

Initiation Git par la pratique

2024

IDGEO



Table des matières

Objectifs	3
I - GIT Introduction	4
II - Précisions et contexte	6
III - Fonctionnement	7
IV - Les commandes	8
V - Le principe général	9
VI - Les outils	10
VII - GIT-GUI	11
VIII - Github	15

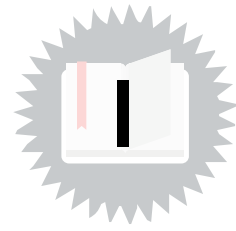
Objectifs



À l'issue de cette formation, les participants seront capables :

- de créer et gérer des repositories Git,
- de comprendre et d'utiliser les fonctionnalités essentielles de Git (commit, branches, merges, etc.) tout en collaborant via GitHub.
- de réaliser un repository complet avec des commits organisés,
- de l'utiliser pour une gestion de projet ou du travail en équipe.

GIT Introduction



GIT est un logiciel de gestion de versions décentralisé.

C'est un logiciel libre créé par Linus Torvalds (une page d'histoire de l'informatique).



git

Petit sondage ?



Remarque

Vous connaissez certainement les sites

Github ? <https://github.com/>



GitHub

Gitlab ? <https://about.gitlab.com/>



GitLab

ou Framgit ? <https://framagit.org/public/projects>



Précisions et contexte



Histoire

- <https://fr.wikipedia.org/wiki/Git>
- 2005 => création par Linus Torvalds
- GIT est Open Source Sous Licence GPL-GNU v2
- La solution de versionnement le plus populaire.
- 93% des développeurs



Définition

GIT permet de suivre les modifications de son code ou de ses documents, qu'on travaille **seul** ou **en groupe**.



Méthode

Vocabulaire

- **GIT** est un logiciel.
- **Github** est une plateforme web qui permet de travailler avec git, comme **Gitlab** ou **Framagit**.
- Un **repository** est un répertoire (un par projet à priori).
- Un répertoire peut être **distant** ou **local** (sur sa machine).



Fondamental

Précisions

Qu'est-ce qu'un gestionnaire de version ?

Comparaison ou somme de contrôle avec la fonction de hachage SHA-1 (générateur de nombres pseudo-aléatoires)

Si un fichier n'est pas modifié, il n'est stocké qu'une fois.



Complément

Avantages

Si on travaille **seul sur son code**, ça permet de conserver l'historique de ses propres modifications pour revenir dessus.

Si on travaille **en équipe**, ça fusionne les modifications des personnes qui travaillent en même temps. On trouve rapidement qui fait quoi et on peut revenir en arrière.

Fonctionnement



Deux structures de données

=> des **objets** et un **cache de répertoires**

Les 4 types d'objets

? *Exemple*

> **blob** (binary large object) = contenu d'un fichier

> **tree** (arbre) = arborescence de fichiers, contient de listes de blobs, contient des détails comme le nom, les permissions

> **commit** (valider une transaction) = arborescence de fichiers enrichie de métadonnées

> **tag** (étiquette) = manière de nommer un commit spécifique

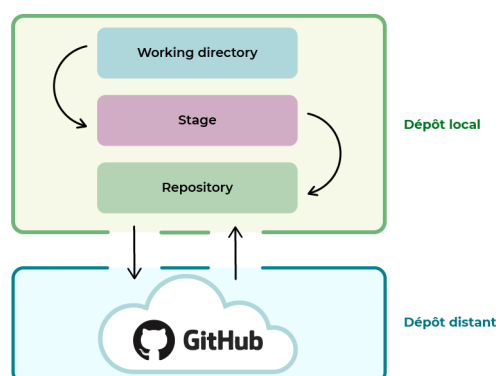
Tous les objets sont identifiés par une somme de contrôle SHA-1

La notion de dépôt

💡 *Fondamental*

On distingue donc un **dépôt local** d'un **dépôt distant**.

Un dépôt local est un entrepôt virtuel. Il faut activer cet entrepôt sinon votre projet reste un dossier classique.



1 Git init

Le dépôt distant est localisé sur une machine distante. Il devient indispensable dès qu'on travaille à plusieurs. Il faut aussi faire un init au départ, mais on va surtout le cloner sur sa machine en local.

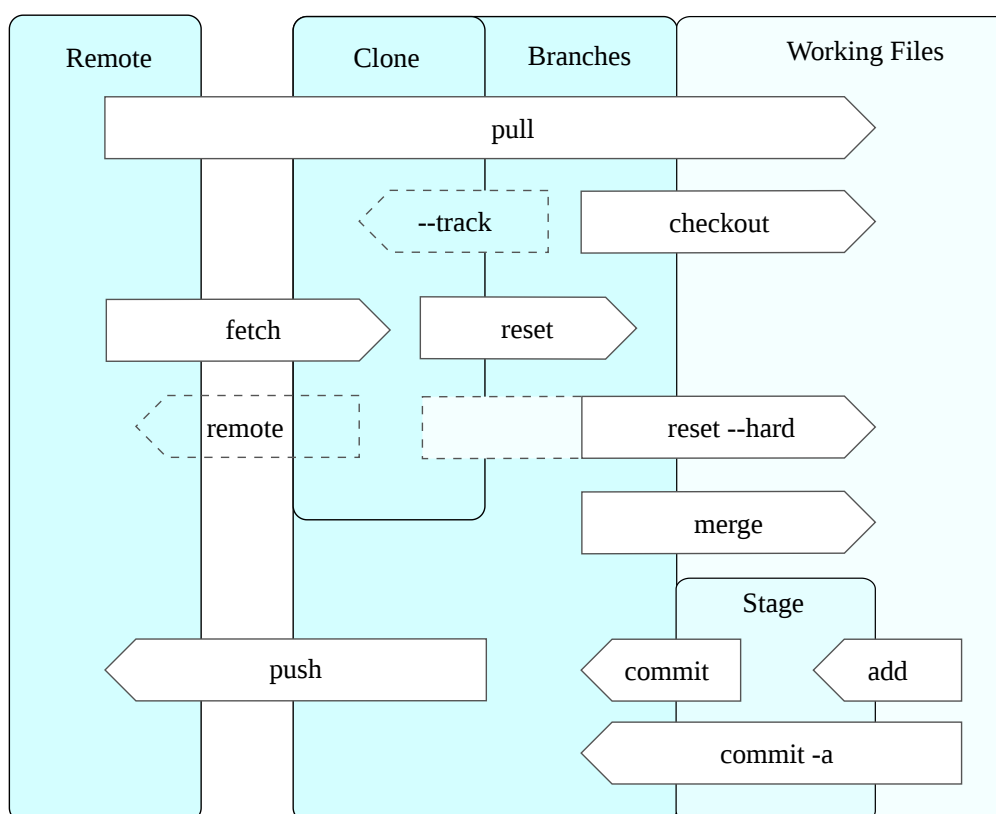
1 Git clone

Les commandes



Description des commandes courantes

On en parle, il faut comprendre ces différentes commandes pour intégrer les principes de fonctionnement de GIT.



Le principe général



Le Dépôt local

- **Le Working Directory (le dossier de travail)**

Notre dossier de travail sur notre ordinateur.

- **Le stage (index)**

Zone intermédiaire qui présente les éléments modifier entre le WD et le repo.

- **Le repository**

Stockage des nouvelles versions.

=> ces trois zones sont sur notre ordinateur

Le dépôt distant

Repository Github

Comment ça marche ?



Définition

Dans notre dossier de travail, nous avons 3 fichiers, si nous en modifions 2, nous avons une version évoluée.

Ces deux fichiers apparaissent donc dans la zone d'index.

Une fois indexé les deux fichiers apparaissent dans la zone du repo.

Les outils



Plateformes

Les plateformes web de Github ou GitLab permettent d'exploiter à fond les capacités de GIT.

- tout est visuel
- tout est push bouton
- des fonctionnalités avancées comme la gestion des paramètres poussés
- la gestion des issues
- la gestion des Pull Request
- possibilité de faire un wiki



Ces plateformes s'installent sur son propre serveur.

GIT sur son PC (multi-plateforme)

Plusieurs possibilités à partir du moment où le logiciel est installé.

<https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-Installation-de-Git>

Mode CLI



A ce stade, GIT s'utilise en ligne de commande.

Logiciels bureautique



Il y en a beaucoup.

Solutions standalone ou intégrée.

Github offre aussi un github bureautique.

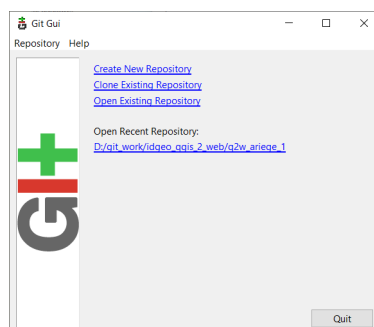
GIT-GUI

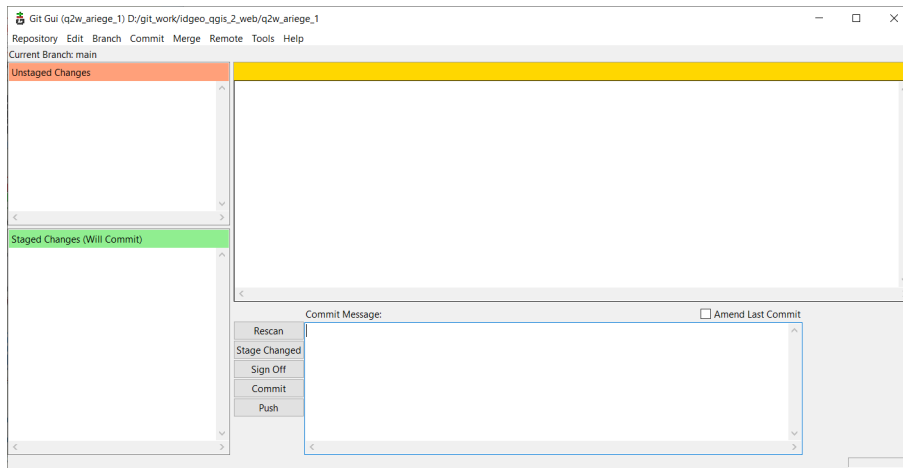


Un exemple d'outil bureautique.

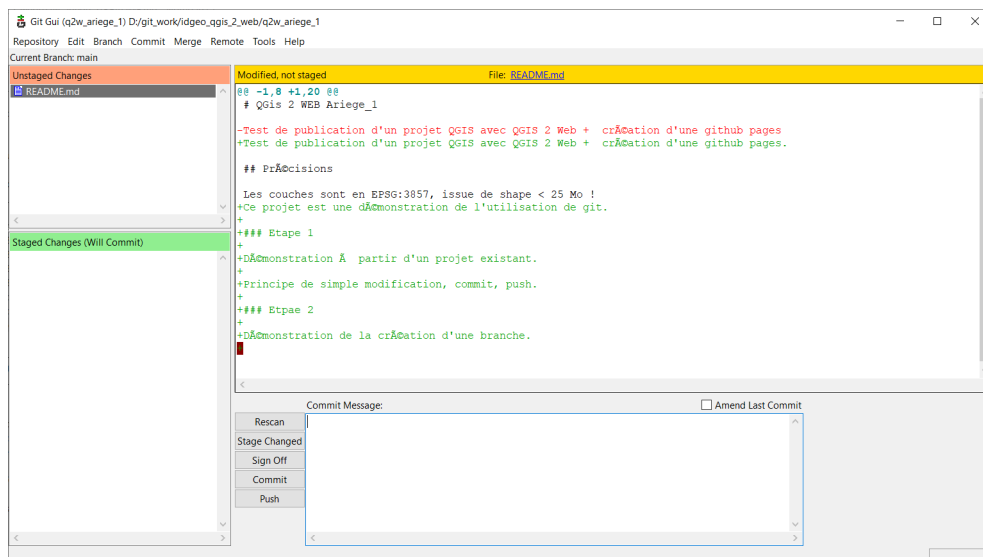


Tout se fait au clic

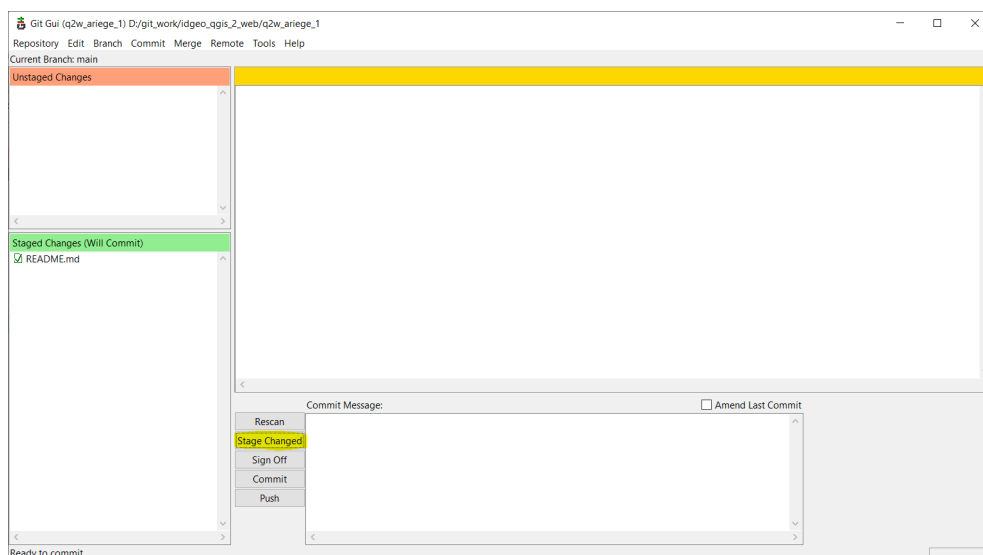




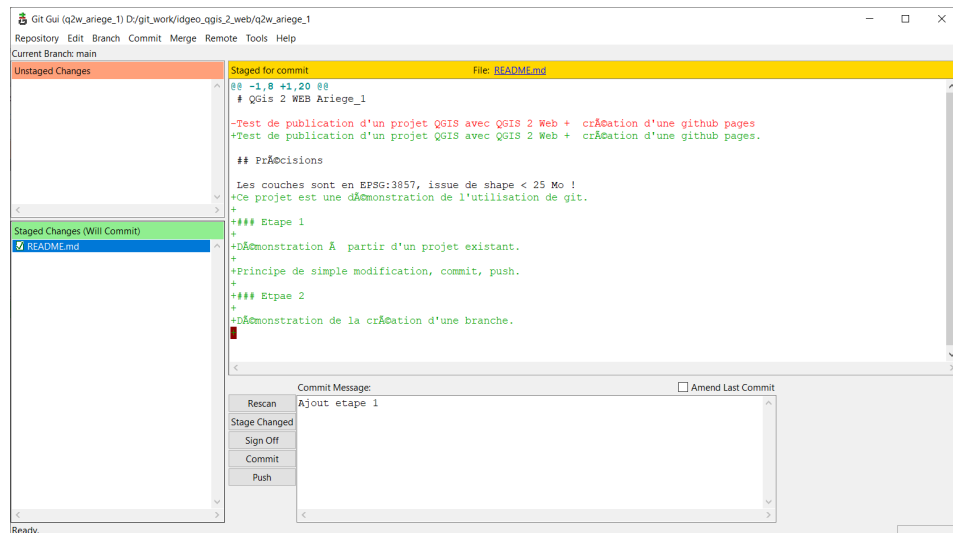
Un petite modification,



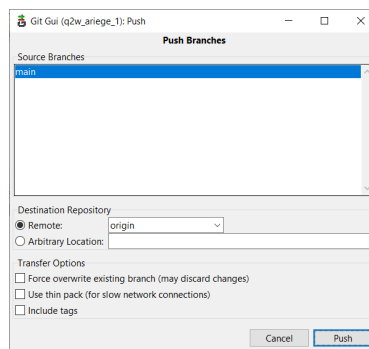
zone « stage » (index de la modification)



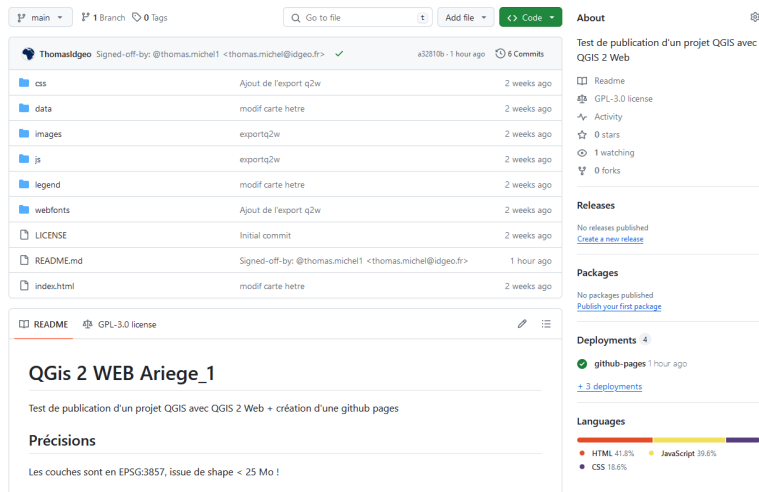
commit



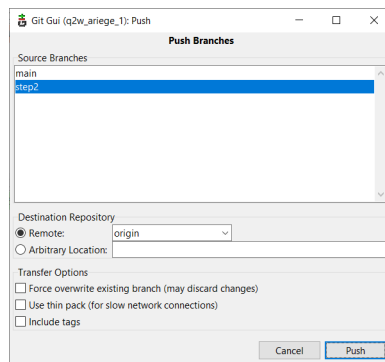
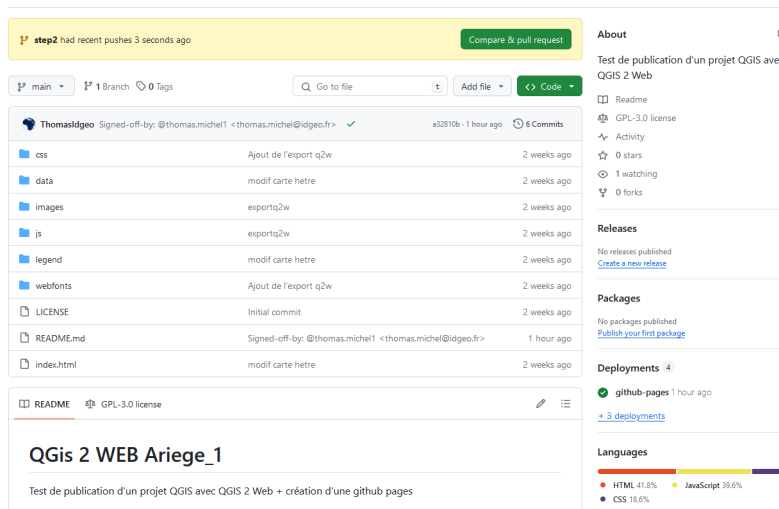
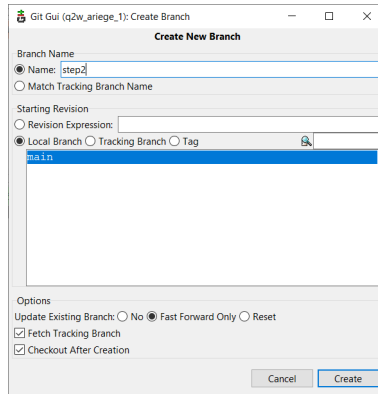
Pour valider le tout on « **push** »



On peut vérifier avant / après le push pour voir comment que ça fonctionne.



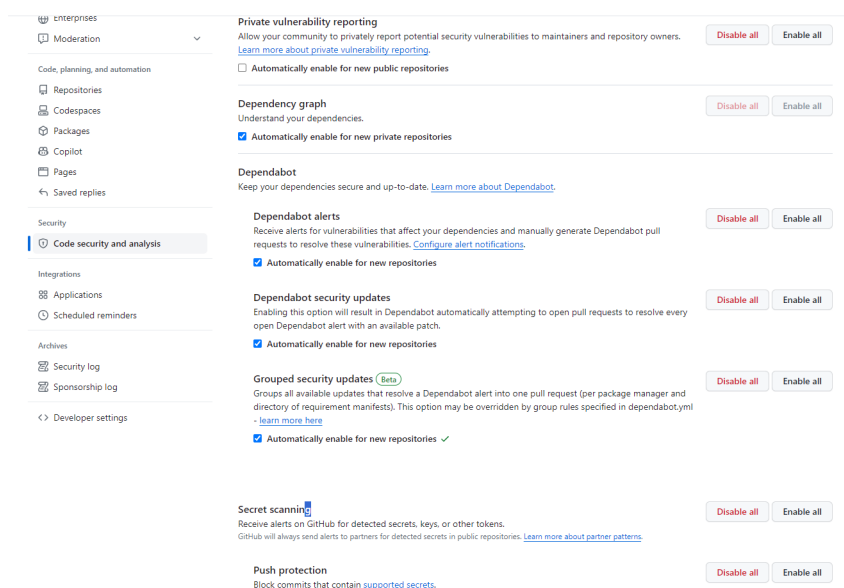
Création d'une branche



Inscription et création d'un compte gratuit sur Github.

Paramétrage des aspects de sécurités. (Bonne pratique)

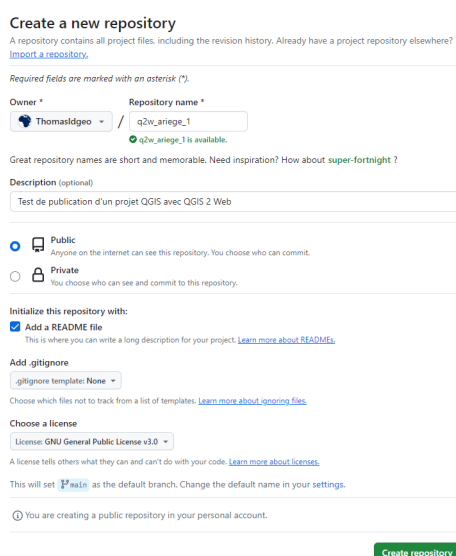
Ce réglage permet d'activer des outils qui détecteraient d'éventuels données sensibles dans notre code. Ce réglage permet à tout nos repository d'activer ces options à la création.



The screenshot shows the 'Code security and analysis' settings for a repository. The left sidebar lists various settings categories. The main content area shows several security-related settings, each with a 'Disable all' and 'Enable all' button.

- Private vulnerability reporting:** Allow your community to privately report potential security vulnerabilities to maintainers and repository owners. [Learn more about private vulnerability reporting.](#) ☐ Automatically enable for new public repositories.
- Dependency graph:** Understand your dependencies.
 - ☒ Automatically enable for new private repositories
- Dependabot:** Keep your dependencies secure and up-to-date. [Learn more about Dependabot.](#)
 - Dependabot alerts:** Receive alerts for vulnerabilities that affect your dependencies and manually generate Dependabot pull requests to resolve these vulnerabilities. [Configure alert notifications.](#)
 - ☒ Automatically enable for new repositories
 - Dependabot security updates:** Enabling this option will result in Dependabot automatically attempting to open pull requests to resolve every open Dependabot alert with an available patch.
 - ☒ Automatically enable for new repositories
 - Grouped security updates (Beta):** Groups all available updates that resolve a Dependabot alert into one pull request (per package manager and directory of requirement manifests). This option may be overridden by group rules specified in dependabot.yml - [learn more here](#).
 - ☒ Automatically enable for new repositories ✓
- Secret scanning:** Receive alerts on GitHub for detected secrets, keys, or other tokens. [Learn more about partner patterns.](#)
- Push protection:** Block commits that contain [supported secrets](#).

Création d'un premier repository sur Github.



The screenshot shows the 'Create a new repository' form. It includes fields for 'Owner', 'Repository name', 'Description (optional)', 'Visibility' (Public/Private), 'Initialize this repository with' (Add a README file), 'Add .gitignore', 'Choose a license', and 'This will set' (main). A green 'Create repository' button is at the bottom right.

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

Required fields are marked with an asterisk (*).

Owner * / Repository name *
✓ q2w_ariege_1 is available.

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

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
-gitignore template:

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set [main](#) as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.



- Langage de balise léger, facile à lire et facile à écrire.

<https://docs.framasoft.org/fr/grav/markdown.html>

- « Utiliser *Markdown* ne doit pas être considéré comme une complication supplémentaire mais fait au contraire partie d'un mouvement *Low-Tech*. »

<https://e-publish.uliege.be/md/chapter/markdown/>