Introduction à Git

Thibault Clérice

Octobre X, 2017

0. Contenus du cours

- 4 cours de 2h
  - Git, commandes de base
  - Github : Collaborer et faire de l'open-source/data
  - Github et Travis : Organisation de projet, tests et intégration continue
  - Point d'étape projet
- 1 conférence 1.5h par Bridget Almas (Perseids & Alpheios)

0. Devoir

- Travail de groupe : 3/4 personnes maximum
- Tâche : Transformation et gestions de texte et de leurs métadonnées parmi une sélection de textes
  - Corpus latins déjà en XML
  - Corpus francais
  - Corpus ENC
  - Possibilité de faire des propositions
- Partage des rôles : Rôles clairs et prédifinis dès le départ
  - Gestionnaire de métadonnées
  - Transformation des textes
  - Gestionnaire de projet (peut-être partagé) : ouverture des billets de tâches, etc.
  - Tenue d'un journal des tâches effectuées
- Notation :
  - A partir des archives GIT
  - A partir du travail effectué et du sérieux dans le travail en équipe
  - Notes différentes suivant les personnes
  - Temps de travail estimé : 20 à 30 h sur 4 mois.

1. Problème

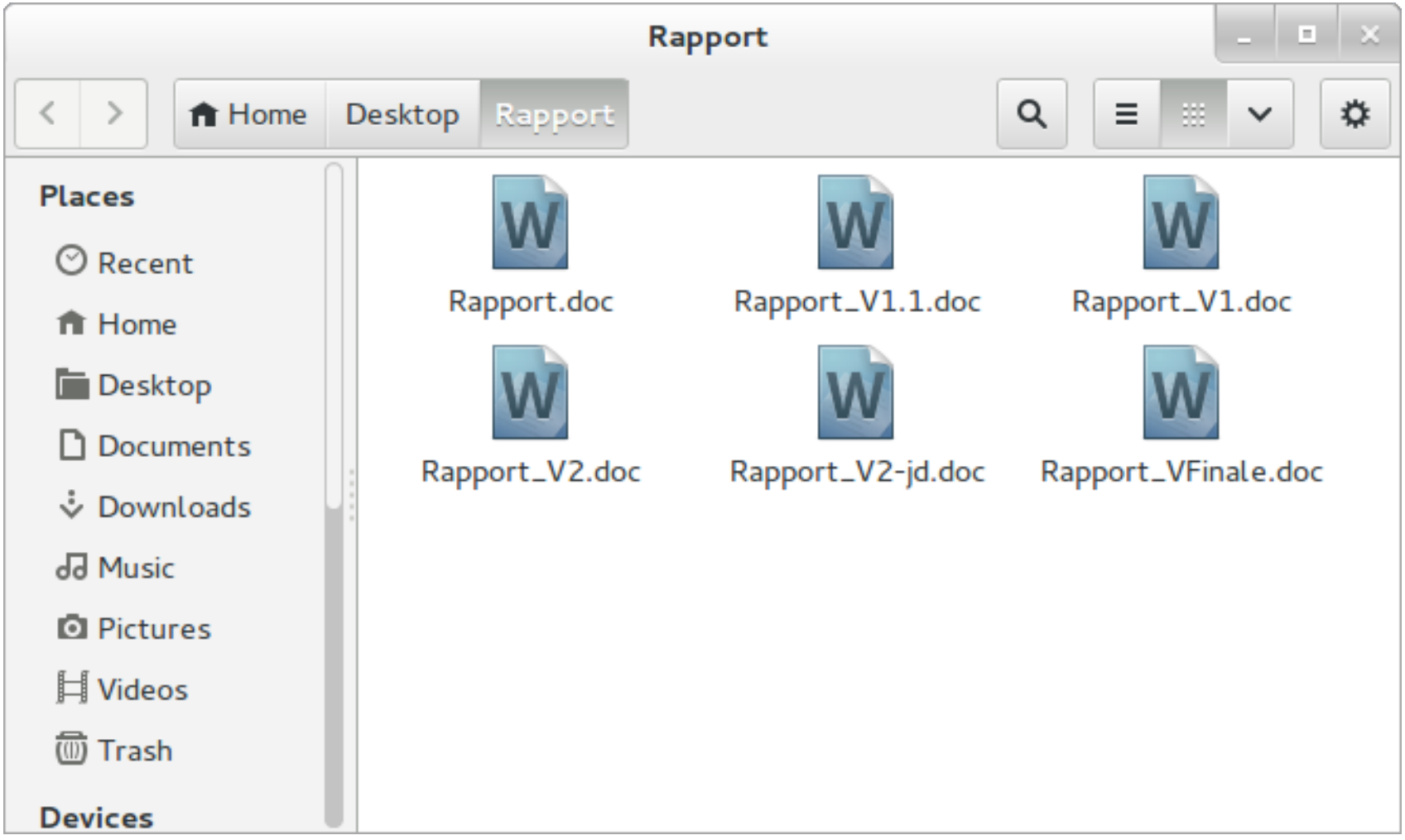

Figure 1: La source du problème

2. Problème

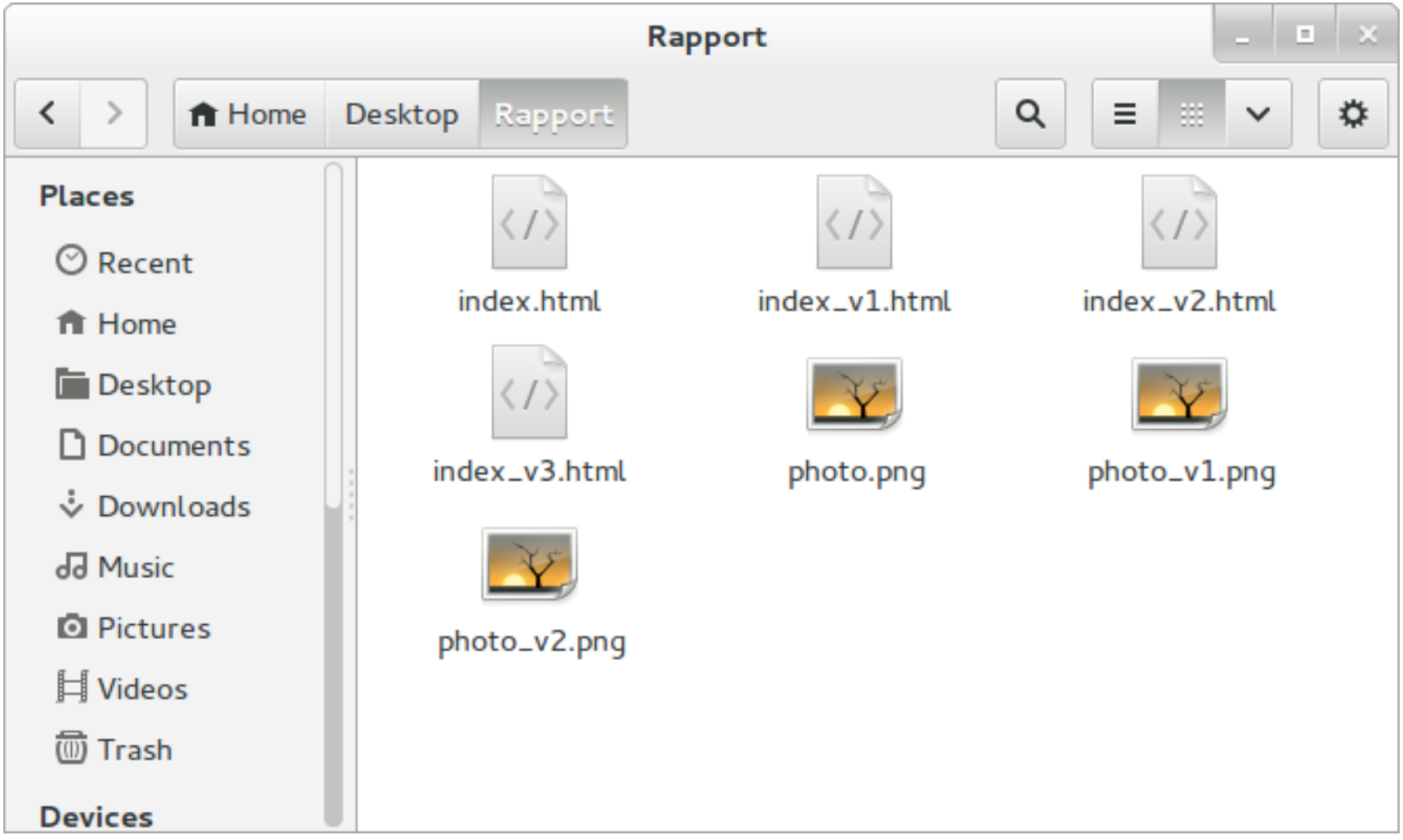

Figure 2: La source du problème (bis)

3. Problème(s)

- A partir de ces documents, les besoins éprouvés peuvent-être :
  - Comprendre les versions
  - Pouvoir revenir en arrière, avoir une “trace”
  - Pouvoir avoir une collaboration simple

4. Git : un outil de versionnage

- Date d'Avril 2005, créé par le créateur du noyau linux Linus Torvalds et par Junio Hamano
- Sous licence libre GNU GPLv2
- Version 2 actuellement
- Autres outils :
  - connus : CVS, SVN (Subversion)
  - moins connus : Mercurial, Bazaar

5. Git : Principes généraux

- Travail dans un repository (dépôt en francais) == un dossier
  - Il contient un dossier (souvent caché) `.git` qui contient toutes les archives enregistrées
- Contrairement à Dropbox ou Google Drive, pour qu'une modification soit archivée, il faut que cela soit explicité
  - Ces modifications archivées sont appelées "commit"
  - Elles portent un message enregistré par l'utilisateur
  - Elles peuvent comporter plusieurs fichiers
  - Les fichiers qui ont subi des modifications doivent y être ajoutés explicitement

6. Git : Principe généraux

On distingue trois “états” des fichiers

- un état de travail : le fichier a subi des modifications mais nous ne l’avons pas encore ajouté (add) à un futur commit
- un état de futur enregistrement : le fichier a été ajouté (add) à un commit mais le commit n’a pas été finalisé avec un message
  - Appelée *staging area*, ou *stage*
- un état archivé : le fichier a subit des modifications enregistrées et n’a pas été modifié depuis lors.

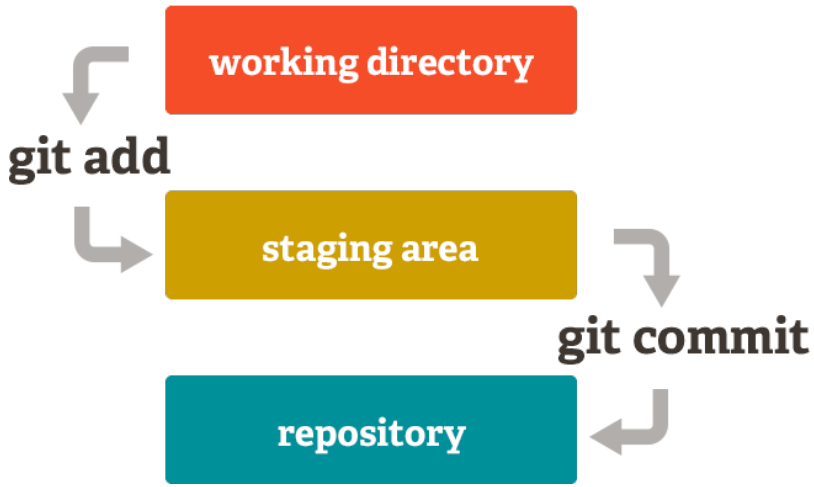

Figure 3: Stage

7. Git : Principe généraux

On distingue trois “états” des fichiers

- un état de travail : le fichier a subi des modifications mais nous ne l’avons pas encore ajouté (add) à un futur commit
- un état de futur enregistrement : le fichier a été ajouté (add) à un commit mais le commit n’a pas été finalisé avec un message
  - Appelée *staging area*, ou *stage*
- un état archivé : le fichier a subit des modifications enregistrées et n’a pas été modifié depuis lors.

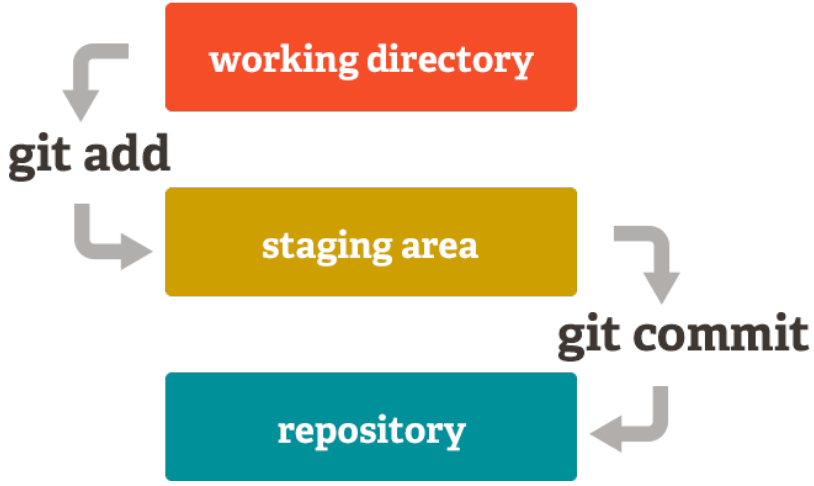

Figure 4: Stage

8. Git : Commandes principales

- **Initialisation** : `git init`
- **Ajout de modifications** : `git add [Nom du fichier]` ou `git add -A` (Ajout de tous les fichiers changés)
- **État du dépôt** : `git status` (Donne la liste des modifications réalisées)
- **Enregistrement de modifications** : `git commit -m "Message du commit"`. N'oubliez jamais le -m à moins de vouloir passer un mauvais moment.
- **Historique du repository** : `git log`
- **Différence entre l'état archivé et l'état actuel** : `git diff`

9. Changer les couleurs si difficile à lire

`color.*`

If you want to be more specific about which commands are colored and how, Git provides verb-specific coloring settings. Each of these can be set to `true`, `false`, or `always`:

```
color.branch
color.diff
color.interactive
color.status
```

In addition, each of these has subsettings you can use to set specific colors for parts of the output, if you want to override each color. For example, to set the meta information in your diff output to blue foreground, black background, and bold text, you can run

```
$ git config --global color.diff.meta "blue black bold"
```

You can set the color to any of the following values: `normal`, `black`, `red`, `green`, `yellow`, `blue`, `magenta`, `cyan`, or `white`. If you want an attribute like bold in the previous example, you can choose from `bold`, `dim`, `ul` (underline), `blink`, and `reverse` (swap foreground and background).

Figure 5: Git Couleurs

10. Importance des messages

Figure 6: https://xkcd.com/1296/

11. Formats “compilés” / “compressés”

- https://ponteineptique.github.io/cours-git/cours-1/images/blanc.png

- https://ponteineptique.github.io/cours-git/cours-1/images/blanc2.png

12. Les branches

- Les branches sont comme des “Sauvegarder sous. . . ” pour l'ensemble du dépôt
  - Sauf qu'on peut les rejoindre/fusionner plus facilement !
- La branche par défaut s'appelle master
  - Dans `git status`, la branche `master` doit être affichée dans votre dossier de notes
- Offre la possibilité de travailler sur différents problèmes en parallèle. Possibilité de travailler sur des problèmes différents en même temps et de changer de tâche rapidement.
  - Une branche Master
  - Une branche bug 1
  - Une branche bug 2 parce que celui-ci est plus pressé
  - etc. . .

13. Les branches : Vocabulaire

- **Créer une branche** : `git branch [nom de la branche]`
- **Se déplacer dans une branche** : `git checkout [nom de la branche]`
- **Se déplacer et créer une branche en même temps** : `git checkout -b [nom de la branche]`
- **Fusionner une branche** : `git merge` (On fusionne toujours depuis la branche réceptrice, celle sur laquelle on veut continuer de travailler ensuite)

14. Branches

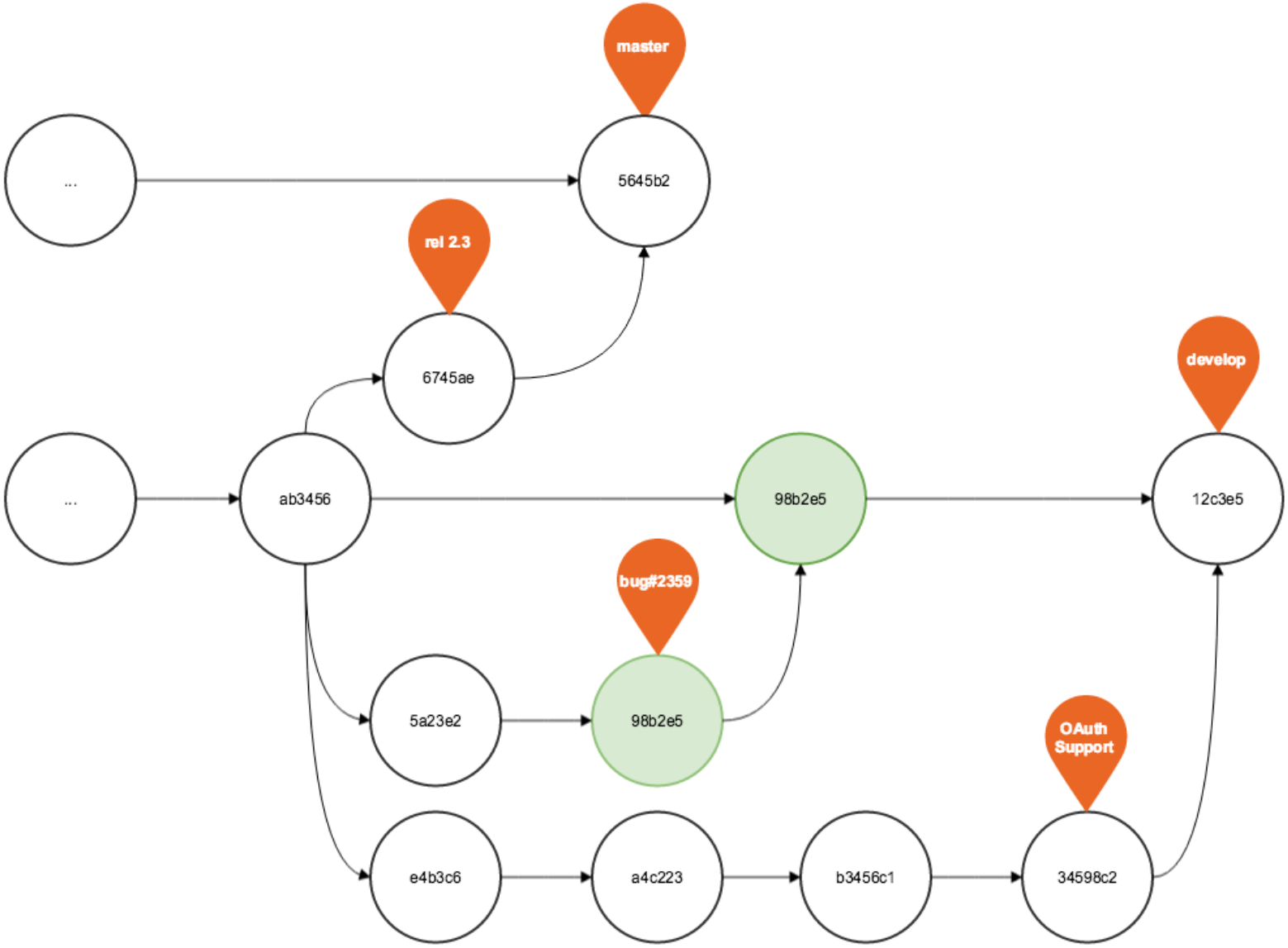

Figure 7: Issue de
http://www.ateam-oracle.com/developercloudserviceworkflow/

Crédits

- Images “Motivation” issues de Introduction à Git sous licence CC BY SA 3.0
- Image “Stage” issue de https://git-scm.com/about/staging-area