

# Bonnes pratiques en bioinformatique : (essayer) d'aller vers plus de reproductibilité

## DUBii - Module 5

Valentin Loux - Cédric Midoux - Olivier Rué

Plateforme bioinformatique Migale

(valentin.loux | cedric.midoux | olivier.rue)@inrae.fr

2020/06/11

# Programme de l'après-midi

- Intro : Reproductibilité, science ouverte
- Partie 1 : Organiser son espace de travail
- Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note
- Partie 3 : Versionner ses documents
  - TP
- Partie 4 : Utilisation de documents computationnels -- Notebook
  - TP
- Conclusion et pour aller + loin

# Tout le monde a déjà eu cette expérience

OPEN  ACCESS Freely available online



## The *Arthrobacter arilaitensis* Re117 Genome Sequence Reveals Its Genetic Adaptation to the Surface of Cheese

Christophe Monnet<sup>1,2\*</sup>, Valentin Loux<sup>3</sup>, Jean-François Gibrat<sup>3</sup>, Eric Spinnler<sup>1,2</sup>, Valérie Barbe<sup>4</sup>, Benoit Vacherie<sup>4</sup>, Frederick Gavory<sup>4</sup>, Edith Gourbeyre<sup>5</sup>, Patricia Siguier<sup>5</sup>, Michaël Chandler<sup>5</sup>, Rayda Elleuch<sup>6</sup>, Françoise Irlinger<sup>1,2</sup>, Tatiana Vallaey<sup>7</sup>

Un Papier interessant

collaboration with the user community. Genome comparisons were performed using Origami, an **in-house** tool developed for microbial genome comparison. Orthologs were defined as reciprocal best hits with an e-value lower than  $10^{-3}$ . Transposases were excluded from the analysis. Core genes were defined as orthologs shared between the four *Arthrobacter* strains. Synteny was studied using an **in-house** developed tool, Align, using dynamic programming to search conserved gene trains allowing gaps and “mismatches” (homology relation instead of orthology). Circular representation of the genome was produced using the Circos software [27].

Un materiel et méthode décevant

# Crise de la reproductibilité

- Problème général ( "Reproducibility crisis")
  - Remis en avant par les sciences sociales ( psychologie )
  - Etendu à l'ensemble des disciplines

Mais un problème qui n'est pas nouveau :

- Expériences de la pompe à vide au XVIIe siècle (von Guericke et Boyle)



# Et en bioinfo ?

Un problème vieux comme la bioinformatique :

- En 2009, moins de la moitié des expériences de transcriptomique parues dans Nature Genetics n'ont pu être reproduites
- Sur 50 articles citant BWA en 2011, 31 ne citent ni version, ni paramètres. 26 ne donnent pas accès aux données sous-jacentes
- Selon un sondage mené auprès de plus de 1500 biologistes
  - 70% ont déjà éprouvé des difficultés à reproduire une analyse ([Baker, 2016](#))
- [Ten Years Reproducibility Challenge](#) refaire ses analyses d'il y a 10 ans ...

# Quelles sont les difficultés ?

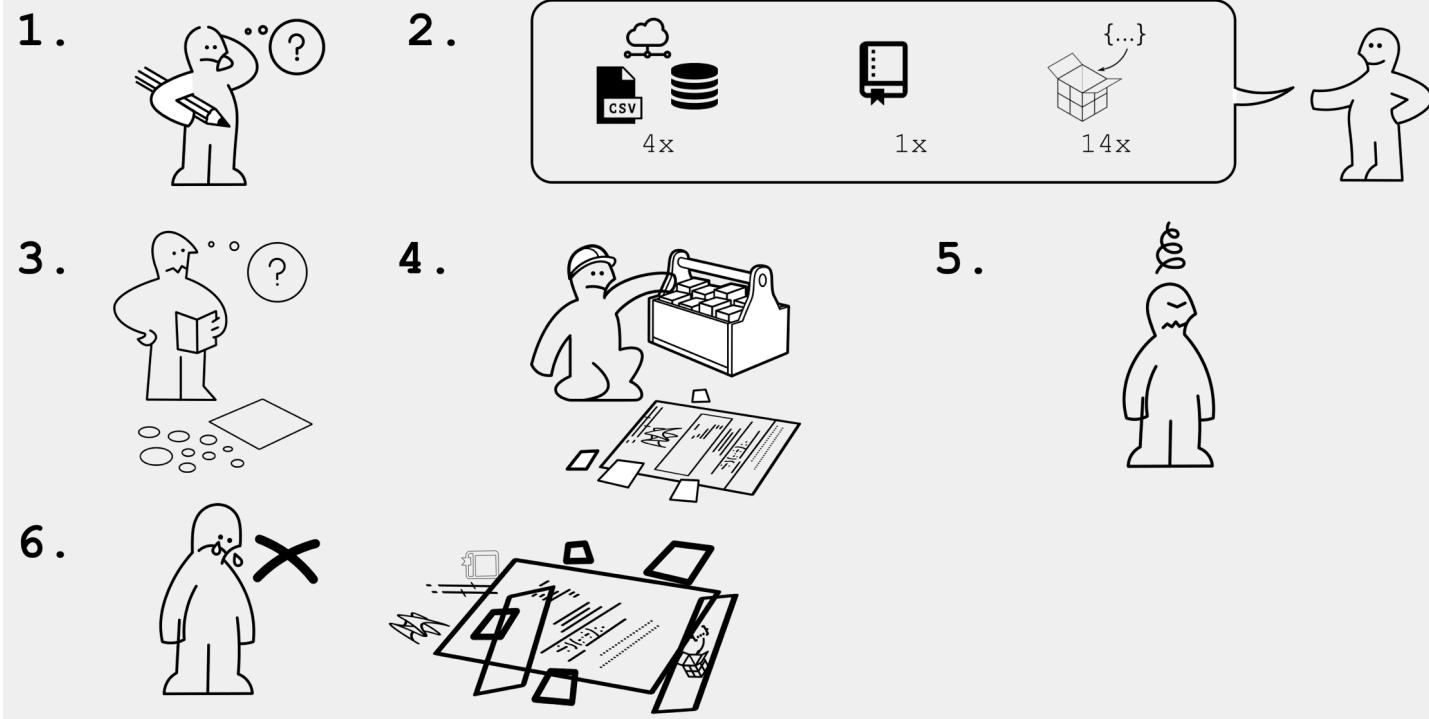
- Problèmes **d'accès aux données** :
  - "data available upon request"
  - méta-données inexistantes ou insuffisantes
- Problèmes **d'accès aux outils** :
  - outils anciens ou obsolètes
  - difficultés à installer
- Problèmes de **paramétrage de l'analyse**
  - version des outil
  - paramètres des outils
  - enchainement des outils
- Problème d'accès aux **ressources nécessaires**
  - calcul
  - stockage

# RéPLICATION ≠ Reproductibilité

- La réPLICATION indépendant d'analyse est à la base de la méthode scientifique
- En complément de **réPLICATION** indépendante ( expérimentation, échantillonnage, analyse, ...), la **rePRODUCTION** d'analyse est indispensable à l'évaluation et à la compréhension de la démarche employée
- Il existe une ambiguïté en anglais entre réPLICATION (*replication*) et reproduction (*reproducibility*). Derrière la *reproducibility crisis* on mélange les deux :
  - Impossibilité de répliquer des résultats de façon indépendante (psychologie, médecine, biologie...)
  - Impossibilité de reproduire des analyses à partir des mêmes données de départ
- Chacun peut déjà, par l'utilisation d'outils conviviaux, améliorer la reproductibilité de ses travaux

Source : ([Allard, 2018](#))

# En pratique, qu'est ce qu'être reproductible ?



<https://github.com/karthik/rstudio2019/blob/master/reproducible-data-analysis.pdf>

# En pratique, qu'est ce qu'être reproductible (2) ?

Avoir accès :

- aux pièces (les **données**)
- aux outils ( les **logiciels**, )
- au mode d'emploi : **paramètres, workflows d'analyse**

Mais aussi :

- à la description des pièces, de la façon dont elles ont été produites (**méta-données**)
- à la documentation technique (**choix techniques explicites**)
- au savoir faire du monteur (**formation**)
- Éventuellement à un atelier équipé pour le montage (**ressources informatiques**)

# FAIR : un pré-requis la reproductibilité

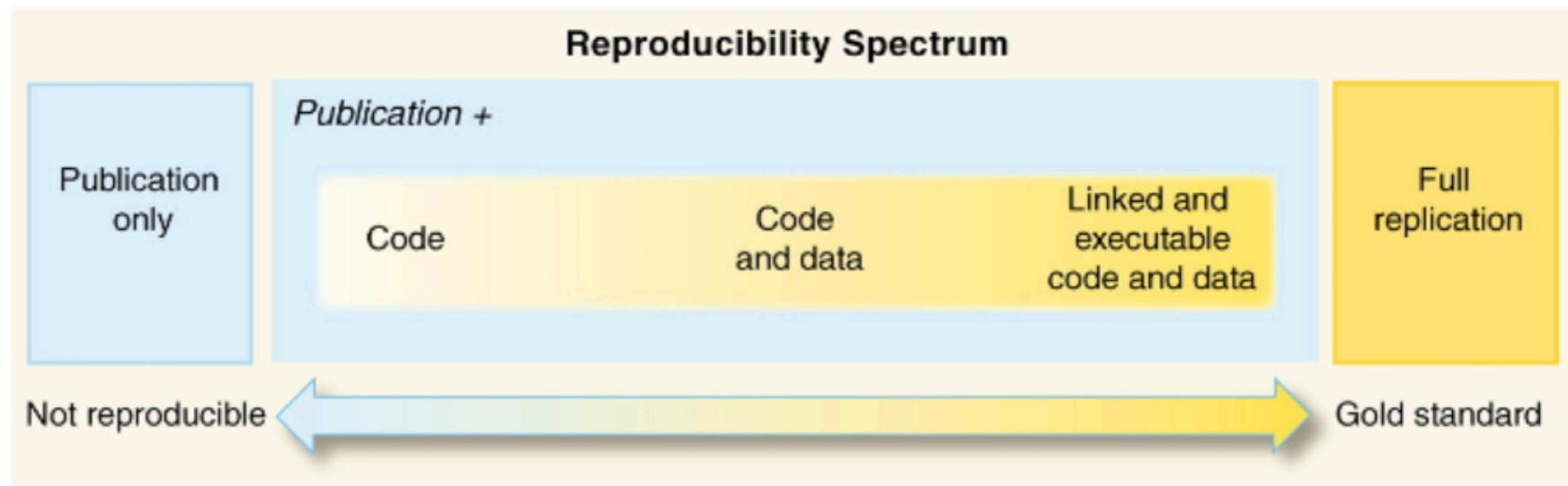


Findable Accessible Interoperable Reusable

Principes autour des données au sens large :

- Facile à trouver : pour les humains et les ordinateurs :
  - id unique et pérennes
  - métadonnées riches
- Accessibles à **long terme**
  - entrepôt "pérenne"
  - licence d'utilisation explicite (FAIR ≠ ouvert)
- Interoperables : faciles à combiner avec d'autres jeux de données
  - formats ouverts et documentés
  - vocabulaire standardisé, ontologies (données et métadonnées)
- Réutilisables :
  - réutilisables par soi, par d'autres
  - réutilisables par des machines

# Le spectre de la reproductibilité



Spectre de la reproductibilité,

Source : (Piazzi, Cerqueira, Manso, et al., 2018)

# L'outillage

Aller de façon pragmatique vers une documentation accrue de ce que l'on fait (comment, pourquoi)

- Rendre accessible ses données à soir, aux partenaires, à tous :
  - Data Management Plan (Opidor)
  - Dépôt internationaux (ENA, NCBI)
  - DataVerse
- Définir les outils utilisés :
  - conda, bioconda
  - singularity, docker
  - Machine Virtuelle
- Décrire son workflow d'analyse, le rendre portable :
  - Galaxy
  - Snakemake, Nextflow
- Gérer les versions de ses codes, les publier :
  - git
  - github / gitlab
- Tracer son analyse dans des documents partageables et réutilisables :
  - Rmd
  - Jupyter Notebooks

# Objectifs du TP

Décomplexifier les problèmes, se décomplexifier, désacraliser la reproductibilité !

Vous fournir des outils, des pistes pour rendre vos projets :

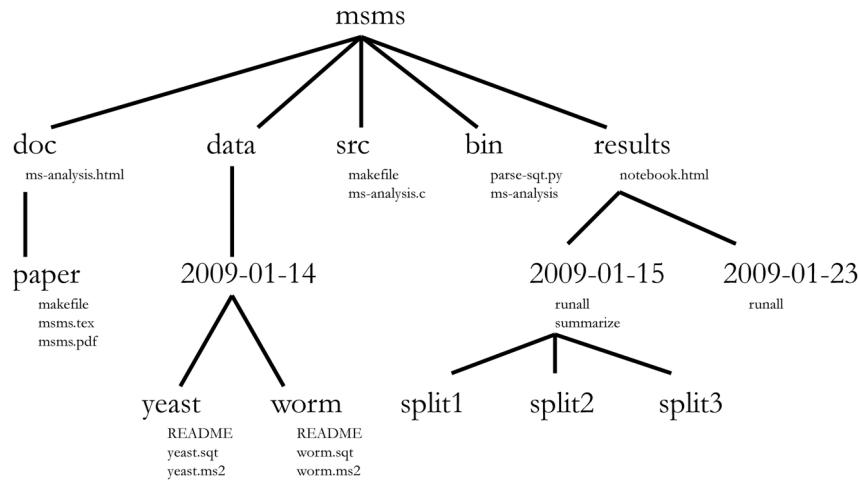
- transparents
- robustes
- réutilisables
- partageables

Bref, *plus* reproductibles.

Parties pratiques sur la versionning des documents (Git et GitHub) et les docs computationnels.

# Partie 1 : Organiser son espace de travail

# Partie 1 : Organiser son espace de travail



Source : ([Noble, 2009](#))

## Séparer

- données
- code
- scripts
- résultats
- Avoir un copie de sauvegarde de ses données
- Mettre le répertoire et les fichiers de données en lecture seule

# Organiser son espace de travail (2)

## Box 1. Summary of practices

1. Data management
  - a. Save the raw data.
  - b. Ensure that raw data are backed up in more than one location.
  - c. Create the data you wish to see in the world.
  - d. Create analysis-friendly data.
  - e. Record all the steps used to process data.
  - f. Anticipate the need to use multiple tables, and use a unique identifier for every record.
  - g. Submit data to a reputable DOI-issuing repository so that others can access and cite it.

Source : ([Wilson, Bryan, Cranston, et al., 2017](#))

## Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note

# Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note

*Comment mettre en forme et structurer simplement un document texte ?*

→ avec un balisage faible tel que *Markdown*

- Permet :
  - Organiser les titres de sections
  - Italique / gras / souligné
  - Générer des listes
  - Ajouter des tableaux
  - Insertion d'image et de blocs de code
- Texte codé en UTF-8 (assure une pérennité, lisibilité et portabilité) facilement versionnable.
- Les langages de balisage permettent de mettre en forme convenablement le fichier pour un meilleur confort de

*Exemple illustrant la simplification du balisage :*

- HTML

```
<ul>
  <li>item1</li>
  <li>item2</li>
</ul>
```

- Markdown

```
- item1
- item2
```

# Partie 2 : Markdown - exemples de commande

```
# Titre H1
## Sous-titre H2
### Sous-titre H3

*italique*, **gras** et `code`

> Citations

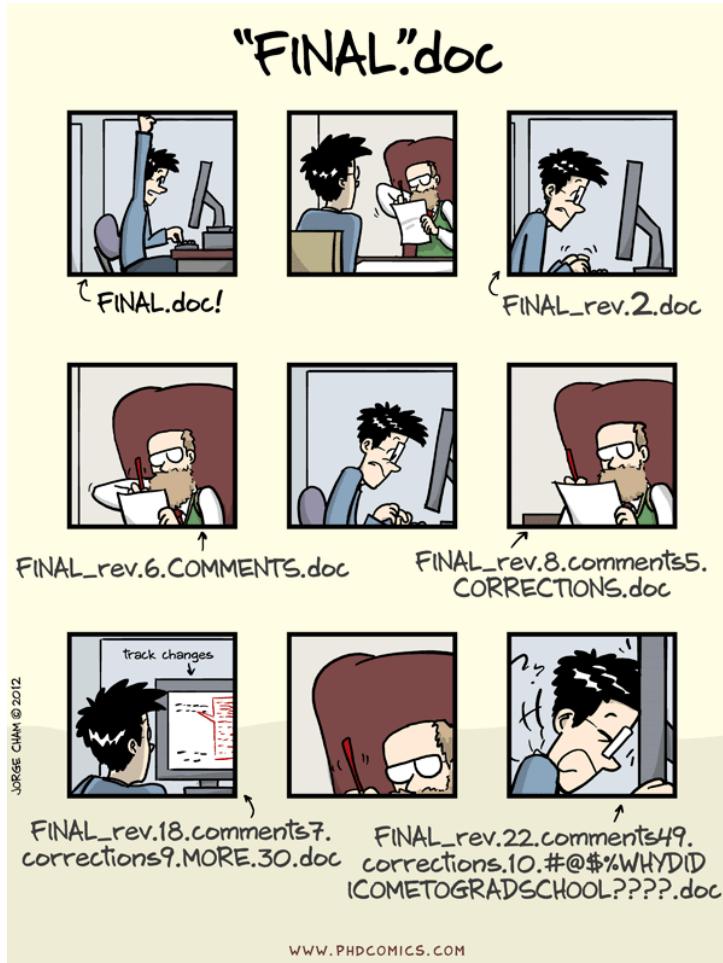
```
bloc de code
```

* liste
  * item
  * item

[lien](https://fr.wikipedia.org)
 #lien vers une image en li
```

Gardez ce **mémo** à porté de main !

# Partie 3 : Versionner ses documents



Piled Higher and Deeper by Jorge Cham. phdcomics

# Partie 3 : Versionner ses documents

- Les documents évoluent, il est nécessaire de suivre les versions
  - On trace toutes les modifications faites
  - On garde chaque version des documents du dossier de travail
  - C'est un peu comme copier-coller son dossier de travail ... mais en beaucoup plus précis et pratique !
  - `git` est un standard dans la gestion des versions distribuée

# Vocabulaire

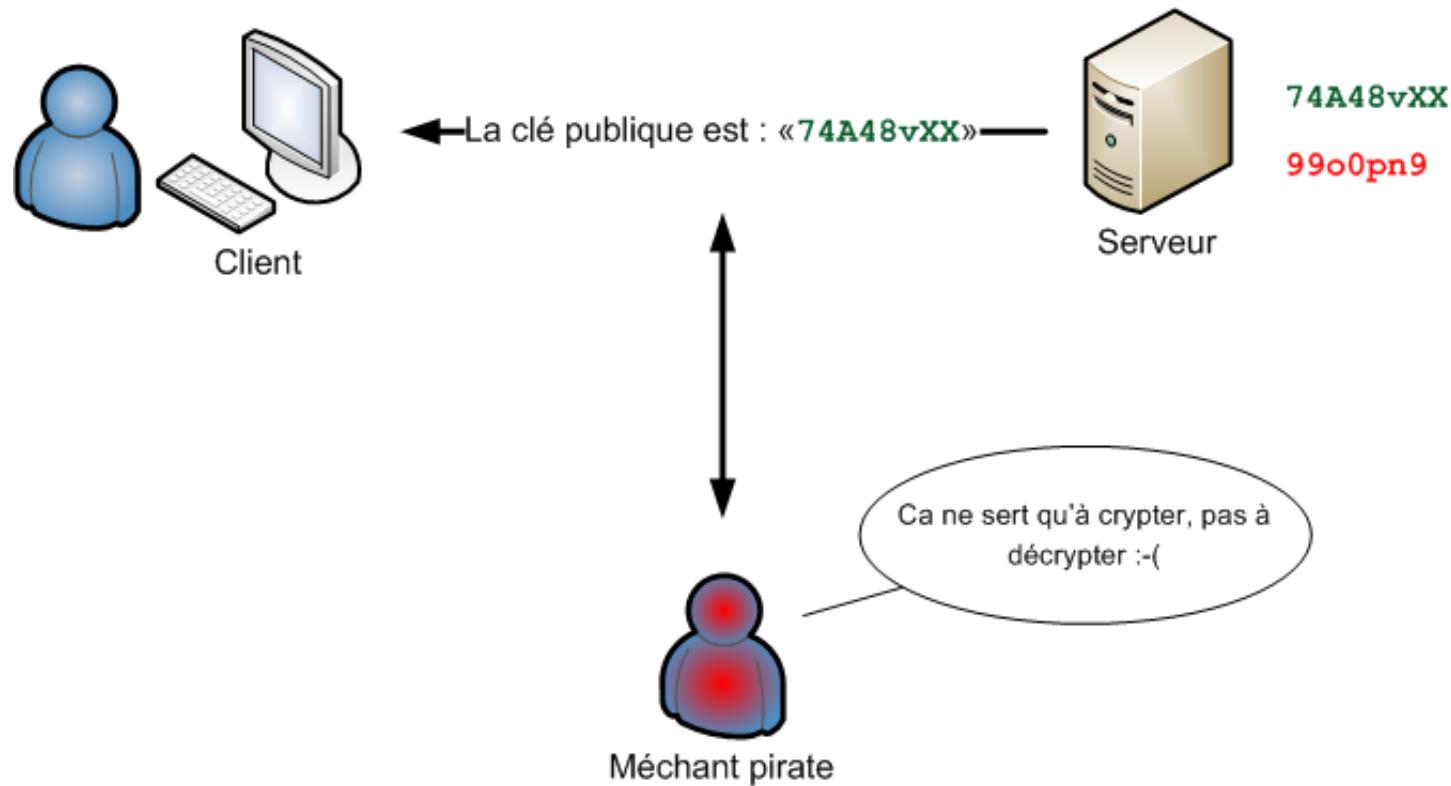
Quelques mots à connaître :

- **Repository / Dépôt** = Dossier / Projet
- **Commit** = Enregistrement d'un ensemble de fichier à un instant T (= photo)
- **Branche** = Ensemble chaîné de commits, par défaut la branche principale s'appelle « master » (cette notion n'est pas primordiale pour débuter)
- **Git** : logiciel *open-source* de gestion de version de document.
- **Github** : site web permettant de centraliser en ligne ses dépôts git et facilitant la collaboration sur les projets.



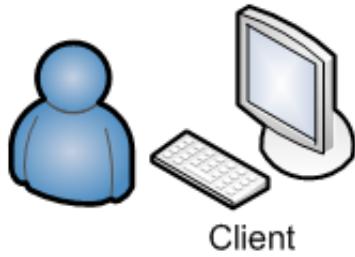
# Retours sur les clés SSH

- Protocole pour la sécurisation des transferts de données.
- Méthode de chiffrement asymétrique qui fonctionne avec une paire de clé :
  - une clé *publique* qui sert à chiffrer (et que vous pouvez partager à l'extérieur)
  - une clé *privée* qui sert à déchiffrer (et que vous gardez précieusement secrète)
- Permet d'établir un tunnel sécurisé entre deux machines





*Crypte une clé symétrique de son choix (topsecret) avec la clé publique « 74A48vXX »*

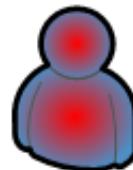


Client

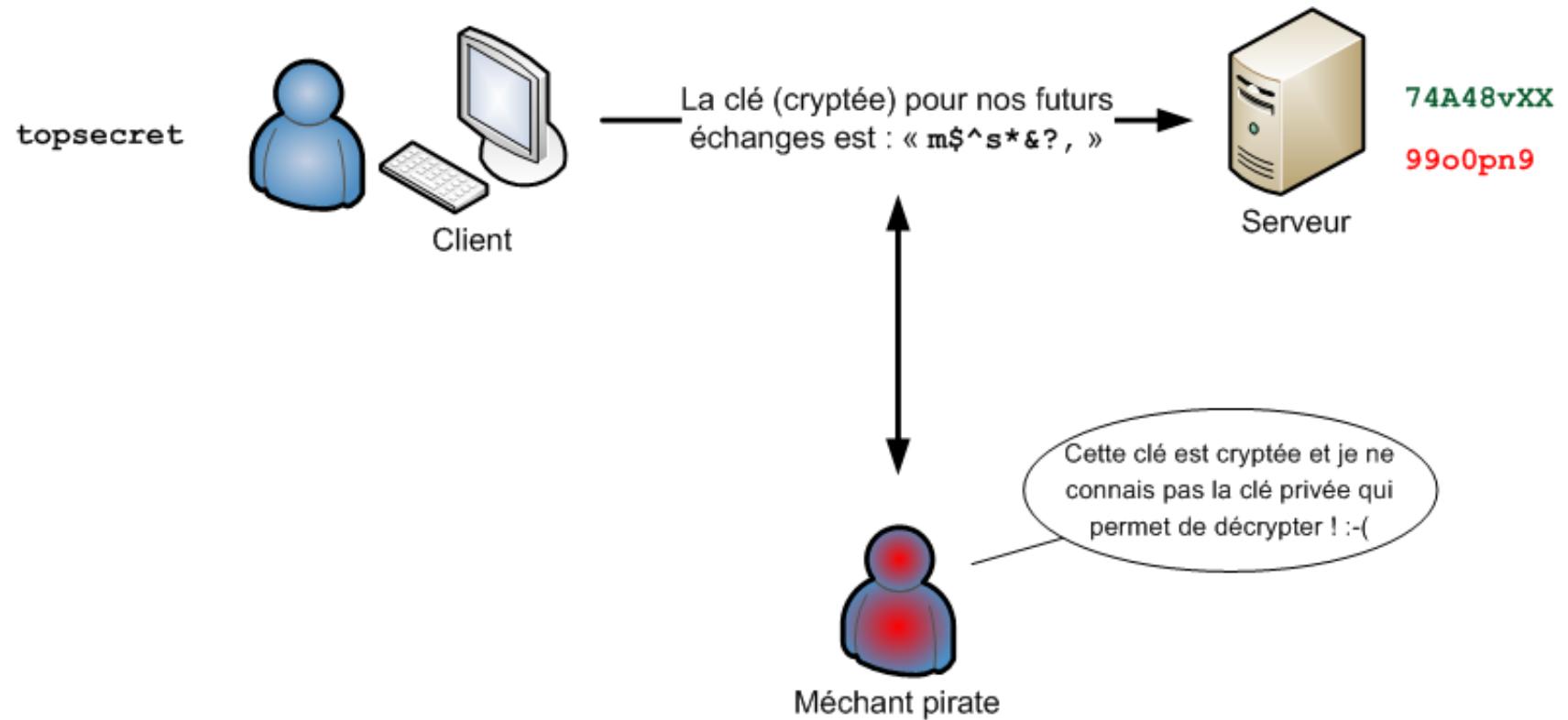


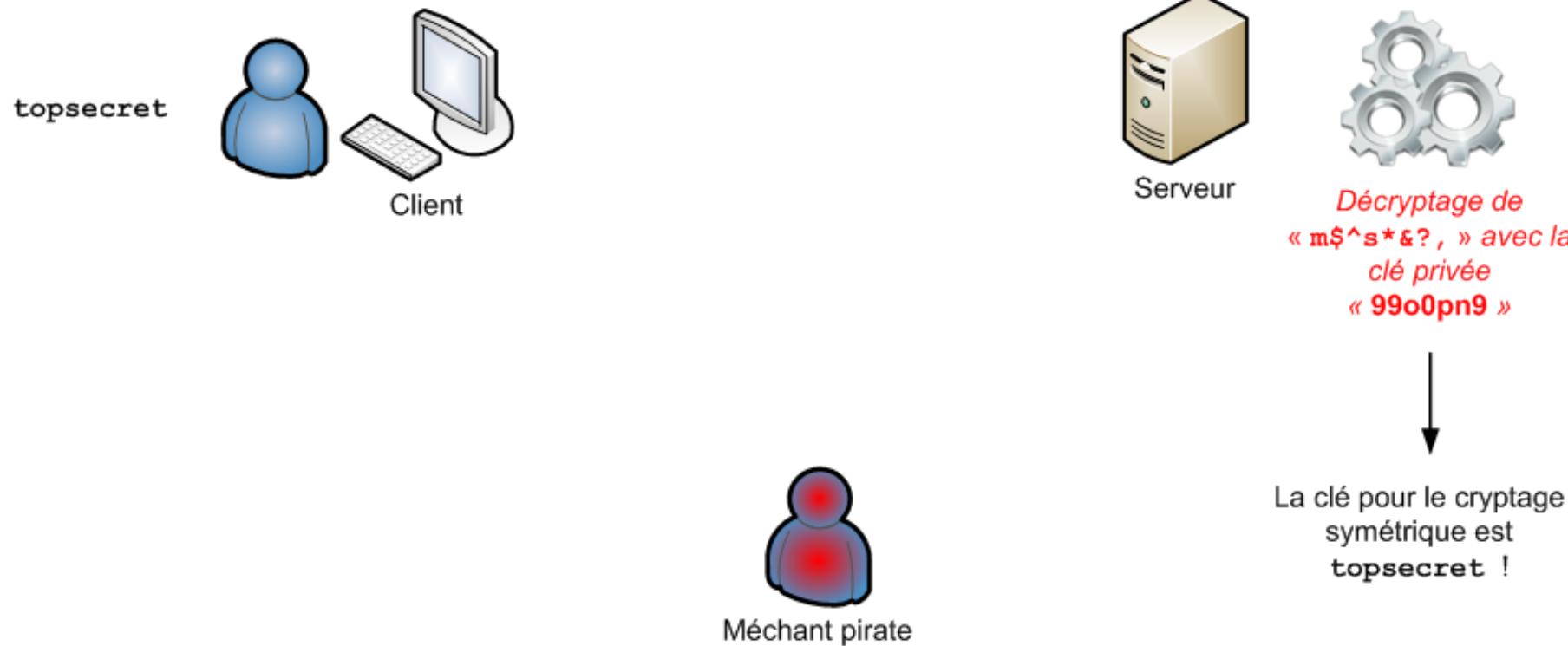
Serveur

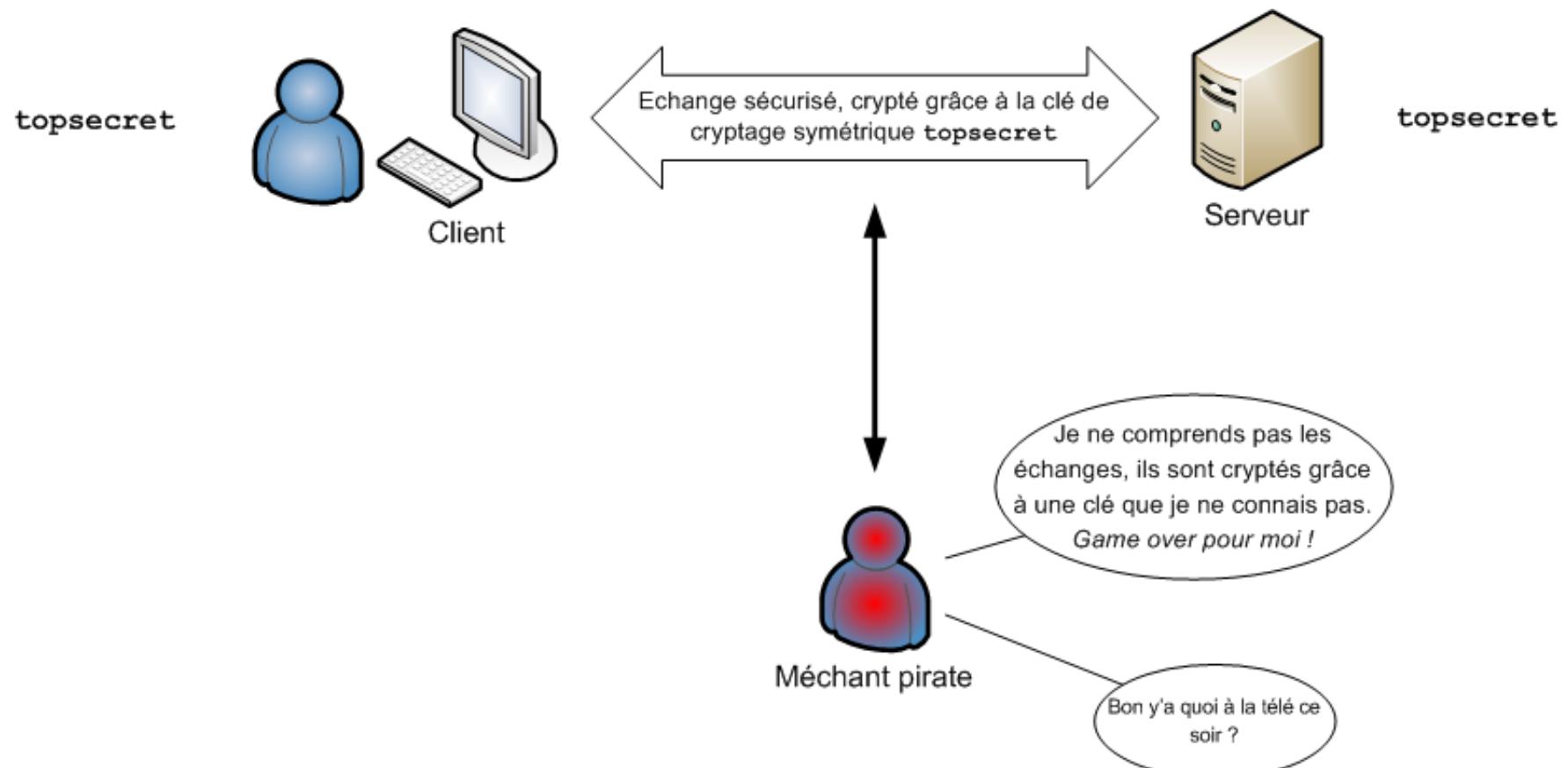
74A48vXX  
99o0pn9



Méchant pirate







# Retours sur les clés SSH

- Protocole pour la sécurisation des transferts de données.
  - Méthode de chiffrement asymétrique qui fonctionne avec une paire de clé :
    - une clé *publique* qui sert à chiffrer (et que vous pouvez partager à l'extérieur)
    - une clé *privée* qui sert à déchiffrer (et que vous gardez précieusement secrète)
  - Permet d'établir un tunnel sécurisé entre deux machines
- 
- On utilise `ssh-keygen -t rsa` pour générer la paire de clé
  - Puis `ssh-copy-id -i id_rsa.pub <login>@<server>` pour envoyer la clé publique
  - Maintenant on peut se connecter de manière sécurisée via `ssh <login>@<server>` sans renseigner de clé
  - Pour GitHub, on renseigne les clés publiques via l'interface graphique

# TP : Utilisation de Git

## 1. Initialiser le dépôt (en local)

```
mkdir testRepo  
cd testRepo  
git init
```

```
vloux@jj-1404-port196 ~/tmp> mkdir testRepo  
vloux@jj-1404-port196 ~/tmp> cd testRepo  
vloux@jj-1404-port196 ~/tmp> git init  
Initialized empty Git repository in /Users/vloux/tmp/testRepo/.git/
```

# TP : Utilisation de Git

## 2. Ajouter un document (en local)

```
echo 'my first line' > firstFile  
git status
```

```
vloux@jj-1404-port196 ~ / t / testRepo > echo 'my first line' > firstFile
```

```
vloux@jj-1404-port196 ~ / t / testRepo > git status  
On branch master
```

No commits yet

Untracked files:  
(use "git add <file>..." to include in what will be committed)  
firstFile

```
nothing added to commit but untracked files present (use "git add" to track)
```

# TP : Utilisation de Git

## 3. Versionner un document (en local)

```
git add firstFile  
git commit firstFile -m "Premier commit"
```

```
git status
```

```
vloux@jj-1404-port196 ~/t/testRepo> git add firstFile  
vloux@jj-1404-port196 ~/t/testRepo> git commit firstFile -m "Premier commit"  
[master (root-commit) 53e1cbc] Premier commit  
 1 file changed, 1 insertion(+)  
  create mode 100644 firstFile  
vloux@jj-1404-port196 ~/t/testRepo> git status  
On branch master  
nothing to commit, working tree clean
```

# TP : Utilisation de Git

## 4. Workflow de modification (en local)

```
echo 'seconde modif' >> firstFile  
  
git status  
git diff  
  
git commit -m "ajout de la deuxième ligne"
```

```
vloux@jj-1404-port196 ~/t/testRepo> git status  
On branch master  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes)  
    modified:   firstFile  
  
vloux@jj-1404-port196 ~/t/testRepo> git diff  
diff --git a/firstFile b/firstFile  
index b1a9f66..d491802 100644  
--- a/firstFile  
+++ b/firstFile  
@@ -1 +1,2 @@  
 my first line  
+second modif  
  
vloux@jj-1404-port196 ~/t/testRepo> git commit  
On branch master  
[master b3a2b4c] ajout de la deuxième ligne
```

# TP : Utilisation de GitHub

## 5. Initialisation du dépôt distant (Github)

- Créer un dépôt distant sur Github (1)

# 5. Initialisation du dépôt distant (Github)

- Créer le dépôt distant sur Github
- Noter l'adresse du dépôt dans l'écran suivant.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner      Repository name \*

 vlux / test ✓

Great repository names are short and memorable. Need inspiration? How about [ideal-guacamole](#)?

Description (optional)

 Public  
Anyone can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾ | Add a license: None ▾ 

**Create repository**

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/vlux/test.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/vlux/test.git
git push -u origin master
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/vlux/test.git
git push -u origin master
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

**Import code**

# TP : Utilisation de GitHub

## 6. Lier le dépôt local et distant

- La commande `git remote` permet de lier notre dépôt à un dépôt distant, ici `test.git`
- Par convention, on appelle ce dépôt `origin`

```
git remote add origin git@github.com:vloux/test.git  
git remote -v
```

```
vloux@jj-1404-port196 ~/t/testRepo> git remote add origin git@github.com:vloux/test.git  
vloux@jj-1404-port196 ~/t/testRepo> git remote -v  
origin    git@github.com:vloux/test.git (fetch)  
origin    git@github.com:vloux/test.git (push)
```

# TP : Utilisation de GitHub

## 7. Pousser les modifications locales sur le dépôt distant

La commande `push` pousse les modifications de la branche `master` (la branche par défaut locale) sur la branche `origin` (le nom de la branche distante). Le paramètre `-u` permet de créer la branche de référence distante et de lier les branches disatntes et locales (et ainsi de se passer des arguments `origin master` par la suite)

```
git push origin master -u
```

```
vloux@jj-1404-port196 ~/t/testRepo> git push origin master -u
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 489 bytes | 489.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To github.com:vloux/test.git
 * [new branch]      master -> master
```

# TP : Utilisation de GitHub

## 8. Vérification sur Github des infos

- Connectez vous sur l'interface de Github :
  - trouvez votre dépôt
  - vérifiez que vous trouvez bien le fichier que vous avez modifié, les commits

# TP : Utilisation de GitHub

## 9. Modification du dépôt par l'interface

- ajouter un `README.md` (avec markdown)
- le commiter depuis l'interface
- l'éditer
- commiter la modification

# TP : Utilisation de Git et GitHub

## Récupérer sur le repo local des infos de la branche distante

git fetch  
git merge

équivalent à :

git pull

```
v loux@jj-1404-port196 ~/t/testRepo> git fetch
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From github.com:v loux/test
      b3a2b4c..c61dab3  master      -> origin/master
v loux@jj-1404-port196 ~/t/testRepo> git merge
Updating b3a2b4c..c61dab3
Fast-forward
  README.md | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 README.md
```

# TP : Utilisation de Git et GitHub

## Nommer des versions (tags)

```
git tag version1 -m "Initial version"
```

```
git push --tags
```

```
vloux@jj-1404-port196 ~/t/testRepo> git tag version1 -m "Initial version"
version1
vloux@jj-1404-port196 ~/t/testRepo> git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 171 bytes | 171.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To github.com:vloux/tesRepository.git
 * [new tag]           version1 -> version1
```

# TP : utilisation de Git et GitHub

Comparer à une version précédente

```
git log
```

```
git diff [CommitNumber] aNewFile
```

```
vloux@jj-1404-port196 ~/t/testRepo> git diff 53e1cbcc5cab26f6c1550acd159ca30ad2acfec
diff --git a/firstFile b/firstFile
index b1a9f66..d491802 100644
--- a/firstFile
+++ b/firstFile
@@ -1 +1,2 @@
 my first line
+second modif
```

# TP : utilisation de Git et GitHub.

## Cloner un dépôt existant

- Dans le cas d'un dépôt distant existant, on peut directement le cloner en local. Il est lié au dépôt distant et les modifications peuvent ainsi être récupérées (`pull`) ou poussées (`push`) sans avoir besoin de d'abord configurer les liens avec le dépôt distant.

```
git clone git@github.com:v loux/test.git
```

```
git remote -v
```

# En résumé

- `git clone` : cloner un dépôt distant
- `git init` : initialiser le versionning sur un dépôt local
- `git commit` : enregister l'état d'un dépôt
- `git status` : afficher l'état des documents du dépôt
- `git diff` : comparer l'état actuel au dernier commit, ou deux commits entre eux ou deux branches

et bien d'autres encore (`blame`, `revert``, ...)

CheatSheet

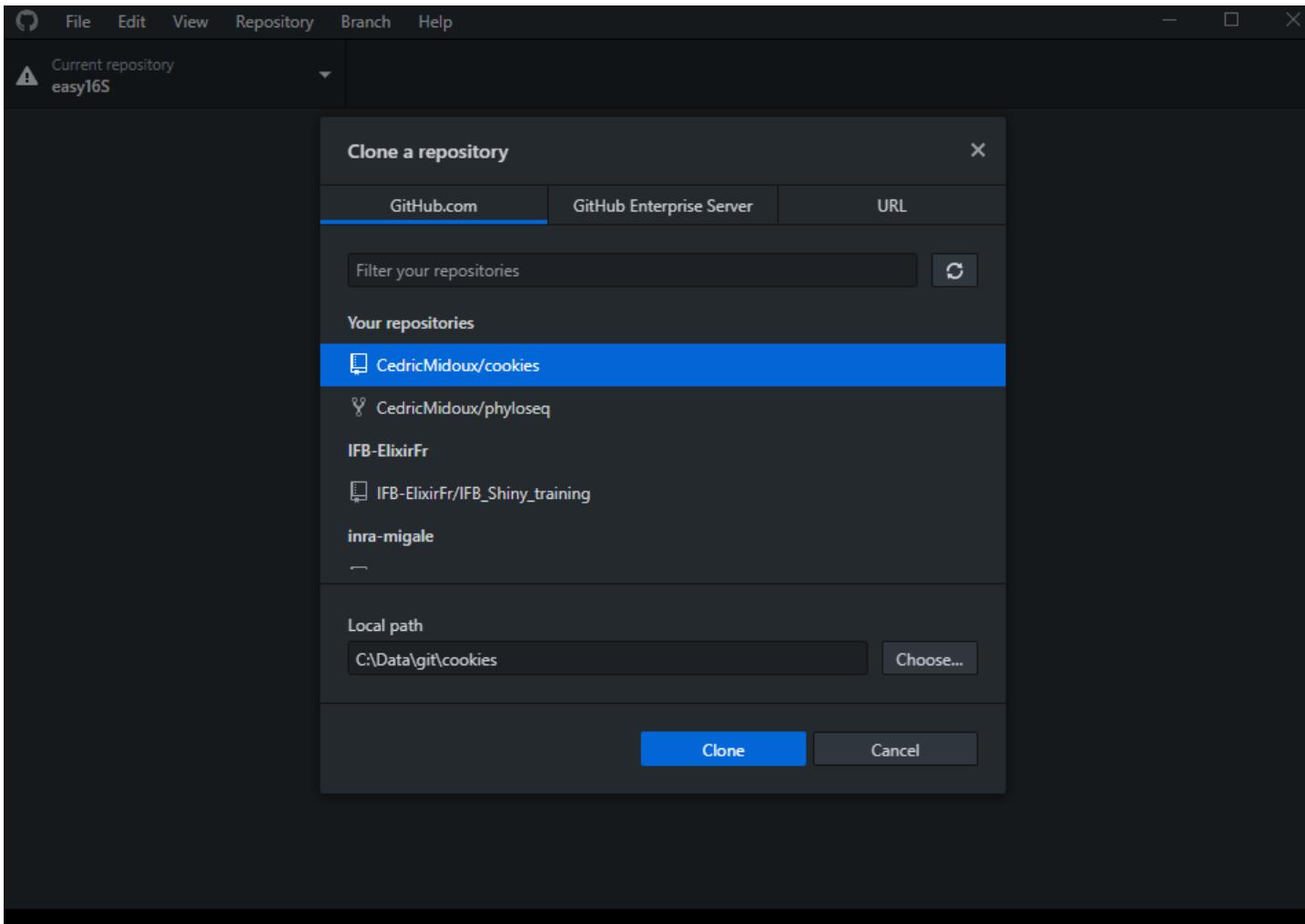
# Pour apprendre progressivement et/ou se simplifier la vie

- Partir de dépôts que l'on crée sur Github et clone en local.

Utiliser une interface conviviale :

- Github Desktop
- Les intégrations aux différents IDE (RStudio and co)





Cloner un dépôt

The screenshot shows a dark-themed Git commit interface. At the top, there are dropdown menus for 'File', 'Edit', 'View', 'Repository', 'Branch', and 'Help'. Below these are three status indicators: 'Current repository: cookies', 'Current branch: master', and 'Fetch origin: Last fetched 2 minutes ago'. The main area has tabs for 'Changes' (1) and 'History'. Under 'Changes', a list shows '1 changed file: README.md'. The content of 'README.md' is displayed in a diff viewer:

```
@@ -4,3 +4,4 @@
 4   4
 5   5
 6   6
 7 +* sucre
```

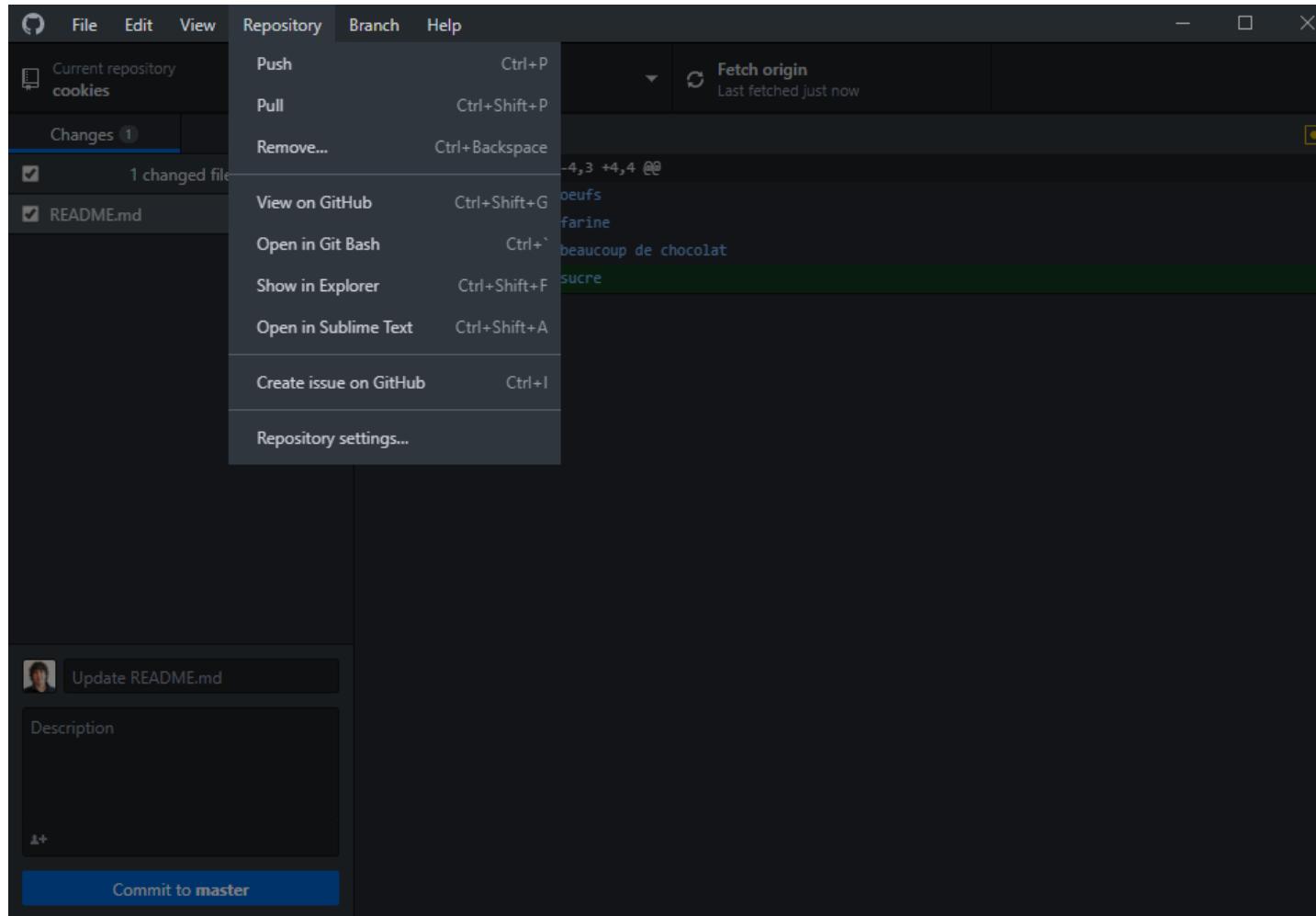
Below the diff viewer, there is a user profile icon and the text 'Update README.md'. A 'Description' field is present with a small '+' icon. At the bottom, a large blue button says 'Commit to master'.

Suivre les changements

The screenshot shows a GitHub repository interface for the 'cookies' repository on the 'master' branch. The 'History' tab is selected. A commit titled 'jolie table' by Cédric Midoux is highlighted. The commit message includes a note about adding the `echo = FALSE` parameter to a code chunk. The diff view shows changes to 'notebook.Rmd' and 'notebook.html'. The 'notebook.Rmd' file has two additions: '-Joli plot' and '+Joli plot !'. The 'notebook.html' file has three additions: '+' (line 31), '+```{r}' (line 36), and '+knitr::kable(head(iris))' (line 37). The commit was made 6 days ago.

File	Line	Change
notebook.Rmd	31	plot(cars)
	32	---
notebook.html	33	33
	34	-Joli plot
	35	+Joli plot !
	36	+
	37	+```{r}
	38	+````
	39	+

Parcourir l'historique des modifications



## Intéragir avec le Repository

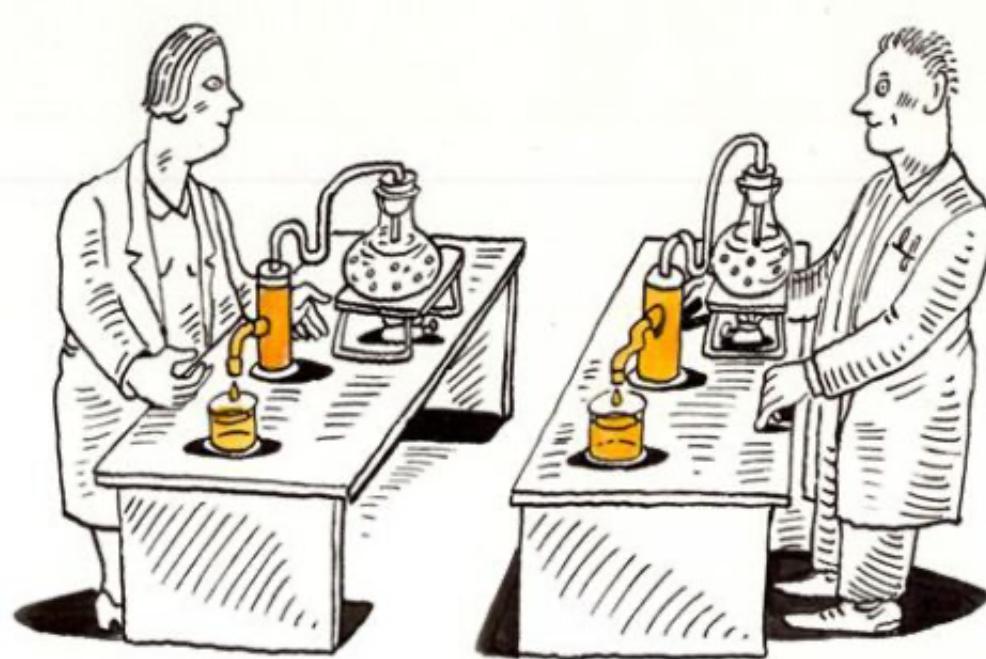


COFFEE  
**BREAK**

# Partie 4 : Documents computationnels - Notebook

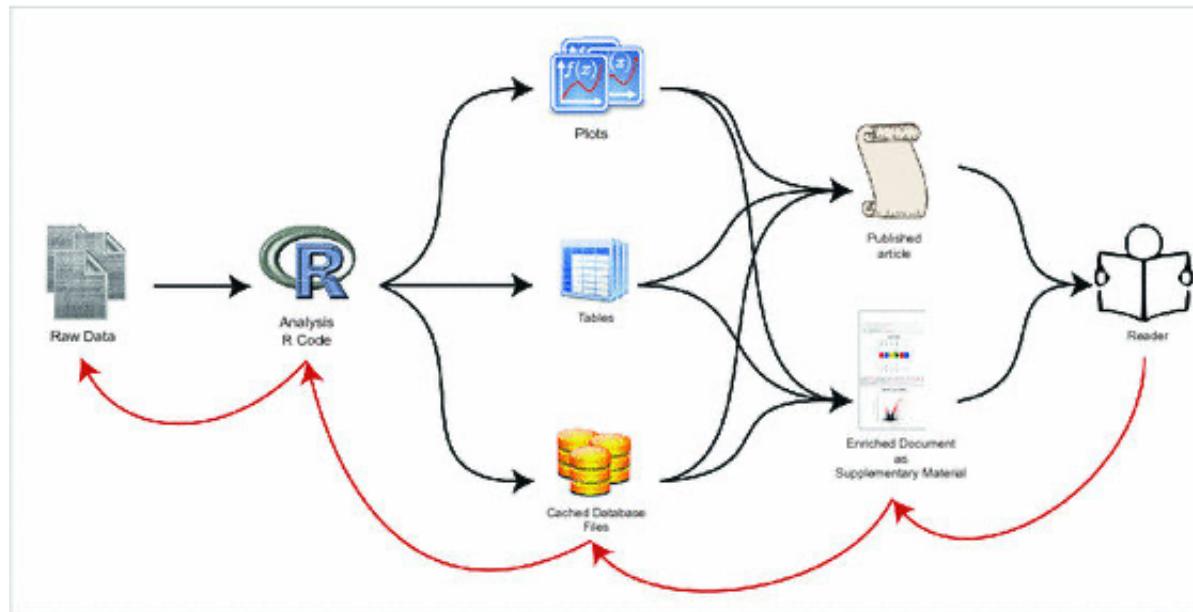
# Partie 4 : Documents computationnels - Notebook

- Il faut se donner les moyens pour qu'autrui puisse inspecter nos analyses
- Expliciter pour augmenter les chances de trouver les erreurs et de les éliminer
  - Inspecter pour justifier et comprendre
  - Refaire pour vérifier, corriger et réutiliser



# Partie 4 : Documents computationnels - Notebook

- Regrouper dans *un unique document*:
  - Les informations, le code, calculs et les résultats
  - Pour assurer leur cohérence et améliorer la traçabilité.
  - Exportable (ex : html) pour une meilleure portabilité et lisibilité.



(Russo, Righelli, and Angelini, 2016)

Encore un joli **mémo** pour R markdown.

# TP : Rédaction d'un notebook avec RStudio

1. Connectez vous au RStudio de l'IFB (<https://rstudio.cluster.france-bioinformatique.fr/>).
2. Clonez le projet :
  - New Project (en haut à droite)
  - Version Control
  - Git
  - Repository URL (clone with SSH : `git@github.com: ...`)
3. Explorez le dépôt depuis ce 3e device remote
4. Créez un document R Markdown :
  - File
  - New file
  - R Markdown

**Dans ce document on a :**

- Une entête générale
- Du texte, mis en forme avec markdown
- Du code R dans des chunks
- Des résultats et outputs

# TP : Rédaction d'un notebook avec RStudio

- Grace à l'onglet git de Rstudio, vous pouvez suivre l'état des fichiers, commit, diff, push/pull, ...
  - Chaque chunk peut être exécuté individuellement grâce à la flèche verte. Les options associées à chaque chunks sont disponible avec la roue crantée.
1. Modifier le document (en plusieurs commit)
    - ajoutez un chunk `knitr::kable(head(iris))` pour visualiser un table
    - ajoutez un chunk `plot(cars)` pour un plot
    - ajoutez des commentaires
    - puis commitez les modifications.
  2. Lorsque vous êtes satisfait de votre rapport, générez la version HTML en cliquant sur Knit. Commitez, pushez, ...
  3. Visualisez les modifications côté GitHub.

**Votre rapport est disponible, versionné et partageable !**

# TP : Partager un document html avec GitHub Pages

- Dans les options du repo, dans le chapitre "GitHub Pages", activer la source correspondant à la branch master.
- la page est disponible à l'adresse [https://\[user\].github.io/\[repo\]/\[page\].html](https://[user].github.io/[repo]/[page].html)

Suivant l'interlocuteur, partagez le `.Rmd` ou le `.html`

# Conclusion :

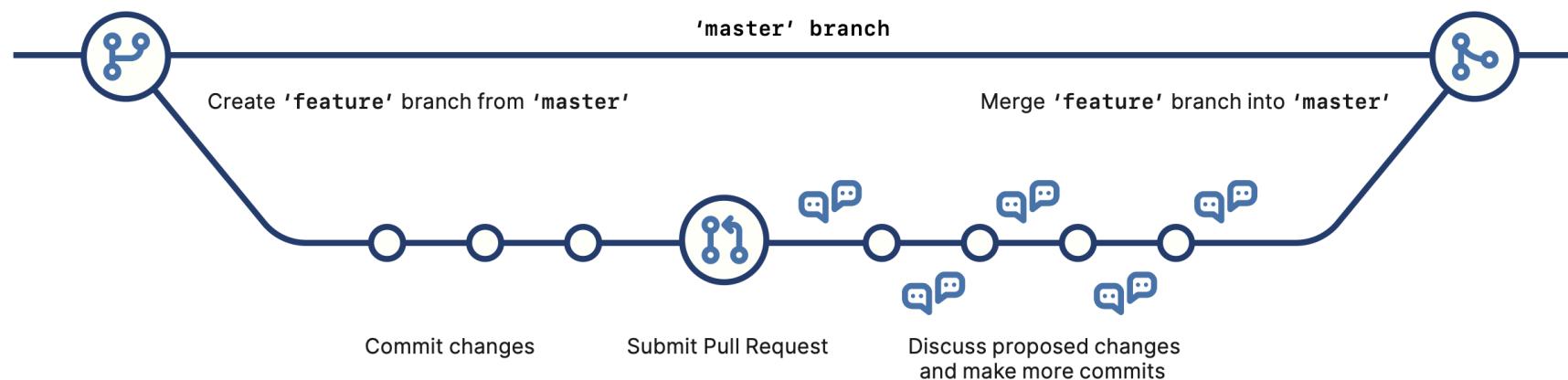
La reproducibilité, comme la "FAIRisation" sont des processus.

Des bonnes pratiques appuyées par des outils qui les facilitent

- Organiser ses analyses
- Décrire correctement ses données et ses processus d'analyse (PGD, FAIR)
- Tracer ses analyses à l'aide de documents computationnels (Rmd, Jupyter Notebooks, ...):
  - Transparents
  - Accessibles
  - Partageables
- Versionner ses documents computationnels (GitHub, GitLab, ...)
  - Traçabilité
  - Accessibilité

# Aller plus loin - travailler en commun avec Git et Github :

## GitHub Flow



- Branche : version parallèle à la version principale
- Pull Request : demande de fusion des modifications d'une branche vers la branche principale

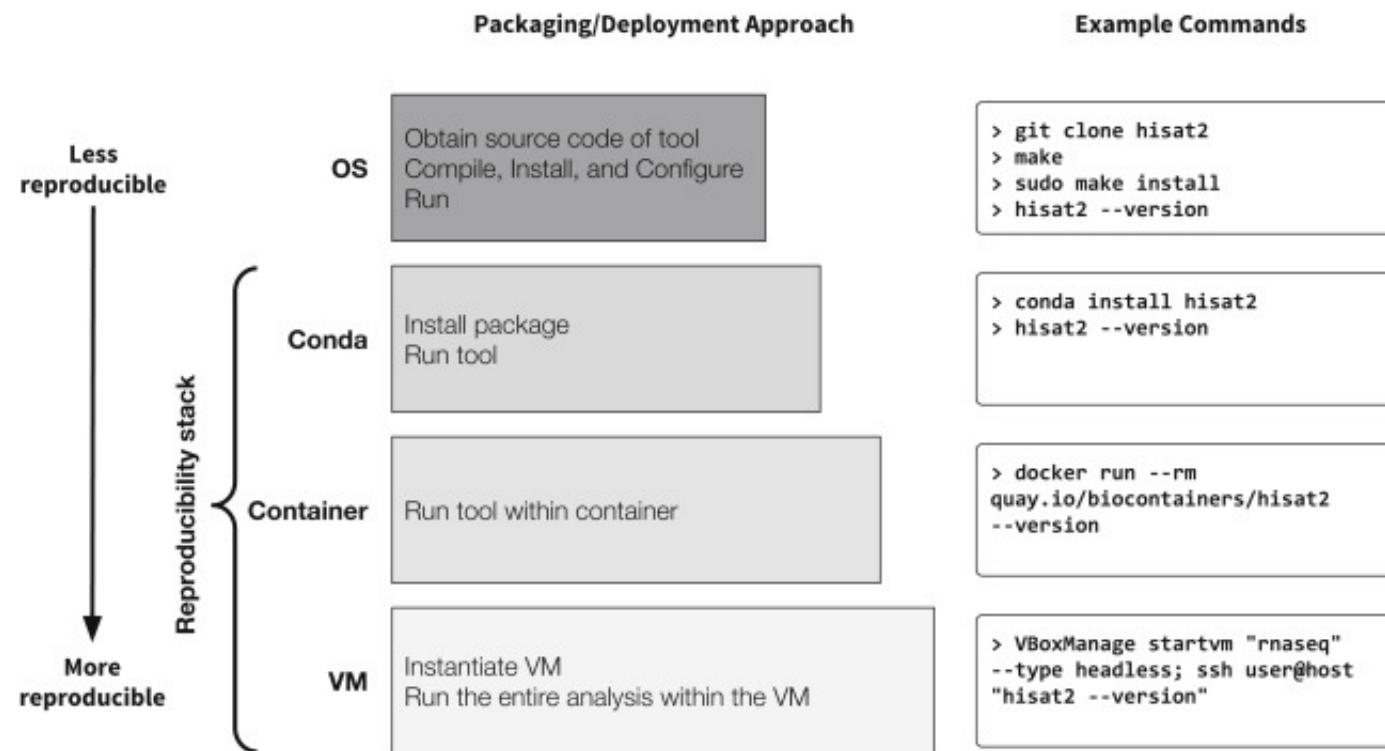
Un exemple de PR

# Pour aller + loin - Fixer et partager son environnement :

- Conda et Bioconda
  - gestion des dépendances, versions
  - Possibilité de créer un environnement par analyse
  - Exporter son environnement dans un fichier `env.yml` et le versionner
  - `conda env export > environment.yml`
- Containers, machines virtuelles
  - Docker, Singularity, VM virtualbox
  - Pour les outils non "condaoisables", les environnements complexes



# Pour aller plus loin - Fixer et partager son environnement (2)



(Grüning, Chilton, Köster, et al., 2018)

# Pour aller + loin - gestionnaires de workflows

SnakeMake, NextFlow pour :

- Définir de façon "simple" et modulaire des workflows d'analyse :
  - Parallelisables : les étapes indépendantes peuvent être jouées en parallèle.
  - Qui assurent la reprise sur erreur : si on refait une analyse, change un paramètre, seul ce qui doit être rejoué est relancé.
  - Portables : un même script peut être joué en local, sur des clusters différents en changeant le fichier de configuration.
  - Partageables : un fichier texte versionné
  - Peut gérer pour vous le versionning et l'installation des outils avec Conda

# Pour aller + loin - exemple de Snakefile

## Bash

```
for sample in `ls *.fastq.gz` do
    fastqc ${sample}
done
```

## Snakefile

```
SAMPLES, = glob_wildcards("./{smp}.fastq.gz")

rule final:
    input:expand("fastqc/{smp}/{smp}_fastqc.zip",smp=SAMPLES)
rule fastqc:
    input: "{smp}.fastq.gz"
    output: "fastqc/{smp}/{smp}_fastqc.zip"
    message: """Quality check"""
    shell: """fastqc {input} --outdir fastqc/{wildcards.smp}"""
```

# Pour aller + loin - FAIRifier ses données

Dépôts dans les dépôts publics :

- Dans les dépôts thématiques internationaux (européens !)
  - données brutes
  - données analysées
  - /!\ méta-données
- dans les dépôts généralistes (dataverse , figshare, ...)
  - fichiers tabulés; "autres" données. ce qu'on mettrait en supplementary material.
  - (éventuels) liens vers les fichiers de données
- Publier un data-paper ?

# Pour aller + loin :

- jusqu'où aller dans la reproductibilité ?
  - Mat et Met électroniques :
  - Galaxy Pages
  - Gigascience, GigaDB :
    - "GigaScience aims to revolutionize publishing by promoting reproducibility of analyses and data dissemination, organization, understanding, and use."

The screenshot shows the Galaxy web interface with the following details:

- Header:** Galaxy, Analyse de données, Workflow, Visualize, Données partagées, Aide, Authentification et Enregistrement.
- Title:** Windshield splatter analysis with the Galaxy metagenomic pipeline: A live supplement
- Authors:** SERGEI KOSAKOVSKY POND<sup>1,2\*</sup>, SAMIR WADHAWAN<sup>1,3†</sup>, FRANCESCA CHIARMONTE<sup>1</sup>, GURUPRASAD ANANDA<sup>1,2‡</sup>, WEN-YU CHUNG<sup>1,2,3‡</sup>, JAMES TAYLOR<sup>1,2</sup>, ANTON NEKRUTENKO<sup>1,2</sup> and THE GALAXY TEAM<sup>1</sup>
- Correspondence:** Correspondence should be addressed to SKP, JT, or AN.
- How to use this document:** This document is a live copy of supplemental materials for the manuscript. It provides access to the exact analyses and workflows discussed in the paper, so you can play with them by re-running, changing parameters, or even applying them to your own data. Specifically, we provide the two histories and one workflow found below. You can view these items by clicking on their name to expand them. You can also import these items into your workspace and start using them; click on the green plus to import an item. To import workflows you must create a Galaxy account (unless you already have one) – a hassle-free procedure where you are only asked for a username and password.
- Galaxy History:** Galaxy History | Galaxy vs MEGAN, Comparison of Galaxy vs. MEGAN pipeline.
- Galaxy Workflow:** Galaxy Workflow | metagenomic analysis, Galaxy Workflow | metagenomic analysis (Generic workflow for performing a metagenomic analysis on NGS data).
- Accessing the Data:** Windshield Splatter datasets analyzed in this manuscript can be accessed through this Galaxy Library. From there they can be re-analyzed through Galaxy using the above workflows or downloaded.
- Supplemental Analysis:** Comparison between Galaxy pipeline and Megan

Live Mat et Met  
<https://usegalaxy.org/u/aun1/p/windshield-splatter>

Au final, toujours se poser la question du rapport coût / bénéfice.

# Ressources

- FUN MOOC Recherche Reproductible
- FAIR Bioinfo
- Cours Git et Github
- Github pages
- Rmd the definitive Guide
- Snakemake
- NextFlow et nf-core
- 
- Les mémo présentés dans ce cours :
  - markdown
  - git
  - R markdown

# References

- Allard, A. (2018). *La crise de la réplicabilité*. URL: <https://laviedesidees.fr/La-crise-de-la-replicabilite.html>.
- Baker, M. (2016). "1, 500 scientists lift the lid on reproducibility". In: *Nature* 533.7604, pp. 452-454. DOI: [10.1038/533452a](https://doi.org/10.1038/533452a). URL: <https://doi.org/10.1038/533452a>.
- Grüning, B, J. Chilton, J. Köster, et al. (2018). "Practical Computational Reproducibility in the Life Sciences". In: *Cell Systems* 6.6, pp. 631-635. DOI: [10.1016/j.cels.2018.03.014](https://doi.org/10.1016/j.cels.2018.03.014). URL: <https://doi.org/10.1016/j.cels.2018.03.014>.
- Noble, W. S. (2009). "A Quick Guide to Organizing Computational Biology Projects". In: *PLOS Computational Biology* 5.7, pp. 1-5. DOI: [10.1371/journal.pcbi.1000424](https://doi.org/10.1371/journal.pcbi.1000424). URL: <https://doi.org/10.1371/journal.pcbi.1000424>.
- Piazzì, A. C, A. S. Cerqueira, L. R. Manso, et al. (2018). "Reproducible research platform for electric power quality algorithms" , pp. 1-6.
- Russo, F., D. Righelli, and C. Angelini (2016). "Advantages and Limits in the Adoption of Reproducible Research and R-Tools for the Analysis of Omic Data". In: Vol. 9874. , pp. 245-258. DOI: [10.1007/978-3-319-44332-4\\_19](https://doi.org/10.1007/978-3-319-44332-4_19).
- Wilkinson, M. D, M. Dumontier, I. J. Aalbersberg, et al. (2016). "The FAIR Guiding