GitHub

THIOUNE Seydina

Fiche technique

Adresse	https://github.com/
Slogan	*Build software better, together.* ; *Where software is built*
Langage de conception	Ruby
Type de site	Gestion de versions collaborative
Nombre d'inscrit	14 millions en 2016
Propriétaire	Microsoft
Lancement	10 Avril 2008

C’est quoi Github?

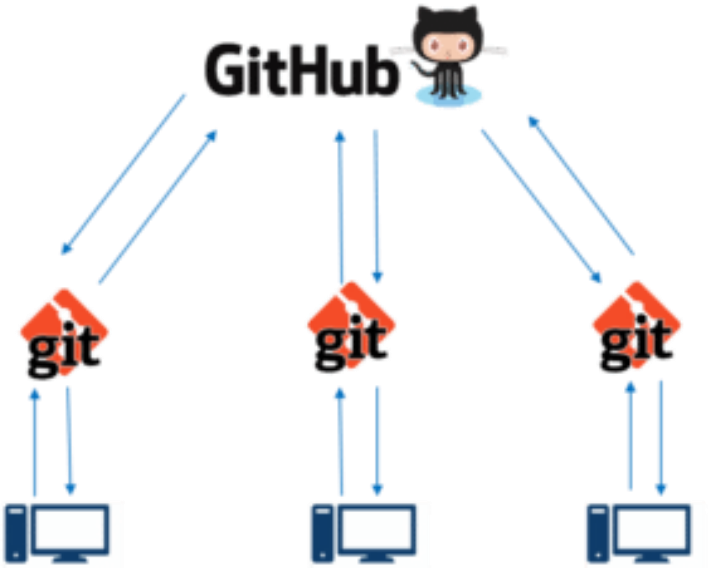
Github est une entreprise de développement logiciel et de service dont le siège est situé aux États-Unis. Github développe notamment la plateforme Github, l'éditeur de texte Atom ou encore le framework Electron.

Mais dans notre situation, nous sommes intéressés par la **plateforme Github**. Ce dernier est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions **Git**. Développé en Ruby, GitHub propose des comptes payants pour professionnels mais aussi des comptes gratuits pour les projets de logiciels libres. Ainsi elle gère 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*.

Avec ses 16 millions d'utilisateurs en 2016, le plateforme Github devient le plus grand hébergeur de code source au monde avec 35 millions de dépôts. Ayant comme logo un autocat (voir ci-dessus), le nom Github est composé de “Git” qui peut signifier ressource (système de gestion open-source) et de “Hub” qui fait référence à un réseau social “Hub” mais aussi qui pouvait signifier un appareil permettant de relier plusieurs données dans une même canal.

10 ans après sa création, Microsoft annonce l’acquisition de l’entreprise pour la somme de 7,5 milliards de dollars américains.

Comment fonctionne Github ?



Contrairement aux autres hébergeurs (traditionnel) open source où t'es obligé de télécharger le code pour ensuite proposer les modifications, Github est tel un réseau social permettant de contribuer direct avec les branchements et nous permettant d'être de propriétaire aussi grâce au principe de fork.
Les "branch" dans Github nous permet d'avoir un timeline des dépôts qu'on a fait surtout quand on travaille à plusieurs sur un même projet et ainsi on aura aussi un historique des "commit".
Alors pour débuter avec Github, il faut déjà se rendre sur https://git-scm.com afin de télécharger l'installeur "**Git**" et l'installer sur votre ordinateur (actuellement on est à la version 2.25.1 pour Windows).
Sans tarder il faut savoir que l'utilisation de **Git** ne sera pas sur un interface graphique mais sur un terminal (en ligne de commande) comme à l'ancienne dont PowerShell, cmd, etc. Du coup, on peut pas parler de ligne de commande sans pour autant savoir les commandes.
Quand on utilise quelque chose de nouveau ça nous arrive d'avoir besoin d'aide à chaque fois et de l'aide, git en propose:

```
git help config   //          pour des aides à la configurations
git help push     //          envoie et mettre à jour le dépôt
git help pull     //          assemblage de branches
git help branch   //          les branches
```

1. Commandes de configuration

- Identifiant

```
git config --global user.name "username"
```

- Email d'identification

```
git config --global user.email "email du compte"
```

- Outil d'édition

```
git config --global core.editor subl
```

2. Commandes de base

- Premier commit permettant d'initialiser notre premier dépôt

```
git init
```

- Premier commit permettant de sélectionner fichier ou dossier

```
git add "nom_dossier/nom_fichier"
git add .        //envoyer tous les dossiers et fichiers du projet
```

- Premier commit permettant d'initialiser notre premier dépôt

```
git commit -m "first commit"
```

- Premier commit permettant d'initialiser notre premier dépôt

```
git reset --hard md5_commit
git push --force
```

- Envoyer ses commits vers le dépôt distant

```
git push
```

- Mettre à jour le dépôt local

```
git pull
```

- Statut des fichiers

```
git status
```

- Lister les branch

```
git branch -a
```

- Création de branch

//créer et basculer sur la nouvelle branch en deux lignes

```
git branch “nom_nvlle_branch”
git checkout “nom_nvlle_branch”
```

//Une seule ligne: créer et basculer

```
 git checkout -b “nom_nvlle_branch”
```

- Supprimer branch

// Si la branch est local et n'est pas créée sur le repo distant

```
git branch -d nom_branch_local
```

// Si la branch est présente sur le repo distant

```
git push origin --delete nom_branch_distante
```

- Changer branch

```
git checkout “nom_branch”
```

3. Autres commandes

- Supprimer un fichier du répertoir

```
git rm nom_fichier
```

- Annuler les commits et perdre tous les changements

```
git reset --hard HEAD^
```

- Initialiser un projet

```
echo "# MON_PROJET" >> README.md
git init
git add README.md
git commit -m "Initial commit"
git remote add origin ....git
git push -u origin master
```

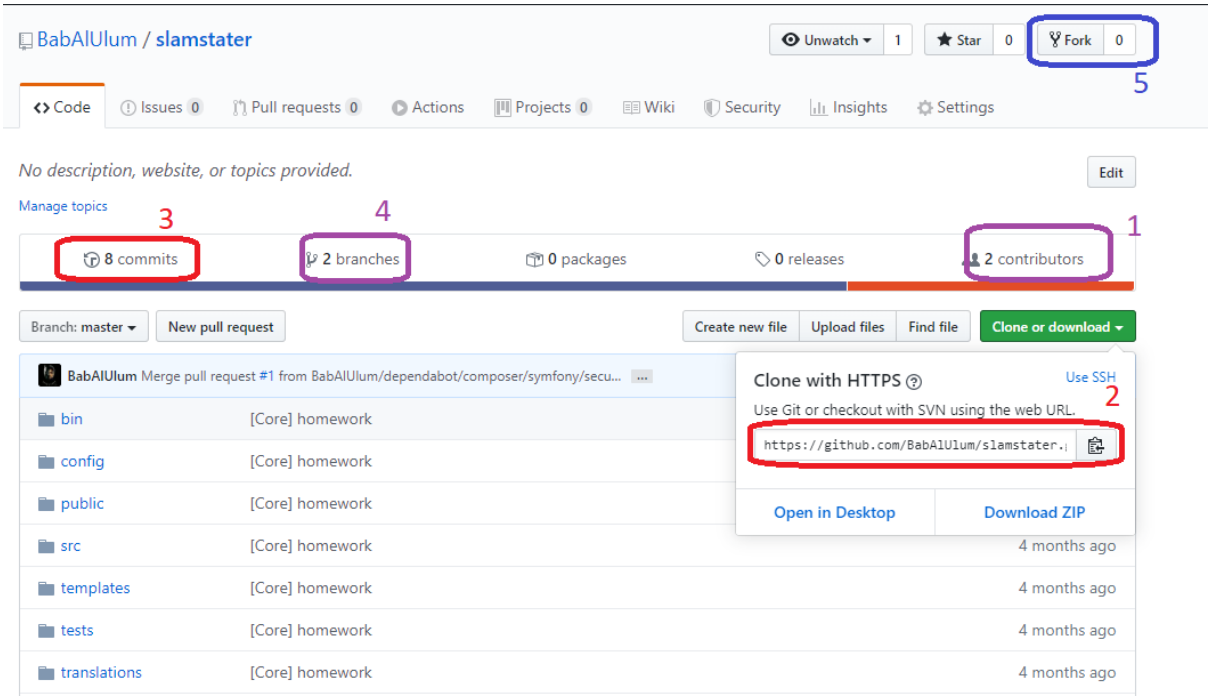
- Récupérer le

```
git clone “lien du repository”
```

Voilà quelques commandes qu’on est obligé de connaître pour l’utilisation de git. Et voilà un exemple de projet sur la plateforme de github où c’est possible

de faire quelques manipulations. Mais on se permet de présenter vite fait notre exemple:

1. ceux qui ont contribué aux projets
2. le lien utile pour faire la commande git clone ……
3. le nombre de commits fait mais aussi qui peut nous donner plus de détails si on veut
4. Le nombre de branch créé dans ce projet
5. York



Github, une aide au déploiement de projet

Google Cloud Platform

Google Cloud Platform (GCP) est une plateforme de cloud computing fournie par Google, proposant un hébergement sur la même infrastructure que celle que Google utilise en interne pour des produits tels que son moteur de recherche.
Alors pour déployer son projet sur GCP à travers Github il faut utiliser un compte utilisateur machine conçu spécialement pour accomplir des tâches automatisées, comme la mise en miroir d'un dépôt. Ce compte doit disposer d'un accès administrateur à notre dépôt.
Après avec accordé les droits de lecture à GCP sur notre projet github, on va ouvrir notre Cloud Source Repository:

- On ajoute un dépôt en sélectionnant “Connecter un dépôt externe”
- On choisit le projet Google Cloud auquel appartient le dépôt mis en miroir.
- On sélectionne GitHub.
- En se connecter à GitHub, on autorise Cloud Source Repositories à stocker nos identifiants puis on accorde à Google Cloud l'accès en lecture de notre dépôt
- Une fois faite, on sera redirigé vers la page “Connecter un dépôt externe.”
- On prends le dépôt à mettre en miroir.
- Et là, la page principale du dépôt s'affiche. Cette page présente la liste des dépôts contenus dans le projet Google Cloud, y compris le dépôt qu’on vient de créer.

Et là tout est bon, même si on peut ajouter une synchronisation automatique.

Heroku

Ce dernier est une entreprise créant des logiciels pour serveur qui permettent le déploiement d'applications web. Heroku gère la plupart des déploiements d'applications avec Git.
Par rapport à GCP, avec Heroku il faut déjà installer CLI Heroku et on débute avec la commande `$ heroku create` dans le dossier racine de notre projet puis `$ git remote -v` pour confirmer la création de la commande heroku.
Par la suite il faut taper la commande `$ git push heroku master` pour initialiser le déploiement. Cette partie semble facile car étant plus simple par

rapport à GCP. Mais cependant, il existe d'autres types de déploiement dans heroku.
Pour des raisons de sécurité, ce déploiement nécessite une authentification HTTP Git. Mais aussi il est possible de faire le transfert en utilisant une clé SSH avec la commande `$ heroku create --ssh-git`.

Existent-ils des alternatives de Github?

1. Cloud Source Repositories

On a un peu parlé précédemment concernant la relation entre github et GCP. Cloud Source Repositories peuvent être connectées à d'autres bases. On peut également utiliser les propres référentiels de Google dans lesquels les fichiers sont enregistrés comme partie intégrante de l'infrastructure. Cela garantit la sécurité des fichiers et surtout du code source. Les Cloud Source Repositories présentent un avantage certain du fait qu'elles permettent d'une recherche direct d'un extrait de code depuis le moteur de recherche de Google. Elles permettent de plus de suivre les bugs pendant que le code s'exécute à l'arrière-plan grâce à son outil Cloud Diagnostics.

2. SourceForge

SourceForge est un site web qui permet aux développeurs d'héberger des projets logiciels et propose des outils pour leur gestion. CVS, SVN, Bazaar, Git ou Mercurial sont des systèmes de gestion de versions qu'il propose. Le site offre aussi un wiki, assure l'accès à une base de données MySQL et offre un sous-domaine pour chaque projet comme " nom-de-projet.sourceforge.net ".

3. Apache Allura

Allura est un programme open-source d'Apache proposant la gestion des codes source et des référentiels de données, le suivi des bugs, la mise en place de forums, de blogs, de pages wiki et autres contenus web prisés des développeurs. Pour le suivi des erreurs, vous pouvez utiliser le formatage Markdown, des fichiers joints ainsi que des tickets d'erreur avec les milestones correspondants.

4. GitKraken

Ce dernier est un logiciel qui va servir de client d'un gestionnaire de version. GitKraken est devenu très utilisé en entreprise. En effet, il est d'une aide précieuse lorsque l'on travaille en équipe sur un même projet. De plus ça permet de rendre son projet récupérable en ligne si on perd les données en local. On peut percevoir plusieurs fonctionnalités très utiles lorsque l'on travaille en équipe.

5. GitLab

Étant un logiciel libre de forge basé sur git, GitLab propose les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue comme presque tous ses concurrents. 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, la GNOME Foundation, Boeing, Autodata, SpaceX, etc.

Conclusion

Pour conclure, il faut savoir que GitHub a de nombreux avantages comme l'offre d'une base de données gratuite et illimitée, une possibilité de créer des organisations et avec un forfait mensuel, c'est possible d'être propriétaire. Par contre on trouve des inconvénients à notre cher GitHub comme sur toute bonne chose. Parmi ses inconvénients, on peut citer le cas où on génère un code hébergé par un serveur privé ce qui pousse à trouver des concurrents de Github comme ce dont on a cité précédemment.

Vocabulaire

Dépôt : Un répertoire ou de l'espace de stockage où vos projets peuvent vivre. Parfois les utilisateurs GitHub raccourcissent ça en « repo ». Il peut être local sur un répertoire de votre ordinateur, ou ce peut être un espace de stockage sur GitHub ou un autre hébergeur en ligne. À l'intérieur d'un dépôt, Vous pouvez conserver des fichiers de code, des fichiers texte, des images.

Fork : terme anglais signifiant « fourche », « bifurcation », « embranchement ». C'est un nouveau logiciel créé à partir du code source d'un logiciel existant lorsque les droits accordés par les auteurs le permettent : ils doivent autoriser l'utilisation, la modification et la redistribution du code source. C'est pour cette raison que les forks se produisent facilement dans le domaine des logiciels libres.

Branche : Comment faire travailler plusieurs personnes sur un projet en même temps sans que Git ne s'embrouille ? Habituellement, elles se « débranchent » du projet principal avec leurs propres versions complètes des modifications qu'elles ont chacune produites de leur côté. Après avoir terminé, il est temps de « fusionner » cette branche pour la ramener vers la branche « master », le répertoire principal du projet.

Commit : C'est la commande qui donne à Git toute sa puissance. Quand vous « committez », vous prenez un « instantané », une « photo » de votre dépôt à ce stade, vous donnant un point de contrôle que vous pouvez ensuite réévaluer ou restaurer votre projet à un état précédent.

HTTP : HyperText Transfer Protocol est un protocole de communication client-serveur développé pour le World Wide Web. HTTPS est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS. HTTP est un protocole de la couche application.
SSH : Secure Shell est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion.