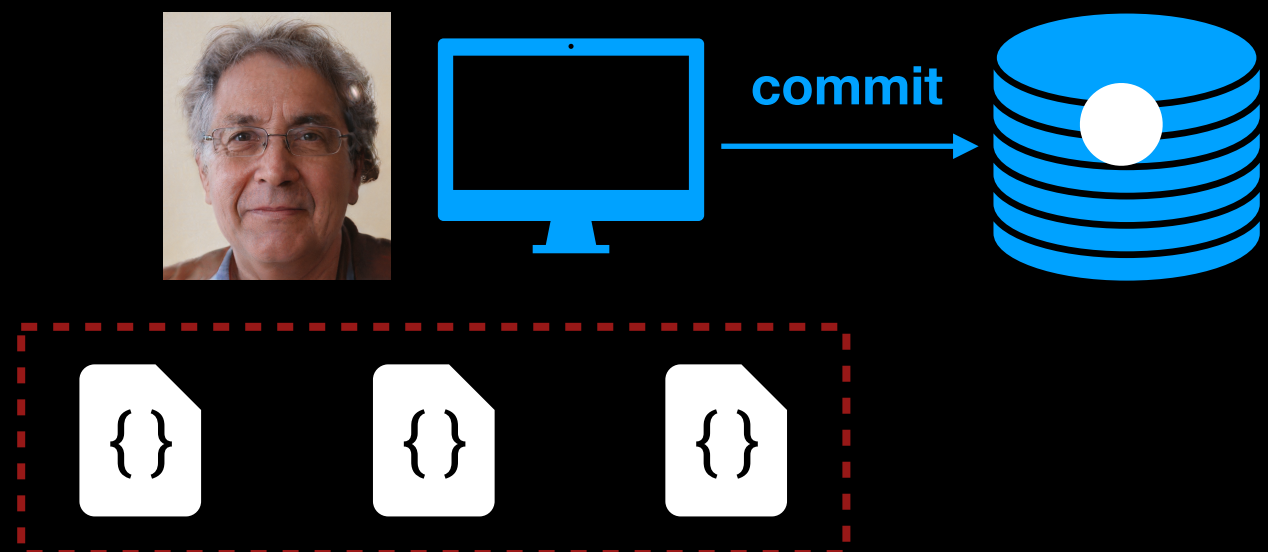


github.com
(Alice)



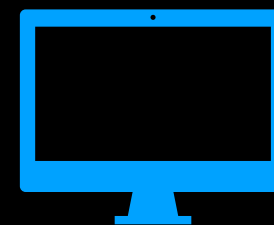
Bob a trouvé un bug (!)

- Il corrige **localement** le logiciel

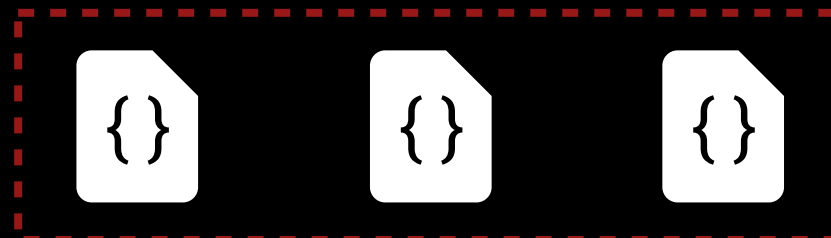
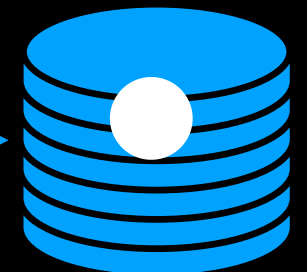


Bob a trouvé un bug (!)

- Il corrige **localement** le logiciel
- Mais il ne peut pas publier sur le dépôt distant de Alice: il doit créer sa **propre copie** publique (« fork »)

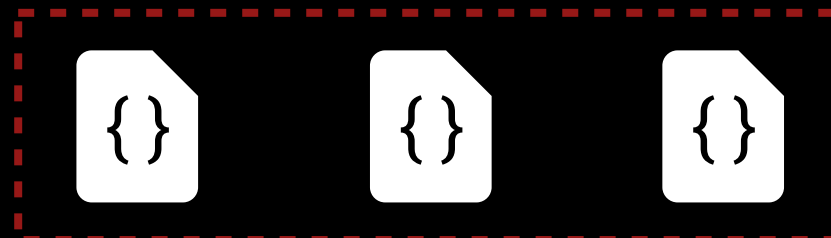
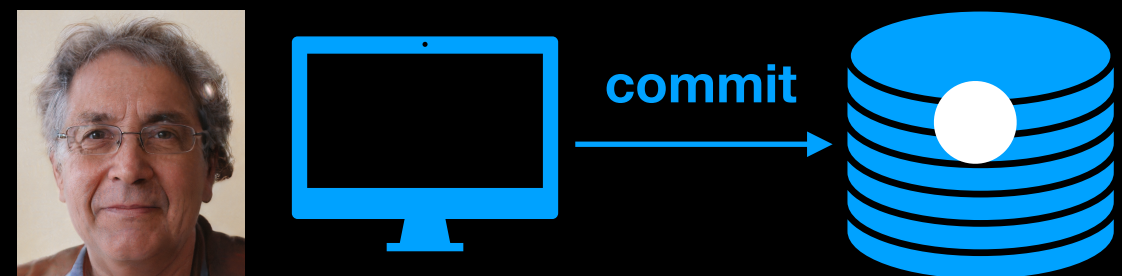
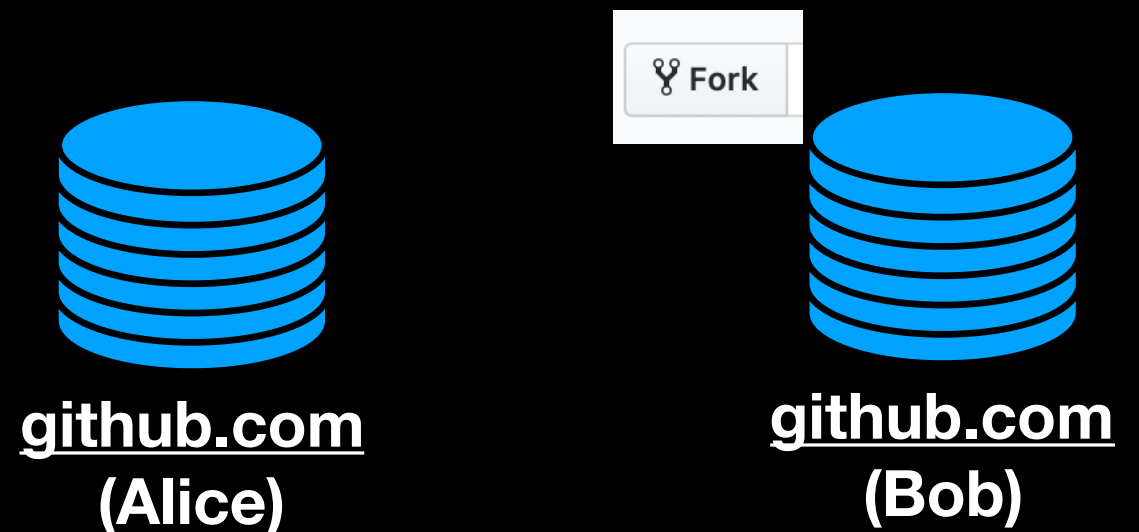


commit



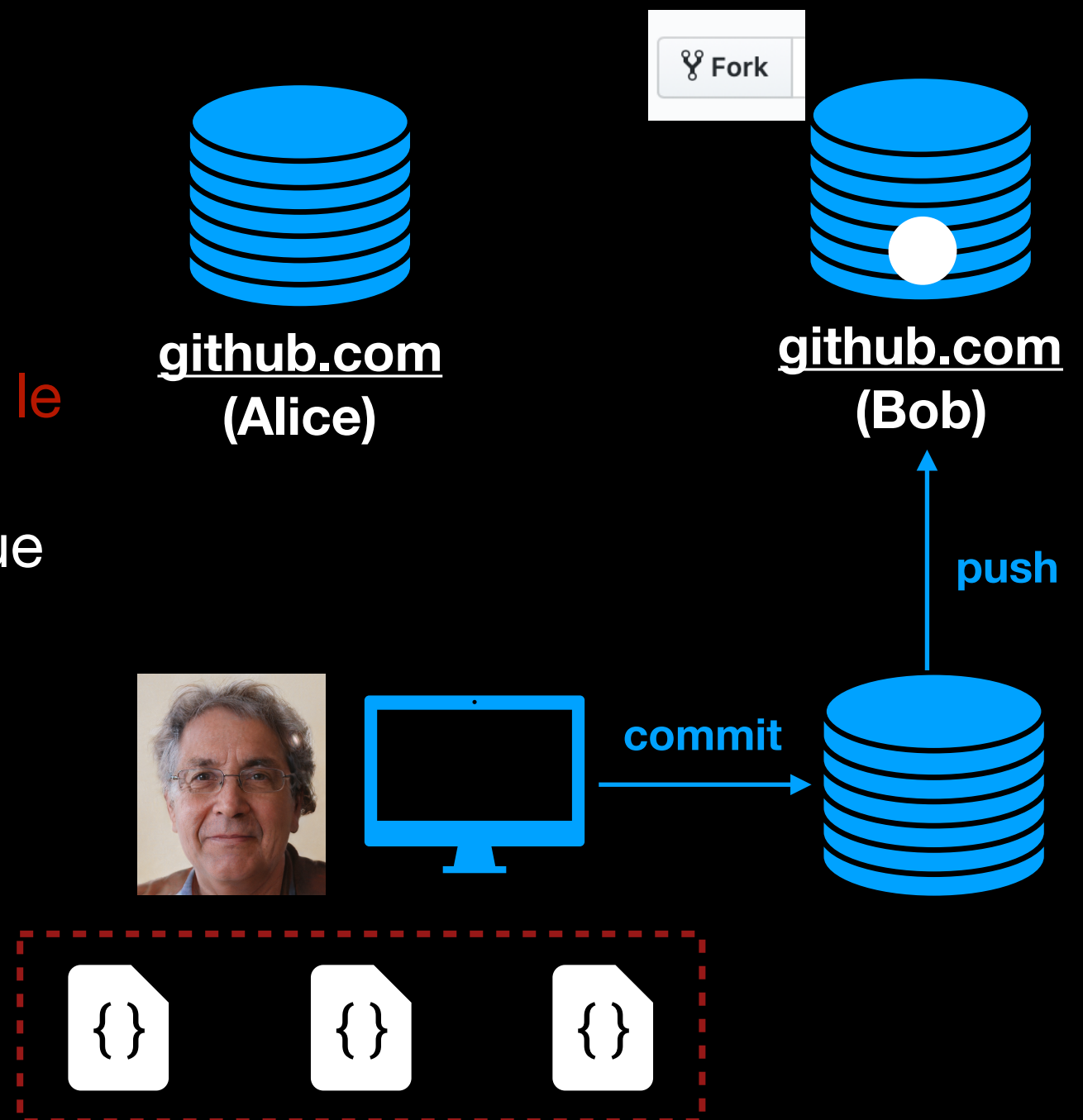
Bob a trouvé un bug (!)

- Il corrige **localement** le logiciel
- Mais il ne peut pas publier sur le dépôt distant de Alice: il doit créer sa **propre copie** publique (« fork »)



Bob a trouvé un bug (!)

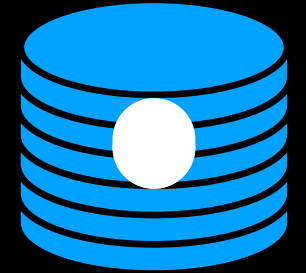
- Il corrige **localement** le logiciel
- Mais il ne peut pas publier sur le dépôt distant de Alice: il doit créer sa propre copie publique (« fork »)
- Il est désormais autorisé à publier ses changements, sur son dépôt **public**



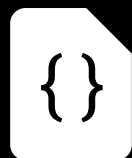
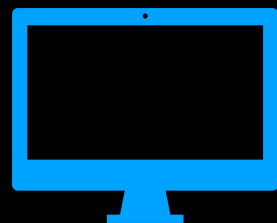
Alice récupère les changements



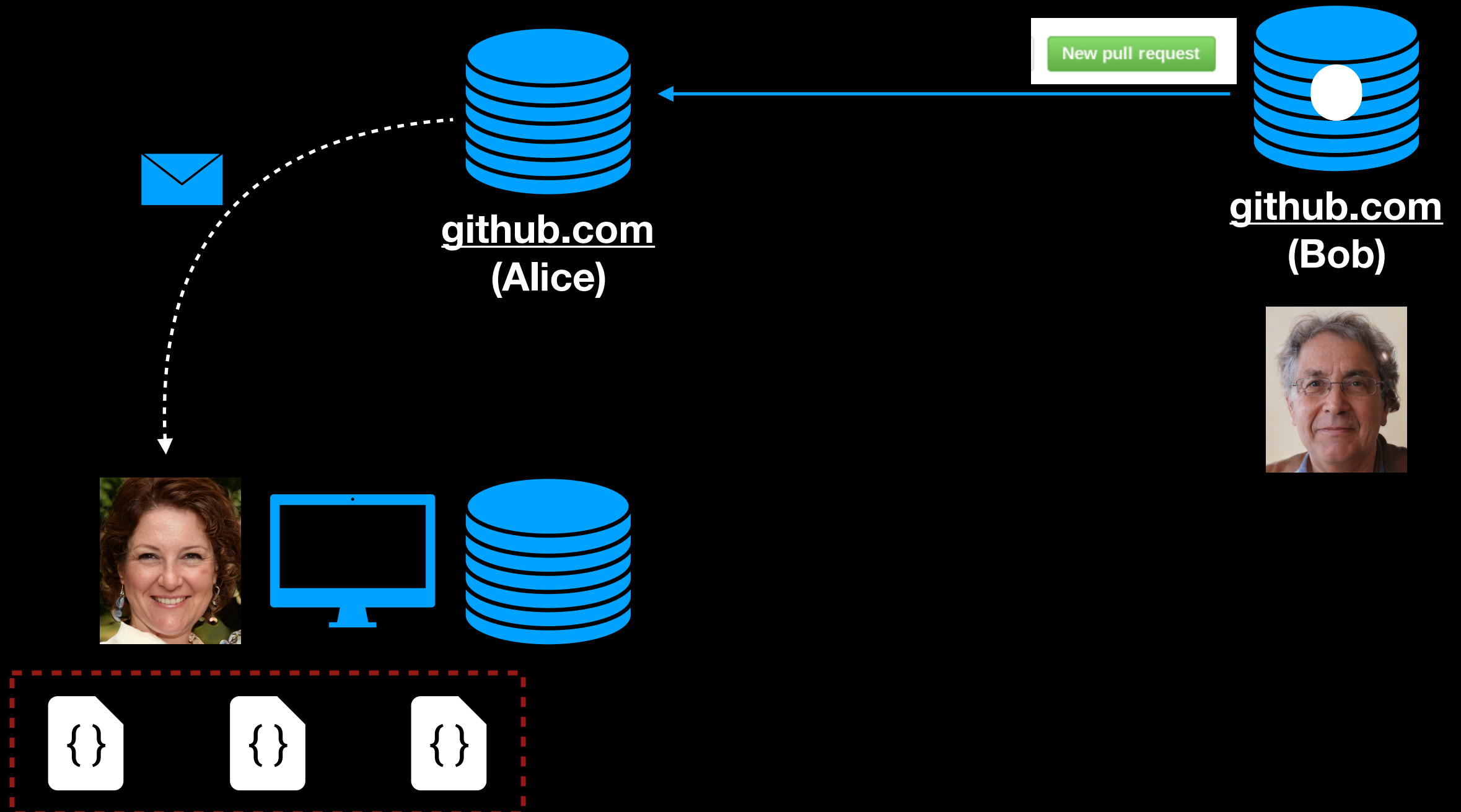
github.com
(Alice)



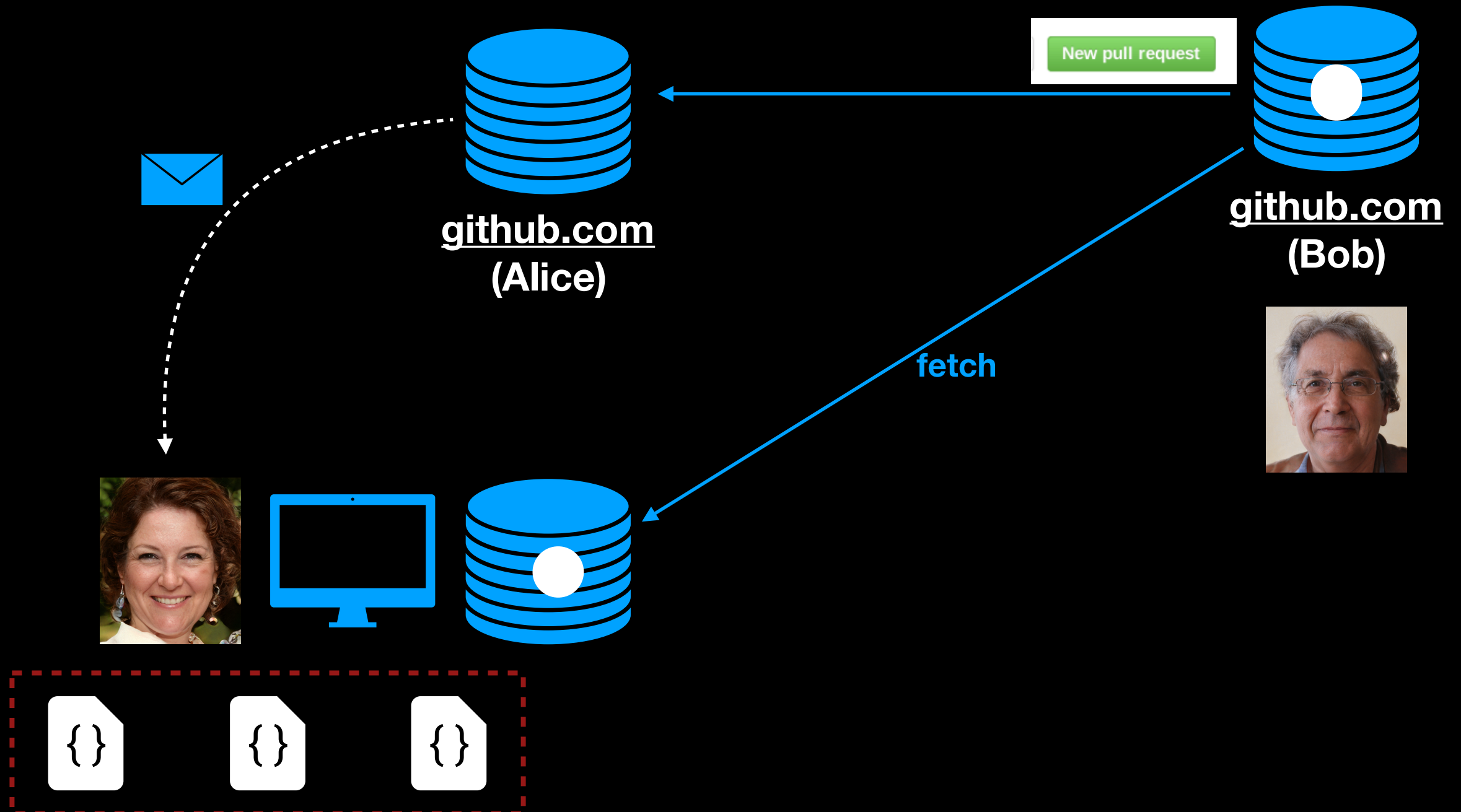
github.com
(Bob)



Alice récupère les changements



Alice récupère les changements



Alice récupère les changements

