Les interactions avec le dépôt distant

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : GitBash / Terminal

Pré-requis : Bases de Git

Contexte

Dans le cadre professionnel, le développement d'applications se fait généralement au sein d'une équipe de développeurs, chacun des membres de l'équipe réalisant des tâches distinctes. Il est ainsi nécessaire que chacun d'entre eux ait accès au travail réalisé par le reste de l'équipe afin de mettre en commun les tâches terminées.

Pour cela, il est devenu habituel d'utiliser des outils de *versionning*, c'est-à-dire des outils permettant de centraliser le code afin de le rendre accessible par tous les membres d'une équipe.

Nous allons voir ici comment utiliser l'un d'entre eux, GitHub, pour créer un espace de travail centralisé et y transmettre nos réalisations ou récupérer un projet.

Complément

En attendant la mise à jour du cours, ce lien peut vous être utile : https://stackoverflow.com/questions/68775869/support-for-password-authentication-was-removed-please-use-a-personal-access-to

II. Débuter sur GitHub

Objectifs

- Créer un dépôt sur GitHub
- Faciliter et protéger l'accès à son compte GitHub

Mise en situation

Afin de partager les développements réalisés avec les membres d'une équipe en utilisant Git, il faut au préalable mettre en place un espace commun sur lequel seront stockées les différentes versions de l'application.

Cet espace commun est appelé **dépôt distant**. On parle aussi couramment de *repository*.

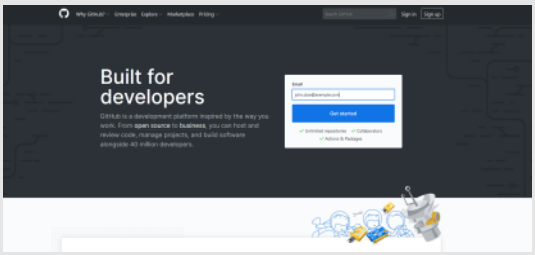
Nous allons voir ici comment créer et accéder à ces espaces en utilisant le service GitHub.

Créer un dépôt distant

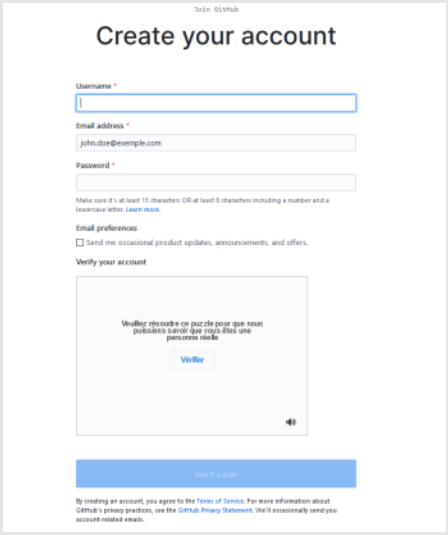
GitHub est un site web proposant des services d'hébergement de projets de développement d'applications grâce à l'utilisation de Git.

La première étape pour utiliser les services proposés par GitHub est de créer un compte, vous pouvez le faire à l'adresse : https://github.com/.

Méthode Créer son accès GitHub



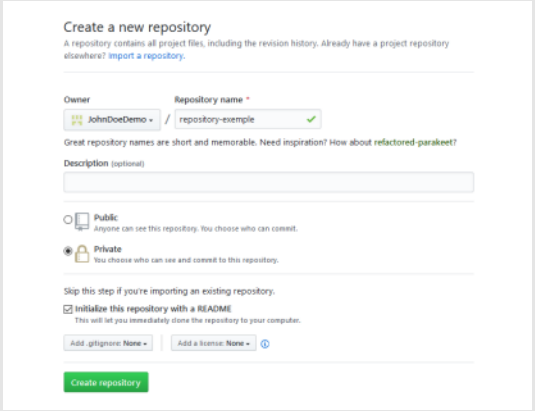
La saisie de votre adresse mail permet d'accéder à la page de création de compte.



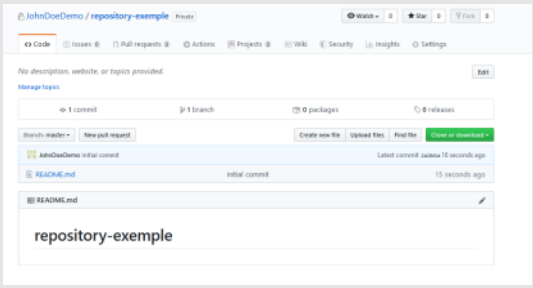
Une fois le formulaire rempli, votre compte sera créé.

À la validation de l'adresse mail, GitHub propose automatiquement de créer un premier dépôt, c'est-à-dire l'espace sur lequel se trouvera le projet.

Méthode Créer son premier dépôt



Lors de la création d'un dépôt, GitHub propose de définir son nom, sa portée (publique ou privée), ainsi que quelques éléments tels que l'ajout d'un fichier `README` (destiné à contenir la documentation nécessaire au fonctionnement du projet), ou encore d'un fichier `.gitignore` (qui permet de répertorier les éléments du projet ne devant pas être versionnés).



En accédant à la page de visualisation du dépôt, on retrouve l'ensemble des fichiers du projet, ainsi qu'une interprétation automatique du fichier `README`.

Conseil **Faciliter et sécuriser l'accès à son compte GitHub**

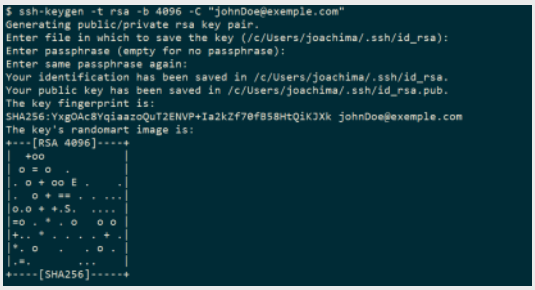
GitHub permet de faciliter et de sécuriser l'accès à un compte en y ajoutant une clé SSH afin de valider l'identité de l'utilisateur.

Pour cela, il suffit de suivre la procédure décrite ici : https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh.

Cela permet de ne pas devoir saisir de mot de passe lorsqu'une action est effectuée vers le dépôt.

Exemple

```
1 ssh-keygen -t rsa -b 4096 -C "johnDoe@exemple.com"
```



La commande `ssh-keygen` permet de générer une paire de clés privée/publique en suivant l'algorithme passé en paramètre.

```
1 eval $(ssh-agent -s)
```

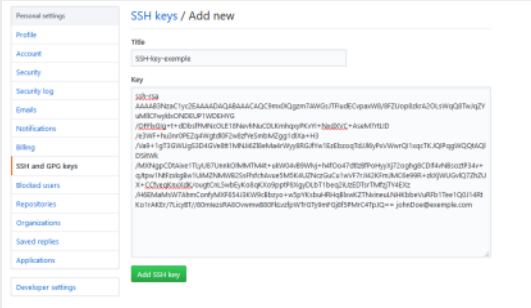
Cette commande permet de lancer l'agent SSH, au cas où celui-ci ne serait pas actif.

```
1 ssh-add ~/.ssh/id_rsa
```

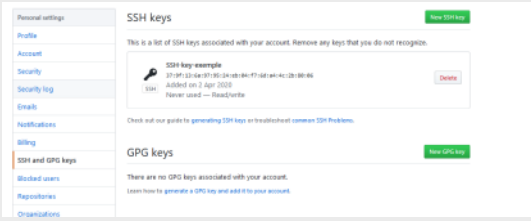
La commande `ssh-add` permet d'ajouter une clé à l'agent SSH en lui spécifiant le chemin où elle se trouve.

```
1 clip < ~/.ssh/id_rsa.pub
```

La commande `clip` permet de copier dans le presse-papier le contenu du fichier `id_rsa.pub` qui contient la clé publique.



Il ne reste plus qu'à coller la clé publique dans le champ permettant l'ajout d'une clé SSH dans les paramètres du compte GitHub.



On peut trouver la liste des clés SSH d'un compte dans l'onglet **SSH and GPG keys** du compte GitHub.

Fondamental **Synchroniser un projet local avec un dépôt distant**

Une fois le dépôt distant créé sur GitHub et l'accès SSH configuré, il faut à présent synchroniser le projet versionné localement avec son dépôt distant.

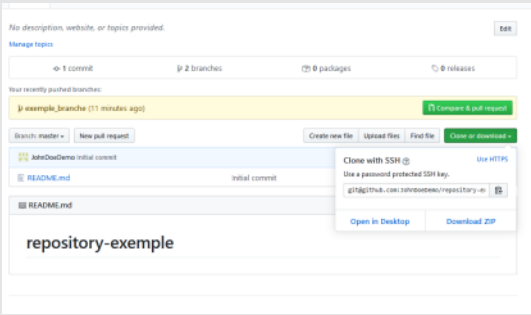
Pour cela, il faut préciser à Git comment s'y interfacer, au moyen de la commande `git remote add`.

Celle-ci attend en paramètre le nom du dépôt distant. Par convention, le dépôt par défaut est nommé **origin**. Il faut également préciser l'URL d'accès à ce dépôt.

Il est possible de lister les différents dépôts auxquels le projet est rattaché, grâce à la commande `git remote -v`.

Méthode **Ajouter un dépôt distant**

```
1 git remote add origin git@github.com:JohnDoeDemo/repository-exemple.git
```



L'URL d'accès au dépôt distant peut être récupérée dans l'interface de GitHub grâce au bouton **Clone or download**.

```
1 git remote -v
```

```
$ git remote -v
origin  git@github.com:JohnDoeDemo/repository-exemple.git (fetch)
origin  git@github.com:JohnDoeDemo/repository-exemple.git (push)
```

La commande `git remote -v` affiche la liste des dépôts associés au projet.

Complément **Afficher les détails d'un dépôt configuré**

Afin de visualiser les détails d'un dépôt déjà configuré, il est possible d'utiliser la commande `git remote show nom_du_depot`.

Cela permet alors d'afficher la liste des URL configurées pour le dépôt distant, ainsi que la liste des branches distantes suivies.

Cette commande informe également l'utilisateur que, s'il est sur la branche `master` et qu'il lance `git pull`, il va automatiquement fusionner la branche `master` du dépôt distant après avoir récupéré toutes les références sur le serveur distant. Cela donne aussi la liste des autres références qu'il aura tirées.

```
remote origin
  Fetch URL: git@github.com:url_depot.git
  Push  URL: git@github.com:url_depot.git
  HEAD branch: master
  Remote branches:
    master     tracked
    new-branch tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)

```

Fondamental **À retenir**

- GitHub est un outil permettant la centralisation du code source d'une application afin de le partager avec tous les membres d'une équipe.
- Il faut pour cela créer un dépôt, qui sera l'espace de stockage distant du projet.
- L'identification d'un compte GitHub peut être faite en SSH grâce à l'utilisation d'une paire de clés publique/privée.
- Il est possible de synchroniser un dépôt Git local avec un dépôt Git distant grâce à la commande `git remote add`.
- Il est possible de lister tous les dépôts d'un projet avec `git remote -v` et les détails d'un dépôt en particulier peuvent être visualisés avec la commande `git remote show nom_du_depot`.

III. Exercice : Appliquez la notion

Sur votre environnement local, créez un nouveau projet, que vous appellerez "`entrepot-distant`".

Initialisez un dépôt Git dans ce répertoire.

```
mkdir entrepot-distant
cd entrepot-distant
git init
```

Question

[solution n°1 p.19]

Il est temps de synchroniser votre premier projet avec votre compte GitHub.

Pour cela :

- Créez un compte GitHub[1] si ce n'est pas encore fait.
- Créez un dépôt intitulé `entrepot-distant` qui contiendra le projet préalablement initialisé.
- Synchronisez ce dépôt distant avec votre projet créé localement. Vous conserverez le nom de dépôt utilisé conventionnellement, à savoir `origin`.
- Vérifiez que votre dépôt a bien été lié à votre projet local.

IV. Travailler avec un dépôt distant

Objectifs

- Partager son code avec Git
- Récupérer le code commun avec Git

Mise en situation

Une fois un projet hébergé sur un dépôt distant, chaque membre de l'équipe va pouvoir y récupérer le code commun et y déposer ses propres modifications. Nous allons voir comment il est possible d'interagir entre un dépôt local et un dépôt distant.

Fondamental **Récupérer le code distant**

Lorsque l'on travaille en équipe avec un gestionnaire de version, la première étape avant de commencer le développement d'une nouvelle tâche consiste à mettre à jour son projet local avec les modifications présentes sur le dépôt distant.

En effet, il faut toujours considérer que l'état du projet sur le dépôt distant constitue la référence de l'avancement d'un projet. Cela signifie qu'il faut à chaque fois synchroniser son projet local avec le projet distant pour s'assurer d'intégrer toutes les modifications ayant pu intervenir depuis la dernière récupération.

Pour cela, Git propose deux commandes permettant de récupérer les données stockées dans le dépôt distant : `git fetch` et `git pull`.

Méthode **La commande git fetch**

La commande `git fetch` compare l'état du projet local et récupère tous les commits manquants de la branche courante depuis le dépôt distant.

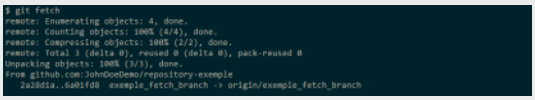
Cette commande ne télécharge pas les fichiers modifiés, mais uniquement les metadata permettant de connaître les différences existantes.

De cette manière, il est possible d'être informé des modifications à venir et de ne décider de les intégrer, avec la commande `git merge`, qu'après s'être assuré qu'elles n'auront pas d'impact sur les développements en cours.

1 https://github.com/

Exemple

```
1 git fetch
```



L'utilisation de la commande `git fetch` récupère les informations liées à la branche `exemple_fetch_branch` présente sur le dépôt distant, mais aucune modification réelle du projet local n'a été effectuée.

```
1 git merge
```

La commande `git merge` va quant à elle intégrer localement les modifications présentes sur le dépôt distant.

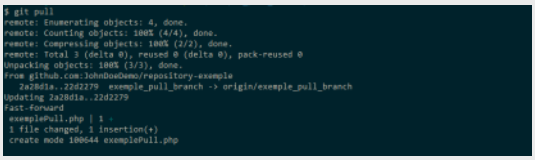
Méthode **La commande git pull**

La commande `git pull` permet d'effectuer les commande `git fetch` et `git merge` en une seule fois.

Ainsi, lorsque `git pull` est effectuée, les modifications présentes sur le dépôt distant sont immédiatement intégrées au projet local.

Exemple

```
1 git pull
```



L'utilisation de la commande `git pull` a bien récupéré et intégré localement les modifications existantes sur le dépôt distant.

Méthode **La commande git push**

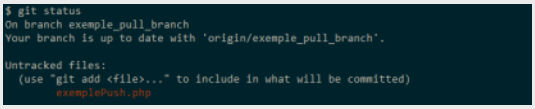
Une fois les développements d'une branche terminés, il faut mettre à jour le dépôt distant afin que les autres membres de l'équipe puissent récupérer les modifications apportées.

Pour cela, il va falloir pousser les commits réalisés en local vers le dépôt distant grâce à la commande `git push`.

Cette commande prend en paramètres le nom du dépôt distant suivi du nom de la branche cible à mettre à jour.

Exemple

```
1 git status
```

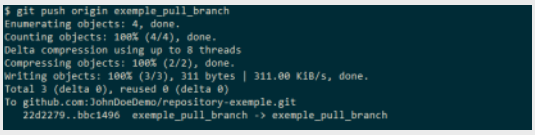


L'utilisation de la commande `git status` montre qu'un nouveau fichier, `examplePush.php`, est présent sur le dépôt local.

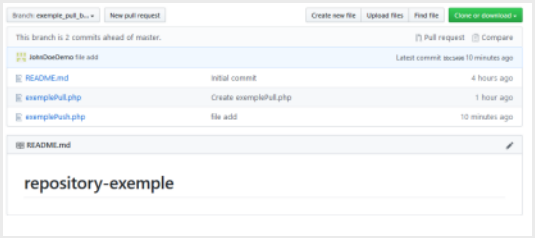
```
1 git add .
2 git commit -m 'file add'
```

Un nouveau commit contenant les modifications est créé.

```
1 git push origin exemple_pull_branch
```



La commande `git push` envoie les modifications locales vers le dépôt distant nommé `origin` et met à jour la branche `exemple_pull_branch`.



Sur l'interface du dépôt distant de GitHub, lorsque l'on se positionne sur la branche modifiée, on peut voir le fichier ajouté par le commit précédent.

Complément

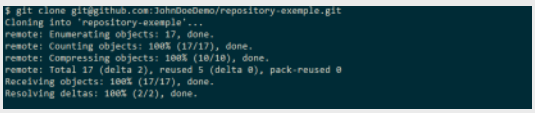
Lorsque le projet a déjà été initié sur un dépôt distant au moment où nous intervenons sur celui-ci, il est possible de créer un dépôt local directement depuis une référence distante.

Pour cela, nous utilisons la commande `git clone` qui attend en paramètre l'URL d'accès au dépôt distant.

L'utilisation de cette commande présente l'avantage de configurer automatiquement la synchronisation vers le dépôt distant.

Exemple

```
1 git clone git@github.com:JohnDoeDemo/repository-exemple.git
```



La commande `git clone` va automatiquement créer un dépôt Git local lié au dépôt distant.

Fondamental **À retenir**

- Travailler avec un dépôt Git distant implique de toujours récupérer la dernière version du projet grâce aux commandes `git fetch` et `git merge` ou `git pull`, avant de pousser ses propres modifications locales vers le dépôt distant grâce à la commande `git push`.
- Lorsque l'on travaille sur un projet existant, il faut utiliser la commande `git clone` pour récupérer le projet localement : de cette manière, un dépôt local automatiquement lié au dépôt distant est créé.

V. Exercice : Appliquez la notion

À ce stade, vous disposez désormais d'un dépôt distant et d'un projet nommé "`entrepot-distant`", configuré lors de l'exercice précédent.

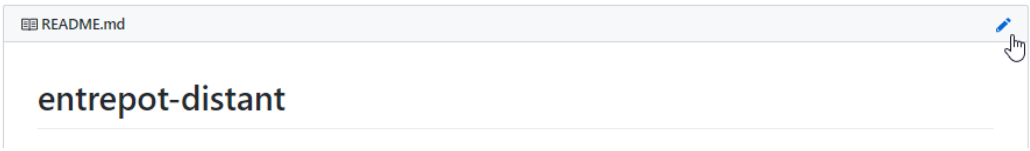
Il est temps d'interagir avec celui-ci.

Question 1

[solution n°2 p.19]

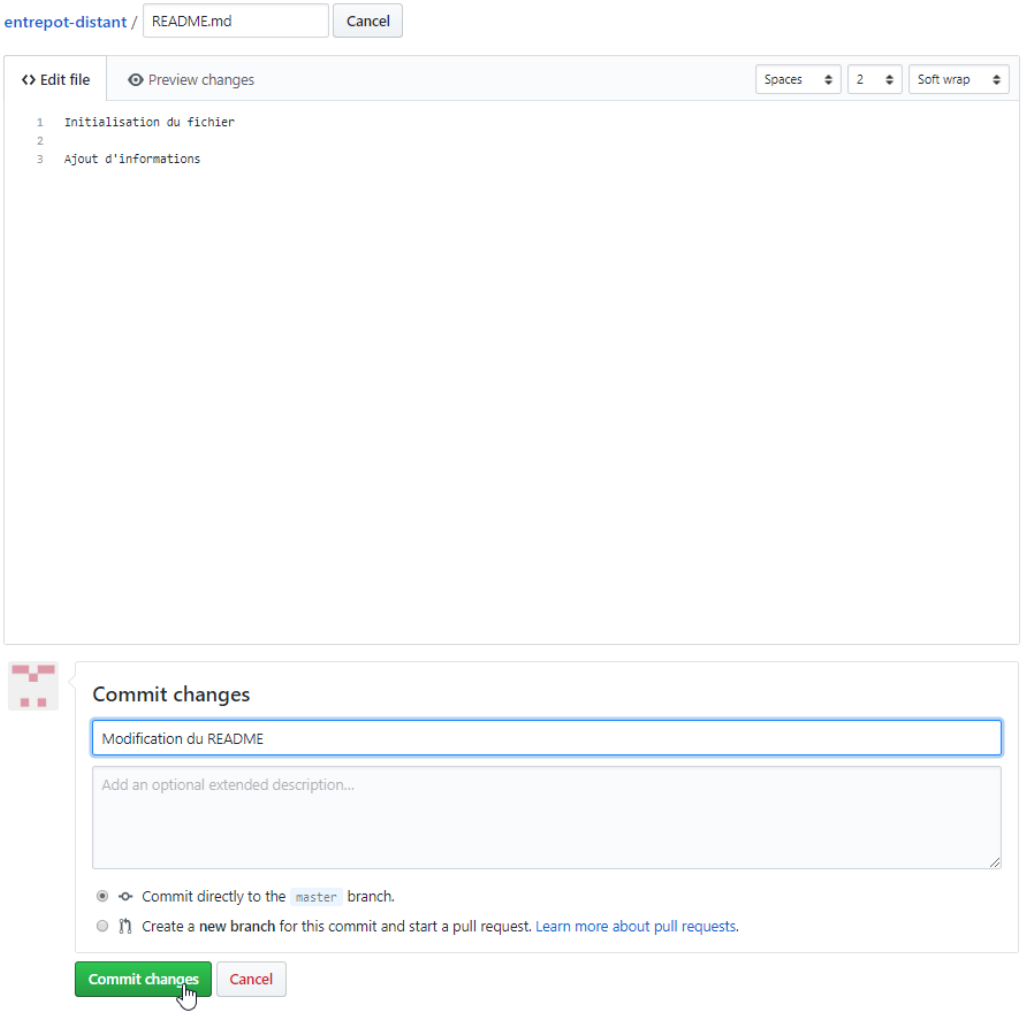
1. Sur votre environnement local, créez un fichier `README.md` contenant le texte "*Initialisation du fichier*".
2. Commitez ces modifications avec le message suivant : "*Initialisation du README*".
3. Mettez à jour votre dépôt distant en poussant ces modifications à l'aide de la commande `git push origin`.
4. Vérifiez la présence de celles-ci.

Il est possible de modifier un fichier directement depuis l'interface de GitHub.



Effectuez une modification dans le fichier `README.md` précédemment créé en y ajoutant la ligne suivante : "*Ajout d'informations*".

Commitez ces modifications avec le message "*Modification du fichier*".



Question 2

[solution n°3 p.19]

Depuis votre environnement local, récupérez les modifications effectuées sur votre dépôt distant à l'aide de la commande `git pull`.

Vérifiez que vous disposez du dernier commit à l'aide de la commande `git log`.

VI. Naviguer dans GitHub

Objectif

- Apprendre à explorer un projet dans GitHub

Mise en situation

GitHub est un outil proposant un service d'hébergement de projets versionnés avec Git, utilisé par de nombreux développeurs.

Il propose une interface complète permettant de naviguer au sein d'un projet : nous allons voir comment retrouver des informations en explorant l'historique d'un projet.

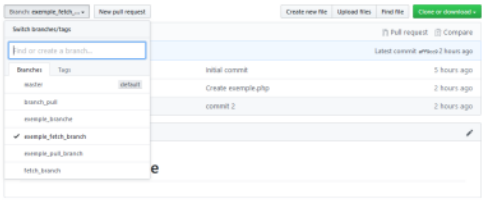
Méthode **Naviguer à travers un projet**

Sur la page principale d'un dépôt, GitHub présente l'arborescence des fichiers de la branche sélectionnée.

Il est donc possible de visualiser les fichiers présents sur chaque branche en utilisant la liste déroulante de sélection d'une branche au-dessus du tableau.

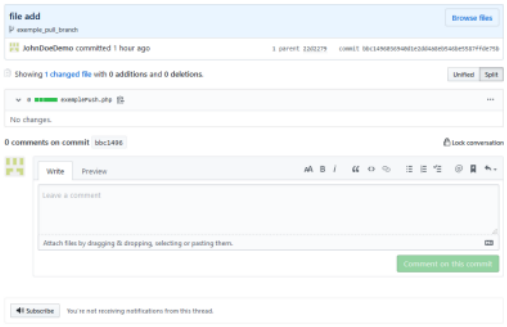
De plus, ce dernier affiche pour chaque fichier le commentaire lié au dernier commit l'ayant modifié, ainsi que la date à laquelle celui-ci a été effectué.

Parcourir les branches



Sur l'exemple présenté, on retrouve la liste des branches sélectionnables, ainsi que le tableau contenant les fichiers associés à la branche en cours.

Naviguer vers un commit spécifique



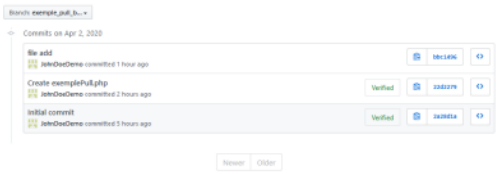
Il est possible de naviguer depuis cet écran vers le contenu d'un commit, en cliquant sur le commentaire du commit souhaité.

Cette page présente le contenu d'un commit et affiche les modifications, ajouts et suppressions qui ont pu être faits.

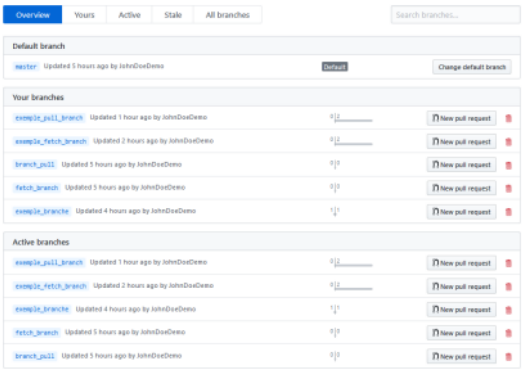
Afficher les commits d'une branche

De plus, depuis la page principale, on peut accéder à la liste des commits de la branche en cours, ainsi qu'à la liste des branches existantes sur le repository distant.

Afficher le détail d'un commit



Depuis la liste des commits, on peut accéder au contenu de chacun d'entre eux. On y retrouve aussi le tag et les identifiants.



Depuis la liste des branches existantes, on peut voir rapidement les branches possédant des commits non mergés sur la branche principale, `master`.

Syntaxe **À retenir**

- L'utilisation d'outils comme GitHub permet d'avoir accès à une interface web complète.
- Il est possible de naviguer à travers les branches et commits existants, ainsi que de visualiser le contenu des fichiers en fonction de leur état dans la branche choisie.

VII. Exercice : Appliquez la notion

Rendez-vous sur le dépôt suivant : https://github.com/symfony/symfony.

Question 1

[solution n°4 p.20]

Identifiez le nombre de commits présents sur ce dépôt.

Question 2

[solution n°5 p.20]

Identifiez le nombre de branches que le dépôt contient.

Question 3

[solution n°6 p.20]

Affichez le détail du dernier commit de la branche 2.8.

Question 4

[solution n°7 p.22]

Affichez les fichiers contenus dans le commit précédent.

VIII. GitLab, une alternative à GitHub

Objectif

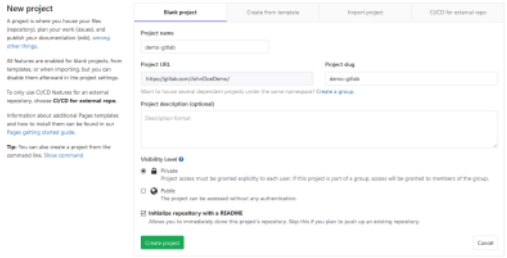
- Découvrir un autre outil d'hébergement

Mise en situation

Il existe différents outils proposant des services d'hébergement de dépôts Git. Parmi eux, on retrouve GitHub, mais aussi d'autres alternatives telles que GitLab, BitBucket, SourceForge et bien d'autres encore.

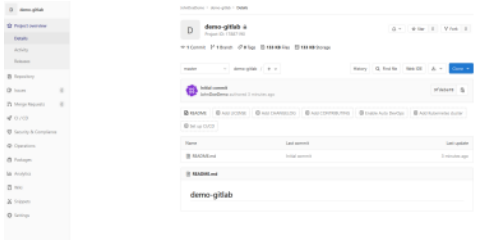
Nous allons ici explorer l'interface proposée par GitLab afin d'en comprendre le fonctionnement et les possibilités.

Création d'un nouveau dépôt



La création d'un dépôt sur GitLab se fait en y créant un nouveau projet et en remplissant les champs du formulaire de création.

Comme sur GitHub, il est possible de définir la portée du projet, privée ou publique, ainsi que d'ajouter automatiquement un fichier `README`.

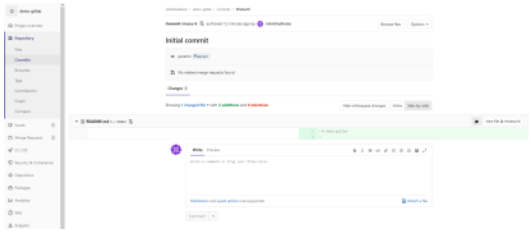


La page principale du dépôt présente l'arborescence de fichiers correspondant à la branche en cours, que l'on peut sélectionner grâce à la liste déroulante de sélection des branches.

Remarque

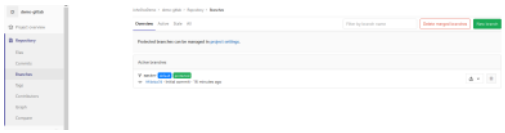
À la différence de GitHub qui préfère afficher un bandeau de navigation horizontal en haut de la page, GitLab affiche un panneau de navigation vertical à gauche de la page.

D'une manière générale, les options de navigation entre les deux outils sont très similaires.



Il est possible d'accéder au contenu d'un commit d'un clic sur le commentaire associé depuis l'arborescence des fichiers d'une branche.

La liste des commits d'une branche peut être visualisée depuis l'onglet **Commits** du panneau de navigation de gauche.



La liste des branches existantes sur un dépôt est accessible depuis l'onglet **Branches** du panneau de navigation de gauche.

Syntaxe **À retenir**

- GitHub et GitLab sont deux outils d'hébergement de dépôts distants Git.
- Malgré leurs différences d'interface, les possibilités qu'ils offrent en termes d'hébergement de code sont très proches.
- Le choix d'un outil par rapport à l'autre revient donc aux préférences de chacun.

IX. Exercice : Appliquez la notion

Question

[solution n°8 p.22]

Il est temps de découvrir Gitlab par vous-même.

- Créez un compte Gitlab[1] si ce n'est pas encore fait.
- Créez un dépôt que vous appellerez "`entrepot-distant`".
- Synchronisez ce dépôt distant avec le projet local créé lors des exercices précédents, intitulé "`entrepot-distant`". Vous indiquerez comme origine "`gitlab`".
- Vérifiez que votre dépôt a bien été lié à votre projet local.

X. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°9 p.22]

Exercice

Qu'est-ce que GitHub ?

- ○ Un service de messagerie
- ○ Un service d'hébergement de dépôts Git distants
- ○ Un service d'analyse de code

Exercice

1 https://gitlab.com/

Quelle commande permet d'ajouter un nouveau dépôt distant ?

- git remote add nom_du_depot
- git add remote nom_du_depot
- git remote new nom_du_depot

Exercice

Quelle commande permet de lister l'ensemble des dépôts distants ?

- git add
- git remote -v
- git list-remote

Exercice

Quelle commande permet d'afficher les détails d'un dépôt distant ?

- git remote show nom_du_depot
- git show remote nom_du_depot
- git remote-details nom_du_depot

Exercice

Grâce à quelle commande est-il possible de mettre à jour le dépôt distant par rapport au dépôt local ?

- git push
- git pull
- git remote update

Exercice

Quelle commande compare l'état du projet local et récupère tous les commits manquants de la branche courante depuis le dépôt distant sans les télécharger ?

- git pull
- git push
- git fetch

Exercice

Quelle commande supplémentaire effectue la commande `git pull` par rapport à la commande `git fetch` ?

- git status
- git add
- git merge

Exercice

Conventionnellement, quel nom est-il donné au dépôt distant par défaut ?

- ○ default
- ○ origin
- ○ init
- ○ source

Exercice

Un projet peut disposer de plusieurs dépôts distants configurés.

- ○ Vrai
- ○ Faux

Exercice

Quelles alternatives à GitHub pouvez-vous citer ?

- ☐ GitLab
- ☐ Microsoft Teams
- ☐ BitBucket
- ☐ SourceForge
- ☐ Zoom

B. Exercice : Défi

On appelle *fork* la duplication d'un dépôt.

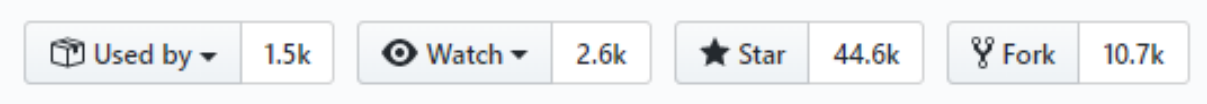
Forker un dépôt permet par exemple d'expérimenter des modifications sur du code, sans apporter de modifications au projet initial.

Les forks peuvent par exemple permettre d'utiliser le projet de quelqu'un d'autre, en s'en servant de point de départ.

GitHub permet de forker simplement un projet. Pour cela, vous pouvez vous rendre sur la page de n'importe quel projet et cliquer sur le bouton correspondant dans l'interface.

Vous pouvez par exemple vous rendre ici : https://github.com/hakimel/reveal.js et cliquer sur le bouton **Fork** présent en haut de la page.

Ce projet propose de présenter des slides à partir de code HTML. Vous pourriez très bien envisager de l'adapter si vous l'utilisiez régulièrement, afin d'y intégrer systématiquement des modifications que vous effectueriez.



Question

[solution n°10 p.24]

À vous de réaliser votre premier fork !

Forkez un projet qui vous tient à cœur (ou celui donné en exemple : https://github.com/hakimel/reveal.js) et clonez ce projet pour le récupérer sur votre environnement local.

Créez un fichier intitulé "`Custom.md`" dans lequel vous indiquerez "`Détail des modifications apportées`". Commitez ces modifications en indiquant le message suivant "*Création du fichier Custom.md*".

Rendez-vous sur GitLab et créez-y un nouveau dépôt.

Mettez à jour la liste de vos dépôts sur votre environnement local, vous allez devoir :

- Renommer le remote "`origin`" en "`old-origin`"
- Définir votre dépôt GitLab comme étant votre nouveau dépôt `origin`
- Affichez et vérifiez la liste de vos remotes
- Affichez les détails du remote "`origin`"
- Supprimez le remote "`old-origin`"

Poussez ensuite votre code sur votre dépôt et vérifiez que tout est en ordre.

Solutions des exercices

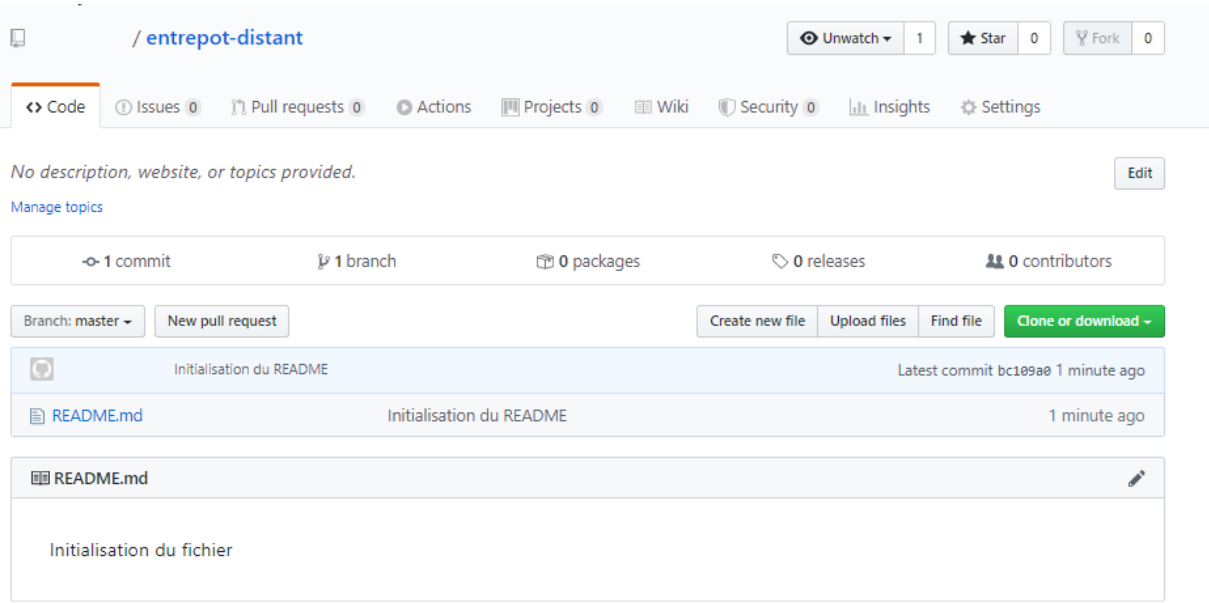
p. 8 Solution n°1

- Votre compte GitHub doit être créé à l'adresse suivante : https://github.com/join.
- Une fois connecté au service, vous pouvez créer un nouveau dépôt en suivant les instructions à l'adresse suivante : https://github.com/new.
- Lorsque votre dépôt aura été créé, vous devez exécuter la commande `git remote add origin url_de_votre_depot`. Pour rappel, l'URL d'accès au dépôt distant peut être récupérée dans l'interface de GitHub grâce au bouton **Clone or download**.
- Une fois la dernière commande exécutée, vous pouvez vérifier que tout est en ordre en exécutant la commande `git remote -v`.

p. 11 Solution n°2

```
# Modification d'un fichier
touch README.md
echo 'Initialisation du fichier' > README.md
git add README.md
git commit -m "Initialisation du README"

# Mise à jour de la branche master sur le dépôt "origin"
git push origin master
```



p. 12 Solution n°3

Vous avez dû effectuer les commandes suivantes :

```
# Récupération de la branche master sur le dépôt "origin"
git pull origin master

# Vérification de la présence du commit
git log

```

```
commit c56b99cc033021965c53f0d0232a3fe04493afcd (HEAD -> master, origin/master)
Date:   Mon May 11 17:34:37 2020 +0200

    Modification du README

commit bc109a02fb47ad0dad934718339e0c9086fe673c
Date:   Mon May 11 17:01:16 2020 +0200

    Initialisation du README
```

p. 13 **Solution n°4**

Ces éléments varieront nécessairement lorsque vous effectuerez cet exercice.

Néanmoins, à ce jour, le dépôt contenait 48 475 commits.

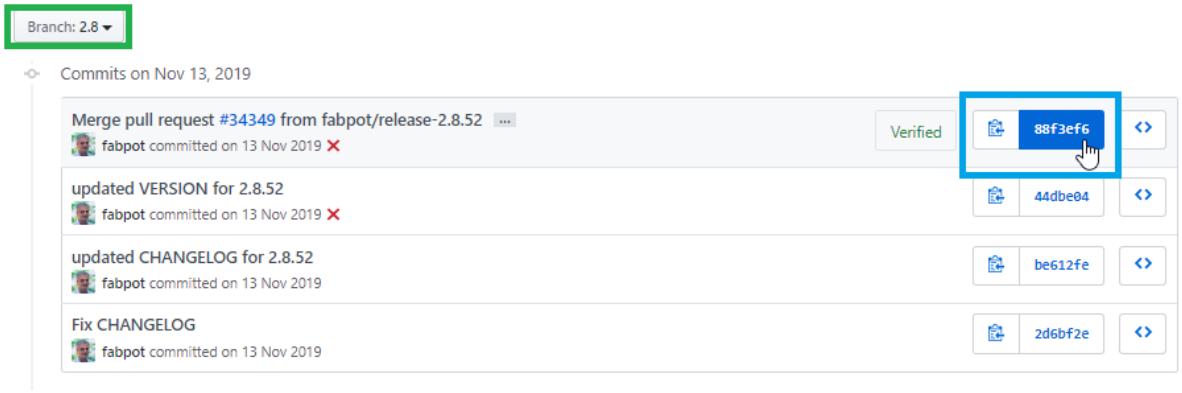
p. 13 **Solution n°5**

Ces éléments varieront nécessairement lorsque vous effectuerez cet exercice.

Néanmoins, à ce jour, le dépôt contenait 21 branches.

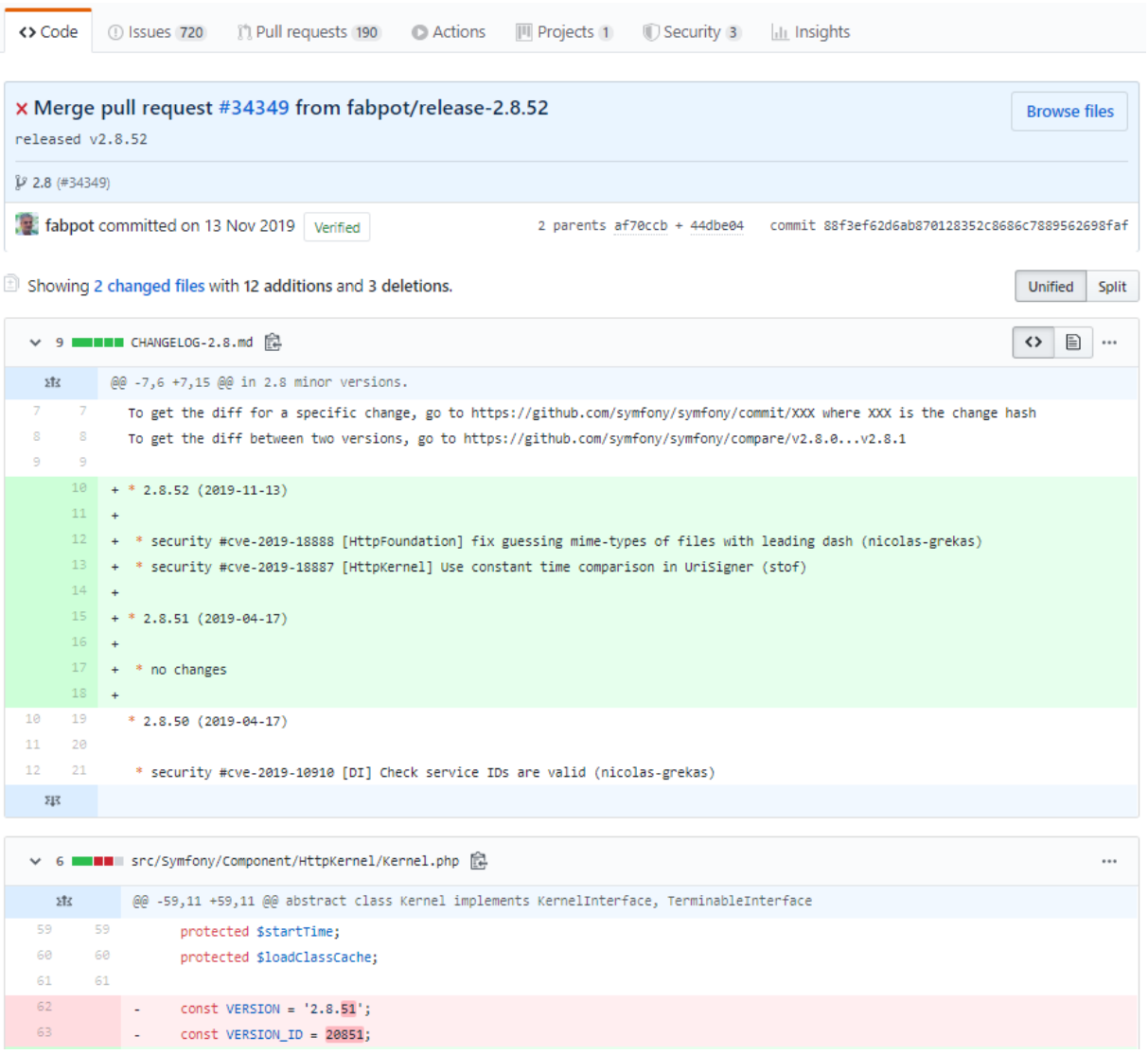
p. 13 **Solution n°6**

Concernant le dernier commit de la branche 2.8, voici comment y accéder :



Il est possible d'accéder à cette branche directement : https://github.com/symfony/symfony/commits/2.8.

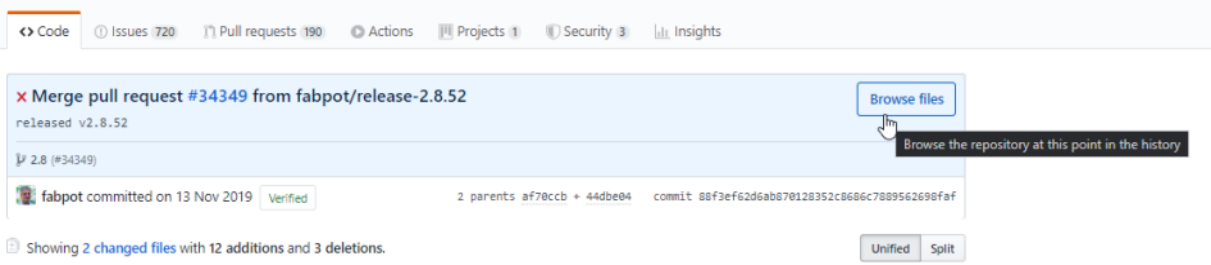
Concernant l'état des derniers fichiers modifiés, on peut accéder au commit via cette URL : https://github.com/symfony/symfony/commit/88f3ef62d6ab870128352c8686c7889562698faf.



p. 13 Solution n°7

On peut afficher le détail en cliquant sur le bouton "**Browse files**".

Le lien direct est le suivant : https://github.com/symfony/symfony/tree/88f3ef62d6ab870128352c8686c7889562698faf.



p. 15 Solution n°8

Votre compte Gitlab doit être créé à l'adresse suivante : https://gitlab.com/users/sign_in#register-pane.

Une fois connecté au service, vous pouvez créer un nouveau dépôt en suivant les instructions à cette adresse : https://gitlab.com/projects/new.

Lorsque votre dépôt aura été créé, vous devez exécuter la commande `git remote add gitlab url_de_votre_depot`. Pour rappel, l'URL d'accès au dépôt distant peut être récupérée dans l'interface de GitHub grâce au bouton **Clone or download**.

Une fois la dernière commande exécutée, vous pouvez vérifier que tout est en ordre en exécutant la commande `git remote -v`.

Exercice p. 15 Solution n°9

Exercice

Qu'est-ce que GitHub ?

- ○ Un service de messagerie
- ◉ Un service d'hébergement de dépôts Git distants
- ○ Un service d'analyse de code

Exercice

Quelle commande permet d'ajouter un nouveau dépôt distant ?

- ◉ git remote add nom_du_depot
- ○ git add remote nom_du_depot
- ○ git remote new nom_du_depot

Exercice

Quelle commande permet de lister l'ensemble des dépôts distants ?

- ○ git add
- ◉ git remote -v
- ○ git list-remote

Exercice

Quelle commande permet d'afficher les détails d'un dépôt distant ?

- ◉ git remote show nom_du_depot
- ○ git show remote nom_du_depot
- ○ git remote-details nom_du_depot

Exercice

Grâce à quelle commande est-il possible de mettre à jour le dépôt distant par rapport au dépôt local ?

- ◉ git push
- ○ git pull
- ○ git remote update

Exercice

Quelle commande compare l'état du projet local et récupère tous les commits manquants de la branche courante depuis le dépôt distant sans les télécharger ?

- ○ git pull
- ○ git push
- ◉ git fetch

Exercice

Quelle commande supplémentaire effectue la commande `git pull` par rapport à la commande `git fetch` ?

- ○ git status
- ○ git add
- ◉ git merge

Exercice

Conventionnellement, quel nom est-il donné au dépôt distant par défaut ?

- ○ default
- ◉ origin
- ○ init
- ○ source

Exercice

Un projet peut disposer de plusieurs dépôts distants configurés.

- (●) Vrai
- ( ) Faux

Exercice

Quelles alternatives à GitHub pouvez-vous citer ?

- [x] GitLab
- [ ] Microsoft Teams
- [x] BitBucket
- [x] SourceForge
- [ ] Zoom

p. 17

Solution n°10

Voici la liste des commandes que vous avez dû effectuer :

```
git clone git@github.com:hakimel/reveal.js.git
cd reveal.js
# Modification d'un fichier
touch Custom.md
echo 'Détail des modifications apportées' > Custom.md
git status
git add .
git commit -m "Création du fichier Custom.md"
# Création du dépôt GitLab
git remote rename origin old-origin
git remote add origin https://gitlab.com/_username_/reveal.js.git
git remote show origin
git remote remove old-origin
git push origin master
```