



# RAPPORT TECHNIQUE



---

# **SOMMAIRE**

[1/ Identification du projet](#)

[2/ Présentation de l'équipe Projet Versionado](#)

[3/Définition des concepts](#)

[4/Fonctionnement du versionning](#)

[Versionning simple : X.Y:](#)

[Le versioning sémantique](#)

[5/ Présentation des outils de versionning](#)

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

[GIT](#)

[GitHub](#)

[GITLAB](#)

[6/ Etude comparative des logiciels de gestion de code](#)

[GIT:](#)

[GitHub:](#)

[GITLAB:](#)

[7/Solution Recommandée](#)

[GitHub Windows10](#)

[GitHub installation linux](#)

[GitHub installation Mac Os](#)

[8/ Conclusion](#)

[9/ Sources](#)



---

## 1/ Identification du projet

Nous avons choisi avec les membres de groupes de prendre le projet Versionado car dans le groupe la plupart des membres veulent faire du développement dans leur futur métier. Nous voulons approfondir nos connaissances sur les possibilités de coder en groupe et de manière décentralisée.

### Énoncé du projet:

La taille des projets informatiques de la société VosRêves est de plus en plus importante. Les équipes de développements comptent, ainsi, de plus en plus de membres. Tout au long de la réalisation des programmes informatiques, de nombreuses personnes y participent, ce pourquoi la société VosRêves souhaite mettre en place un système de versioning.

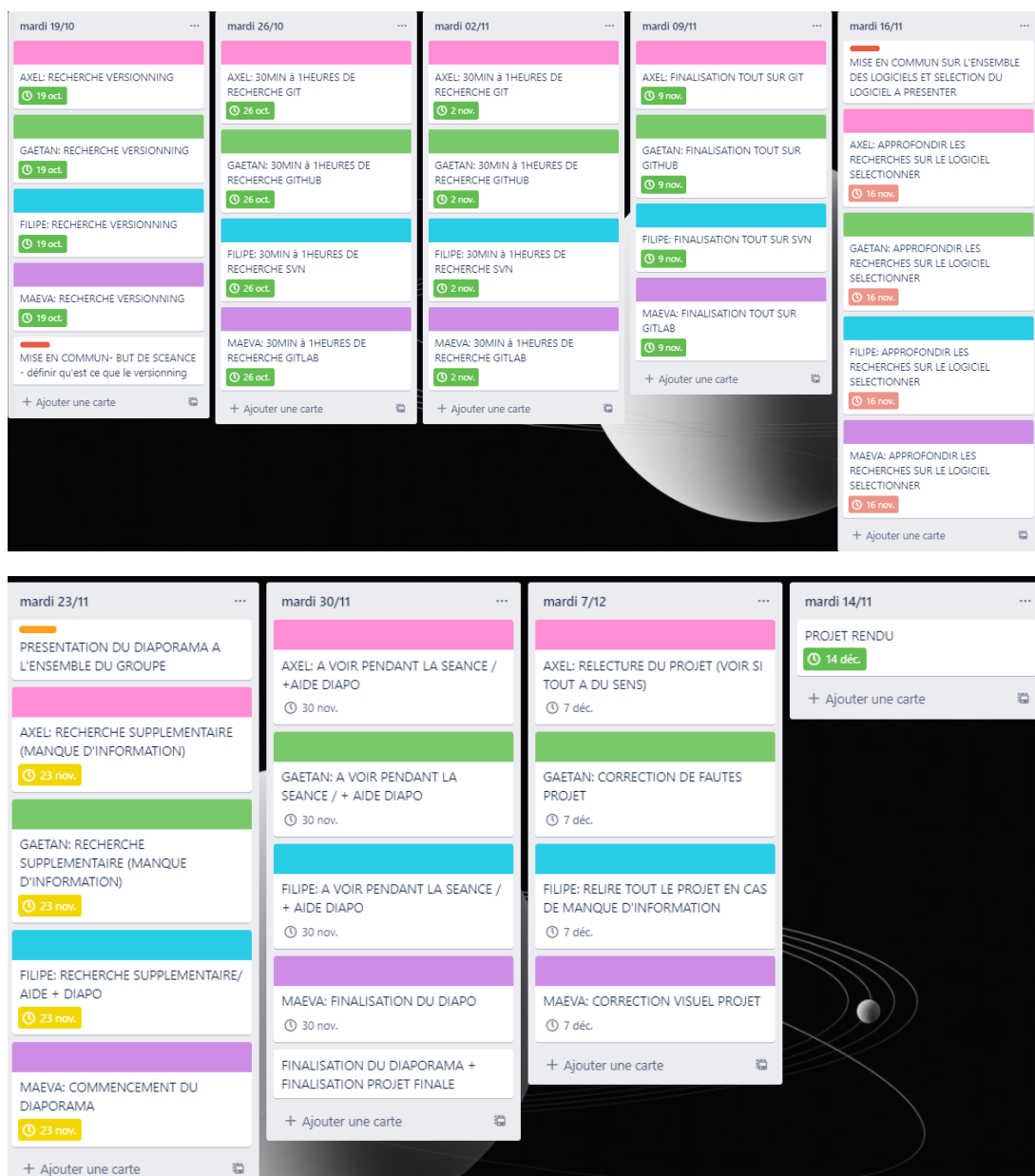
L'équipe en charge devra présenter à la direction une étude exhaustive des différentes méthodes, technologies et solutions de contrôle des versions avec ces avantages et ces inconvénients et proposer une solution concrète.



## 2/ Présentation de l'équipe Projet Versionado

Nous sommes quatre étudiants : Gaëtan GOMEZ, Axel GRAZIANI, Filipe GONCALVES MONTEIRO, Maéva PHAN est chef de projet.

Pour nous organiser, nous avons utilisé Trello afin de mettre au clair ce que chacun allait faire. Cela nous a permis d'être très organisé et de répartir les tâches de façon optimale.

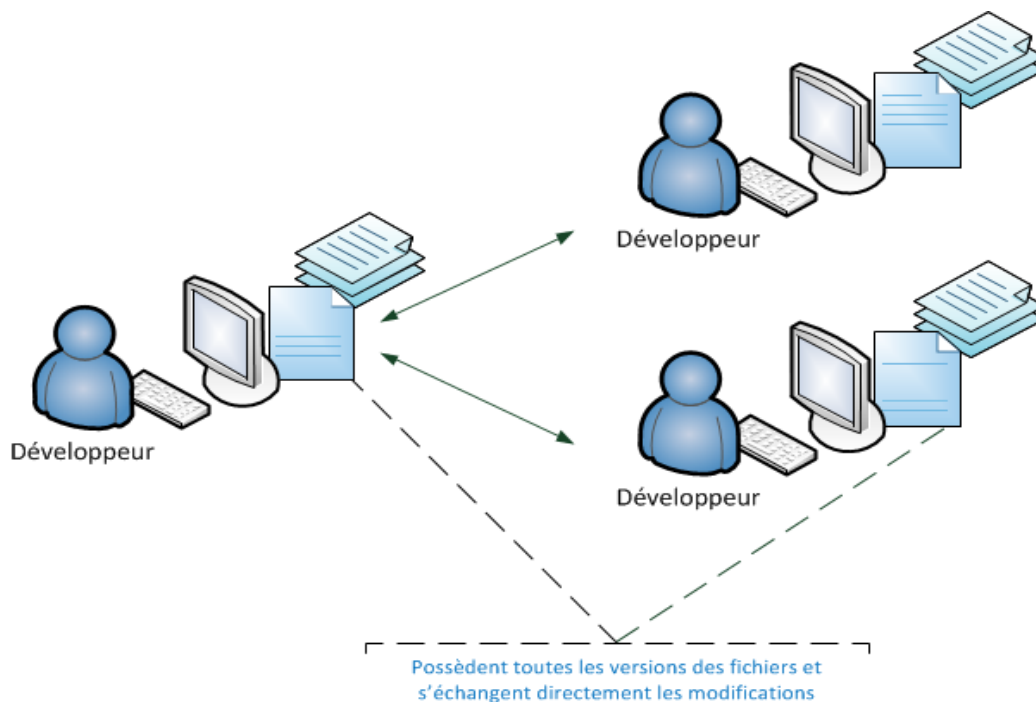


### 3/Définition des concepts

Le versioning est une méthode de gestion des versions d'un même produit. Il consiste à travailler directement sur le code source du projet, en gardant toutes les versions précédentes enregistrées sur le réseau local ou sur le cloud.

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, car toutes les modifications effectuées sont conservées chronologiquement. De cette façon, il n'y a aucun risque d'altérer ou de perdre les données déjà enregistrées sur le dépôt commun.

L'utilisation d'un logiciel de versioning est devenue quasi indispensable pour tout développeur, même s'il travaille seul. Les développeurs peuvent avoir accès à toutes les parties du projet commun, en fonction de la sensibilité des données qui y sont traitées.



---

## 4/Fonctionnement du versionning

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.

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 (potentielles mises à jour) et la version finale.

### Versionning simple : X.Y:

X.Y c'est « la base » : Version - Révision.

La version, c'est le découpage fonctionnel dans lequel se trouve un logiciel à un instant T. Ainsi il peut être prévu de découper une application de blogging.

La révision c'est l'état d'un lot fonctionnel.

La première *release* officielle sera donc identifiée par le numéro de version 1.0 (version 1 révision zéro). En cas de correctifs, ce même lot d'évolutions sortira sous le numéro de version 1.1 (version 1 révision 1). Cela indique que les versions sont identiques mais que des corrections ou améliorations y ont été apportées. Cela continue jusqu'à ce qu'il n'y ait plus de bugs.



---

La seconde *release* officielle sortira sous le numéro 2.0 (même si la version 1 est en révision 1.4).

Ce processus se poursuit jusqu'à ce que le logiciel ait atteint les fonctionnalités souhaitées. Ce type de versionning peut être utilisé pour n'importe quel projet, de n'importe quelle taille. Pour des projets aux nombreuses fonctionnalités, il peut cependant être utile d'utiliser un versionning sémantique.

### Le versioning sémantique

Le versioning sémantique est une méthode très utilisée pour l'attribution des numéros de version : *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 et la version finale.*

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 :

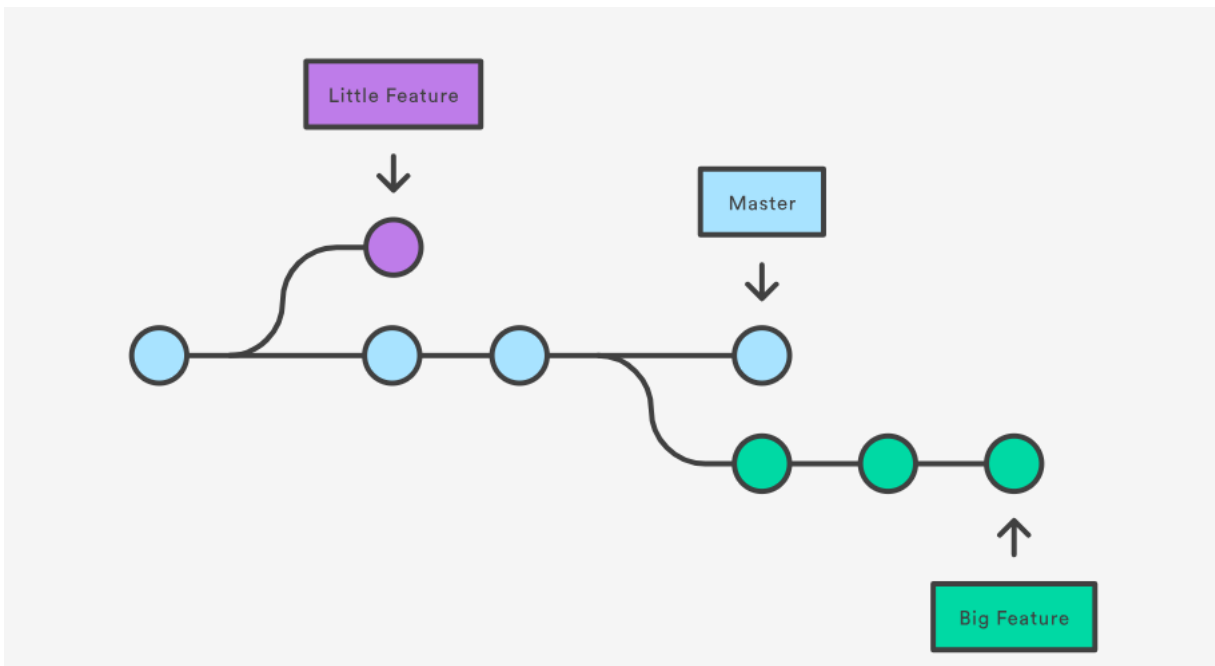
X – Majeur : changements non rétro-compatibles, suppression d'une fonctionnalité obsolète, modification d'interfaces, renommages...

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...



Sur cette image ci-dessous, vous pouvez voir comment un projet peut être amélioré avec les différentes étapes. Les différentes étapes que vous pouvez voir servent à améliorer une fonctionnalité d'une application mais aussi à corriger les bugs qui pourraient apparaître pendant son utilisation.





---

## 5/ Présentation des outils de versionning



### 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 (Version Control System) 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 systèmes de gestion les plus connus : GITHUB. Vous apprendrez à l'installer et à l'utiliser.

Un tutoriel est disponible en annexe : Guide d'utilisation GitHub



---

## Qu'est-ce que le 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 (Version Control System) 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 générées. 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 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. 36 millions de développeurs/utilisateurs dans le monde en 2020.



---

## Qu'est-ce que GitHub ?



GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

Ce site est développé en Ruby on Rails et Erlang par Chris Wanstrath, PJ Hyett et Tom Preston-Werner.

GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

Le nom GitHub est composé du mot « git » faisant référence à un système de contrôle de version open-source et le mot « hub » faisant référence au réseau social bâti autour du système Git, mais aussi à une plate forme de correspondance qui est appelée en anglais un « hub ».

Octocat est la mascotte de la marque. Il a été dessiné par Simon Oxley (également créateur du logo de Twitter ) dans un style épuré inspiré par les arts populaires japonais (manga).

On voit une partie de son visage dans un déguisement lui donnant des oreilles de chat et des tentacules de céphalopode.



---

GitHub est centré vers l'aspect social du développement.

En plus d'offrir l'hébergement de projets avec Git, le site offre de nombreuses fonctionnalités habituellement retrouvées sur les réseaux sociaux comme les flux, la possibilité de suivre des personnes ou des projets ainsi que des graphes de réseaux pour les dépôts (en anglais repository).

GitHub offre aussi la possibilité de créer un wiki et une page web pour chaque dépôt.

Le site offre aussi un logiciel de suivi de problèmes (de l'anglais issu tracking system).

GitHub propose l'intégration d'un grand nombre de services externes, tels que l'intégration continue, la gestion de versions, badges, chat basés sur les projets, etc.

Les documentations des projets sont écrites en langage Markdown (fichiers .md).

L'option webhook permet de communiquer avec des outils d'intégration continue.

19,4 millions de référentiels actifs, 5,8 millions d'utilisateurs actifs et 331 000 organisations actives.



---

## Qu'est-ce que GitLab ?



GitLab est une plateforme de développement collaborative open source éditée par la société américaine du même nom.

Elle couvre l'ensemble des étapes du DevOps (le DevOps est un mouvement qui vise à concilier deux corps de métier : le développeur logiciel (dev) d'une part, l'administrateur de systèmes et d'architectures (ops) d'autre part.).

Se basant sur les fonctionnalités du logiciel Git, elle permet de piloter des dépôts de code source et de gérer leurs différentes versions. Son usage est particulièrement indiqué pour les développeurs qui souhaitent disposer d'un outil réactif et accessible.

GitLab revendique plus de 100 000 organisations utilisant sa plateforme à travers le monde. Ce qui représente un total de plus de 30 millions d'utilisateurs enregistrés.

Open source, l'application fait l'objet d'un développement communautaire. Selon la société GitLab, 3 000 contributeurs collaborent à son évolution.



---

L'interface de GitLab reste très similaire à celle de GitHub. Toutefois, GitLab propose des options pour le moins pratiques :

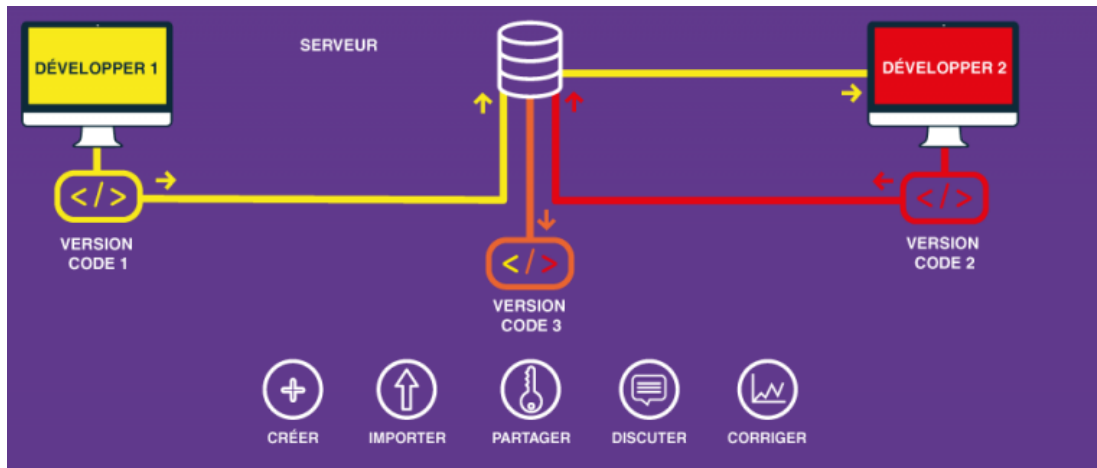
- Gestion de projet
- Planification / priorisation
- Build
- Test logiciel
- Sécurité applicative
- Gestion des configurations
- Monitoring
- Intégration et déploiement continu, etc.

Pour un emploi ergonomique, GitLab se situe sur une machine virtuelle, elle-même hébergée sur un serveur web. Cet outil de plateforme collaborative s'appuie sur une base de données. L'interface d'administration, notamment pour la création de comptes utilisateurs, passe par une configuration en ligne.



## 6/Etude comparative des logiciels de gestion de code

### GIT :



### 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)

## GITLAB:

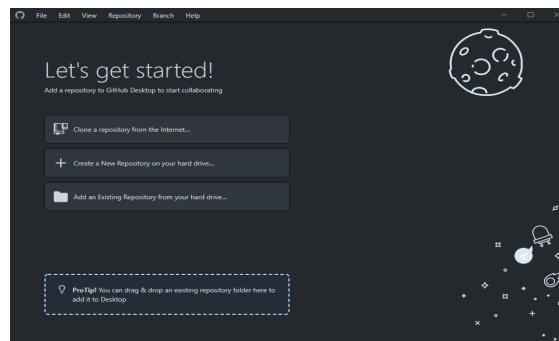
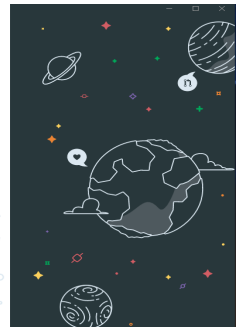
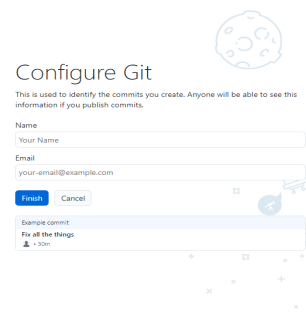
Free Develop with a team of any size	Bronze / Starter Control what goes into production	Silver / Premium Plan across multiple teams	Gold / Ultimate Secure & monitor production
<b>\$0</b> /user/month	<b>\$4</b> /user/month <small>(USD, billed annually at \$48)</small>	<b>\$19</b> /user/month <small>(USD, billed annually at \$228)</small>	<b>\$99</b> /user/month <small>(USD, billed annually at \$1188)</small>
<a href="#">Start now</a>	<a href="#">Buy now</a>	<a href="#">Buy now</a>	<a href="#">Buy now</a>
Includes	All the benefits of Free +	All the benefits of Bronze / Starter +	All the benefits of Silver / Premium +
All stages of the DevOps lifecycle	Single-team project management	Cross-team project management	Company wide portfolio management
Bring your own CI runners	More control over your code	Code integrity controls	Advanced application security
Bring your own production environment	Next business day support	Multi-region support	Executive level insights
400 CI/CD minutes	2000 CI/CD minutes	Priority support	Compliance automation
		10000 CI/CD minutes	Free guest users
			50000 CI/CD minutes



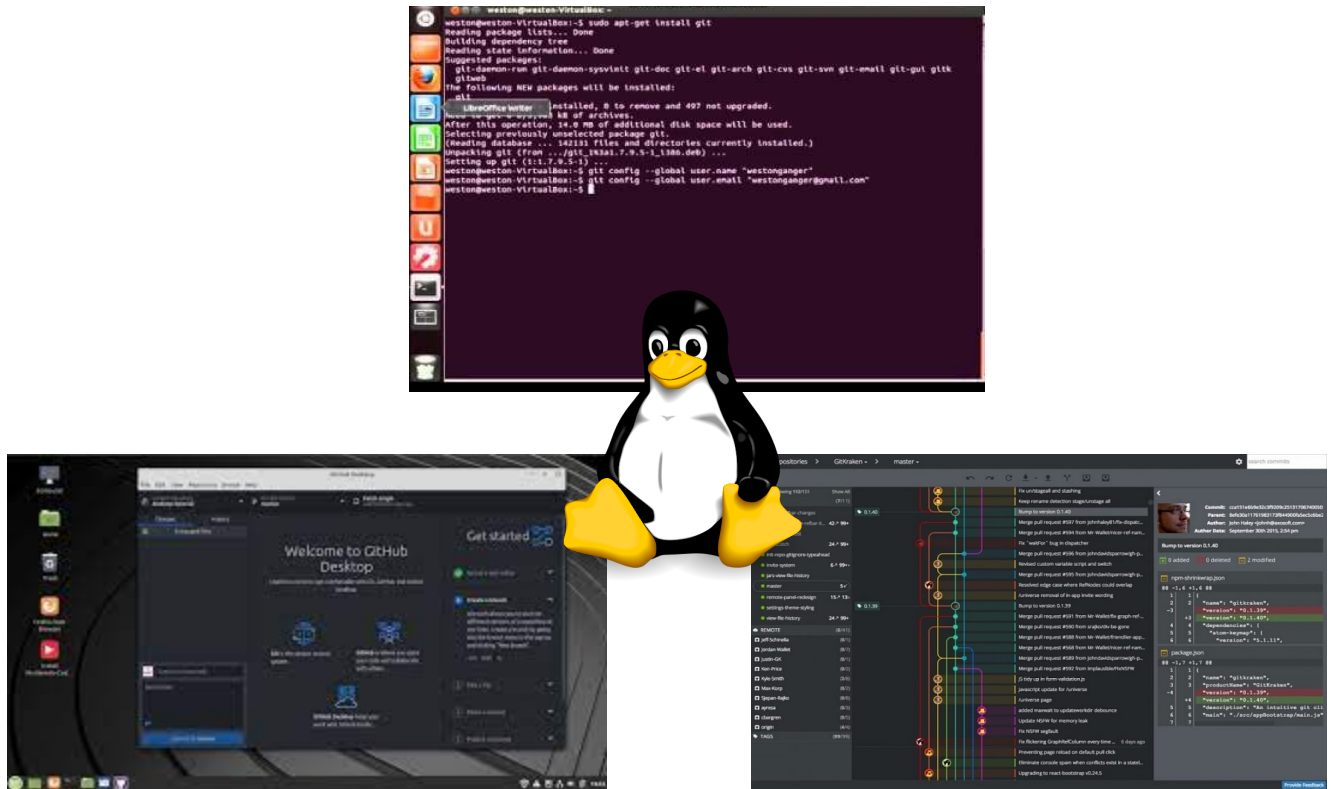
## 7/Solution Recommandée

<https://desktop.github.com/> (site officiel Github)

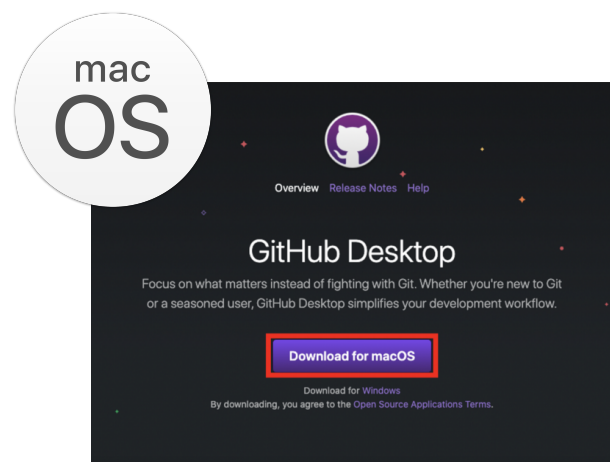
### GitHub pour Windows 10



## GitHub Installation Linux



## Github installation Mac Os



---

## Conclusion sur les installations

Dans un premier temps, il faut savoir que github est un service en ligne mais aussi un logiciel.

Nous avons installé Github sur Windows 10, Linux et Mac OS. Nous avons réalisé cela pour démontrer l'accessibilité de GITHUB. En effet, il est accessible sous tous les systèmes d'exploitation mais surtout il est accessible sur le web cela permet à github de toucher un maximum de personnes.

Mode opératoire : aller sur le site officiel pour télécharger la version officielle. Puis suivre l'installation pour chacun des systèmes d'exploitation. Mais dans notre expérience nous avons rencontré des difficultés surtout dans Linux mais pour Windows et Mac OS cela a été très facile.

Pour Linux l'expérience fut très compliquée, d'une part on ne savait pas trop comment Linux fonctionnait et d'autre part on a peu de connaissance de ce système. Egalement Linux est un système d'exploitation peu connu. Il fallait faire des commandes pour télécharger Github puis il fallait faire des commandes spéciales pour bien utiliser Github.

Pour utiliser GitHub en groupe, il faut créer un compte pour chaque utilisateur dans le service Web de GitHub. Nous arrivons dans un menu qui nous laisse la possibilité soit de travailler dans un serveur local ou dans un serveur sur internet.



---

## 8/ Conclusion

Selon notre étude, après avoir travaillé, recherché, débattu les 3 outils de Versionning nous avons décidé que ce serait GitHub que nous allons recommander car il a des avantages que les autres outils de versionning n'ont pas.

En effet, si on compare la version team ou entreprise, GitHub est nettement plus avantageux 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.



---

## 9/ Sources

Nowteam

<https://www.nowteam.net/versioning-assurer-le-fonctionnement-de-vos-logiciels-dentreprise/>

Projet-isika

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

Wikipedia

[https://fr.wikipedia.org/wiki/Logiciel\\_de\\_gestion\\_de\\_versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions)

Apprendre-la-programmation

<https://apprendre-la-programmation.net/versionner-son-code-git/>

Dynamic-mess

<http://www.dynamic-mess.com/developpement/introduction-a-la-gestion-semantique-des-versions-de-logiciels-2-75/>

stph.scenari-community

<https://stph.scenari-community.org/doc/ecm/co/gcU023versioning.html>

markdown.data-ensta.

<https://markdown.data-ensta.fr/s/introduction-versioning-git>



# Guide d'utilisation GitHub

## Introduction

GitHub est une plate-forme d'hébergement de code pour le contrôle de version et la collaboration. Il permet de travailler ensemble sur des projets communs depuis n'importe où.

Ce guide d'utilisation GitHub vous apprend les éléments essentiels de GitHub tels que les référentiels, les branches, les commits et les pull request.

Vous allez créer votre propre référentiel BTSSIO et apprendre le workflow de demande d'extraction de GitHub, un moyen populaire de créer et de réviser du code.

Dans ce guide d'utilisation GitHub, vous allez :

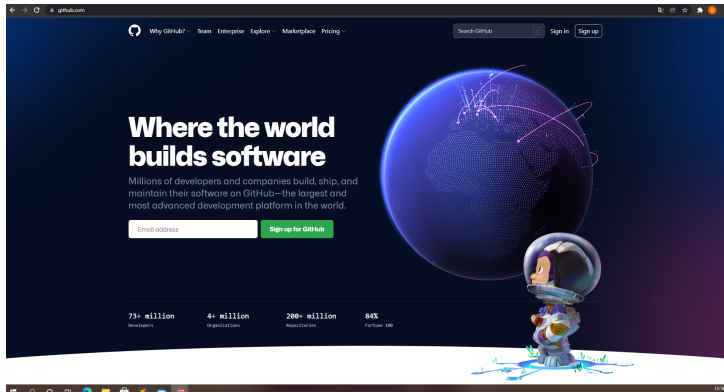
- Créer un compte
- Créer et utiliser un dépôt
- Démarrer et gérer une nouvelle succursale
- Apportez des modifications à un fichier et transférez-le vers GitHub en tant que commits
- Ouvrir et fusionner une pull request

Pour commencer, vous avez besoin d'un compte GitHub (<https://github.com/login>) et d'un accès Internet.

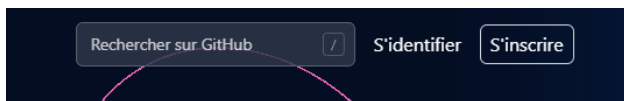


## Création d'un compte GITHUB

Tout d'abord il faut se rendre sur le site internet de GITHUB ( <https://github.com/> )



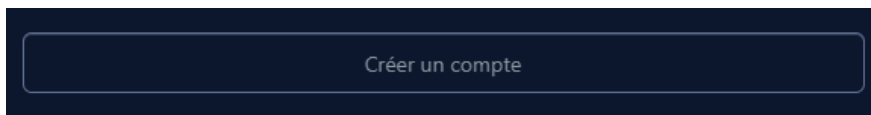
Et cliquer sur s'inscrire en haut à droite



Puis suivez les instructions que le site vous donne



Et pour finir cliquer sur créer un compte tout en bas de la page.





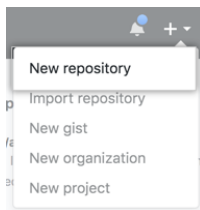
---

## Création d'un dépôt

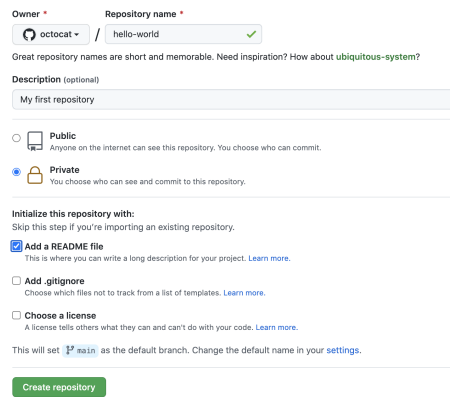
Un dépôt est généralement utilisé pour organiser un seul projet. Les dépôts peuvent contenir des dossiers et des fichiers, des images, des vidéos, des feuilles de calcul et des ensembles de données, tout ce dont votre projet a besoin. Souvent, les référentiels incluent un README fichier, un fichier contenant des informations sur votre projet. GitHub permet d'en ajouter un en même temps que vous créez votre nouveau référentiel. Il offre également d'autres options courantes telles qu'un fichier de licence.

Votre hello-world référentiel peut être un endroit où vous stockez des idées, des ressources, ou même partagez et discutez avec d'autres.

1. Dans le coin supérieur droit de n'importe quelle page, utilisez le menu déroulant et sélectionnez **Nouveau référentiel**.



2. Dans la zone **Nom du référentiel**, saisissez **hello-world**.
3. Dans la zone **Description**, écrivez une brève description.
4. Sélectionnez **Ajouter un fichier README**.
5. Cliquez sur **Créer un référentiel**

A screenshot of the GitHub 'Create new repository' form. The 'Owner' is 'octocat' and the 'Repository name' is 'hello-world'. The description is 'My first repository'. The 'Public' option is selected. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. The 'Create repository' button is at the bottom.

---

## Création d'une succursale

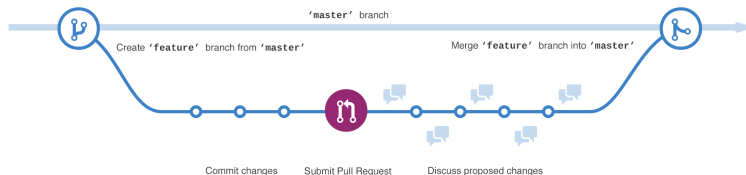
Le branchement vous permet d'avoir différentes versions d'un référentiel en même temps.

Par défaut, votre référentiel a une branche nommée main qui est considérée comme la branche définitive. Vous pouvez utiliser des branches pour expérimenter et apporter des modifications avant de les valider main.

Lorsque vous créez une branche à partir de la main branche, vous faites une copie, ou un instantané, de main ce qu'elle était à ce moment-là. Si quelqu'un d'autre a apporté des modifications à la main branche pendant que vous travaillez sur votre branche, vous pouvez récupérer ces mises à jour.

Ce schéma montre :

- La main branche
- Une nouvelle branche appelée feature
- Le voyage qui feature prend avant qu'il ne se fonde dans main



Avez-vous déjà enregistré différentes versions d'un fichier ? Quelque chose comme:

- story.txt
- story-joe-edit.txt
- story-joe-edit-reviewed.txt

Les branches accomplissent des objectifs similaires dans les référentiels GitHub.

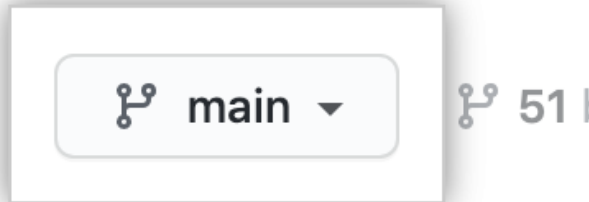
Chez GitHub, nos développeurs, rédacteurs et concepteurs utilisent des branches pour séparer les corrections de bogues et le travail des fonctionnalités de notre main branche (de production). Lorsqu'un changement est prêt, ils fusionnent leur branche dans main.



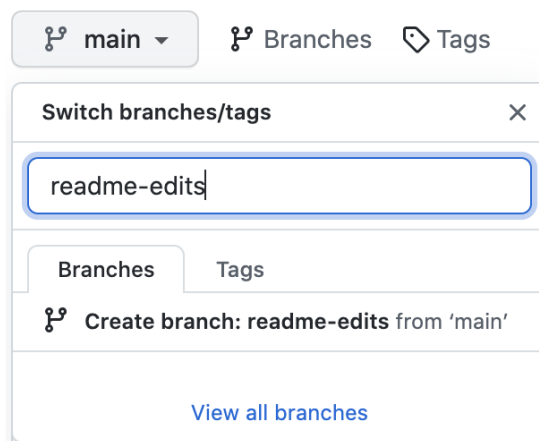
---

## Mode opératoire > Créer une succursale

1. Cliquez sur l'onglet Code de votre BTS SIO référentiel.
2. Cliquez sur le menu déroulant en haut de la liste des fichiers qui dit main .



- 3.
4. Tapez un nom de branche, readme-edits, dans la zone de texte.
5. Cliquez sur Créer une branche : readme-edits à partir du fichier principal .



Vous avez maintenant deux branches, main et readme-edits. En ce moment, ils se ressemblent exactement. Ensuite, vous ajouterez des modifications à la nouvelle branche.

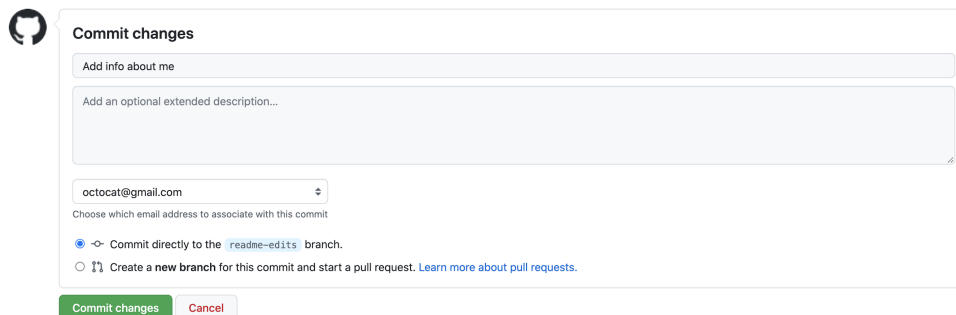
---

## Apporter et valider des modifications

Lorsque vous avez créé une nouvelle branche à l'étape précédente, GitHub vous a amené à la page de code de votre nouvelle readme-edit branche, qui est une copie de main.

Vous pouvez apporter et enregistrer des modifications aux fichiers de votre référentiel. Sur GitHub, les modifications enregistrées sont appelées commits. Chaque commit a un message de commit associé, qui est une description expliquant pourquoi une modification particulière a été apportée. Les messages de validation capturent l'historique de vos modifications afin que les autres contributeurs puissent comprendre ce que vous avez fait et pourquoi.

1. Cliquez sur le README.md fichier.
2. Cliquez pour modifier le fichier.
3. Dans l'éditeur, écrivez un peu sur vous-même.
4. Dans la zone Valider les modifications , écrivez un message de validation décrivant vos modifications.
5. Cliquez sur Valider les modifications .



The screenshot shows the 'Commit changes' dialog box in GitHub. It includes a text input field for 'Add info about me', a larger text area for 'Add an optional extended description...', a dropdown menu showing 'octocat@gmail.com', and a note to 'Choose which email address to associate with this commit'. Below this, there are two radio button options: 'Commit directly to the readme-edits branch.' (which is selected) and 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' At the bottom, there are two buttons: 'Commit changes' (green) and 'Cancel' (red).

6.

Ces modifications seront apportées uniquement au fichier README de votre readme-edit branche, donc maintenant cette branche contient un contenu différent de main.



---

## Ouvrir une pull request

Maintenant que vous avez des changements dans une branche de main, vous pouvez ouvrir une pull request.

Les demandes de tirage sont au cœur de la collaboration sur GitHub. Lorsque vous ouvrez une demande d'extraction, vous proposez vos modifications et demandez à quelqu'un de réviser et d'extraire votre contribution et de les fusionner dans sa branche. Les demandes d'extraction affichent les différences, ou les différences, du contenu des deux branches. Les modifications, ajouts et soustractions sont affichés dans différentes couleurs.

Dès que vous effectuez un commit, vous pouvez ouvrir une pull request et démarrer une discussion, avant même que le code ne soit terminé.

En utilisant la @mentionfonctionnalité de GitHub dans votre message de demande d'extraction, vous pouvez demander les commentaires de personnes ou d'équipes spécifiques, qu'elles soient au bout du couloir ou à 10 fuseaux horaires.

Vous pouvez même ouvrir des pull request dans votre propre référentiel et les fusionner vous-même. C'est un excellent moyen d'apprendre le flux GitHub avant de travailler sur des projets plus importants.

1. Cliquez sur l'onglet Demandes d' extraction de votre hello-world référentiel.
2. Cliquez sur Nouvelle demande de tirage
3. Dans la zone Exemples de comparaisons , sélectionnez la branche que vous avez créée, readme-edits, à comparer avec main(l'original).
4. Examinez vos modifications dans les différences sur la page Comparer, assurez-vous qu'elles correspondent à celles que vous souhaitez soumettre.



- 
5. Cliquez sur Créer une demande d'extraction .
  6. Donnez un titre à votre pull request et rédigez une brève description de vos modifications. Vous pouvez inclure des emojis et faire glisser et déposer des images et des gifs.
  7. Cliquez sur Créer une demande d'extraction .

Vos collaborateurs peuvent désormais revoir vos modifications et faire des suggestions.

### Fusion de votre pull request

Dans cette dernière étape, vous fusionnerez votre readme-edit branche dans la main branche.

1. Cliquez sur Fusionner la demande d'extraction pour fusionner les modifications dans main.
2. Cliquez sur Confirmer la fusion .
3. Continuez et supprimez la branche, puisque ses modifications ont été intégrées, en cliquant sur Supprimer la branche .

En suivant ce didacticiel, vous avez appris à créer un projet et à effectuer une pull request sur GitHub.

Voici ce que vous avez accompli dans ce tutoriel :

- Création d'un référentiel open source
- Création et gestion d'une nouvelle succursale
- Modification d'un fichier et validation de ces modifications sur GitHub
- Ouverture et fusion d'une pull request

Jetez un œil à votre profil GitHub et vous verrez votre travail se refléter sur votre graphique de contribution.

Pour donner accès à des collaborateurs à votre dépôt github, procédez comme suit:



- 
1. Demandez le nom d'utilisateur de la personne que vous invitez en tant que collaborateur. Si elle n'a pas encore de nom d'utilisateur, elle doit s'inscrire sur GitHub.
  2. Sur GitHub, accédez à la page principale du dépôt.
  3. Sous le nom de votre projet, cliquez sur Paramètres.
  4. Dans la barre latérale de gauche, cliquez sur Collaborateurs.
  5. Sous Collaborateurs, commencez à taper le nom d'utilisateur du collaborateur.
  6. Sélectionnez le nom d'utilisateur du collaborateur dans le menu déroulant.
  7. Cliquez sur Ajouter un collaborateur.
  8. L'utilisateur recevra un e-mail l'invitant au repository. Une fois qu'il aura accepté votre invitation, il aura un accès collaborateur à votre dépôt.
  9. Remarque: C'est uniquement à partir du moment où la personne aura accepté l'invitation que vous pourrez gérer ses droits d'accès.

