

RAPPORT TECHNIQUE



Nom du Projet : Versionado

Personne à contacter dans le Projet :

- Duquenne Melwin
- Cassagne Bertrand
- Doueil Hugo

[1-Présentation de l'équipe](#)

[2-Contexte et Définition](#)

[3-Objectif du Projet](#)

[4-Description du fonctionnement](#)

[5-Ressources du Projet](#)

[Git](#)

[Tuto GIT pour une prise en main rapide!](#)

[Introduction](#)

[Qu'est-ce que GIT?](#)

[Étape 1 – Installation de GIT sur différents systèmes](#)

[Option 1 – Installation de GIT sous Windows:](#)

[Option 2 – Installation de GIT sur MacOS:](#)

[Option 3 – Installation de GIT sous Linux:](#)

[Étape 2 – Utilisation de GIT](#)

[Conclusion](#)

[GitHub](#)

[Exploration de L'interface GitHub](#)

[Un exemple de fork de code](#)

[Atlassian Bitbucket](#)

[Tuto BitBucket](#)

[GitLab](#)

[Tuto GitLab](#)

[6-Enveloppe budgétaire](#)

[GitHub:](#)

[Version gratuite:](#)

[Version team](#)

[version entreprise](#)

[Bitbucket:](#)

[version free:](#)

[version Standard](#)

[version Premium](#)

[GitLab:](#)

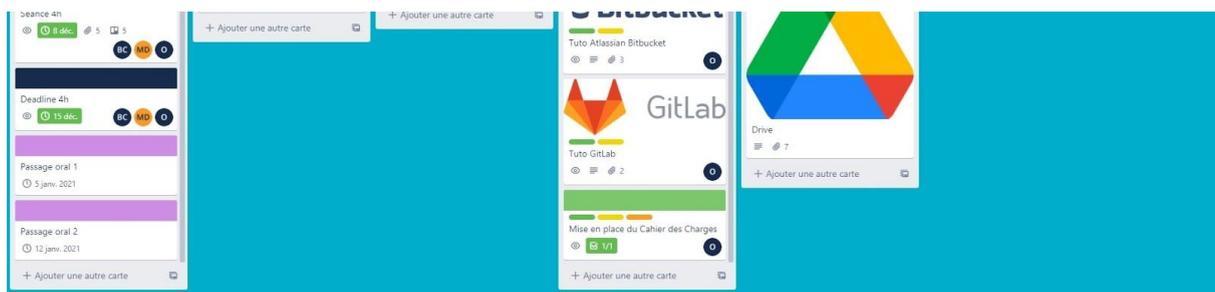
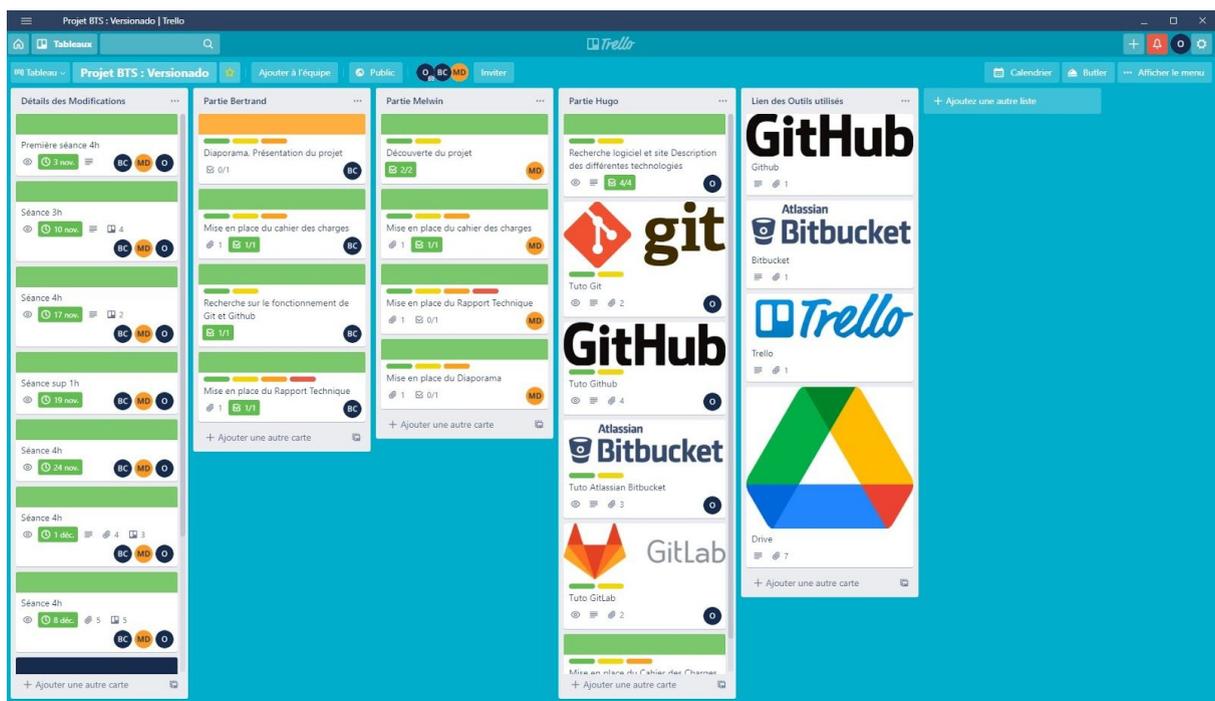
[7-Solution adopter](#)

[8-Source](#)

1-Présentation de l'équipe

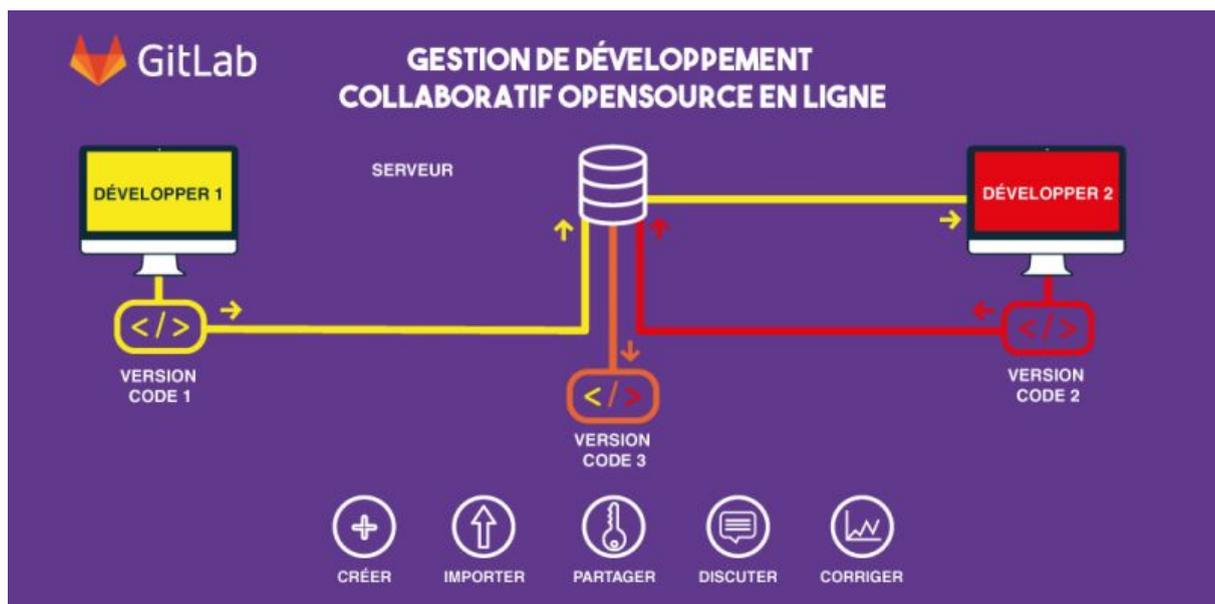
Le Chef de projet est CASSAGNE BERTRAND, les autres participants du projet Versioning sont DOUEIL Hugo et DUQUENNE Melwin.

Les répartitions des tâches de chaque participants au projet sont détaillés sur le trello :



2-Contexte et Définition

Le versioning consiste à travailler directement sur le code source du projet, en gardant toutes les versions précédentes. Les outils du versioning aident les développeurs à travailler parallèlement sur différentes parties du projet et à revenir facilement aux étapes précédentes de leur travail en cas de besoin. L'utilisation d'un logiciel de versioning est devenue quasi-indispensable pour tout développeur, même s'il travaille seul car non seulement ça leur permet de garder toutes les versions mais aussi ça leur permet d'avoir un gain de temps dans leur travail et potentiellement améliorer la qualité du travail, cela permet d'apporter un certain confort pour les développeur.



3-Objectif du Projet

Trouver un système de versioning pour cinq utilisateurs permettant à la société "VosRêves" de réaliser ses projets actuels et futurs de manière ergonomique.

4-Description du fonctionnement

Le versioning permet, grâce à un système de clone (copie du projet sur un réseau local ou sur le cloud), d'avancer simultanément sur différentes tâches.

Il permet également d'archiver un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. De cette façon, il n'y a aucun risque d'altérer ou de perdre les données déjà enregistrées sur le dépôt commun.

Certains logiciels anticipent même les conflits : ils avertissent l'utilisateur qu'un autre développeur travaille en même temps que lui sur la même ligne de code et évitent ainsi que les deux travaux se parasitent.

Le versioning est utilisé tout au long du cycle de vie d'un logiciel, le long des grandes étapes que sont la maquette, le prototype, la version alpha, la version bêta, la release candidate (potentiel mises à jour) et la version finale.

la méthode sémantique:

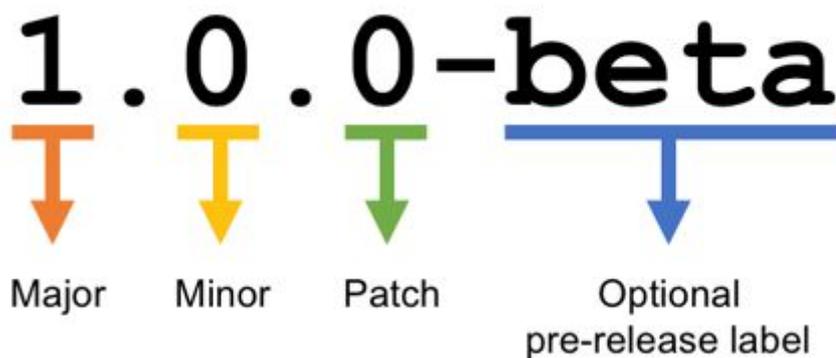
Le versioning sémantique, une méthode très utilisée pour l'attribution des numéros de version.

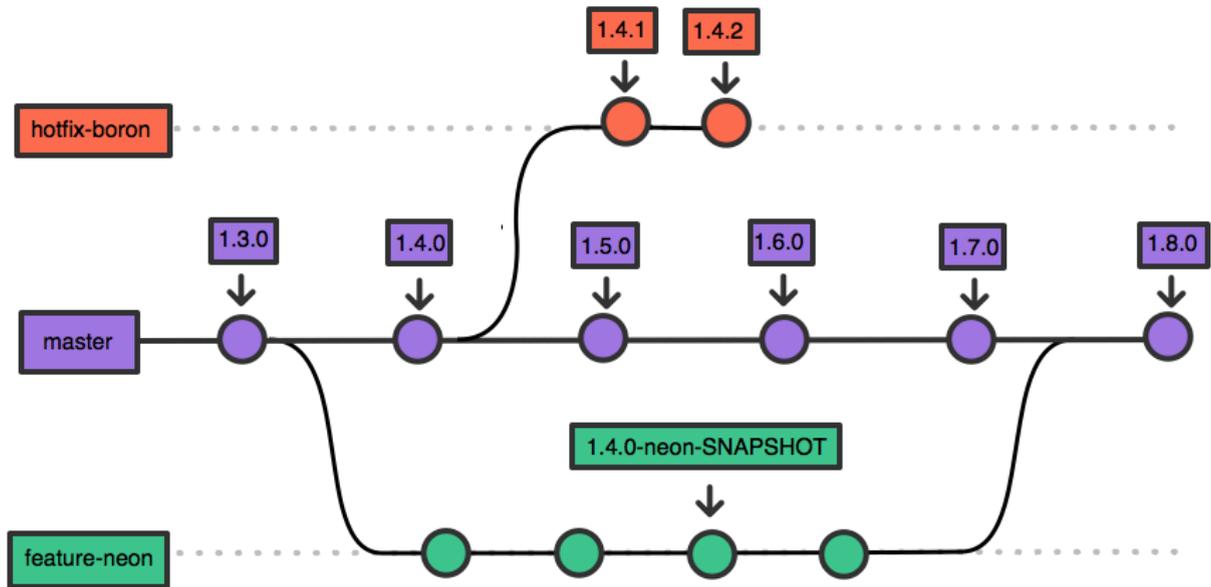
Les modifications apportées au logiciel sont communément rassemblées et désignées par **des numéros de version sous la forme "x.y.z"**. On formule ainsi des degrés d'importance de ces changements de gauche à droite, du plus significatif au moindre correctif :

X - Majeur : changements non rétro-compatibles, suppression d'une fonctionnalité obsolète, modification d'interfaces, renommages...

Y - Mineur : changements rétro-compatibles, introduction de nouvelles fonctionnalités, fonctionnalité marquée comme obsolète...

Z - Correctif : corrections d'anomalies rétro-compatibles, modification/correction d'un comportement interne, failles de sécurité...





Par exemple, le logiciel Windows 8 de Microsoft correspondait en version sémantique à la version 6.2.9200.

D'autres nomenclatures sont utilisées pour la gestion de version. Certaines entreprises optent pour une nomenclature qui cible le grand public, séparant les aspects marketing et technique.

*nomenclature : désigne une instance de classification.

5-Ressources du Projet

On utilisera pour ce projet un ensemble de ressources disponibles sur le web :

Git



Tuto pour les nuls Git :

https://www.youtube.com/watch?v=gp_k0UVOYMw

Tuto GIT pour une prise en main rapide!

Introduction

Les systèmes de contrôle de version aident les développeurs à analyser plus facilement les modifications et les contributions apportées à du code collaboratif. Un VCS est un élément essentiel d'un système de gestion de développement logiciel. Les modifications / révisions / mises à jour effectuées sont identifiables avec des lettres ou des chiffres. Des informations comme la date de modification et l'identité du modificateur sont également renseignées. Dans ce tutoriel, nous vous apprendrons

à utiliser un des système de gestion les plus connus : GIT. Vous apprendrez à l'installer et à l'utiliser.

Qu'est-ce que GIT?

En 2005, Linus Torvalds (l'homme connu pour la création du noyau Linux OS) a développé GIT et depuis, il est activement maintenu par Junio Hamano, un ingénieur logiciel japonais. Aujourd'hui, GIT est l'un des systèmes de contrôle de version open source les plus célèbres et des millions de projets à travers le monde s'appuient sur GIT pour la gestion de ceux ci (cela inclut des projets commerciaux et open-source). GIT est un logiciel entièrement gratuit et peut être téléchargé pour Mac, Linux, Windows et Solaris depuis le [site officiel](#). Certaines des caractéristiques de GIT:

- Un système de contrôle de version distribué, GIT a une approche **peer to peer** contrairement à d'autres tels que Subversion (SVN) qui optent pour l'approche **client-serveur**.
- GIT permet aux développeurs d'avoir une pléthore de branches de code indépendantes. La création, la suppression et la fusion de ces branches est transparente et prend peu de temps.
- Dans GIT, toutes les opérations sont atomiques; Cela signifie qu'une action peut soit réussir soit échouer (sans aucune altération). Cela est important parce que dans certains systèmes de contrôle de version (comme CVS) où les opérations sont non-atomiques, si une opération dans le dépôt est laissée en suspens, elle peut laisser le dépôt dans un état instable.
- Dans GIT, tout est stocké dans le dossier `.git`. Ce n'est pas la même chose dans d'autres VCS comme SVN et CVS où les métadonnées des fichiers sont stockées dans des dossiers cachés (par exemple `.cvs`, `.svn`, etc.)
- GIT utilise un modèle de données qui aide à garantir l'intégrité cryptographique de tout ce qui est présent dans un dépôt. Chaque fois qu'un fichier est ajouté ou qu'une validation est effectuée, des sums de contrôle sont générés. Aussi, ils sont récupérés via leurs sums de contrôle.
- Une autre excellente caractéristique présente dans GIT est sa **zone de classement** ou **index**. Dans l'index, les

développeurs peuvent formater les modifications et les faire réviser avant de les appliquer réellement.

GIT est très simple à utiliser. Pour commencer, vous pouvez soit créer un dépôt soit en rejoindre un. Après l'installation, un simple `git-init` mettra tout ce qu'il faut en place. `git clone` vous permettra de créer une copie du dépôt pour le modifier en local.

Étape 1 – Installation de GIT sur différents systèmes

Voici les manières les plus simples d'installer GIT:

Option 1 – Installation de GIT sous Windows:

L'installation de GIT sous Windows OS est devenue aussi simple que de télécharger un programme d'installation et de l'exécuter. Procédez comme suit pour configurer GIT sur une machine Windows:

- Visitez ce [site Web](#) et téléchargez l'installateur GIT pour Windows.
- Une fois téléchargé, double-cliquez sur l'exécutable pour lancer l'assistant d'installation. Il suffit de suivre les instructions à l'écran, de continuer à cliquer sur **Suivant** et enfin **Terminer** pour terminer l'installation.



- Ouvrez une invite de commande. Dans celles-ci, entrez les commandes suivantes:

```
git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"
```
- Note: N'oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations. Toute modification faite ultérieurement sera associée à ces détails.

C'est tout ce qu'il vous faut pour installer GIT sur Windows !

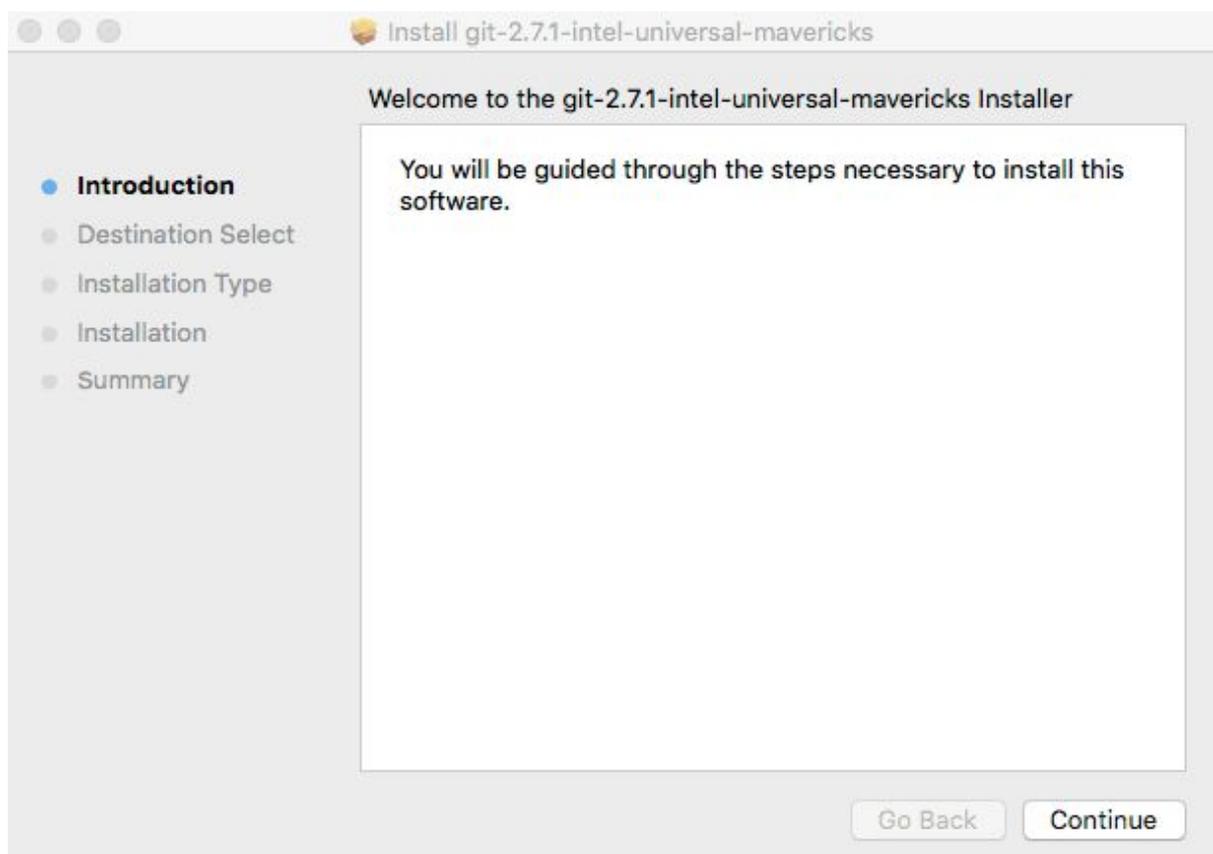
Option 2 – Installation de GIT sur MacOS:

Il existe plusieurs façons d'installer GIT sur un périphérique Mac. Il y a une chance que GIT soit déjà présent sur votre ordinateur si vous avez XCode d'installé. Exécutez la commande suivante sur un terminal pour vérifier:

```
git --version
```

Si vous obtenez une sortie comme git version 2.7.0 (Apple Git-66) , alors vous avez GIT d'installé. Mais si vous n'avez rien à l'écran, effectuez les étapes suivantes:

- Visitez ce site Web et téléchargez le dernier programme d'installation pour Mac.
- Suivez les instructions à l'écran et terminez l'installation.



- Essayez de nouveau la commande `git --version` afin de confirmer que l'installation a réussi.
- Exécutez les commandes suivantes sur un terminal pour configurer votre mail et nom d'utilisateur qui doivent être associés à votre compte GIT:

```
git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"
```
- Note: N'oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations.

Toute modification faite ultérieurement sera associée à ces détails.

Option 3 – Installation de GIT sous Linux:

Si vous êtes un utilisateur Linux, vous devriez être habitué à installer des logiciels et des paquets sur votre ordinateur avec des commandes d'installation telles que apt-get ou yum install. L'installation de GIT est similaire:

Pour les utilisateurs Debian / Ubuntu (apt-get):

- Ouvrez un terminal et exécutez les commandes suivantes:

```
sudo apt-get update Sudo apt-get install git
```

-
- Vérifiez que vous avez bien installé GIT avec `git -version` .
- Exécutez les commandes suivantes sur un terminal afin de configurer l'email et le nom d'utilisateur qui doivent être associés à votre compte GIT:

```
git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"
```

- Note: N'oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations. Toute modification faite ultérieurement sera associée à ces détails.

Étape 2 – Utilisation de GIT

Maintenant que GIT est configuré sur votre périphérique Windows / Mac / Linux, nous allons explorer les bases de GIT. Et comment vous pouvez commencer à utiliser GIT.

- Création / configuration / extraction d'un dépôt:

Le dépôt est la chose la plus importante d'un projet. Pour transformer n'importe quel dossier en un dépôt GIT, on peut

utiliser la commande simple `git init <dossier>` . Un dossier nommé `.git` sera créé quand la commande aura été exécutée.

Inversement, si vous avez déjà un dossier et que vous souhaitez le cloner, vous pouvez utiliser la *commande* `git clone` . Si vous essayez de consulter un dépôt local, utilisez la commande suivante:

```
git clone /chemin/vers/le/dossier/local
```

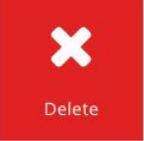
Si vous avez l'intention de vérifier un dépôt stocké à distance, utilisez

```
git clone utilisateur@hote:/chemin/vers/le/dépôt
```

Si vous avez un compte Hostinger, vous pouvez facilement cloner et gérer des dépôts via la zone Membres -> GIT . Par exemple, si vous voulez cloner un dépôt GIT, entrez simplement son adresse, choisissez une branche, choisissez un chemin et cliquez sur le bouton Créer.

Create a New Repository	
Repository Address	<input type="text" value="https://github.com/laravel/laravel"/> example: https://github.com/WordPress/WordPress.git
Branch	<input type="text" value="master"/> Usually this is the master branch
Install path	<input type="text" value="laravel"/> Leave blank to deploy directly to public_html. Directory must be empty.
<input type="button" value="✔ Create"/>	

Une fois la création terminée, vous pourrez gérer votre dépôt dans la même section.

Manage Repositories					
10	Search..				
	Status ▲	Repository	Branch	Install Path	Actions
☰	Success	https://github.com/laravel/laravel.git	master	laravel	Manage
		 Deploy	 View latest build output	 Delete	

- Le Workflow:

Maintenant qu'un dépôt a été mis en place, parlons de la structure de GIT. Chaque dépôt local se compose de trois "arbres" : le dossier de travail qui contient les fichiers actuels; l'index qui joue le rôle d'une zone de transit et le HEAD qui est un pointeur vers le dernier commit effectué par l'utilisateur. Voici le workflow : l'utilisateur ajoute un fichier ou des modifications à l'index (la zone de déploiement) et une fois revues, le fichier ou les modifications sont finalement confiées au HEAD .

- Les commandes add et commit:

Les modifications proposées ou les ajouts de fichiers sont ajoutés à l'index à l'aide de la commande add. Pour ajouter n'importe quel fichier, la commande est:

```
git add <nom_fichier>
```

Si vous êtes assez confiant pour effectuer ces changements dans votre HEAD , vous pouvez utiliser la commande commit:

```
git commit -m "Message pour décrire le commit"
```

Remarque: Une fois la commande `commit` exécutée (à partir du dossier de travail), le fichier est affecté au HEAD , mais il n'est toujours pas envoyé au dépôt distant.

- Pousser les changements plus loin:

Une fois que vous avez validé les modifications (et pensez qu'elles sont prêtes à être envoyées au dépôt d'origine), vous pouvez utiliser la commande `push`.

Une fois que `git push origin master` est exécuté à partir du dossier de travail, les changements présents dans HEAD sont envoyés au dépôt distant. Dans la commande mentionnée ci-dessus, le master peut être changé par le nom de la branche à laquelle vous souhaitez que les modifications soient effectuées.

Cependant, si un dépôt existant n'a pas encore été cloné et que vous souhaitez établir une connexion entre votre dépôt et un serveur distant, procédez comme suit:

```
git remote add origin <serveur>
```

Remarque: Remplacez `<serveur>` par l'adresse du serveur distant.

Une fois cloné, les modifications effectuées seront envoyées au serveur concerné.

- Branches:

Une autre caractéristique brillante (et avancée) de GIT est la possibilité de créer plusieurs branches indépendantes au sein d'un même projet. Le but principal d'une branche est de développer des fonctionnalités tout en les maintenant isolées l'une de l'autre. La branche par défaut dans tout projet est

toujours la branche master. On peut créer autant de branches que nécessaire et finalement fusionner avec la branche master.

Une nouvelle branche peut être créée à l'aide de la commande suivante:

```
git checkout -b fonctionnalité_n *
```

fonctionnalité_n est le nom de la branche

Si vous désirez revenir à la branche master principale, vous pouvez utiliser la commande suivante:

```
git checkout master
```

Toute branche peut être supprimée en utilisant la commande suivante:

```
git checkout -b fonctionnalité_n
```

Pour rendre la branche disponible aux autres utilisateurs, vous devrez la push vers le dépôt distant; Pour ce faire, utilisez la commande suivante:

```
git push origin fonctionnalité_n
```

- Mise à jour et fusion:

Dans le cas où vous souhaitez mettre à jour votre dossier de travail local à jour, la commande `git pull` peut être utilisée.

Pour fusionner (merge) une autre branche dans celle actuellement active, utilisez: `git merge fonctionnalité_n` .

Avec push ou merge, GIT essaie toujours de gérer les conflits lui-même, mais parfois il ne peut pas. En cas d'échec en raison de conflits, l'utilisateur doit résoudre les conflits manuellement. Une fois que vous avez édité les fichiers (pour éliminer les conflits), marquez-les comme fusionnés en utilisant:

```
git add <nom.fichier>
```

Si avant de fusionner vous souhaitez afficher les modifications, la commande suivante peut être exécutée:

```
git diff <nom_de_branche_source> <nom_de_branche_cible>
```

- Étiquetage:

Avant de faire des mises à jour/modifications, il vous est recommandé de créer des étiquettes. Pour ce faire sur GIT, utilisez la commande suivante:

```
git tag 1.1.0 1c2d2d56fa
```

Le 1c2d2d56fa dans la commande ci-dessus se réfère aux 10 premiers caractères du commit-id référencé avec la balise. L'ID de validation peut être trouvé à partir du journal.

- Log:

L'historique du dépôt peut être étudié avec le log. La commande `git log` le récupère. Pour récupérer les commits effectués par un seul utilisateur, vous pouvez utiliser:

```
git log --author =Smith
```

Une version compressée du journal (un commit par ligne) peut être visualisée en utilisant:

```
git log --pretty=oneline
```

Pour afficher uniquement les fichiers qui ont été modifiés:

```
git log --nom-statut
```

- Remplacement des modifications locales:

Si vous avez fait une couille et souhaitez revenir sur les modifications apportées à n'importe quel fichier, faites-le en utilisant la commande suivante:

```
git checkout -- <nomfichier>
```

Cela remplacera les changements d'arbre de travail par les dernières données présentes dans le HEAD . Tous les changements qui ont été ajoutés à l'index ne seront pas pris en compte.

À l'inverse, si tous les changements / commits locaux doivent être supprimés et que la branche master locale doit pointer vers l'historique le plus récent du serveur, exécutez les commandes suivantes:

```
git fetch origin git reset --hard origin/master
```

Conclusion

Il est important dans un système de développement d'avoir un logiciel qui vous aide à gérer les modifications. Ce tutoriel de base sur GIT devrait permettre à tout développeur d'utiliser facilement GIT. Nous vous rappelons que c'est un système de contrôle de version rigoureux (et très utile) avec tout un tas de fonctionnalités utiles. Si vous avez besoin d'autres informations, la documentation officielle de la GIT pourra vous être utile !

GitHub



- **GitHub** est une société à but lucratif qui offre un service d'hébergement de référentiel Git basé sur le cloud. Essentiellement, il est beaucoup plus facile pour les individus et les équipes d'utiliser Git pour le contrôle de version et la collaboration.

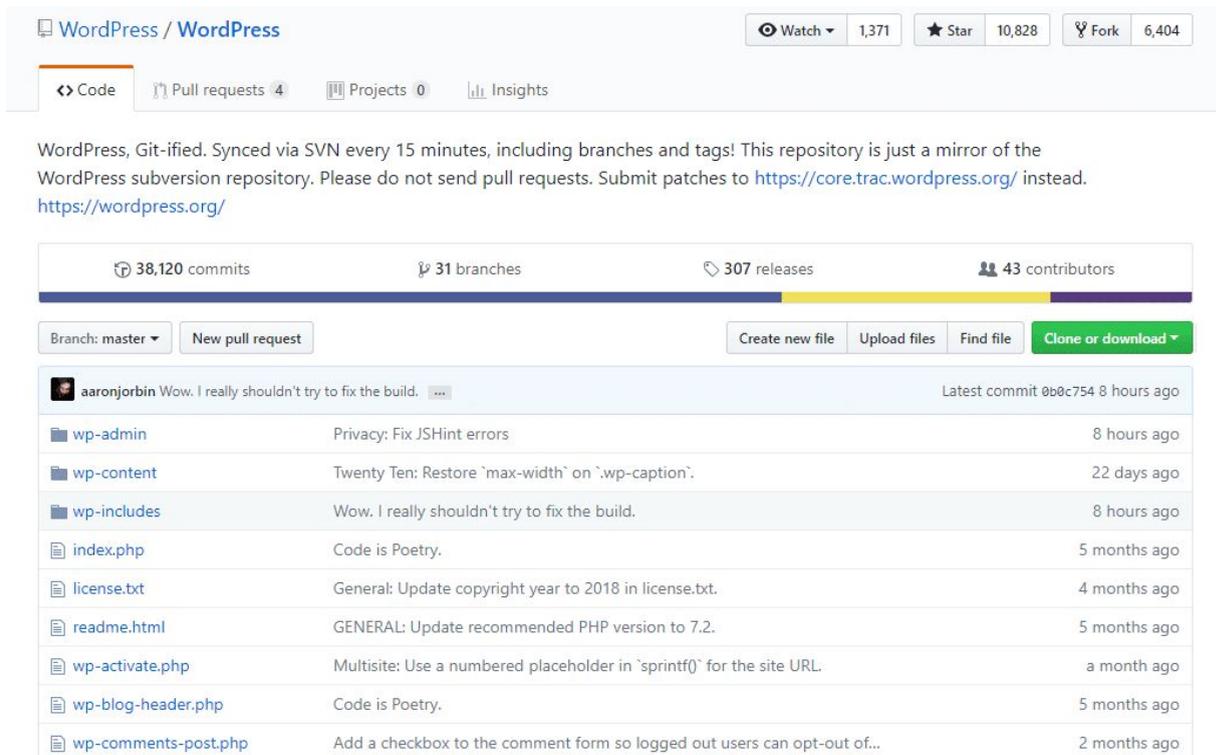
L'interface de GitHub est suffisamment conviviale pour que même les codeurs débutants puissent profiter de Git. Sans GitHub, l'utilisation de Git nécessite généralement un peu plus de connaissances techniques et l'utilisation de la ligne de commande.

GitHub est si convivial, cependant, que certaines personnes utilisent même GitHub pour gérer d'autres types de projets – comme écrire des livres.

De plus, n'importe qui peut s'inscrire et héberger gratuitement un dépôt de code public, ce qui rend GitHub particulièrement populaire auprès des projets open-source. En tant qu'entreprise, GitHub gagne de l'argent en vendant des référentiels de codes privés hébergés, ainsi que d'autres offres orientées métier qui facilitent la gestion des membres de l'équipe et la sécurité pour les entreprises. Nous utilisons Github de manière intensive chez Kinsta pour gérer et développer des projets internes.

Exploration de L'interface GitHub

Pour vous donner une compréhension de base de ce à quoi ressemble l'interface GitHub, voici le code source de WordPress hébergé dans un dépôt GitHub :



The screenshot shows the GitHub repository page for WordPress. At the top, it displays the repository name 'WordPress / WordPress' and statistics: 1,371 watches, 10,828 stars, and 6,404 forks. Below this, there are tabs for 'Code', 'Pull requests (4)', 'Projects (0)', and 'Insights'. A description states: 'WordPress, Git-ified. Synced via SVN every 15 minutes, including branches and tags! This repository is just a mirror of the WordPress subversion repository. Please do not send pull requests. Submit patches to <https://core.trac.wordpress.org/> instead. <https://wordpress.org/>'. Below the description, it shows 38,120 commits, 31 branches, 307 releases, and 43 contributors. There are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A list of recent commits is shown, including one by 'aaronjorbin' with the message 'Wow. I really shouldn't try to fix the build.' and a commit hash '0b0c754' from 8 hours ago. Other commits include 'Privacy: Fix JSHint errors', 'Twenty Ten: Restore `max-width` on `wp-caption`', 'Code is Poetry.', 'General: Update copyright year to 2018 in license.txt.', 'GENERAL: Update recommended PHP version to 7.2.', 'Multisite: Use a numbered placeholder in `sprintf()` for the site URL.', and 'Add a checkbox to the comment form so logged out users can opt-out of...'.

Le code WordPress chez GitHub

Ici, vous pouvez voir les différentes branches sur lesquelles on travaille, ainsi que quand quelqu'un a fait un commit (c'est un peu comme « enregistrer » un fichier). Selon la façon dont un référentiel est configuré, vous pouvez également créer votre propre branche et y effectuer vos propres commits.

Et une fois que vous avez fait quelques changements, vous pouvez soumettre ce code à une branche en faisant une pull request. Cela consiste essentiellement à demander à la personne responsable d'inclure votre code. Et cela aide aussi cette personne à voir exactement ce que vous avez changé dans le code.

Si vous vouliez modifier tout ou partie du code source de WordPress pour votre propre compte sur une base plus

permanente, vous pourriez aussi le forker en cliquant sur le bouton Fork (un fork est similaire en concept à une branche, mais est plus permanent) :



Forking WordPress/WordPress

It should only take a few seconds.



Un exemple de fork de code

WordPress lui-même a été à l'origine un fork de b2/cafeblog. Si vous voulez un peu plus d'informations sur la façon dont vous pouvez utiliser GitHub, Le guide Hello World de GitHub offre un tutoriel convivial pour les débutants.

Atlassian Bitbucket



- **Atlassian Bitbucket**, un service web d'hébergement et de gestion de développement logiciel utilisant le logiciel de gestion de versions Git (et par le passé également le logiciel Mercurial). Il s'agit d'un service freemium dont la version gratuite permet déjà de créer jusqu'à un nombre illimité de dépôts privés, accessibles par cinq utilisateurs au maximum.

The screenshot shows the Bitbucket web interface. At the top, there's a navigation bar with 'Bitbucket' logo and menu items: Teams, Projects, Repositories, Snippets. A search bar 'Find a repository...' is on the right. Below the navigation bar, the user 'Nemanja Popovic' is logged in, and the repository 'GitCommit' is selected. The main content area is titled 'Source' and shows the file structure of the repository. The current branch is 'master'. The file list includes folders like 'DDLTriggers', 'DataTypes', 'FulltextCatalogs', 'Functions', 'Procedures', 'Schemas', 'Tables', 'Triggers', 'Views', and 'XmlSchemaCollections'. At the bottom, there are two files: '.gitignore' (0 B, yesterday) and 'DatabaseSettings.xml' (136 B, yesterday, ApexSQL Script commit).

[Tuto BitBucket: Annexe 1](#)

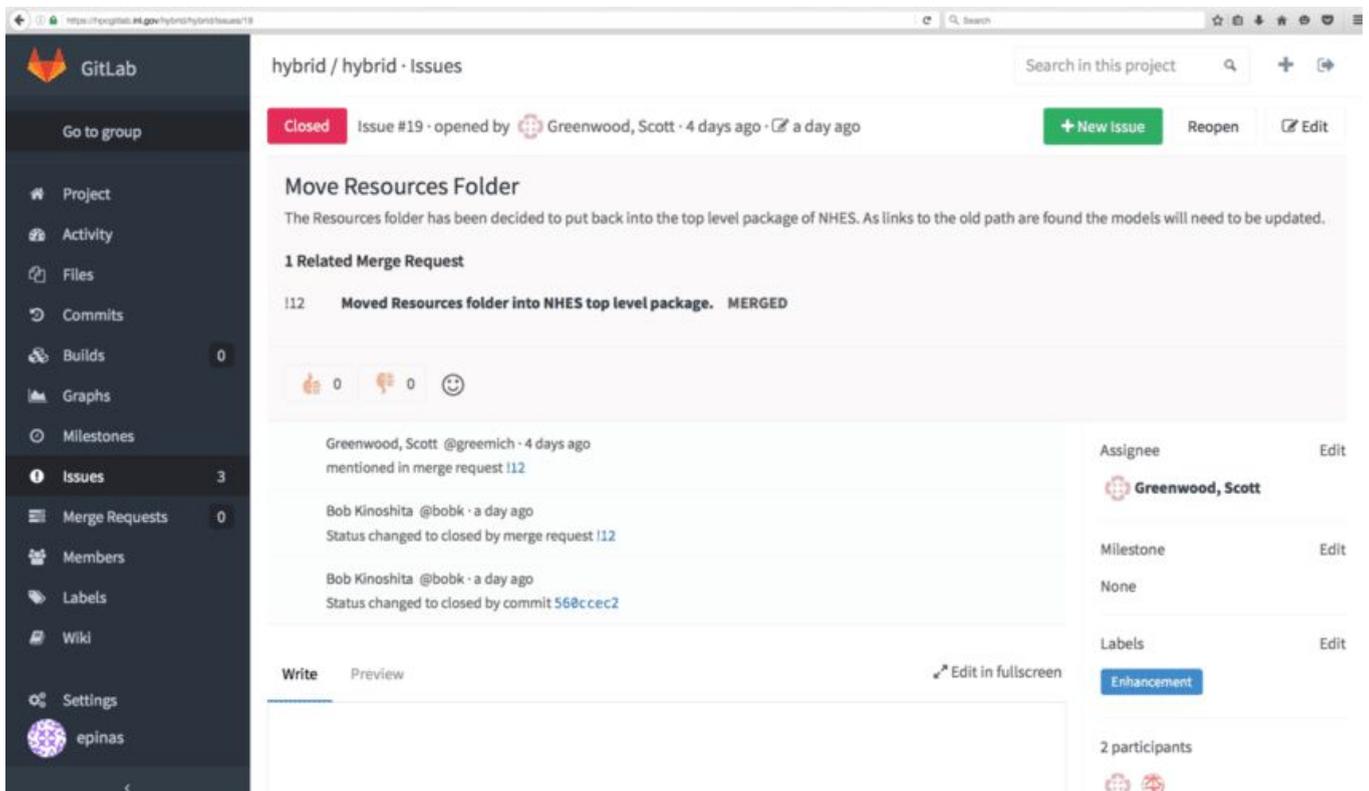
GitLab



- **GitLab**, un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue. Développé par GitLab Inc et créé par Dmitriy Zaporozhets et par Valery Sizov, le logiciel est utilisé par plusieurs grandes entreprises informatiques incluant IBM, Sony, le centre de recherche de Jülich, la NASA, Alibaba, Oracle, Invincea, O'Reilly Media, Leibniz Rechenzentrum, le CERN, European XFEL, la GNOME Foundation, Boeing, Autodata, SpaceX et Altares.

The screenshot shows the GitLab web interface for the 'GitLab Community Edition' project. The page features a navigation bar at the top with 'GitLab', 'Projects', 'Groups', 'Snippets', and 'Help'. The main content area displays the project title 'GitLab Community Edition' and a description: 'GitLab Community Edition (CE) is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. Self-host GitLab CE on your own servers, in a container, or on a cloud provider.' Below the description are several status indicators: 'pipeline passed', 'coverage 75.47%', 'ci best practices passing', 'maintainability B', and 'chat on gitter'. The page also shows a 'Star' button with '4786' stars and a 'HTTPS' button with the URL 'https://gitlab.com/gitlab-or'. At the bottom, there is a table of files with columns for 'Name', 'Last commit', and 'Last update'.

Name	Last commit	Last update
.github	Address feedback about wording.	2 years ago
.gitlab	Merge Templates updates	2 days ago
app	Merge branch 'frozen-string-enable-apps-servi...	9 hours ago
bin	Truncate filenames created by bin/changelog to ...	1 week ago



Gitlab, Github, Bitbucket se ressemblent énormément et ont quasiment la même utilisation, le point commun qu'il on tousse c'est que chacun ont pour base Git.

Tuto GitLab : Annexe 2

6-Enveloppe budgétaire

GitHub:

Version gratuite:

- dépôts public/ Privés illimités
- Nombre illimité de collaborateurs
- 2000 minutes de traitement GitHub d'action par mois
- 500MB de stockage
- Support de la communauté

Version team

- 3.37 euros par utilisateur/mois:
- Tout ce qui est inclus dans l'offre free
- Revue requise par un tiers
- 3000 minutes de traitement GitHub d'action par mois
- 2 Go de stockage
- propriétaires de codes

version entreprise

- 17.70 euros par utilisateur/mois:
- Tout ce qui est inclus dans l'offre Team
- Authentification unique SAML (SSO)
- 50 000 minutes de traitement GitHub d'action par mois
- 50 Go de stockage
- Audit avancés (opération qui vise à vérifier l'ensemble des comptes et les rapports annuels d'une entreprise)

Free	Team	Enterprise
Les bases pour les équipes et les développeurs	Des outils de collaboration et de gestion avancés pour les équipes	Sécurité, conformité et déploiement pour les organisations
<ul style="list-style-type: none"> ∞ Repositories publics/ privés illimités ∞ Nombre illimité de collaborateurs ✓ 2000 minutes de Processing GitHub Actions par mois Les repositories publics sont gratuits ✓ 500MB de stockage pour GitHub Packages Les repositories publics sont gratuits ✓ Support de la communauté 	<ul style="list-style-type: none"> ← Tout ce qui est inclus dans l'offre Free ∞ Revue requise par un tiers ✓ 3000 minutes de Processing GitHub Actions par mois Les repositories publics sont gratuits ✓ 2 Go de stockage de paquets GitHub Les repositories publics sont gratuits ✓ Propriétaires de codes 	<ul style="list-style-type: none"> ← Tout ce qui est inclus dans l'offre Team ✓ Authentification unique SAML (SSO) ✓ 50 000 minutes de Processing GitHub Actions par mois Les repositories publics sont gratuits ✓ 50 Go de stockage de paquets GitHub Les repositories publics sont gratuits ✓ Audit avancé
\$0 par mois	\$4 par utilisateur/mois	\$21 par utilisateur/mois
Je choisis Free	Je choisis Team	Contactez l'équipe des ventes

Bitbucket:

version free:

- jusqu'à cinq utilisateurs
- 50 minute de traitement par mois
- 1 Go de stockage
- dépôts privés illimités
- intégration à jira logiciel
- intégration Trello
- CI/CD

def CI/CD: permet d'augmenter la fréquence de distribution des applications grâce à l'introduction de l'[automatisation](#) au niveau des étapes de développement des applications.

CI:intégration continue

CD:distribution continue ou/et déploiement continu

- support standard

version Standard

- 2.53 euros:
- limite d'utilisateurs illimité
- 2500minute de traitement par mois
- 5 Go de stockage
- dépôts privés illimités
- intégration à jira logiciel
- intégration Trello
- CI/CD
- support standard

version Premium

- 5.06 euros:
- limite d'utilisateurs illimité
- 3500minute de traitement par mois
- 10 Go de stockage
- dépôts privés illimités
- intégration à jira logiciel
- intégration Trello
- CI/CD
- support standard
- contrôles des merges obligatoires
- Autorisations de déploiement
- mise en liste verte des IP
- vérification en deux étapes obligatoire

Free	Standard	Premium
Lancez-vous	J'essaie	J'essaie
0 \$US Illimité	3 \$US Prix de départ /utilisateur/mois	6 \$US Prix de départ /utilisateur/mois
Jusqu'à cinq utilisateurs	Illimité	Illimité
50 min/mois	2 500 min/mois	3 500 min/mois
1 Go	5 Go	10 Go

GitLab:

<p>Free</p> <p>Develop with a team of any size</p> <p>\$0</p> <p>/user/month</p> <p>Start now</p> <p>Includes</p> <ul style="list-style-type: none">All stages of the DevOps lifecycleBring your own CI runnersBring your own production environment400 CI/CD minutes	<p>Bronze / Starter</p> <p>Control what goes into production</p> <p>\$4</p> <p>/user/month <small>(USD, billed annually at \$48)</small></p> <p>Buy now</p> <p>All the benefits of Free +</p> <ul style="list-style-type: none">Single-team project managementMore control over your codeNext business day support2000 CI/CD minutes	<p>Silver / Premium</p> <p>Plan across multiple teams</p> <p>\$19</p> <p>/user/month <small>(USD, billed annually at \$228)</small></p> <p>Buy now</p> <p>All the benefits of Bronze / Starter +</p> <ul style="list-style-type: none">Cross-team project managementCode integrity controlsMulti-region supportPriority support10000 CI/CD minutes	<p>Gold / Ultimate</p> <p>Secure & monitor production</p> <p>\$99</p> <p>/user/month <small>(USD, billed annually at \$1188)</small></p> <p>Buy now</p> <p>All the benefits of Silver / Premium +</p> <ul style="list-style-type: none">Company wide portfolio managementAdvanced application securityExecutive level insightsCompliance automationFree guest users50000 CI/CD minutes
---	--	---	--

7-Solution adopter

Selon notre étude nous avons décidé que se serait GitHub car au niveau budgétaire si on sélectionne la version team ou entreprise, GitHub est nettement plus avantageuse pour l'entreprise au niveau de l'espace de travail, du prix, le temps de traitement par mois et le nombre de collaborateurs qui est illimité.

Une prise en main rapide qui permettra au développeur de manager leur projet, d'éviter des erreurs qui peuvent coûter très cher, il gagneront grâce à cela du temps et en qualité dans leur travail.

8-Source

Contexte et définition:

<http://projet-isika.com/index.php/2018/11/13/le-versioning/>

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

GitHub: <https://fr.github.com/>

Bitbucket: <https://bitbucket.org/>

Tuto Git : <https://www.hostinger.fr/tutoriels/tuto-git/>

Tuto Github :

[https://www.christopheducamp.com/2013/12/15/github-pour-nuls-p-
artie-1/](https://www.christopheducamp.com/2013/12/15/github-pour-nuls-p-
artie-1/)