Présentation Git

Un outil de collaboration puissant

Denis Pettens Pablo Gonzalez Alvarez

6 octobre 2016

Louvain-li-Nux

Table des matières

Introduction

Gérer un projet

Comment gérez-vous actuellement un projet ?

- L'envoyer à travers un message sur Facebook, ... (**Très mauvaise idée**)
- L'envoyer par mail (**Un peu moins**)
- Utiliser une Dropbox, Google Drive, ... (**Déjà mieux mais toujours risqué ou manque de fonctionalités**)

Solution : Utiliser un **VCS**

Monsieur, c'est quoi Git ?

- `Git` a été créé en 2005 par **Linus Torvalds** (auteur du kernel `Linux`) ;
- Il est le logiciel de gestion de versions décentralisé (**VCS**) le plus utilisé au monde en 2016 ;
- Il permet donc de suivre dans le temps l'avancement d'un projet de son début à sa fin ;
- Il garde en mémoire tout ce que vous avez fait un jour dans votre projet et permet de les éditer facilement ;
- ...[1]

1. Tiré en partie du site (`https://git-scm.com`)

Pourquoi l'utiliser ?

- Le plus connu et utilisé (communauté très présente) ;
- Vitesse ;
- Conception simple ;
- Facile d'utilisation mais aussi très puissant ;
- Support pour les développements non linéaires (milliers de branches parallèles) ;
- Complètement distribué ;
- Capacité à gérer efficacement des projets d'envergure tels que le noyau Linux (vitesse et compacité des données). [2]

2. Tiré en partie du site (`https://git-scm.com`)

GitKraken

- `git` ne possède pas d'interface graphique, tout se fait en ligne de commande ;
- Simple d'utilisation ;
- Idéale pour avoir une vue globale de son projet ;
- Cross-plaform ;
- Malheureusement, il n'est pas Libre.

Autres alternatives : `gitg`, `GitHub Desktop`, `SourceTree`, ...

GtiKraken : Image

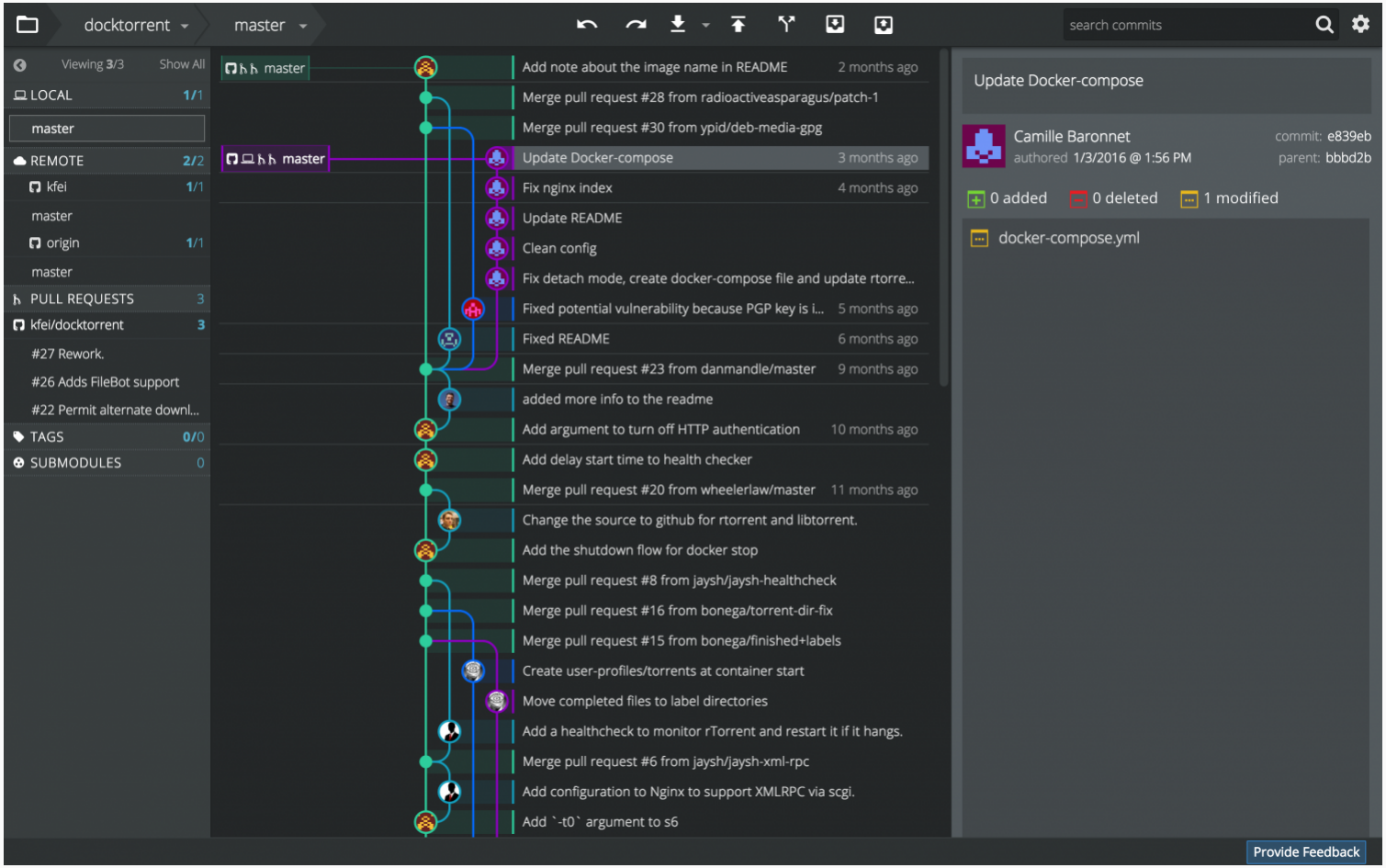

Figure 1 – Tiré de `https://www.camillebaronnet.fr/blog/fr/gitkraken-enfin-outil-visuel-moderne-pour-git`

Instalation et configuration

Installer Git et GiKraken

Git

- **Ubuntu** : `sudo apt-get install git`
- **OS X** : `https://sourceforge.net/projects/git-osx-installer/`
- **Windows** : `https://git-for-windows.github.io/`

GitKraken

- **Toutes plateformes** : `https://www.gitkraken.com/`

Configuration de base

Git a besoin de deux informations de base sur vous pour pouvoir travailler efficacement :

- **Nom et Prénom**

```
1 git config --global user.name "Jules Dupont"
```

- **Email**

```
1 git config --global user.email "jules.
    dupont@email.fr"
```

L'option `--global` permet de configurer `git` pour tous vos autres projets sur votre PC.

GitKrakren demande de créer un compte par défaut pour pouvoir l'utiliser, ensuite il vous introduit rapidement sur son interface.

Premier pas avec Git

git init

- Cette commande permet d'initialiser un dossier en un nouveau dépot local `git` où tout le projet versionné sera contenu dedans
- Cela crée un sous-dossier `.git` où tout la magie de git se fait
- Exemple

```
1 mkdir newProject
2 cd newProject/
3 git init
```

Les trois zones de git

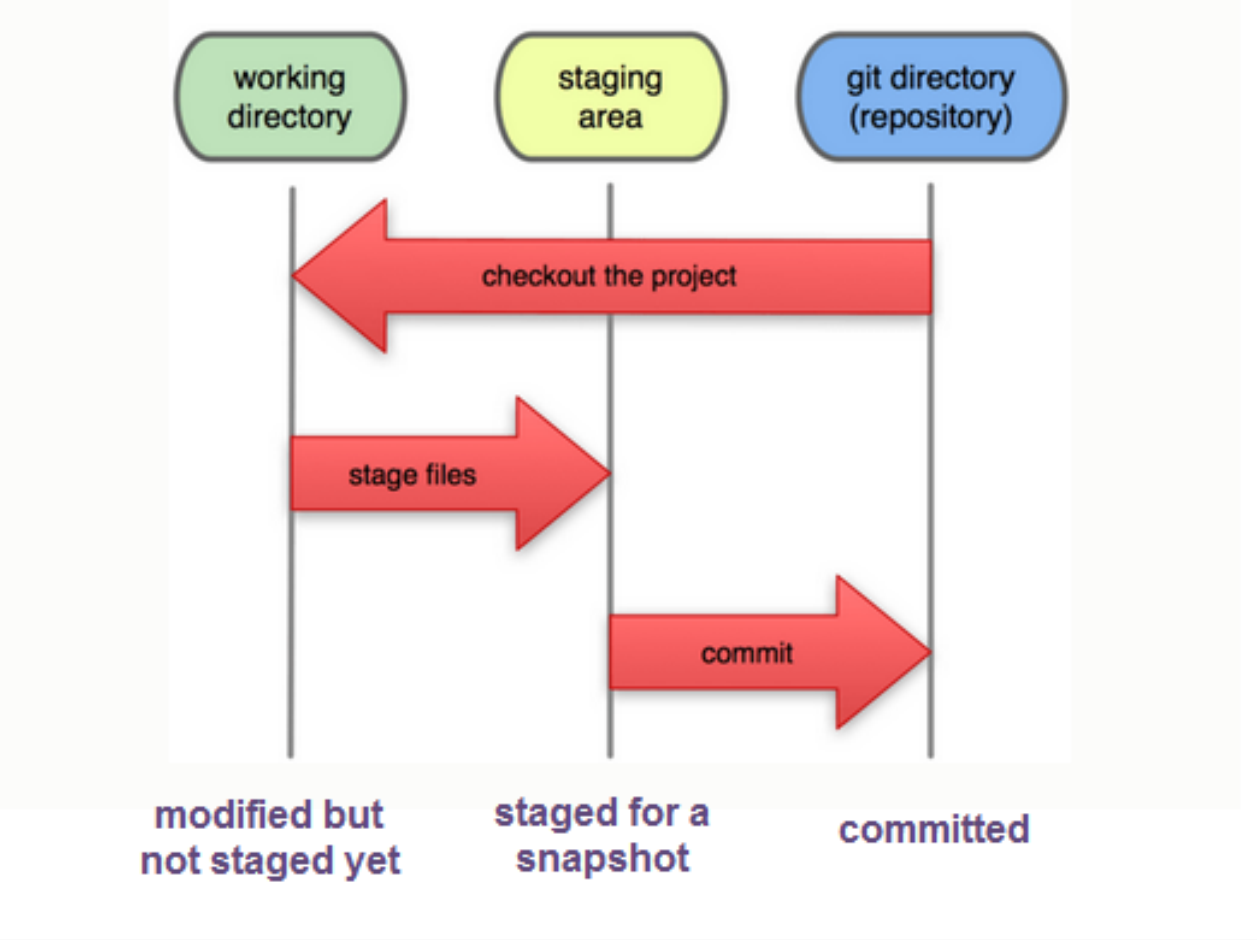

Figure 2 – Tiré du site `https://www.camillebaronnet.fr/blog/fr/gitkraken-enfin-outil-visuel-moderne-pour-git`

Le cycle de vie d'un fichier

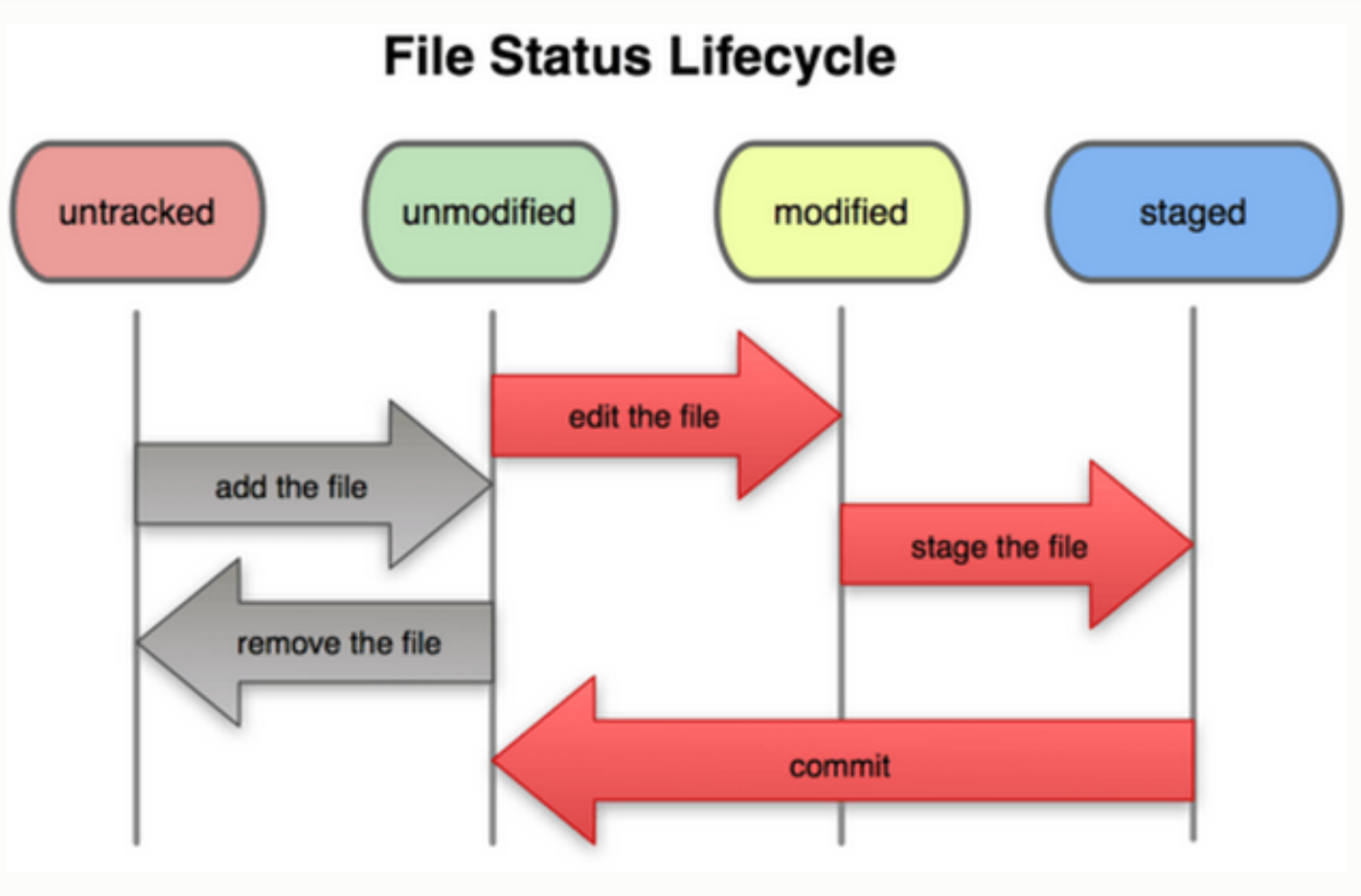

Figure 3 – Tiré du site `https://www.camillebaronnet.fr/blog/fr/gitkraken-enfin-outil-visuel-moderne-pour-git`

git status

- Cette commande permet de connaître l'état du projet sous `git`
- Elle permet d'obtenir des informations sur les différentes zones
- Conseil : Utiliser là dés que vous avez un doute sur l'état de votre projet
- Exemple

```
1 git status
```

git add

- Cette commande permet d'ajouter un ou plusieurs fichier(s)/dossier(s) que l'on veut versionné(s) dans la zone de **staging** afin de préparer le prochain **commit**
- Attention, les fichiers ne sont pas encore dans l'historique de `git`
- Exemple

```
1 touch test.txt
2 git add test.txt
```

git mv

- Cette commande permet de déplacer un ou plusieurs fichier(s)/dossier(s) dans un autre dossier qui est/sont déjà versionné(s) par `git`
- Attention, `git` suit cette modification et la place dans la zone de **staging**
- Exemple

```
1 mkdir folder
2 git mv test.txt folder/test.txt
```

git rm

- Cette commande permet de supprimer un ou plusieurs fichier(s)/dossier(s) du projet qui est/sont déjà versionné(s) par `git`
- Attention, `git` suit cette modification et la place dans la zone de **staging**
- Exemple

```
1 git rm folder/test.txt
```

git commit

- Cette commande permet de créer un snapshot de l'état actuel du projet
- Elle se fait généralement après des modifications de fichiers et placés dans la zone de **staging**
- On l'identifie avec un message complet et précis
- Exemple

```
1 git commit # Ouvre l'editeur pour le message
2 git commit -m "Mon message"
```

- Quand **commit** vos fichiers ?

Voir/Manipuler l'historique

git log

- Cette commande permet de visualiser et/ou d'obtenir des informations sur les commits du projet
- On peut remarquer que chaque commit est identifé par une clé unique qui est utile pour différentes commmandes de `git`
- Exemple

```
1 git log
```

- Options utiles
  - `--oneline` Afficher chaque commit sur une seule ligne
  - `-n N` Afficher les N derniers commits
  - `-p file` Afficher les commits liés à un fichier
  - `-author name` Afficher les commits liés à une personne
  - `--pretty` et `--graph` Affichage plus joli

git diff

- Cette commande permet de comparer les différences entre le **dernier commit** et votre projet actuel d'un ou de tous les fichiers

```
1 git diff
2 git diff <file>
```

- Pour voir celles liées à la zone de **staging**, rajoutez l'option `--staged`

```
1 git diff --staged
```

- Pour voir entre un commit particulier et votre projet actuel ou entre deux commits

```
1 git diff <commit>
2 git diff <commit>..<commit>
```

git commit –amend

- Cette commande permet d'éditer le dernier commit de votre projet
- Permet de corriger une erreur ou ajouter un fichier oublié
- Attention, ne jamais le faire une fois le commit envoyé sur **Github** ou autre serveur
- Exemple

```
1 git commit --amend
```

git checkout

- Cette commande permet de revenir en tant que **spectateur** sur un commit précédent
- Attention, `git` ne modifie en rien quelque chose, vous ne faites que regarder
- Exemple

```
1 git checkout fa1751b
2 git checkout master # Revenir au dernier
      commit
```

git revert

- Cette commande permet d'inverser un commit
- Elle va crée un **nouveau commit** pour supprimer les changements du commit sélectionné
- Exemple

```
1 git revert fa1751b
```

Autres commandes utiles

- `git reset` (Commande **dangeureuse**)
- `git rebase` (Commande **dangeureuse**)
- ...

Les branches

Les Branches

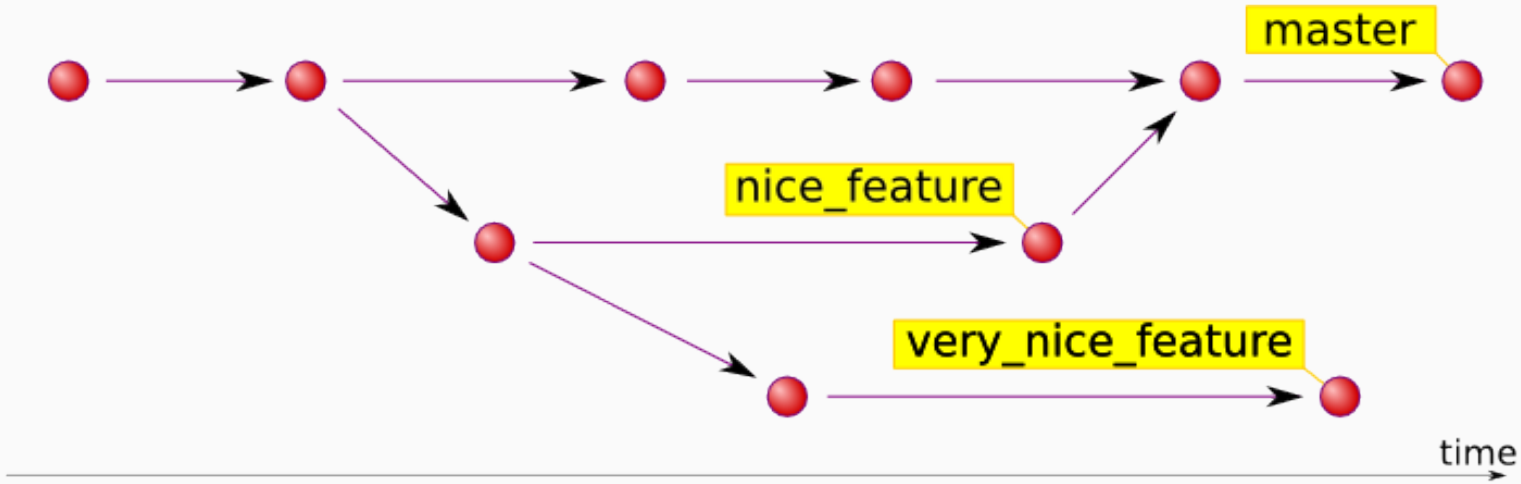

Figure 4 – Tiré du site
`http://hades.github.io/media/git/git-history.png`

git branch

- Cette commande permet de gérer les branches de façon globale
- Permet de lister les branches

```
1 git branch
```

- Permet de créer une nouvelle branche <branche>

```
1 git branch <branche>
```

- Renomme la branche courante en <branche>

```
1 git branch -m <branche>
```

- Permet de supprimer une branche

```
1 git branch -d <branche>
```

git checkout

- Cette commande permet aussi (en plus de voir des anciens commits) de changer de branches
- Exemple

```
1 git branch develop
```

- En revanche, vous pouvez directement l'utiliser pour créer une nouvelle branche et se déplacer dessus

```
1 git checkout -b <branche>
2 # equivaut a
3 git branch <branche>
4 git checkout <branche>
```

git merge

- Cette commande permet de de ramener une branche sur une autre et ainsi de la fusionner.
- Attention, la fusion de 2 branches se fait toujours à partir de la branche principale.
- Crée un commit de fusion
- La branche fusionnée ne sera pas affectée
- Exemple

```
1 git merge <branche>
```

Le travail en groupe

Github vs Bitbucket

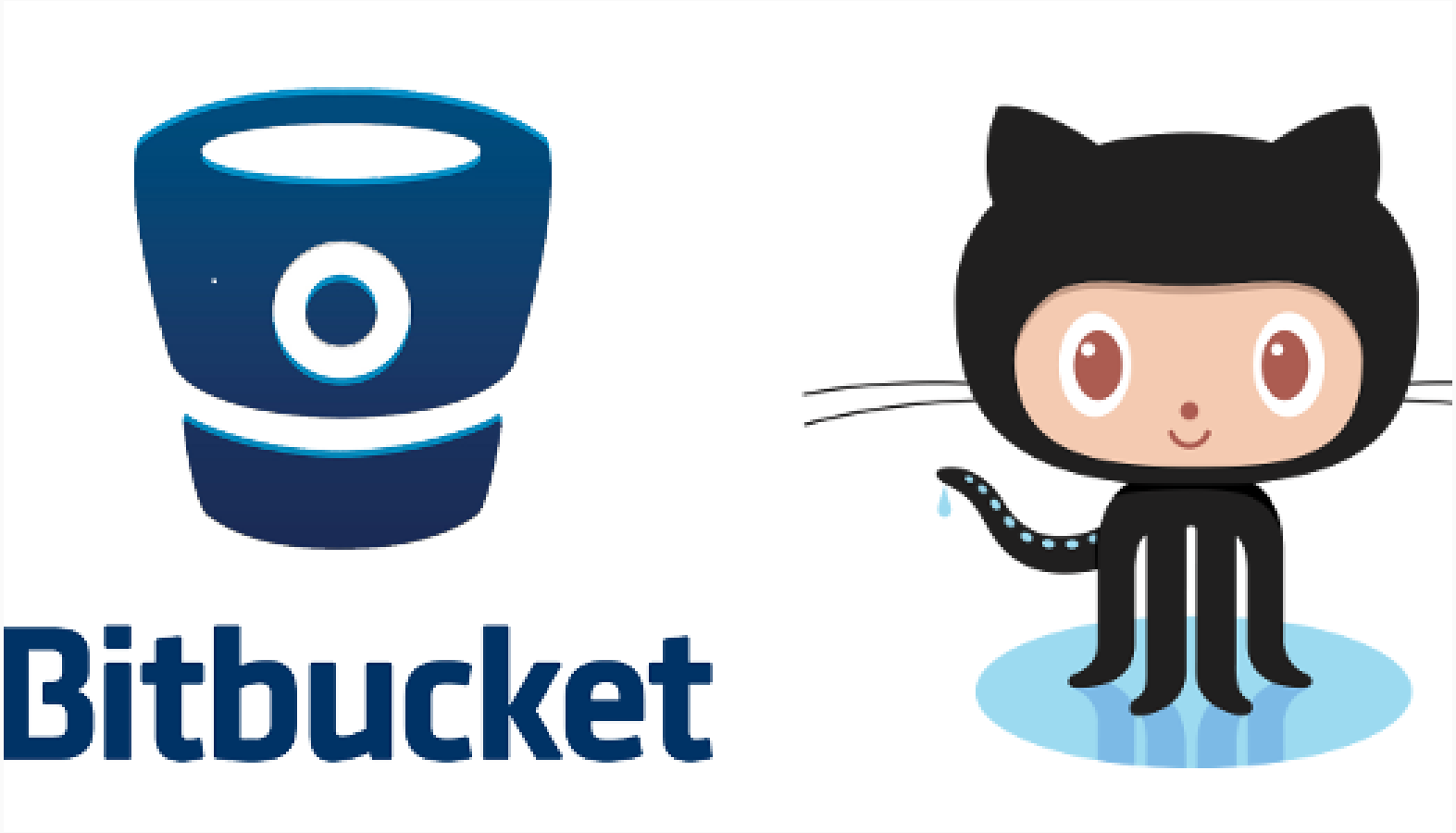

Figure 5 – Tiré du site `https://media.licdn.com`

git clone

- Cette commande permet de cloner un répertoire `git` depuis un serveur principal
- Exemple

```
1 git clone <url>
```

git remote

- Cette commande permet d'ajouter un serveur distant à votre répertoire `git` si vous n'avez pas cloner le répertoire depuis le serveur
- Exemple

```
1 git remote add origin <url>
```

git pull

- Cette commande permet de récupérer les dernières modifications depuis le serveur principal
- Exemple

```
1 git pull origin
```

git push

- Cette commande permet d'envoyer les dernières modifications locales sur le serveur principal
- Exemple

```
1 git push origin master
```

Pour aller plus loin ...

Idées intérésantes

- git flow
- Github student pack
- Travis CI
- ...

Liens utiles

- https://git-scm.com/
- http://rogerdudler.github.io/git-guide/
- https://fr.atlassian.com/git/tutorials/
- https://www.grafikart.fr/formations/git

Questions ?