

# Contrôle de Source

Eric Marcon

16 février 2024

# Principes

git est un gestionnaire de sources :

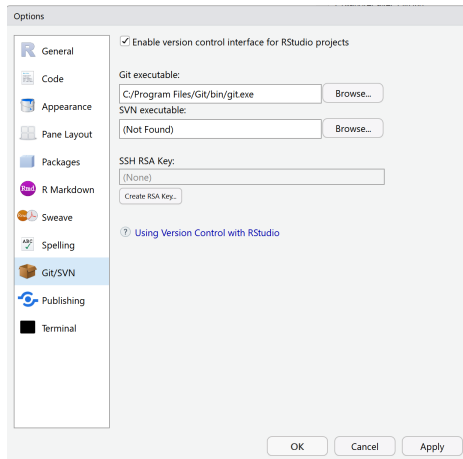
- Suivi des modifications : bien plus qu'une sauvegarde !
- Possibilité de revenir en arrière ;
- Plusieurs versions concurrentes : branches.

GitHub est une plateforme pour la collaboration basée sur Git.

- Possibilité de développer à plusieurs : la fin des pièces jointes !
- Présentation des résultats : pages GitHub ;
- Intégration continue: GitHub Actions.

git

## RStudio doit détecter Git



Sinon, l'installer.

Les fichiers modifiés sont dans la fenêtre Git de RStudio.

Utiliser `.gitignore` pour masquer les fichiers non suivis.

- Créer un fichier R avec une ligne de code et l'enregistrer.

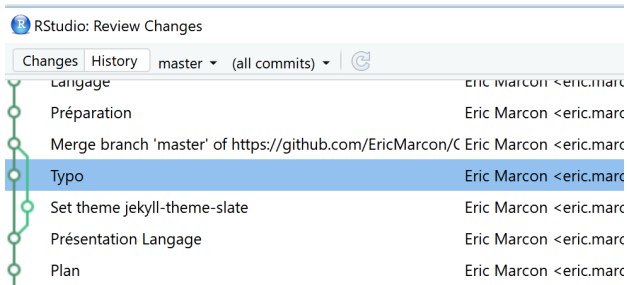
Après chaque séance de travail, livrer le résultat (*Commit*)

Sélectionner les fichiers à livrer.

Saisir un message clair : résumé sur la première ligne.

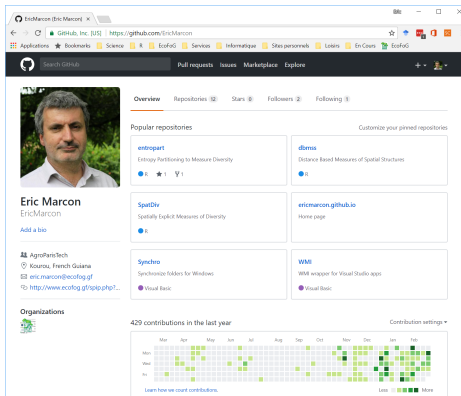


## Icône en forme d'horloge dans la fenêtre Git



# GitHub

## Ouvrir un compte sur GitHub



The screenshot shows the GitHub profile page for Eric Marcon. The browser address bar displays the URL <https://github.com/EricMarcon>. The profile header includes a profile picture of a man with grey hair and a beard, the name **Eric Marcon**, and the username `EricMarcon`. Below the name is a link to [Add a bio](#). The profile is associated with the organization **AgroParisTech**, located in Kourou, French Guiana, with email [eric.marcon@ecofog.gf](mailto:eric.marcon@ecofog.gf) and website <http://www.ecofog.gf/sp.php?>. The page shows 429 contributions in the last year, represented by a calendar heatmap. The 'Popular repositories' section lists four repositories: **entropart** (Entropy Partitioning to Measure Diversity), **dismas** (Distance Based Measures of Spatial Structures), **SpatDiv** (Spatially Explicit Measures of Diversity), and **ericmarcon.github.io** (Home page). The 'Synchro' repository is also listed with the description 'Synchronize folders for Windows' and 'Visual Basic'. The 'WMI' repository is described as 'WMI wrapper for Visual Studio apps' and 'Visual Basic'. The 'Contributions' section shows a heatmap for the last year, with a legend indicating 'Less' and 'More' contributions.

Créer un jeton d'accès personnel :

- Paramètres du compte utilisateur ;
- Developer Settings > Personal Access Tokens > Tokens (classic) ;
- Générer un jeton, le décrire en tant que “git-RStudio” et lui donner l'autorisation “repo”.

Sauvegarder le jeton ! Il sera demandé à la première tentative de pousser des modifications sur GitHub.

La branche principale s'appelle historiquement master mais **récemment** main par défaut.

Choisir :

- Paramètres du compte utilisateur ;
- Repositories : Repository default branch ;

La branche s'appelle “master” dans toute cette présentation.

Dans le terminal de RStudio:

```
git config --global init.defaultBranch master
```

## A partir de rien :

- Dans GitHub :
  - *New Repository*
  - Choisir le nom (pas de caractères spéciaux)
  - Ne rien ajouter : le projet doit être vide.
- Copier l'URL à partir de *Clone or Download*
- Dans RStudio : nouveau projet à partir de Git, coller l'URL.

A partir d'un projet RStudio existant :

- Passer le projet sous contrôle de version :
  - *Tools /Version Control /Project Setup...*
  - Sélectionner *Git*.
- Créer un dépôt sur GitHub, récupérer son URL :  
<https://github.com/MonCompte/MonDepot.git>
- Dans le Terminal de RStudio, exécuter :

```
git remote add origin https://github.com/MonCompte/MonDepot.git
git branch -M master
git push -u origin master
```

# Pratique



A chaque nouvelle séance de travail :

- Tirer (*Pull*) pour récupérer les modifications sur GitHub.

Pendant la séance de travail, après chaque tâche élémentaire :

- Livrer (*Commit*) les modifications.

A la fin de chaque nouvelle séance de travail :

- Pousser (*Push*) ses modifications vers GitHub pour les rendre publiques.

Déclarer un collaborateur.

Travailler à deux ou plus sur le même fichier.

Contenu d'une séance de travail :

- Tirer ;
- Modifier ;
- Livrer ;
- Pousser.

L'information élémentaire est la ligne.

Modifications contradictoires = conflit.

Minimiser les conflits : dans un texte, une phrase = une ligne.

En cas de conflit, trancher.

Ajouter un fichier README.md

Prendre en compte, valider et pousser.

Editer la même ligne de README.md :

- en ligne sur GitHub ;
- localement.

Tirer, constater le conflit, le résoudre.

Le voisin de gauche invite celui de droite sur GitHub.

Les deux modifient le projet.

- Bien penser à tirer avant de modifier.
- Pousser rapidement pour limiter les conflits.

# Branches

Modifier le projet sans perturber son état stable.

Application :

- développer une nouvelle fonctionnalité ;
- la tester, corriger les bugs ;
- la rendre visible quand elle est terminée.



## Cliquer sur New Branch

Travailler dans la nouvelle branche (pull, commit, push).

Se placer dans la branche master.

Exécuter:

```
git merge branche_a_fusionner
```

# Fork et Pull Request

Objectif : modifier le dépôt d'un autre

**Fourcher**: création d'une copie du dépôt

Commencer une nouvelle branche, la modifier.

Demande d'intégration de la branche: *Pull Request*.

Sur GitHub.

Dialogue possible.

Si acceptation, fusion de la branche et suppression.

## Contrôle de Source

Eric Marcon

Principes

git

GitHub

Pratique

Branches

Fork et Pull  
Request