GRENOBLE INP UGA Esisar VALENCE

Outils de développement – suite

Stéphanie CHOLLET

En pratique…

- Le cycle de vie logiciel est un processus qui permet de produire un logiciel qui fonctionne.

Pas seulement coder !

Normalisation du code

Gestion de la documentation

Scripts pour la compilation,
le packaging et le déploiement
d'applications de manière reproductible

Gestion des versions
Version Control System

Plate-forme de tests

Suivi des problèmes
Bug tracker

Analyse statistique
du code

Et …

stephanie.chollet@grenoble-inp.fr

Journalisation des événements

Utilisation d'un Logger

Contexte – 1/2

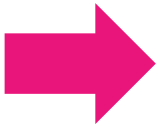
```
public class Calculateur {

  public static void main(String[] args) {
    Ajouter ajouter = new Ajouter();
    Soustraire soustraire = new Soustraire();
    Diviser diviser = new Diviser();

    Calculatrice calculatrice1 = new Calculatrice();
    calculatrice1.ajouterOperation(ajouter);
    calculatrice1.ajouterOperation(soustraire);
    calculatrice1.ajouterOperation(diviser);
    System.out.println("Calculatrice avec des operations");
    System.out.println(calculatrice1.chercherOperation("/"));
    try {
      System.out.println(calculatrice1.calculer("+", 2.0, 4.5));
      System.out.println(calculatrice1.calculer("-", 2.0, 4.5));
      System.out.println(calculatrice1.calculer("/", 2.0, 4.5));
      System.out.println(calculatrice1.calculer("/", 2.0, 0.0));
      System.out.println(calculatrice1.calculer("/", 2.0, 2.0));
    } catch (CalculatriceException e) {
      e.printStackTrace();
    }
  }
}
```


stephanie.chollet@grenoble-inp.fr

Contexte – 2/2

- Résultat de l'exécution :

```
Calculatrice avec des operations
fr.esisar.calculatrice.operations.binaires.Diviser@372f7a8d
Vérification nombre d'opérandes
doCalculer de Ajouter
6.5
Vérification nombre d'opérandes
Soustraire : doCalculer
-2.5
Vérification nombre d'opérandes
0.4444444444444444
Vérification nombre d'opérandes
CalculatriceException [message=Division par zero]
at fr.esisar.calculatrice.operations.binaires.Diviser.doCalculer(Diviser.java:16)
at fr.esisar.calculatrice.operations.OperationBinaire.calculer(OperationBinaire.java:13)
at fr.esisar.calculatrice.Calculatrice.calculer(Calculatrice.java:44)
at fr.esisar.calculatrice.Calculateur.main(Calculateur.java:24)
```

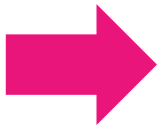


Programme fini ou en cours de développement ou en phase de débogage ?

stephanie.chollet@grenoble-inp.fr

Problématique

- D'où viennent les messages ?
  - De quelle classe ? De quelle ligne ?
- Comment différencier les traces d'exécution « normales » des messages de debogage ?
  - Serait-il possible d'avoir un contrôle du format des messages ?
  - Serait-il possible de modifier le niveau d'affichage des messages en fonction des besoins (debug vs. production) ?
- Serait-il possible d'avoir un moyen de stocker les messages ailleurs que dans la console ?
  - Contenu de la console éphémère
  - Difficile de comparer/d'analyser des exécutions



Utilisation d'un système de journalisation des événements (*logging*)

stephanie.chollet@grenoble-inp.fr

Objectif de la journalisation/logging

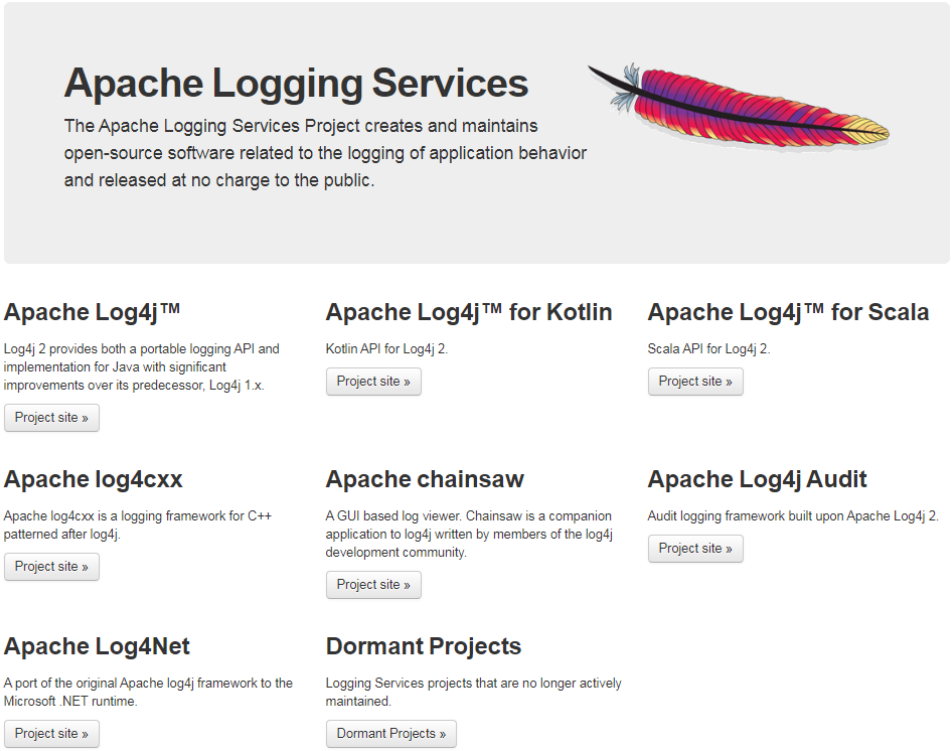
- Journaliser/historiser les événements normaux et anormaux survenus au cours de l'exécution d'un processus logiciel :
  - Application, activité réseau informatique…
- Lors du développement :
  - Pour aider au débogage du code et à la compréhension d'une application
  - Pour faciliter les anomalies remontées par la QA (Assurance Qualité)
- Lors de l'exécution dans l'environnement de production :
  - Pour comprendre et résoudre d'éventuels problèmes

Frameworks de logging Java

- Java Logging
- Apache Log4j et Apache Log4j2
- slf4j
- JLog
- LogBack
- Protomatter

Apache Log4j™ 2

- Bibliothèque logicielle open source programmée en Java et développée par Apache Software Foundation
  - Fournit des fonctions permettant de gérer des traces et des historiques d'applications
  - Fait partie de Apache Logging Services



https://logging.apache.org/log4j/2.x/index.html

stephanie.chollet@grenoble-inp.fr

Principaux concepts

- Logger :
  - Emission d'un message généralement avec un niveau de gravité
- Formatter :
  - Définition du format du message
- Appender :
  - Cible de stockage du message (console, fichier, base de données, email…)

stephanie.chollet@grenoble-inp.fr

Niveau de gravité des messages

+ sévère

- sévère

Niveau	Description
OFF	Niveau le plus élevé destiné à désactiver les traces
FATAL	Erreurs graves provoquant la fin du programme de manière prématurée. Ex : indisponibilité d'une base de données
ERROR	Autres erreurs ou conditions inattendues
WARN	Utilisation d'API obsolètes ou mauvais usage d'API. Ex : erreur de login, données invalides
INFO	Evénements importants comme le démarrage ou la fin normale du programme. Ex : chargement d'un fichier de configuration.
DEBUG	Informations très détaillées sur le déroulement du programme. Ex : affichage de valeur de données
TRACE	Informations de trace supposées n'apparaître que dans les fichiers de logs. Ex : entrée/sortie de méthodes

stephanie.chollet@grenoble-inp.fr

Le format des messages

- Plusieurs layouts :
  - HTMLLayout : formate le message en HTML dans un tableau contenant les colonnes (date/heure, niveau de gravité, thead, logger et message)
  - PatternLayout : layout le plus puissant puisqu'il permet de préciser le format du message grâce à un motif
  - SimpleLayout : layout le plus simple qui ne contient que le niveau de gravité et le message
  - XMLLayout : formate le message en XML

PatternLayout – 1/2

Motif	Rôle
%c	Le nom du logger qui a émis le message
%C	Le nom de la classe qui a émis le message
%d	Le timestamp de l'émission du message. Il est possible de fournir un format pour la date/heure en utilisant les motifs de la classe SimpleDateFormat. Exemple : %d{dd MMM yyyy HH:MM:ss }
%m	Le message
%n	Un saut de ligne dépendant de la plate-forme
%p	Le niveau de gravité du message
%r	Le nombre de millisecondes écoulées entre le lancement de l'application et l'émission du message
%t	Le nom du thread
%L	Le numéro de ligne dans le code émettant le message
%F	Le nom du fichier émettant le message
%M	Le nom de la méthode émettant le message

stephanie.chollet@grenoble-inp.fr

PatternLayout – 2/2

- Le caractère # représente une des lettres du tableau précédent, n représente un nombre de caractères.

Motif	Rôle
%#	Aucun formatage (par défaut)
%n#	Alignement à droite, des blancs sont ajoutés si la taille du motif est inférieure à n caractères
%-n#	Alignement à gauche, des blancs sont ajoutés si la taille du motif est inférieure à n caractères
%.n	Tronque le motif s'il est supérieur à n caractères
%-n.n#	Alignement à gauche, taille du motif obligatoirement de n caractères (troncature ou complément avec des blancs)

stephanie.chollet@grenoble-inp.fr

Recommandations

- Bonnes pratiques pour le logging :
  - Chaque message doit contenir une information d'horodatage ainsi que le nom de la classe émettrice
  - Utiliser le niveau de gravité en adéquation avec le message
  - Eviter les messages émis trop fréquemment :
    - Contre-exemples : dans une boucle avec de nombreuses itérations, dans une méthode très fréquemment invoquée…
  - Ne jamais utiliser de System.out pour afficher des messages
  - Ne jamais utiliser la méthode printStackTrace() de la classe Exception pour afficher des messages

stephanie.chollet@grenoble-inp.fr

Stockage des messages avec les *Appenders*

- Plusieurs appenders possibles avec des configurations spécifiques :
  - ConsoleAppender : messages envoyés sur la console
  - JDBCAppender : messages envoyés dans une base de données (attention à son utilisation)
  - JMSAppender : messages envoyés vers une destination utilisant JMS
  - SMTPAppender : messages envoyés par mail
  - FileAppender : messages envoyés dans un fichier. La classe FileAppender possède deux classes filles : DailyRollingAppender et RollingFileAppender
  - …
- Un logger peut avoir plusieurs appenders : un message sera envoyé à chacun des appenders

stephanie.chollet@grenoble-inp.fr

Exemple de configuration de Apache Log4j2 – 1/3

- Avec une fichier de configuration
  - Au format XML :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Root level="trace">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

log4j2.xml

stephanie.chollet@grenoble-inp.fr

Exemple de configuration de Apache Log4j2 – 2/3

- Avec une fichier de configuration
  - Au format JSON :

```
{
  "configuration": {
    "appenders": {
      "Console": {
        "name": "STDOUT",
        "PatternLayout": {
          "pattern": "%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"
        }
      }
    },
    "loggers": {
      "root": {
        "level" : "trace",
        "AppenderRef": {
          "ref": "STDOUT"
        }
      }
    }
  }
}
```

log4j2.json

stephanie.chollet@grenoble-inp.fr

Exemple de configuration de Apache Log4j2 – 3/3

- Avec une fichier de configuration
  - Aux formats YAML ou properties
- Ou de manière programmatique

stephanie.chollet@grenoble-inp.fr

Utilisation du Logger

```java
package fr.esisar.demo;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class TestLog4j2 {

  private static final Logger LOGGER = LogManager.getLogger();

  public static void main(String[] args) {
    LOGGER.info("Message d'info");
    for(int i = 0 ; i < 3 ; i++) {
      LOGGER.debug("Message debug {}", i);
    }
    LOGGER.error("Message d'erreur");
  }
}
```

Déclaration d'une constante

Utilisation des méthodes de l'API

Concaténation d'informations

```
10:02:50.071 [main] INFO  fr.esisar.demo.TestLog4j2 - Message d'info
10:02:50.084 [main] DEBUG fr.esisar.demo.TestLog4j2 - Message debug 0
10:02:50.084 [main] DEBUG fr.esisar.demo.TestLog4j2 - Message debug 1
10:02:50.084 [main] DEBUG fr.esisar.demo.TestLog4j2 - Message debug 2
10:02:50.084 [main] ERROR fr.esisar.demo.TestLog4j2 - Message d'erreur
```

Notes de sécurité pour Apache Log4j™ 2

Apache Log4j™ 2

Apache Log4j 2 is an upgrade to Log4j that provides significant improvements over its predecessor, Log4j 1.x, and provides many of the improvements available in Logback while fixing some inherent problems in Logback's architecture.

Important: Security Vulnerability CVE-2021-44832

Summary: Apache Log4j2 vulnerable to RCE via JDBC Appender when attacker controls configuration.

Details

Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack where an attacker with permission to modify the logging configuration file can construct a malicious configuration using a JDBC Appender with a data source referencing a JNDI URI which can execute remote code. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.

Mitigation

Upgrade to Log4j 2.3.2 (for Java 6), 2.12.4 (for Java 7), or 2.17.1 (for Java 8 and later)

Reference

Please refer to the Security page for details and mitigation measures for older versions of Log4j.

stephanie.chollet@grenoble-inp.fr

Versions d’un logiciel

Qu'est-ce qu'une version d'un logiciel ?

Une **version d'un logiciel** correspond à un **état donné** de l'évolution d'un produit logiciel utilisant le versionnage (*versioning*). Elle est généralement associée à une numérotation qui permet de l'identifier, voire dans certains cas à un nom symbolique.

- Types d'évolution d'un logiciel :
  - Evolutions majeures :
    - Apport de nouvelles fonctionnalités
    - Restructuration de l'application
  - Evolutions mineures :
    - Correction de bugs
    - Ajouts de fonctionnalités secondaires

stephanie.chollet@grenoble-inp.fr

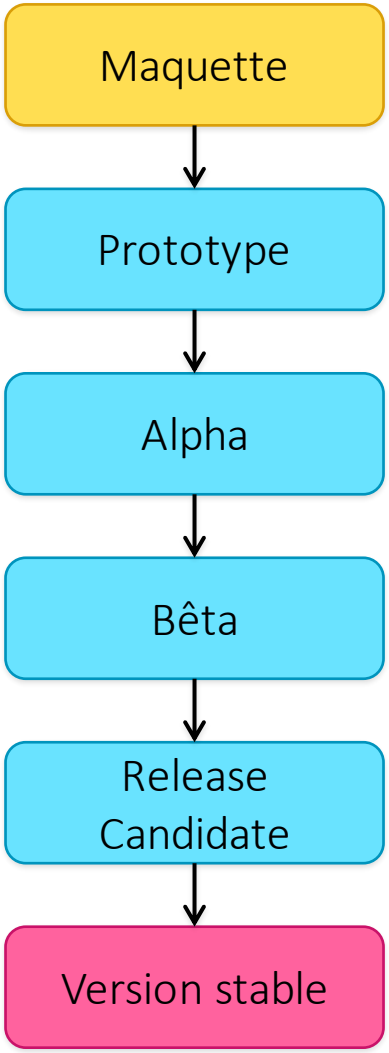
Numérotation des logiciels

- Formalisation :
  - Un ou plusieurs chiffres séparés par des points : 2.12.4
    - $2^{ème}$ version publiée, $12^{ème}$ ajout de fonctionnalités, $4^{ème}$ révision
    - Identification : majeur.mineur.micro
  - Année de sortie du logiciel : Windows 2010
  - Règle mathématique :
    - Version de TeX tend de manière asymptotique vers π
    - Version de Metafont tend de manière asymptotique vers e

- Règle de changement de la numérotation :
  - Modification mineure : +1 sur la valeur micro
  - Modification majeure : +1 sur la valeur mineure (ou majeure) + mise à 0 de la valeur micro (et mineure)

stephanie.chollet@grenoble-inp.fr

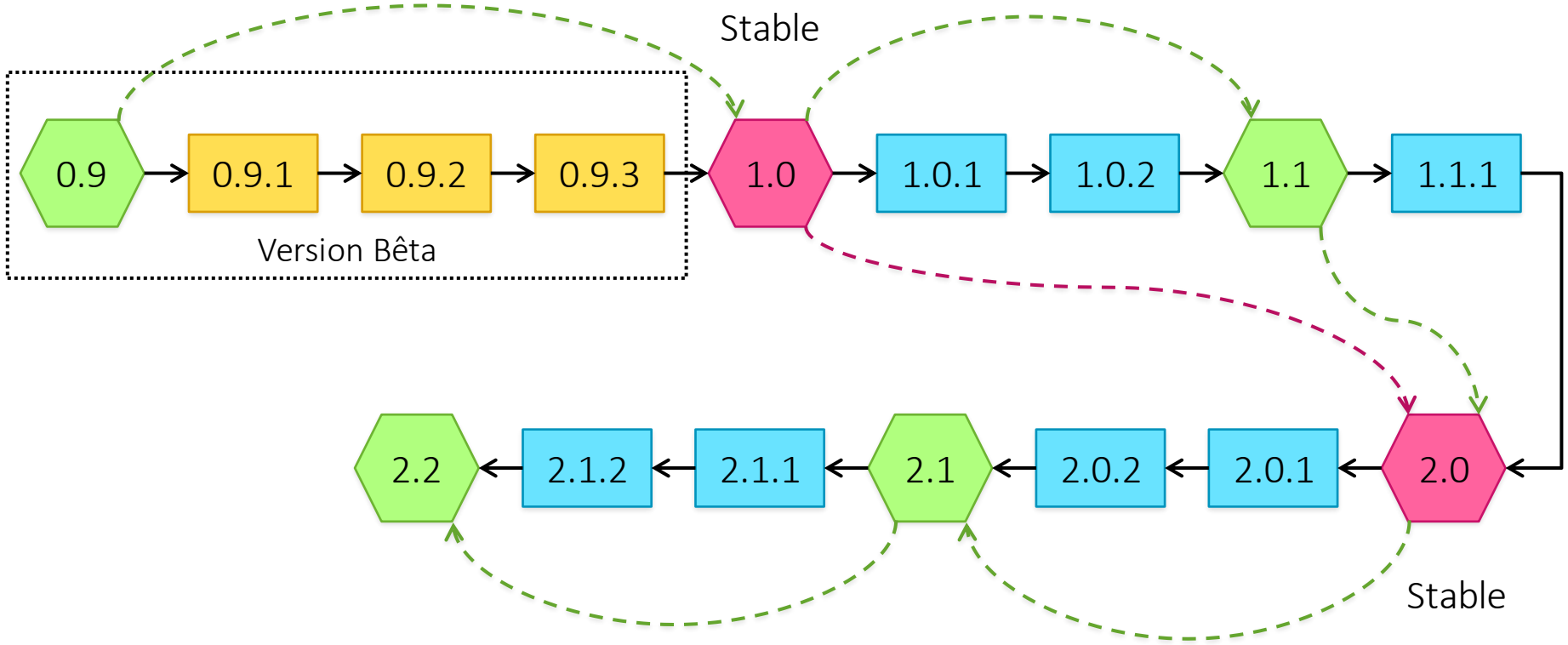
Phases d'un logiciel



- Maquette : aperçu visuel de l'objectif recherché sans réelle fonctionnalité
- Prototype : quelques fonctionnalités et sert de démonstrateur
- Alpha : version interne en phase de tests, manque certaines fonctionnalités et contient des bugs
- Bêta : version soumise à des utilisateurs
- RC : correspond à la version finale du logiciel, mise à disposition pour les qualifieurs
- Stable : version finale du logiciel avec les fonctionnalités prévues et « sans » bug

stephanie.chollet@grenoble-inp.fr

Arbre des versions d'un logiciel



stephanie.chollet@grenoble-inp.fr

Exemples de versions de logiciels – 1/2

- 2021-06, June 16, 2020
- 2021-03, March 17, 2020
- 2020-12, December 16, 2020
- 2020-09, September 16, 2020
- 2020-06, June 17, 2020
- 2020-03, March 18, 2020
- 2019-12, December 18, 2019
- 2019-09, September 19, 2019
- 2019-06, June 19, 2019
- 2019-03, March 20, 2019
- 2018-12, December 19, 2018
- 2018-09, September 19, 2018
- Photon, June 27, 2018
- Oxygen, June 28, 2017
- Neon, June 22, 2016
- Mars, June 24, 2015
- Luna, June 25, 2014
- Kepler, June 26, 2013
- Juno, June 27, 2012
- Indigo, June 22, 2011
- Helios, June 23, 2010
- Galileo, June 24, 2009
- Ganymede, June 25, 2008
- Europa, June 27, 2007
- Callisto, June 26, 2006

Eclipse 2021-06

- R Packages
- RC1 Packages
- M3 Packages
- M2 Packages
- M1 Packages

Eclipse Photon

- R Packages
- RC3 Packages
- RC2 Packages
- RC1 Packages
- M7 Packages
- M6 Packages
- M5 Packages
- M4 Packages
- M3 Packages
- M2 Packages

stephanie.chollet@grenoble-inp.fr

Exemples de versions de logiciels – 2/2

android

stephanie.chollet@grenoble-inp.fr

Gestionnaires de versions

Problématique

- Trace de l'évolution :

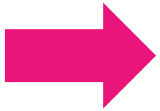
Calculatrice.java

```
public Integer ajouter(Integer operande1, Integer operande2) {
        return operande1 + operande2;
}
```

Passage à une deuxième version

Calculatrice.java

```
public Float ajouter(Float operande1, Float operande2) {
        return operande1 + operande2;
}
```



Quelles modifications ? Pourquoi ? Par qui ?

stephanie.chollet@grenoble-inp.fr

Problématique

- Retour en arrière :
  - Première version :

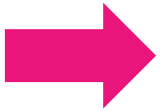
Calculatrice.java

```
public Float ajouter(Float operande1, Float operande2) {
        return operande1 + operande2;
}
```

  - Deuxième version :

Calculatrice.java

```
public Integer ajouter(Integer operande1, Integer operande2) {
        return operande1 + operande2;
}
```



Impossible si on n'a pas gardé une copie du premier fichier

stephanie.chollet@grenoble-inp.fr

Problématique

- Equipe de développement :

```
public Integer ajouter(Integer
operande1, Integer operande2) {
   return operande1 + operande2;
}
```

Calculatrice.java

```
public Integer soustraire(Integer
operande1, Integer operande2) {
   return operande1 - operande2;
}
```

Calculatrice.java

Fusion

```
public Integer ajouter(Integer
operande1, Integer operande2) {
   return operande1 + operande2;
}
public Integer soustraire(Integer
operande1, Integer operande2) {
   return operande1 - operande2;
}
```

Calculatrice.java

stephanie.chollet@grenoble-inp.fr

Gestion des versions

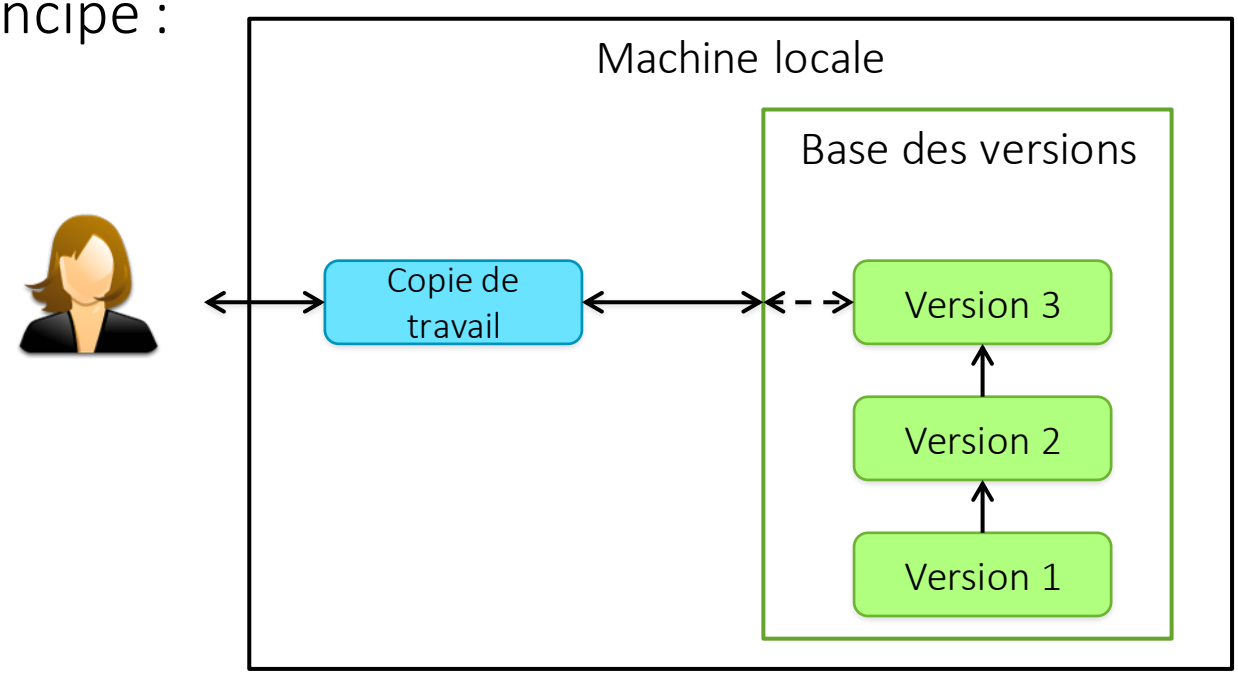
La **gestion de versions** (en anglais *version control* ou *revision control*) consiste à maintenir l'ensemble des versions d'un ou plusieurs fichiers (généralement en texte).

- Une **version** (ou révision) est une étape d'avancement
- Le passage d'une version à une autre se fait par des **modifications** : ajouts, suppressions, mises à jour de données
- Trois types de gestions de versions :
  - Locale, centralisée ou décentralisée

stephanie.chollet@grenoble-inp.fr

1- Gestion de versions locale

- Principe :



- Exemple : *Source Code Control System* (SCCS)

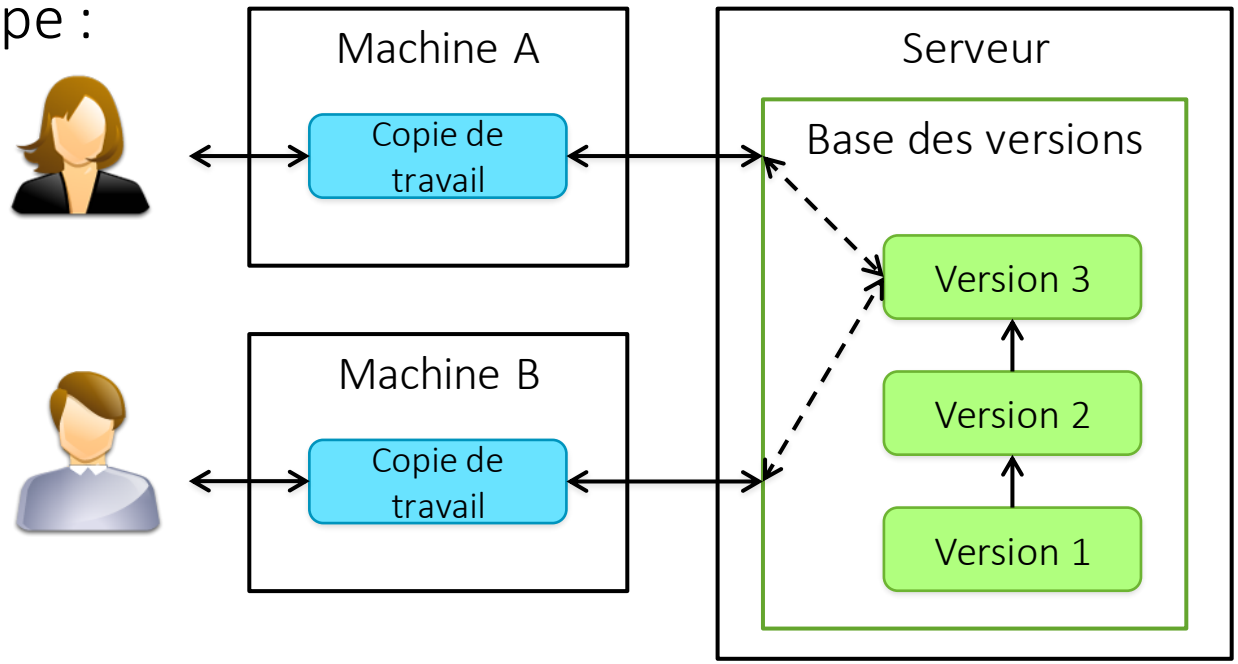
Gestion et utilisation très simples

Sensible aux pannes, pas de collaboration

stephanie.chollet@grenoble-inp.fr

2- Gestion de versions centralisée

- Principe :



- Exemples : CVS et Subversion

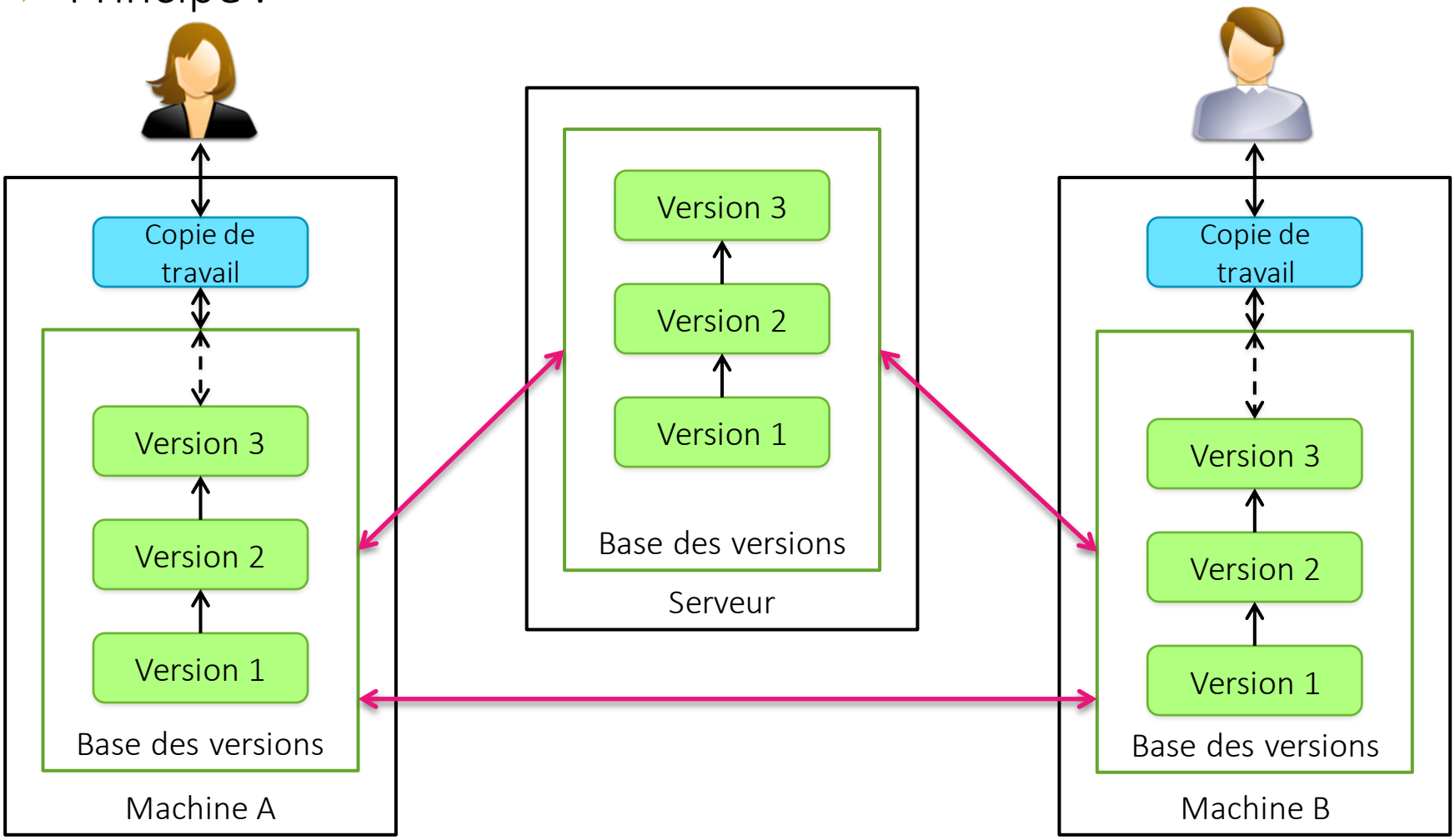
Gestion et utilisation simples

Sensible aux pannes, nécessite une connexion réseau

stephanie.chollet@grenoble-inp.fr

3- Gestion de versions décentralisée – 1/2

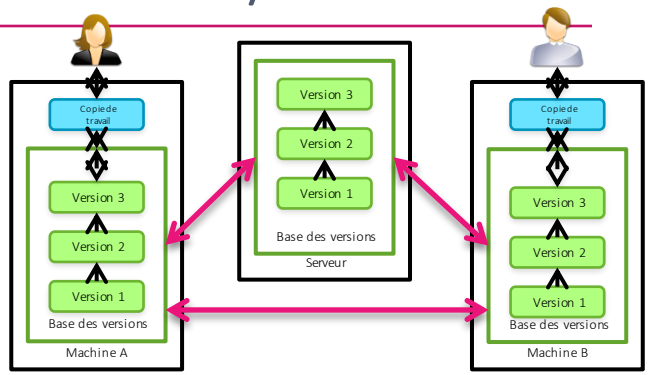
- Principe :



stephanie.chollet@grenoble-inp.fr

3- Gestion de versions décentralisée – 2/2

- Exemples : Git et Mercurial



- Avantages :
  - Moins sensible aux pannes
    - Pas de dépendance à une seule machine
  - Permet de travailler même si on n'est pas connecté au réseau
  - Opérations du contributeur plus rapides car locales
  - Permet le travail privé (brouillon) sans devoir publier ses modifications aux autres contributeurs
  - Permet de garder un dépôt de référence contenant les versions livrées d'un projet

- Inconvénients :
  - Opération longue pour le clonage d'un dépôt (tout l'historique est copié)
  - Pas de système de lock donc il faut gérer les fusions (limites pour les données binaires)

stephanie.chollet@grenoble-inp.fr

Git

Historique (Wikipedia)

- De 1991 à 2002, le noyau Linux était développé sans utilisé de système de gestion de versions
- A partir de 2002, la communauté a commencé à utiliser BitKeeper, un système de gestion de versions propriétaire

- En 2005, suite à un contentieux, BitKeeper retire la possibilité d'utiliser gratuitement son produit. Linus Torvalds lance le développement de Git et après seulement quelques mois de développement, Git héberge le développement du noyau Linux
- Le magazine *PC World* nous apprend que « quand on lui a demandé pourquoi il avait appelé son logiciel "*git*", qui est à peu près l'équivalent de "connard" en argot britannique[8,9], Linus Torvalds a répondu "je ne suis qu'un sale égocentrique, donc j'appelle tous mes projets d'après ma propre personne. D'abord Linux, puis Git."[10] ».

stephanie.chollet@grenoble-inp.fr

Git

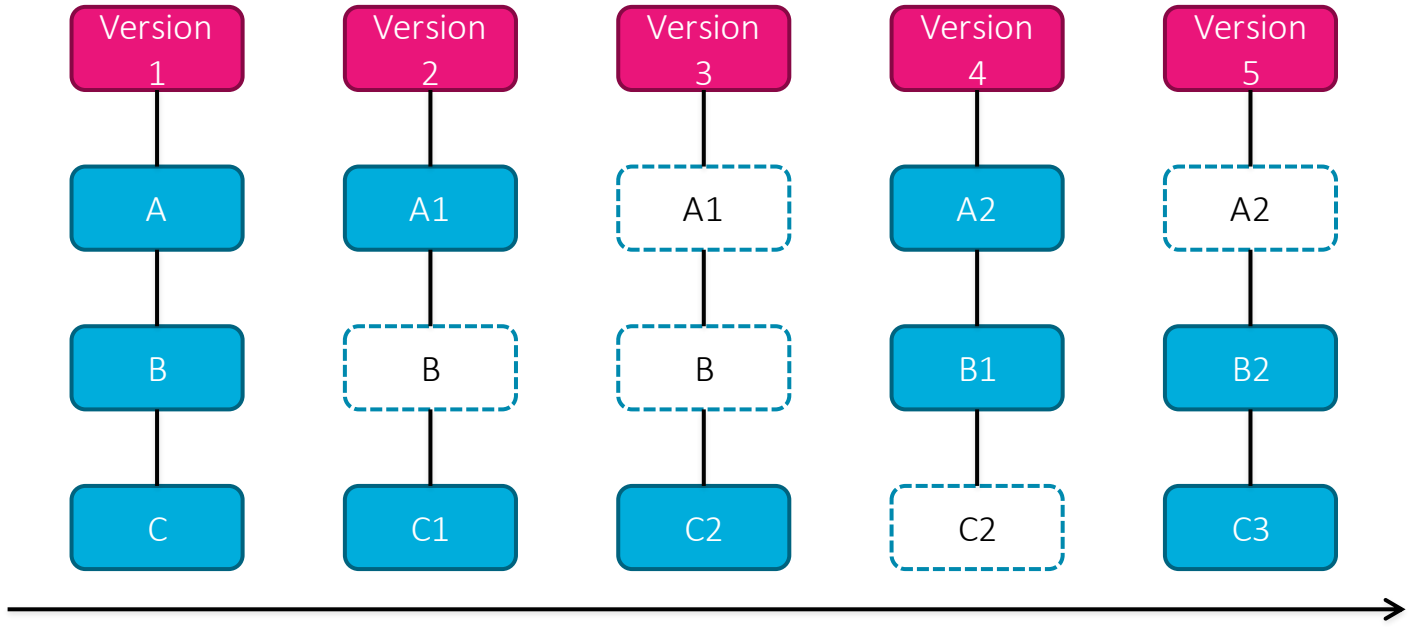
Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre et distribué selon les termes de la licence publique générale GNU version 2.
Il est écrit en C, Shell Unix, Perl, Tcl et GNU Bash.

- https://git-scm.com/

stephanie.chollet@grenoble-inp.fr

Principes de base

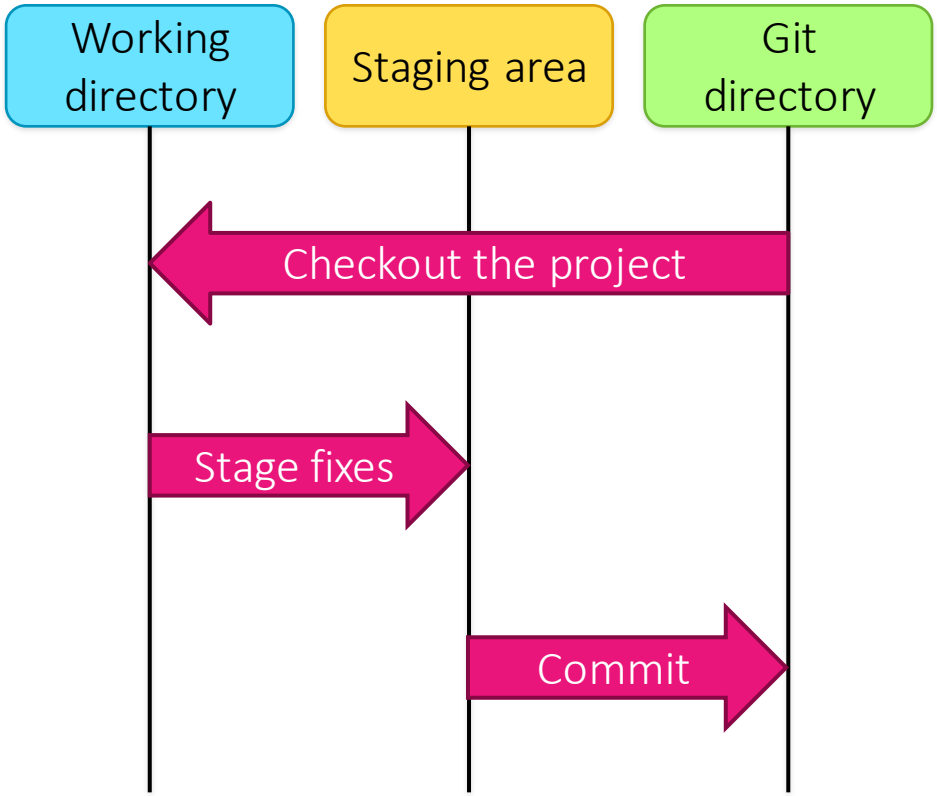
Un dépôt Git est une sorte de système de fichiers (base de données), enregistrant les versions de fichiers d'un projet à des moments précis au cours du temps sous forme d'instantanés.



stephanie.chollet@grenoble-inp.fr

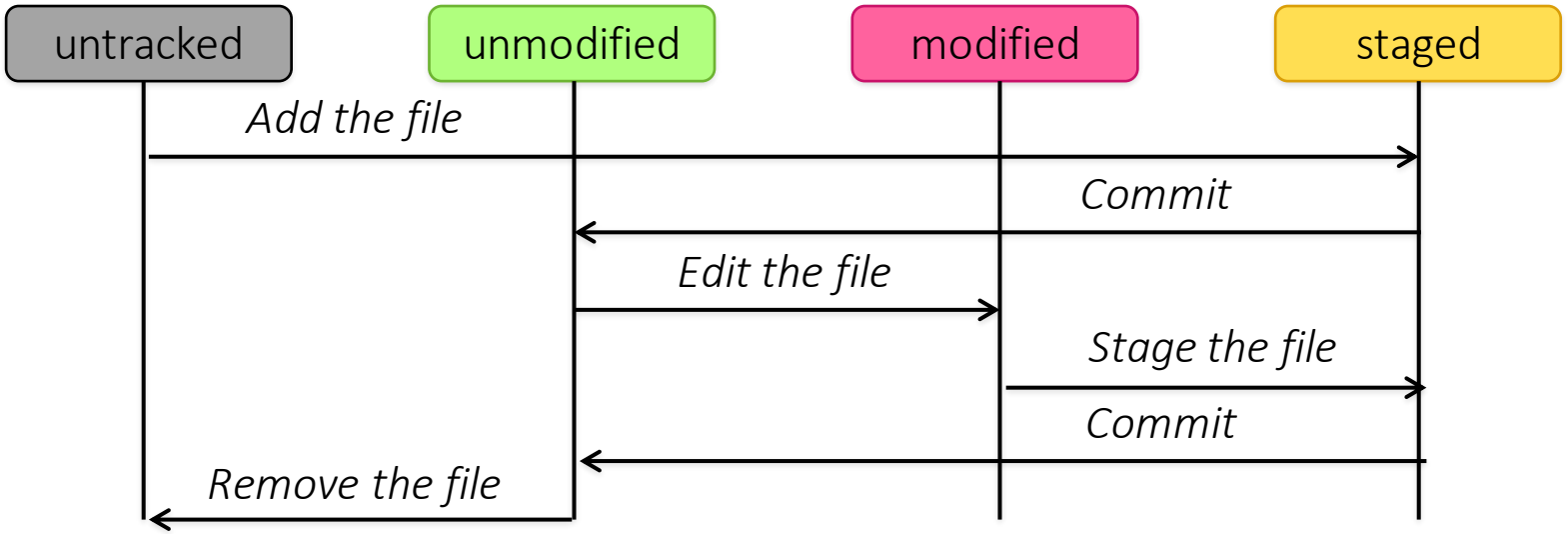
Projet versionné localement avec Git

- Le **répertoire de travail** courant :
  - Extraction unique d'une version du projet depuis la base de données du dépôt
- Le **répertoire Git/dépôt (.git)** :
  - Contient les méta-données et la base de données des objets du projet
- La **zone de transit/d'index** :
  - Fichier contenant des informations à propos de ce qui sera pris en compte lors de la prochaine soumission



stephanie.chollet@grenoble-inp.fr

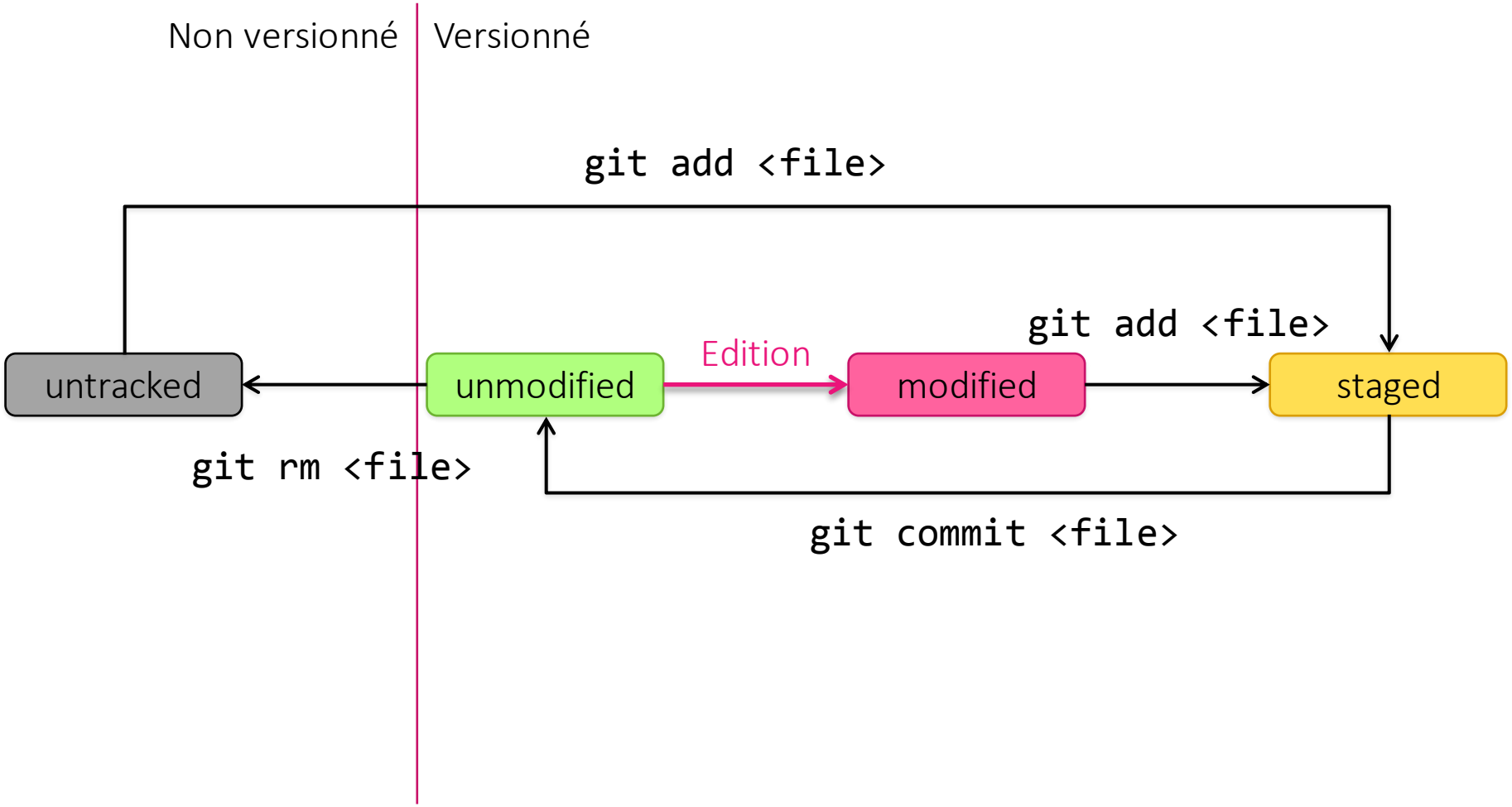
Etats d'un fichier



- **Non versionné** :
  - Fichier n'étant pas ou plus géré par Git
- **Non modifié** :
  - Fichier sauvegardé de manière sûre dans sa version courante dans la base de données du dépôt
- **Modifié** :
  - Fichier ayant subi des modifications depuis la dernière soumission
- **Indexé (*staged*)** :
  - Idem, sauf qu'il en sera pris un instantané dans sa version courante lors de la prochaine soumission (commit)

stephanie.chollet@grenoble-inp.fr

Avec les commandes git



stephanie.chollet@grenoble-inp.fr

Commandes de base – 1/3

- Initialiser un dépôt :
  - `git init`

- Afficher l'état des fichiers du répertoire courant :
  - `git status`
    - *Untracked files* : fichiers non versionnés
    - *Changes to be committed* : modifications (ajout, suppression, changement) chargées en zone de transit (staging area) ou indexées
    - *Changes not staged for commit* : modifications n'ayant pas été chargées en zone de transit (ou indexées)

stephanie.chollet@grenoble-inp.fr

Commandes de base – 2/3

- Indexer l'ajout ou les changements d'un fichier
  - `git add <fichier>`
- Annuler les modifications indexées d'un fichier
  - `git reset <fichier>`
- Annuler les modifications non encore indexées d'un fichier
  - `git checkout [--] <fichier>`
- Indexer la suppression d'un fichier
  - `git rm <fichier>`
- Déversionner un fichier
  - `git rm –cached <fichier>`

stephanie.chollet@grenoble-inp.fr

Commandes de base – 3/3

- Afficher le détail des modifications non indexées
  - `git diff`
- Afficher le détail des modifications indexées
  - `git diff --staged`
- Soumettre les modifications indexées en zone de transit
  - `git commit`
- Voir l'historique des soumissions
  - `git log`

stephanie.chollet@grenoble-inp.fr

Git

Les branches

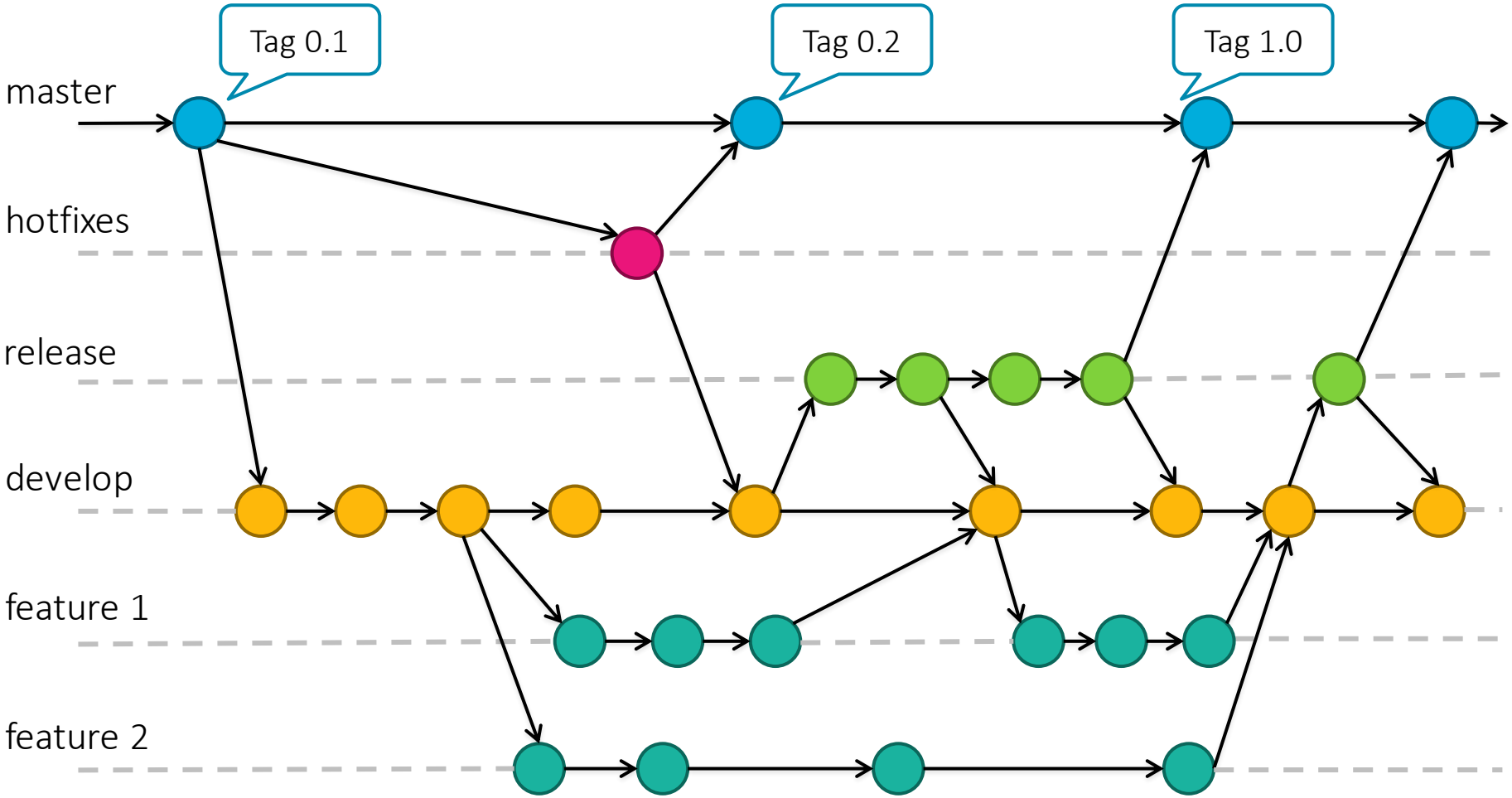
Branche d'un projet

Une **branche d'un projet** est une ligne d'évolution divergeant de la ligne d'évolution courante, celles-ci se poursuivant **indépendamment** l'une de l'autre.

- Intérêt :
  - Possibilité d'avoir des évolutions de versions en parallèle
- Deux stratégies :
  - **Une branche par produit** (*Branch by Product*) : un client veut une modification qui lui est spécifique (contrainte externe)
  - **Une branche par objectif** (*Branch by Purpose*) : nécessité de continuer de développer tout en permettant la correction de bugs sur une branche en cours de tests (contrainte interne)

stephanie.chollet@grenoble-inp.fr

Exemple de git flow – 1/2



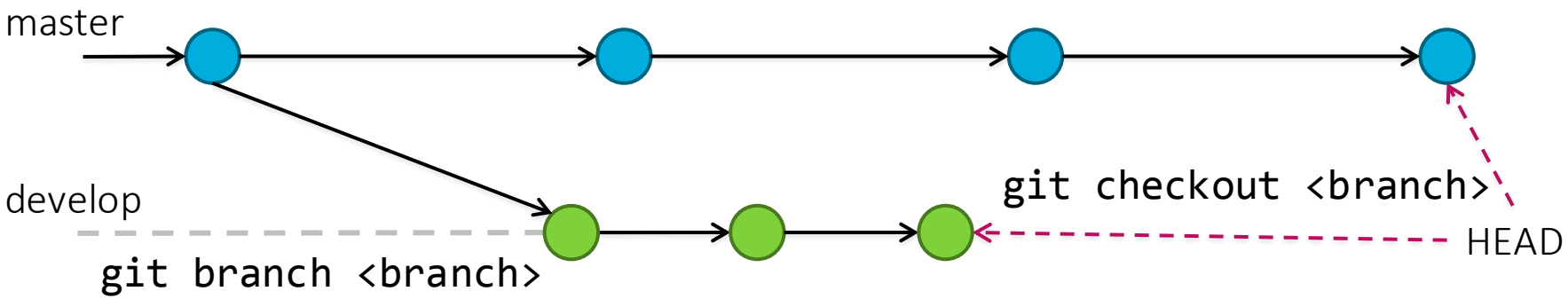
stephanie.chollet@grenoble-inp.fr

Exemple de git flow – 2/2

- Idée :
  - Une branche develop pour le développement
  - Une branche release pour préparer les livraisons
  - Une branche master qui ne contient que des commits « stables »
  - Autant de branches hotfix que de bugs sur ce qui a été livré
  - Autant de branches (éphémères) que de features/développeurs

Branche avec Git

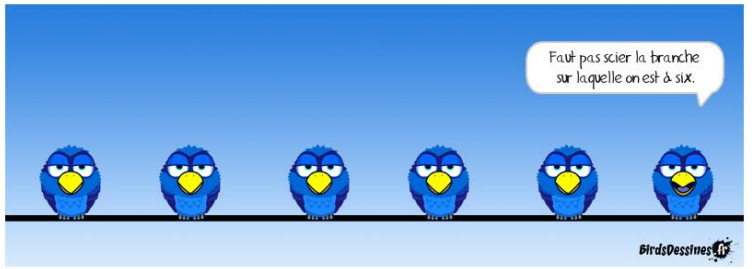
- Une branche est un pointeur vers un objet commit
- Par défaut, il existe une seule branche nommée « master »
- HEAD est un pointeur spécial vers la branche sur laquelle on travaille



stephanie.chollet@grenoble-inp.fr

Commandes pour les branches

- Créer une nouvelle branche :
  - `git branch <branche>`
- Voir les branches du dépôt local :
  - `git branch`
- Supprimer une branche :
  - `git branch -d <branche>`

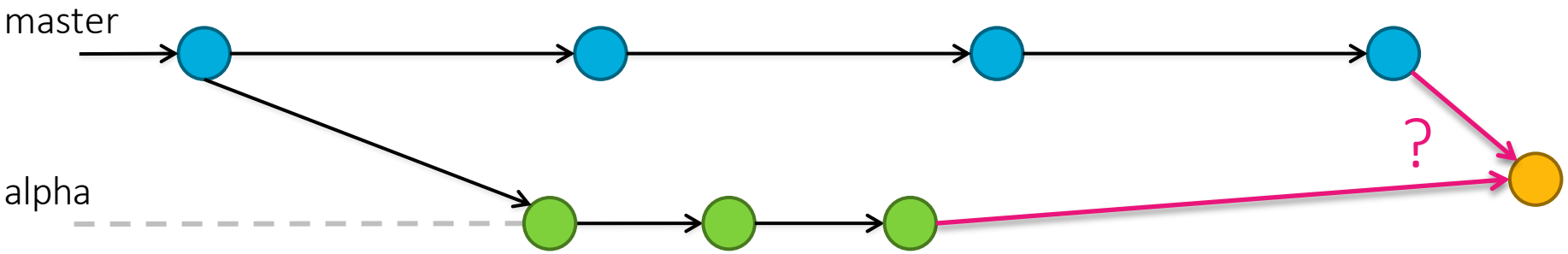


- Passer à une branche donnée (mise à jour de l'index et du répertoire de travail, ainsi que du pointeur HEAD) :
  - `git checkout <branche>`

stephanie.chollet@grenoble-inp.fr

Problématique : incorporer des modifications

- Comment incorporer des modifications venant d'une autre branche ?

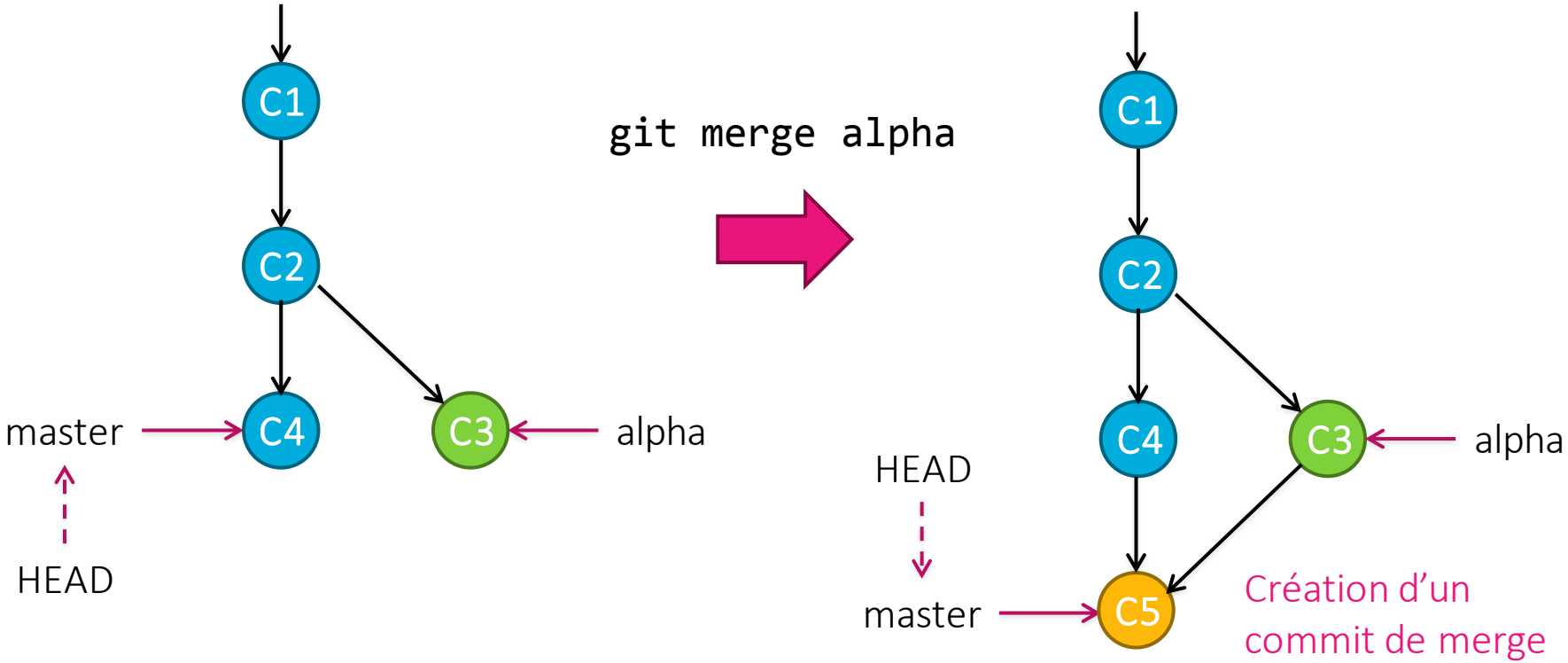


- Deux techniques :
  - Fusionner (merge)
  - Rebaser (rebase)

stephanie.chollet@grenoble-inp.fr

Fusionner deux branches : `git merge <branche>`

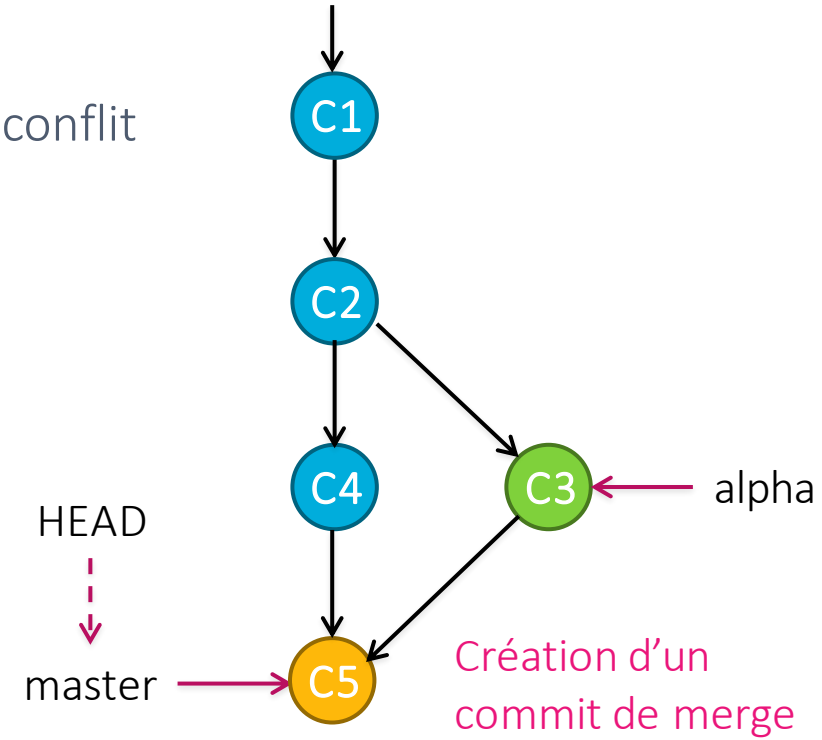
- Fusionner deux branches :
  - Rapatrier les modifications d'une branche dans une autre
  - Par défaut, la fusion concerne tous les *commits* depuis la dernière fusion ou la création de la branche



stephanie.chollet@grenoble-inp.fr

Fusionner deux branches : `git merge <branche>`

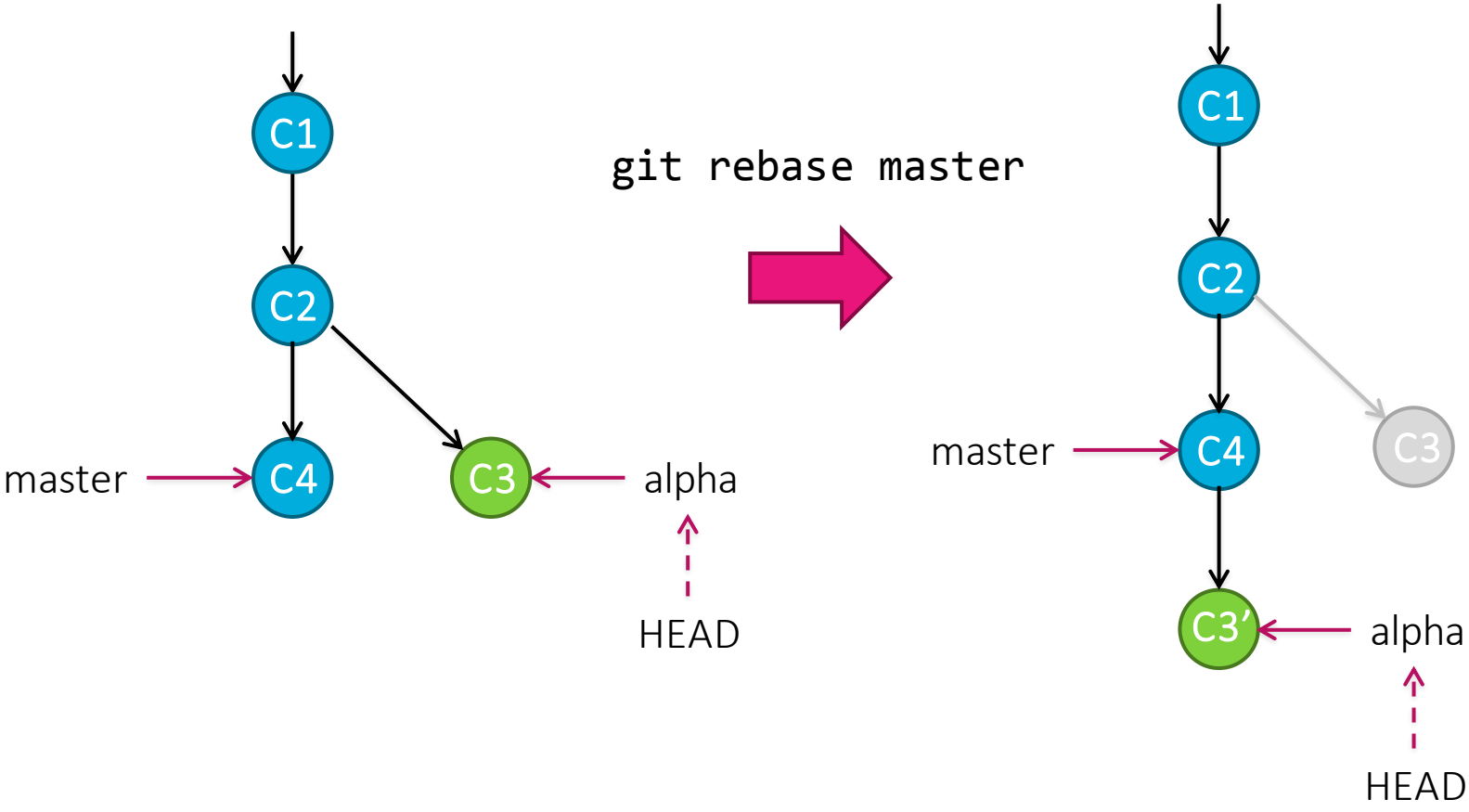
- En principe : c'est facile
- En réalité : gestion de conflits
- Exemple : si C3 et C4 modifient la même ligne d'un fichier
  - Git ne sait pas quoi choisir
  - Il faut manuellement résoudre le conflit



stephanie.chollet@grenoble-inp.fr

Rebaser deux branches : `git rebase <branche>`

- Modifie et/ou réécrit l'historique des commits



stephanie.chollet@grenoble-inp.fr

Merge vs. Rebase

- Merge :
  - Pour la réintégration de branches de feature
  - On veut garder l'historique des commits indépendants sans polluer l'historique de la branche principale

- Rebase :
  - Pour la mise à jour de branches comme la correction d'anomalies
  - Rend votre arbre de commits très propre puisqu'il ressemble à une ligne droite mais il modifie l'historique (apparent) de l'arbre des commits.

  - Ne jamais rebaser des commits qui ont déjà été poussés sur un dépôt public

stephanie.chollet@grenoble-inp.fr

Revenir en arrière

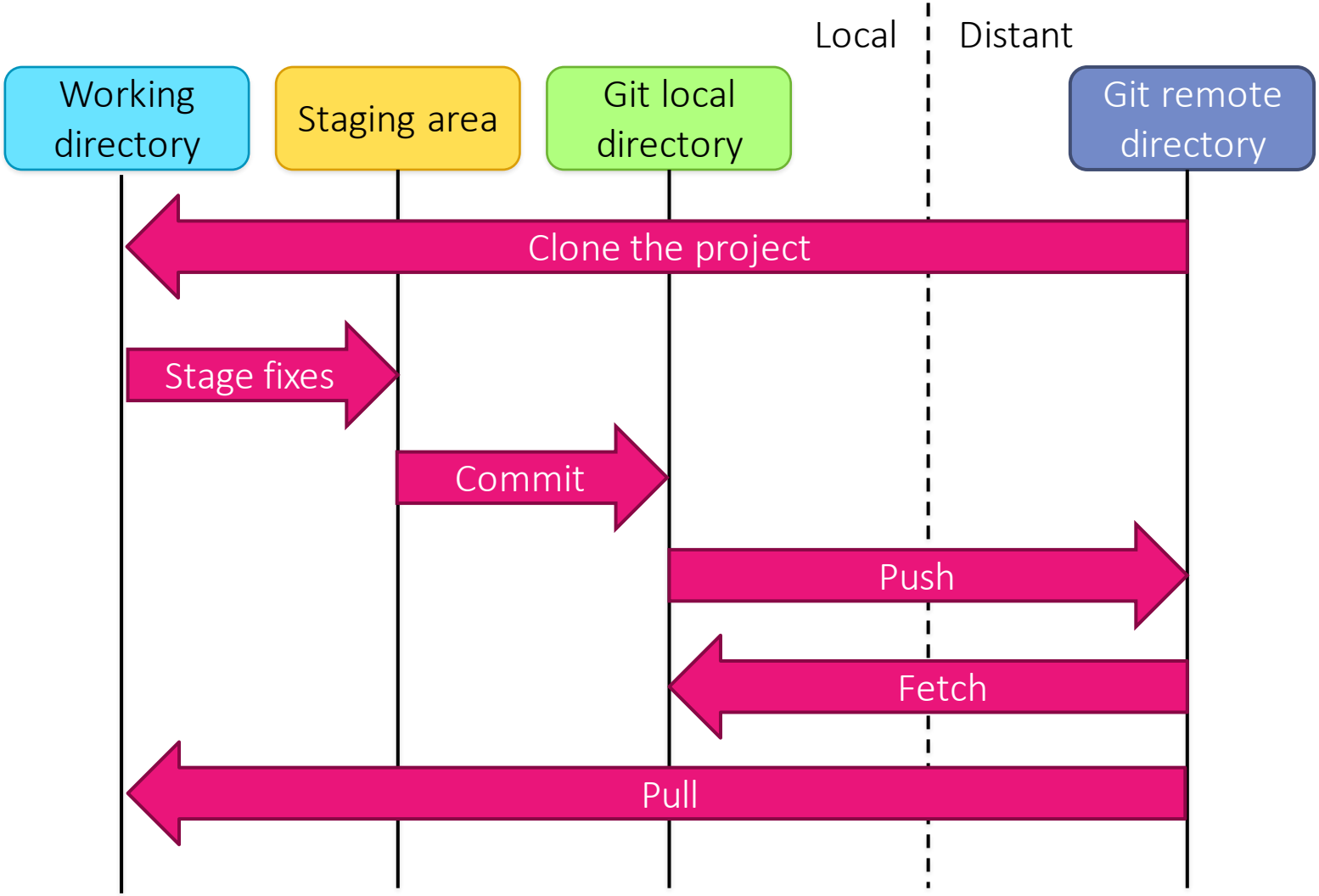
- Cas de modifications commitées :
  - amend : modifier le dernier commit:
    - Ajout de fichiers au commit
    - Changement de message du commit
  - revert : annuler un commit par un autre commit
  - reset : rétablir la situation d'un ancien commit

stephanie.chollet@grenoble-inp.fr

Git

Dépôts distants

Dépôt distant



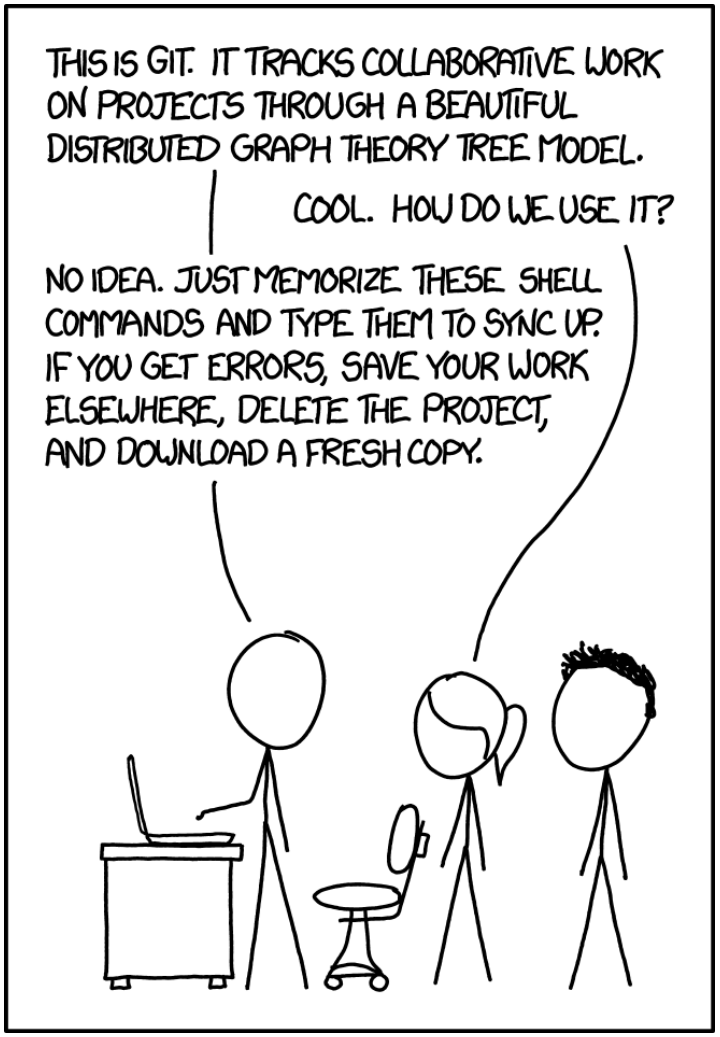
stephanie.chollet@grenoble-inp.fr

Commandes pour les dépôts distants

- Cloner un dépôt distant dans un nouveau répertoire :
  - `git clone <url_dépôt> <répertoire>`
- Récupérer/télécharger les branches ou les commits d'un autre dépôt
  - `git fetch <dépôt>`
- Rapatrier et intégrer un autre dépôt ou une branche locale
  - `git pull <dépôt>`
- Mettre à jour les références distantes ainsi que les objets associés
  - `git push <dépôt>`

stephanie.chollet@grenoble-inp.fr

Gestion des conflits – Bonne pratique

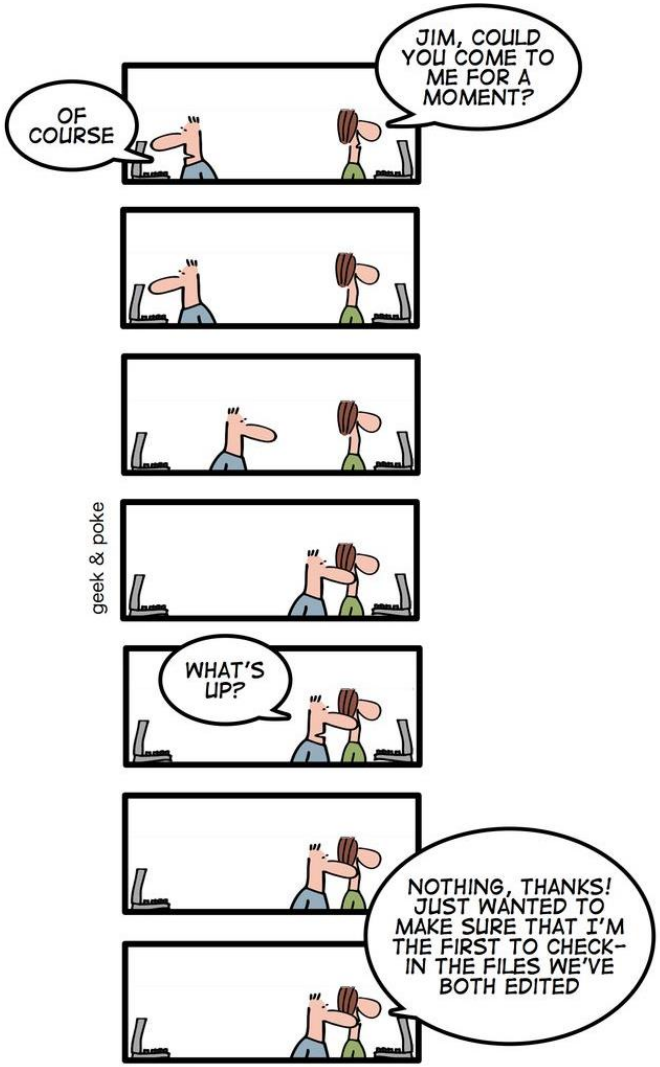


1. Je fais des modifications
2. Je peux faire *git fetch* pour comparer mais pas git pull
3. J'indexe mes modifications (*git add/git commit*)
4. Je fais *git pull*

5. J'ai un conflit
6. J'édite pour résoudre
7. J'indexe (*add/commit*) ce qui a été résolu
8. Je peux faire *git push* (on n'oublie pas le pull)

stephanie.chollet@grenoble-inp.fr

Gestion des conflits – Mauvaise pratique



https://geekandpoke.typepad.com/geekandpoke/2010/10/being-a-code-made-easy-chapter-1.html

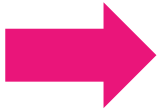
stephanie.chollet@grenoble-inp.fr

Services d'hébergement de dépôts distants

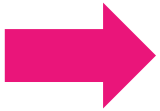
- Github :
  - https://github.com/

- GitLab :
  - https://about.gitlab.com/

- Bitbucket :
  - https://bitbucket.org/



Lequel choisir ?



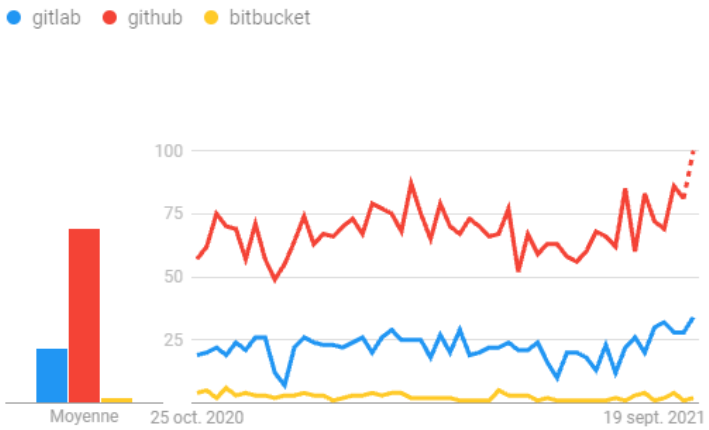
Dépôt GitLab hébergé à l'Ensimag :
https://gitlab.ensimag.fr/users/sign_in

stephanie.chollet@grenoble-inp.fr

Critères

- En version Cloud :

		Github	GitLab	Bitbucket
Open source	Jusqu'à 5 utilisateurs	0$	0$	0$
	Supérieur à 5 utilisateurs	0$	0$	2$/utilisateur/Mo
Répertoire privé	Jusqu'à 5 utilisateurs	25$/Mo	0$	0$
	Supérieur à 5 utilisateurs	$9/utilisateur/Mo	0$	2$/utilisateur/Mo



Facilité d'intégration avec les outils d'intégration continue (CI/CD) ?

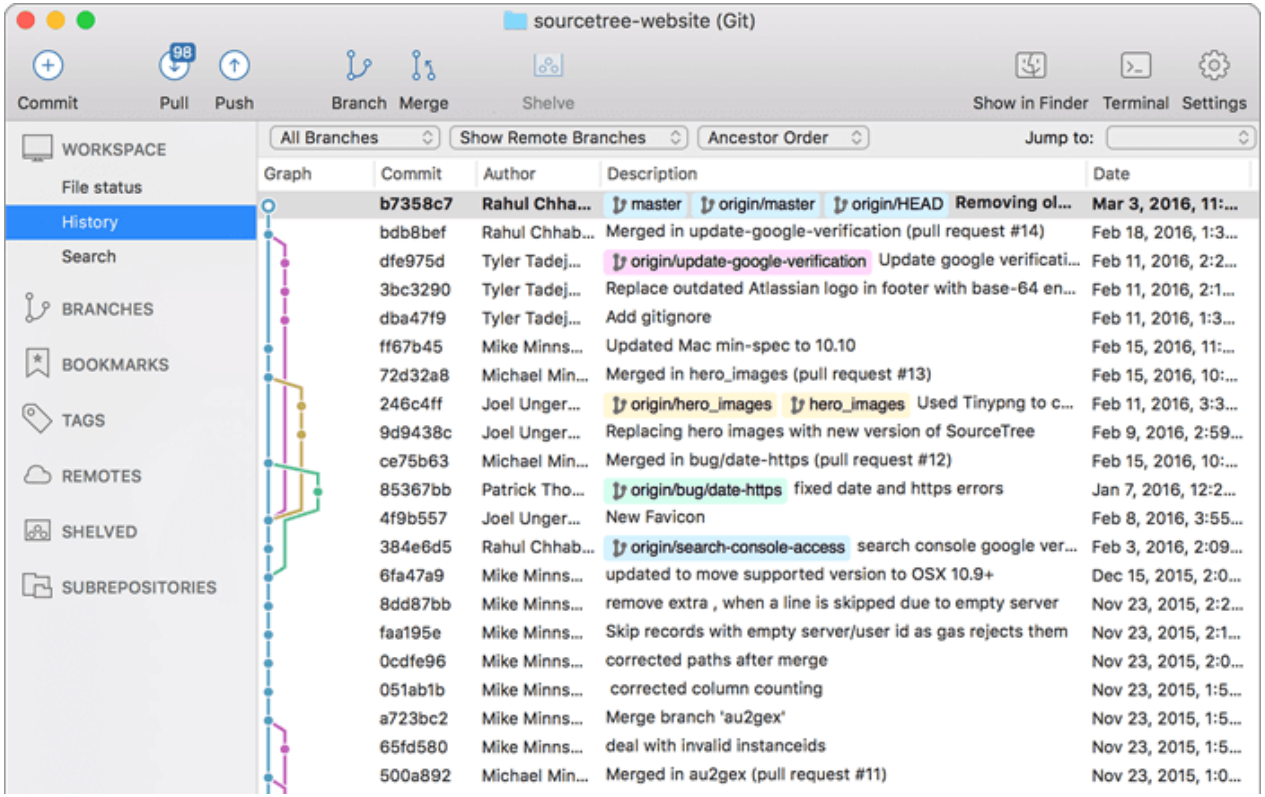
stephanie.chollet@grenoble-inp.fr

Git

Les outils

GUI Clients

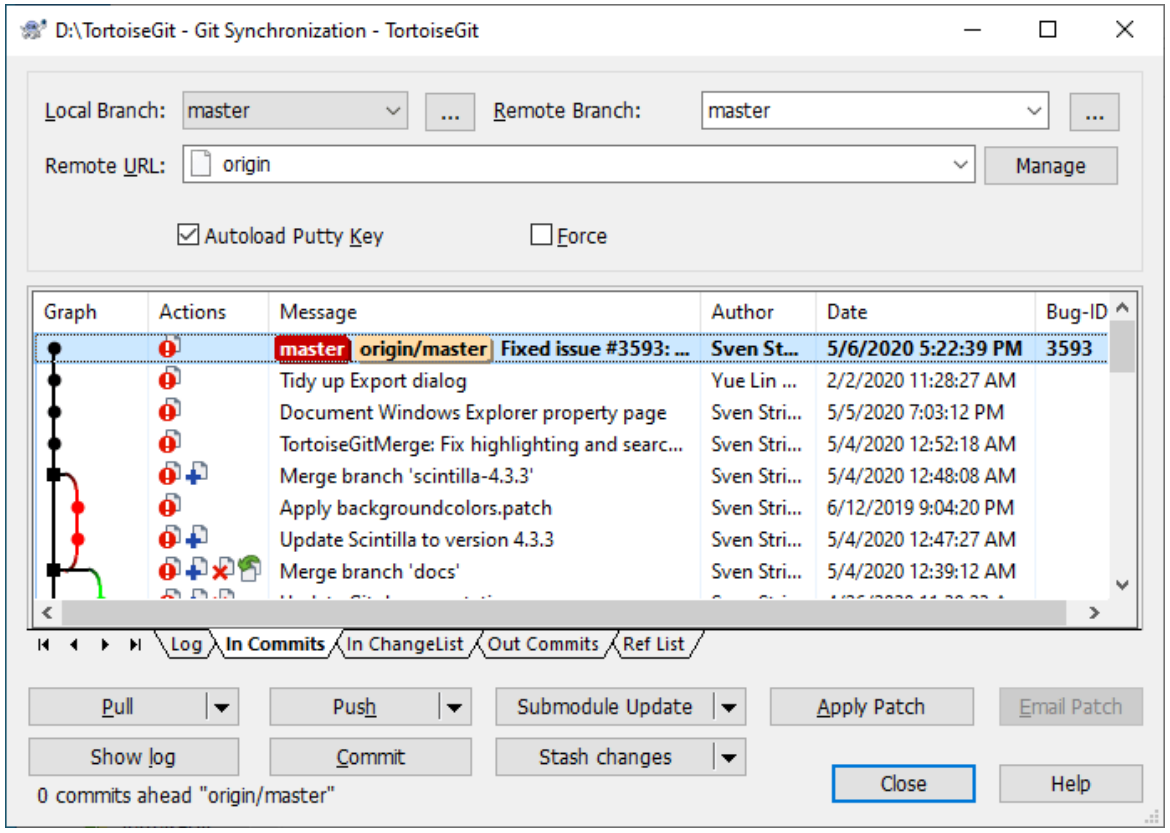
- Sourcetree :
  - https://www.sourcetreeapp.com/



stephanie.chollet@grenoble-inp.fr

GUI Clients

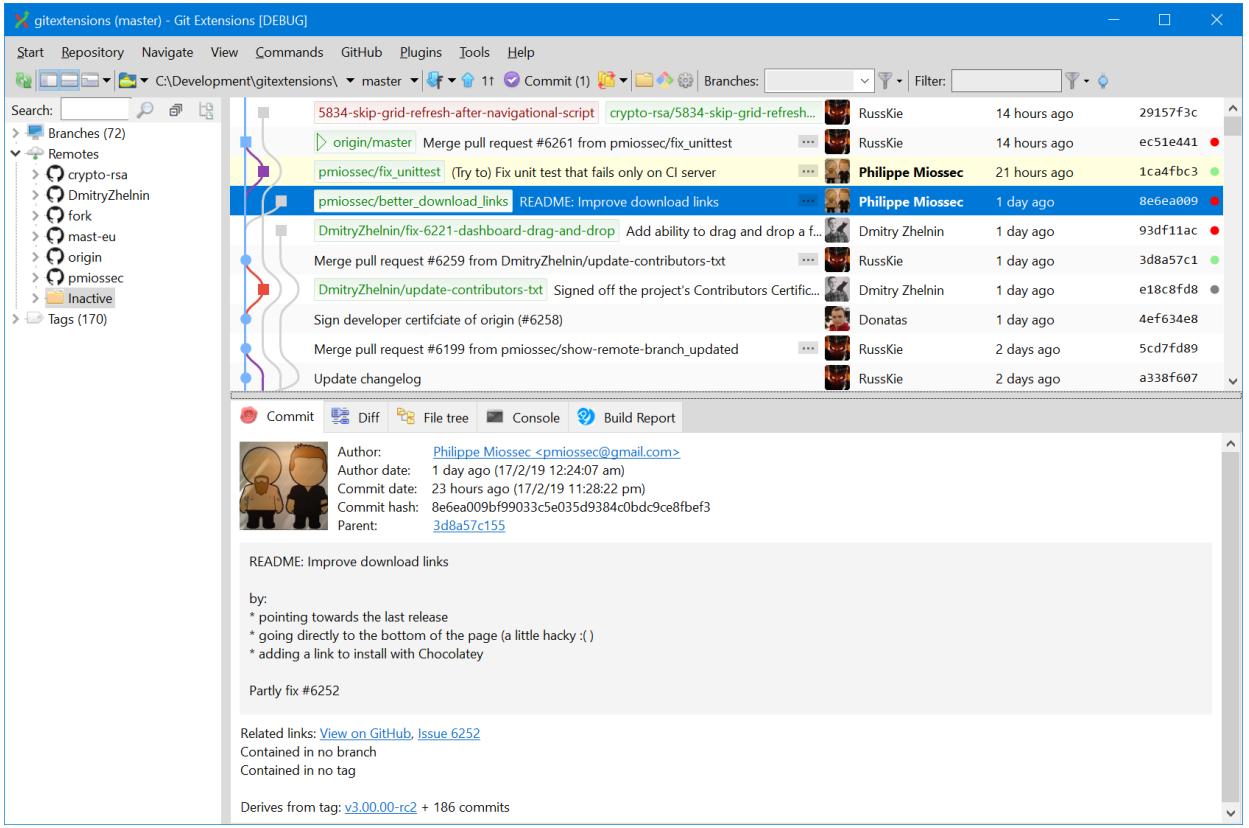
- TortoiseGit :
  - https://tortoisegit.org/



stephanie.chollet@grenoble-inp.fr

GUI Clients

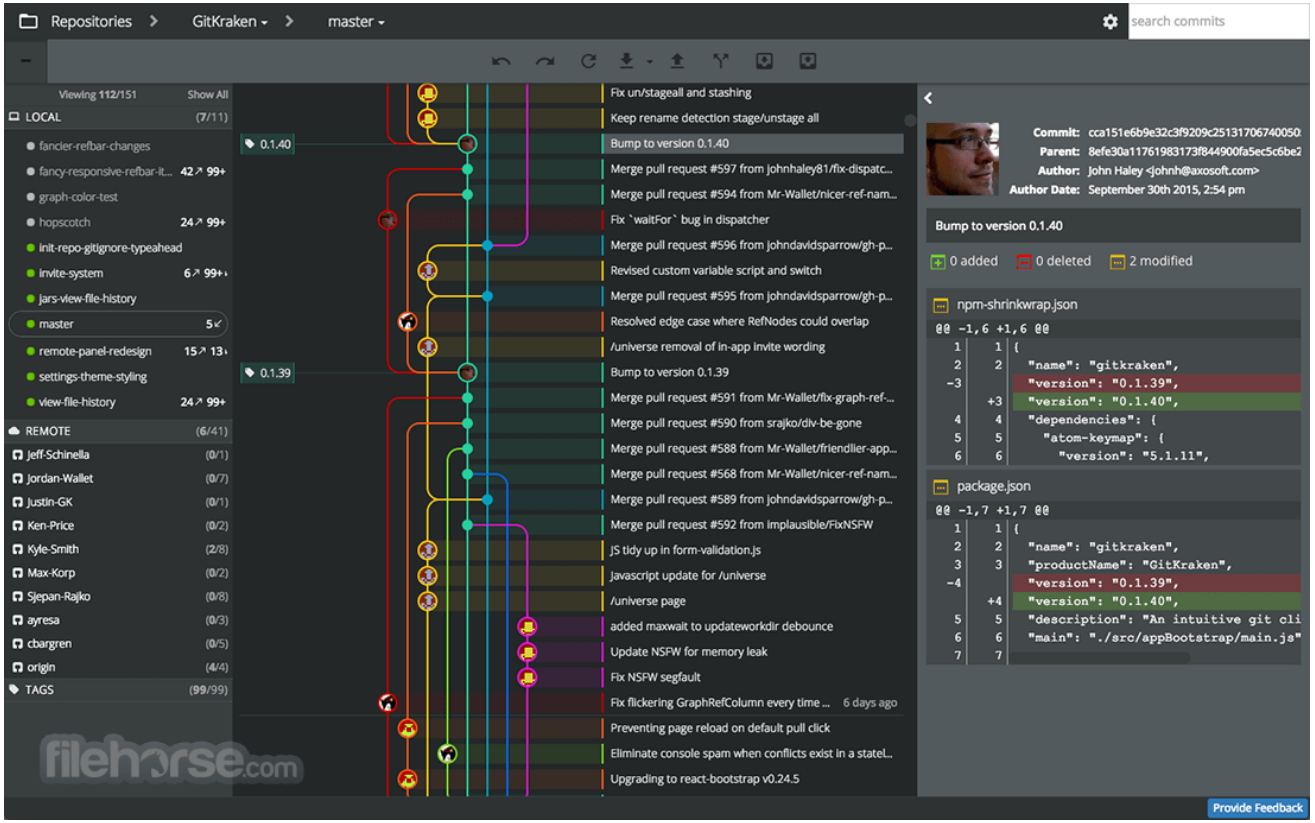
- Git Extensions :
  - https://gitextensions.github.io/



stephanie.chollet@grenoble-inp.fr

GUI Clients

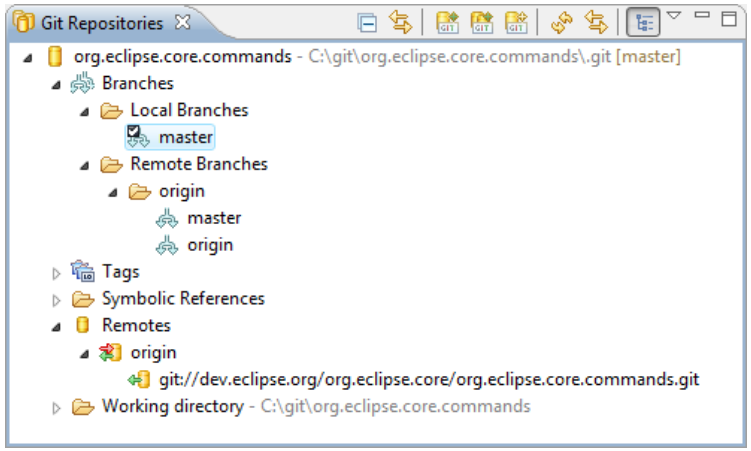
- GitKraken :
  - https://www.gitkraken.com/

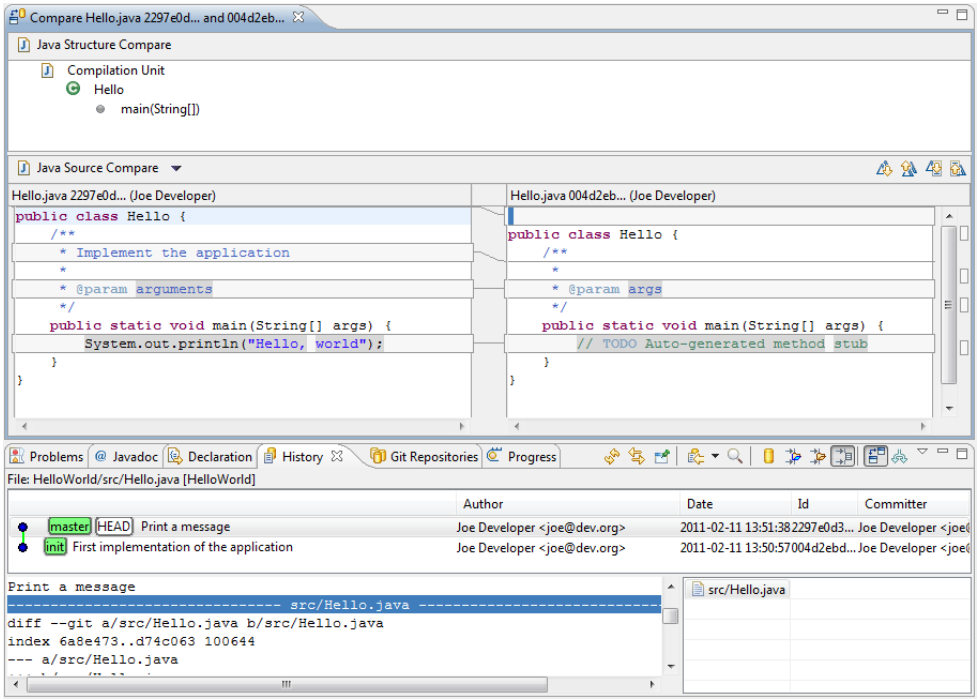


stephanie.chollet@grenoble-inp.fr

Dans Eclipse

- Plugin EGit :
  - https://www.eclipse.org/egit/
  - Installé par défaut dans Eclipse
  - Documentation : https://wiki.eclipse.org/EGit/User_Guide

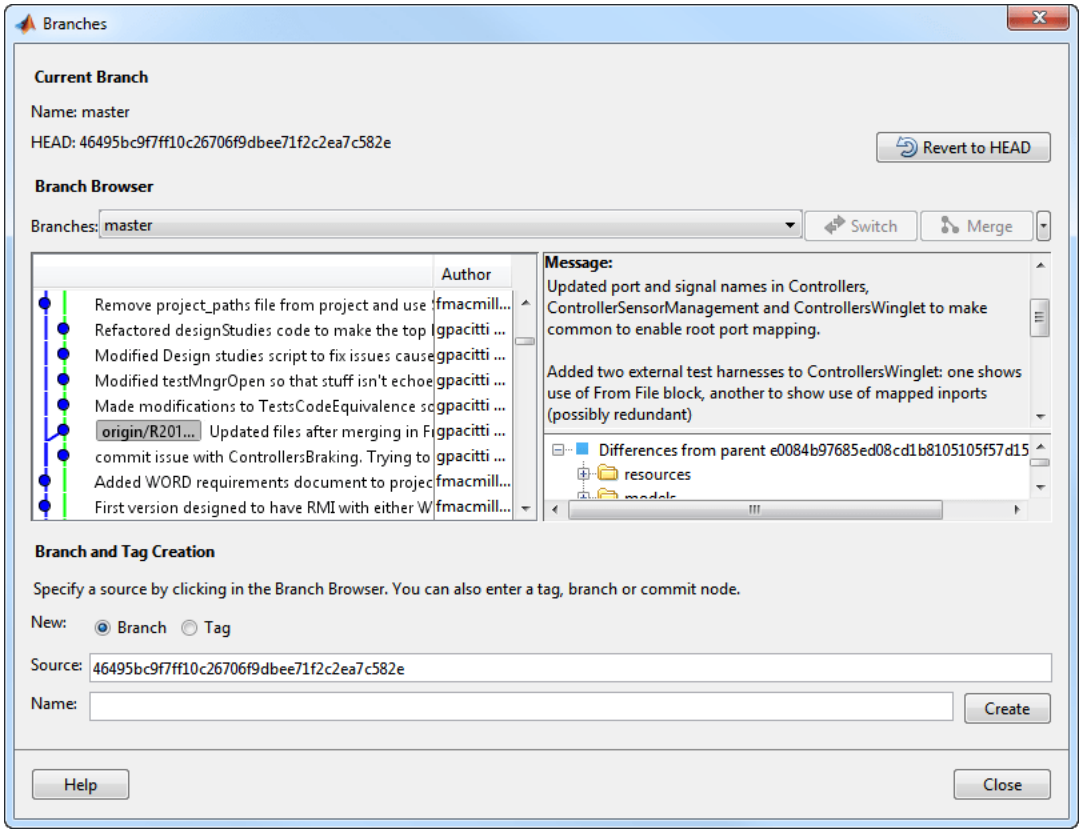




stephanie.chollet@grenoble-inp.fr

Dans Matlab

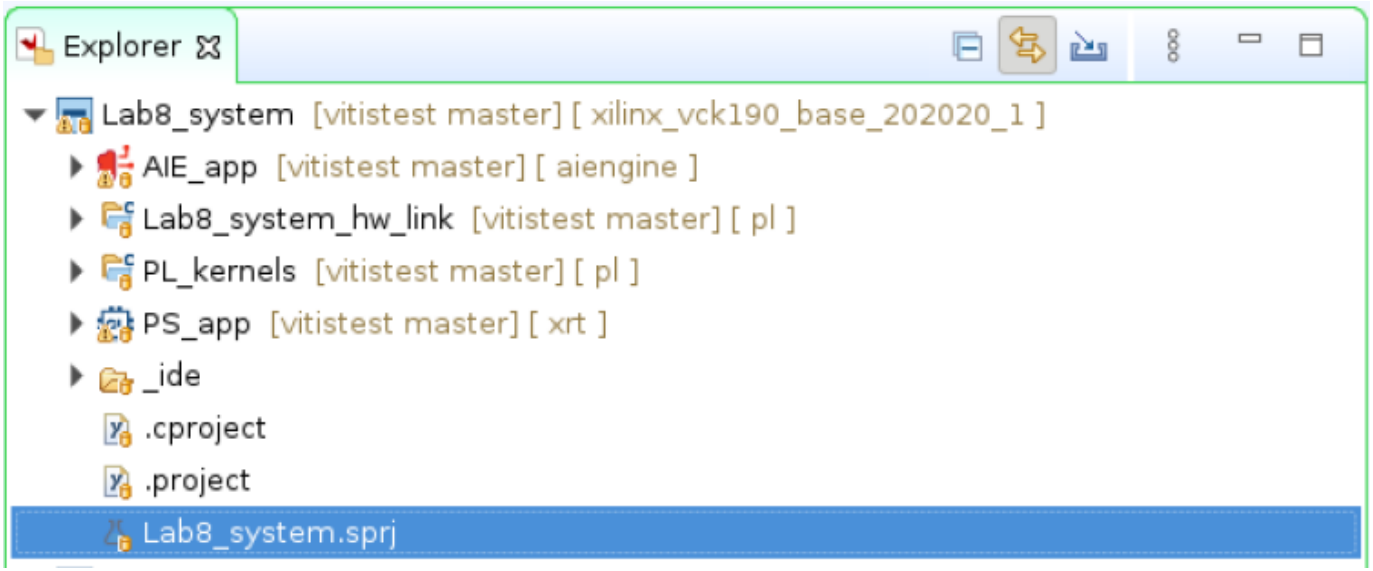
- Documentation :
- https://fr.mathworks.com/help/matlab/source-control.html?s_tid=CRUX_lftnav



stephanie.chollet@grenoble-inp.fr

Dans Xilinx

- Documentation : https://www.xilinx.com/html_docs/xilinx2021_1/vitis_doc/versioncontrol.html



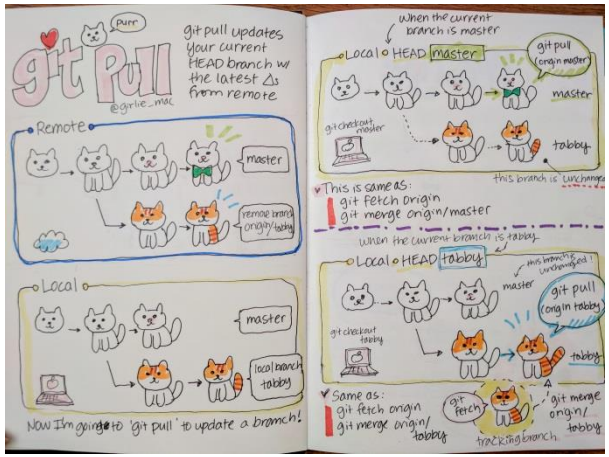
stephanie.chollet@grenoble-inp.fr

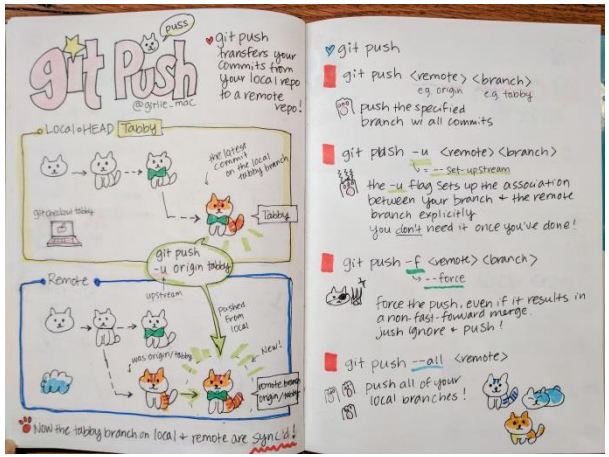
Git

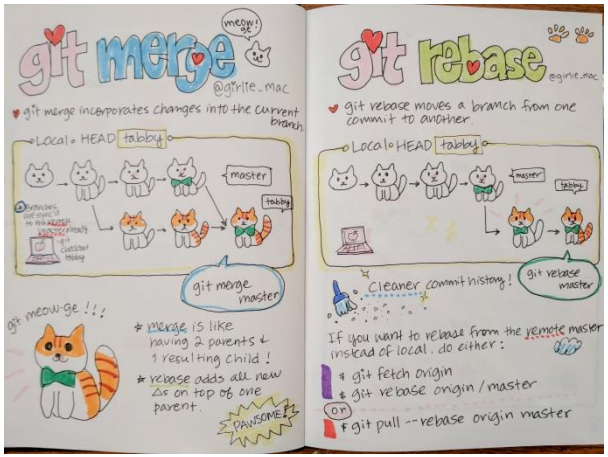
Conclusion

Conclusion

- Documentation officielle :
  - https://git-scm.com/
- D'autres documentations :
  - https://training.github.com/downloads/fr/github-git-cheat-sheet.pdf?source=korben.info
  - http://rogerdudler.github.io/git-guide/index.fr.html
  - https://girliemac.com/blog/2017/12/26/git-purr/



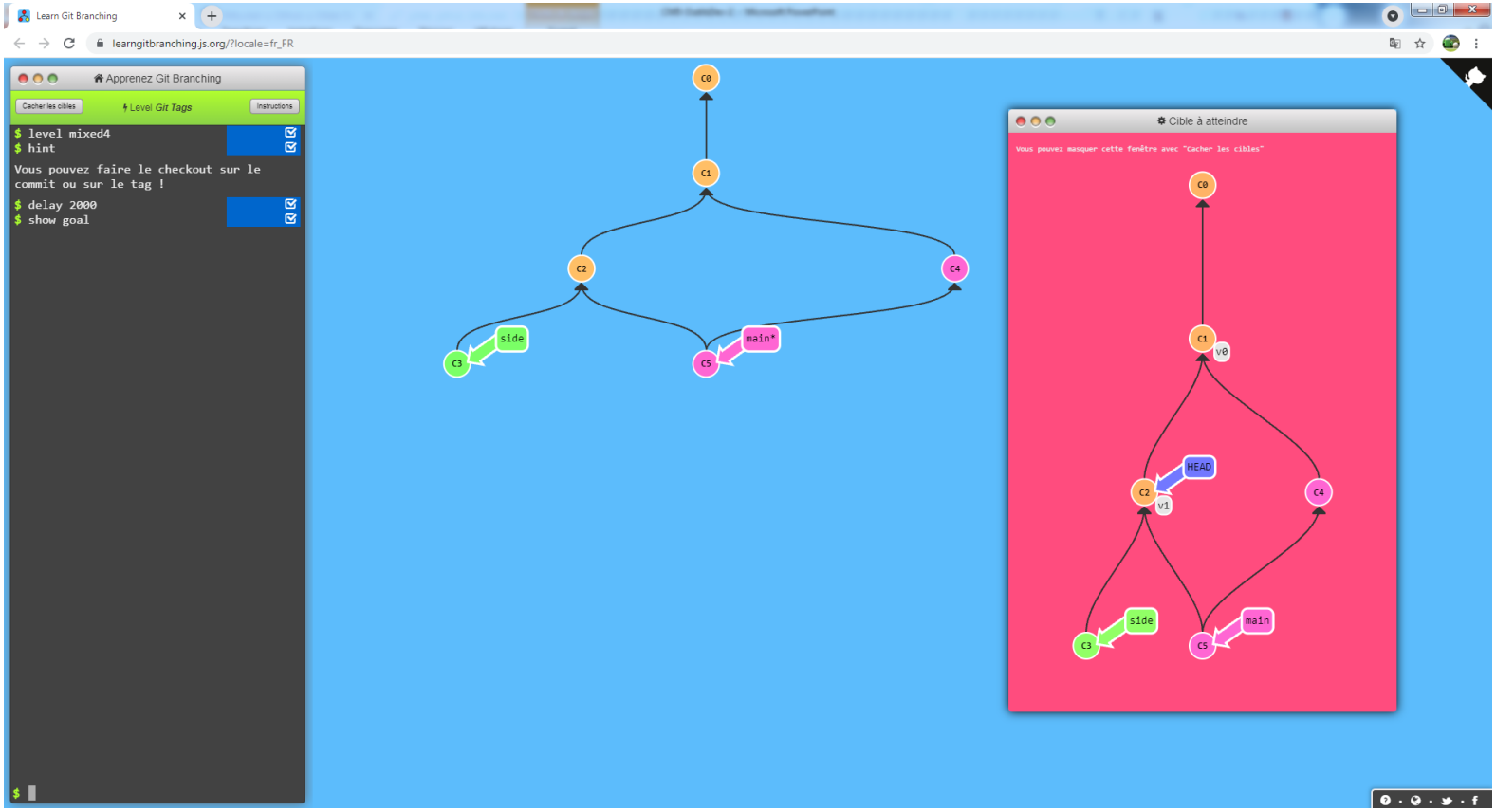




stephanie.chollet@grenoble-inp.fr

A faire

- Pas de TD sur Git mais à faire en travail personnel :
  - https://learngitbranching.js.org/?locale=fr_FR



stephanie.chollet@grenoble-inp.fr