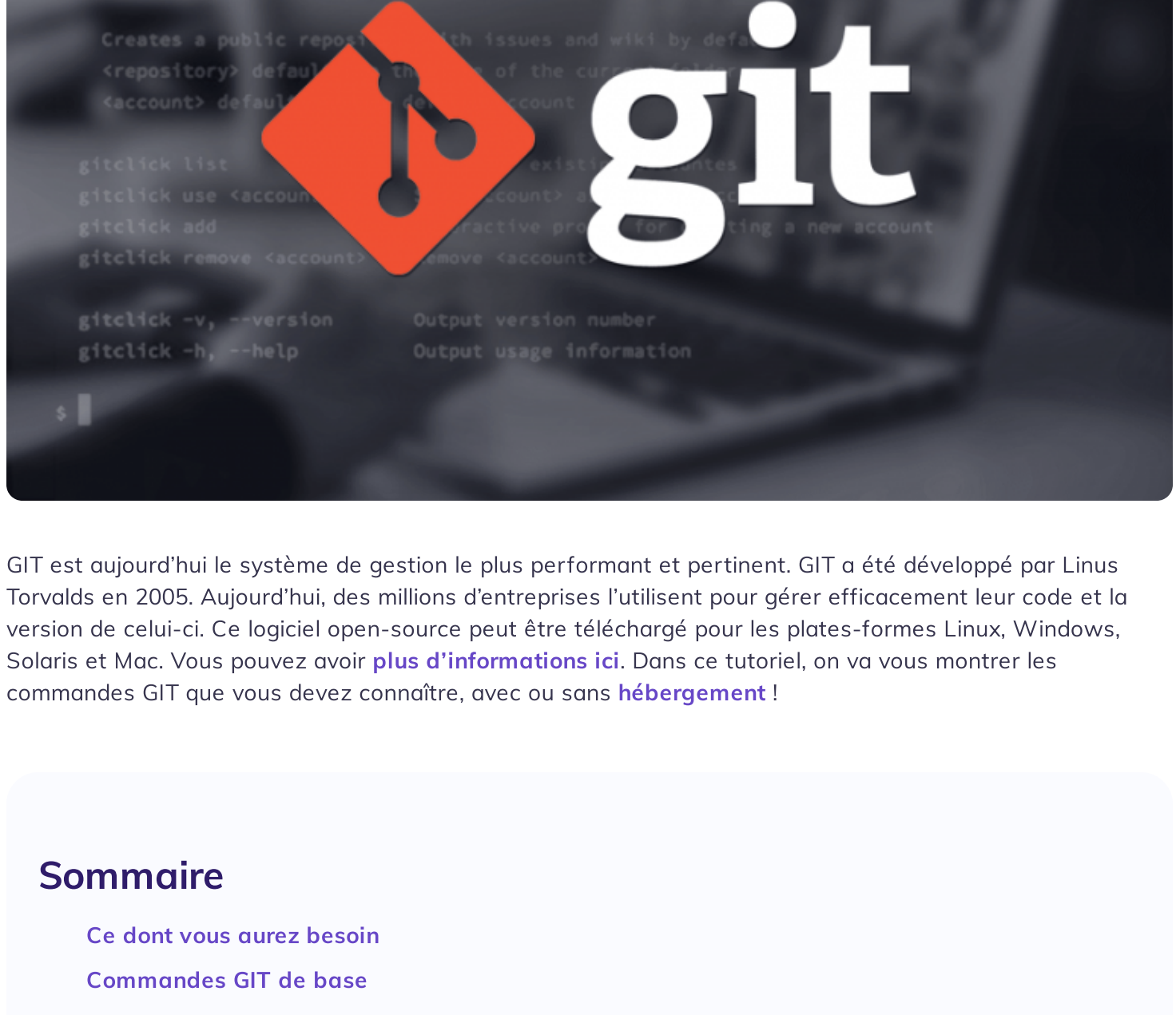


# Les commandes GIT que vous devez absolument connaître !

🔗 🐦 📘 🔖 ✉



GIT est aujourd'hui le système de gestion le plus performant et pertinent. GIT a été développé par Linus Torvalds en 2005. Aujourd'hui, des millions d'entreprises l'utilisent pour gérer efficacement leur code et la version de celui-ci. Ce logiciel open-source peut être téléchargé pour les plates-formes Linux, Windows, Solaris et Mac. Vous pouvez avoir plus d'informations ici. Dans ce tutoriel, on va vous montrer les commandes GIT que vous devez connaître, avec ou sans hébergement !

Sommaire

[Ce dont vous aurez besoin](#)


[Commandes GIT de base](#)

[Conclusion](#)

## Ce dont vous aurez besoin

Avant de nous attaquer aux commandes à proprement parler, assurez-vous d'avoir les éléments suivants:

- GIT d'installé sur votre système ou VPS

 Offre d'une durée limitée ! Jusqu'à **77 % de réduction** sur l'hébergement VPS aujourd'hui.

DECOUVRIR

## Commandes GIT de base

### Git config

- L'une des commandes git les plus utilisées est **git config**. On l'utilise pour configurer les préférences de l'utilisateur : son mail, l'algorithme utilisé pour diff, le nom d'utilisateur et le format de fichier etc. Par exemple, la commande suivante peut être utilisée pour définir le mail d'un utilisateur:

```
git config --global user.email sam@google.com
```

### Git init

- Cette commande est utilisée pour créer un nouveau dépôt GIT :

```
git init
```

### Git add

- La **commande git add** peut être utilisée pour ajouter des fichiers à l'index. Par exemple, la commande suivante ajoutera un fichier nommé temp.txt dans le répertoire local de l'index:

```
git add temp.txt
```

### Clone git

- La **commande git clone** est utilisée pour la vérification des dépôts. Si le dépôt se trouve sur un serveur distant, utilisez:

```
git clone alex@93.188.160.58:/chemin/vers/dépôt
```

- Inversement, si une copie de travail d'un dépôt local doit être créée, utilisez:

```
git clone /chemin/vers/dépôt
```

### Git commit

- La **commande git commit** permet de valider les **modifications apportées** au HEAD. Notez que tout commit ne se fera pas dans le dépôt distant.

```
git commit -m "Description du commit"
```

### Git status

- La **commande git status** affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés. Usage:

```
git status
```

### Git push

- **Git push** est une autre commandes GIT de base. Un simple push envoie les modifications locales apportées à la branche principale associée :

```
git push origin master
```

### Git checkout

- La **commande git checkout** peut être utilisée pour créer des branches ou pour basculer entre elles. Par exemple nous allons créer une branche:

```
command git checkout -b <nom-branch>
```

- Pour passer simplement d'une branche à une autre, utilisez:

```
git checkout <non-branch>
```

### Git remote

- Cette commande **remote** permet à un utilisateur de se connecter à un dépôt distant. La commande suivante répertorie les dépôts distants actuellement configurés:

```
git remote -v
```

- Cette commande permet à l'utilisateur de connecter le dépôt local à un serveur distant:

```
git remote add origin <93.188.160.58>
```

### Branche git

- La **commande git branch** peut être utilisée pour répertorier, créer ou supprimer des branches. Pour répertorier toutes les branches présentes dans le dépôt, utilisez:

```
git branch
```

- Pour supprimer une branche:

```
git branch -d <nom-branch>
```

### Git pull

- Pour fusionner toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local, la commande pull est utilisée. Usage:

```
git pull
```

### Git merge

- La **commande git merge** est utilisée pour fusionner une branche dans la branche active. Usage:

```
git merge <nom-branch>
```

### Git diff

- La **commande git diff** permet de lister les conflits. Pour visualiser les conflits d'un fichier, utilisez

```
git diff --base <nom-fichier>
```

- La commande suivante est utilisée pour afficher les conflits entre les branches à fusionner avant de les fusionner:

```
git diff <branche-source> <branche-cible>
```

- Pour simplement énumérer tous les conflits actuels, utilisez:

```
git diff
```

### Git tag

- Le marquage est utilisé pour marquer des commits spécifiques avec des poignées simples. Un exemple peut être:

```
git tag 1.1.0 <insert-commitID-here>
```

### Git log

- L' **exécution** de cette commande génère le log d'une branche. Un exemple de sortie :

```
commit 15f4b8c44b3c8344ca5adac9e4be13246e21sadb
Author: Alex Hunter <alex@hga11.com>
Date: Mon Oct 1 12:56:29 2016 -0600
```

### Git reset

- Pour réinitialiser l'index et le répertoire de travail à l'état du dernier commit, la **commande git reset** est utilisée :

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

### Git rm

- **Git rm** peut être utilisé pour supprimer des fichiers de l'index et du répertoire de travail. Usage:

```
git rm nomfichier.txt
```

### Git stash

- L'une des moins connues, **git stash** aide à enregistrer les changements qui ne doivent pas être commit immédiatement. C'est un commit temporaire. Usage:

```
git stash
```

### Git show

- Pour afficher des informations sur tout fichier git, utilisez la **commande git show** . Par exemple:

```
git show
```

### Git fetch

- **Git fetch** permet à un utilisateur d'extraire tous les fichiers du dépôt distant qui ne sont pas actuellement dans le répertoire de travail local. Exemple d'utilisation:

```
git fetch origin
```

### Git ls-tree

- Pour afficher un fichier arborescent avec le nom et le mode de chaque élément, et la valeur SHA-1 du blob, utilisez la **commande git ls-tree** . Par exemple:

```
git ls-tree HEAD
```

### Git cat-file

- À l'aide de la valeur SHA-1, affichez le type d'un fichier à l'aide de la **commande git cat-file** . Par exemple:

```
git cat-file -p d67b460b4b4aece5915caf5c68d12f568a9fe3e4
```

### Git grep

- **Git grep** permet à un utilisateur de rechercher dans les arbres de contenu des expressions et / ou des mots. Par exemple, pour rechercher **www.hostinger.com** dans tous les fichiers, utilisez:

```
git grep "www.hostinger.com"
```

### Gitk

- **Gitk** est l'interface graphique du dépôt local. Vous pouvez l'appeler en exécutant:

```
gitk
```

### Git instaweb

- Avec la **commande git instaweb** , un serveur Web peut être exécuté par interface avec le dépôt local. Qui redirige directement vers un serveur web. Par exemple:

```
git instaweb -http=webrick
```

### Git gc

- Pour optimiser le dépôt en supprimant les fichiers inutiles et les optimiser, utilisez:

```
git gc
```

### git archive

- La **commande git archive** permet à un utilisateur de créer un fichier zip ou tar contenant les composants d'un arbre du dépôt. Par exemple:

```
git archive --format=tar master
```

### Git prune

- Via la **commande git prune** , les fichiers qui n'ont pas de pointeurs entrants seront supprimés. Usage:

```
git prune
```

### Git fsck

- Pour effectuer une vérification d'intégrité du système de fichiers git, utilisez la commande **git fsck** . Tous les fichiers corrompus seront identifiés:

```
git fsck
```

### Git rebase

- La **commande git rebase** est utilisée pour la réapplication des commits sur une autre branche. Par exemple:

```
git rebase master
```

## Conclusion

Nous venons de vous montrer les commandes de base de GIT. Assurez-vous de consulter [notre tutoriel complet sur GIT](#) pour savoir comment configurer GIT.

J'espère vous avoir été utile et vous dis à très bientôt pour un futur tutoriel !



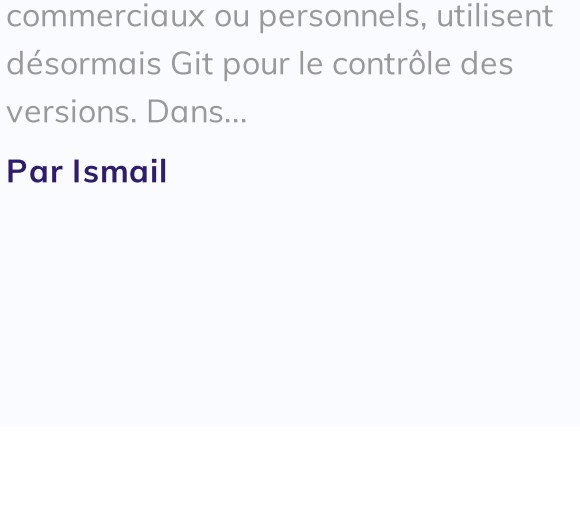
L'AUTEUR

Fatima Zahra

Fatima Zahra est une passionnée du marketing digital et de l'IT, elle fait partie de l'équipe du contenu de notre site web pour apporter à nos chers internautes une meilleure expérience client. Ses passes- temps impliquent le blogging et l'apprentissage des langues pour se connecter véritablement au cœur avec les autres.

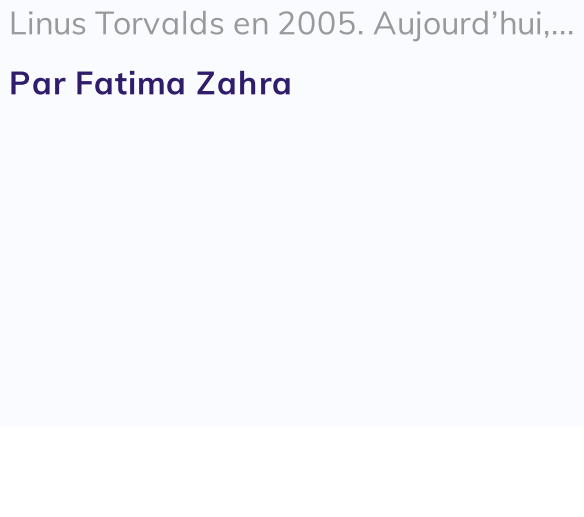
Plus d'informations sur Fatima Zahra

## Tutoriels relatifs



**13 Mai • GIT**  
**Les meilleurs clients Git GUI de 2021 : Toutes les plateformes incluses**  
Presque tous les projets de développement et de logiciels, commerciaux ou personnels, utilisent désormais Git pour le contrôle des versions. Dans...

Par Ismail



**25 Avr • GIT**  
**Les commandes GIT que vous devez absolument connaître !**  
GIT est aujourd'hui le système de gestion le plus performant et pertinent. GIT a été développé par Linus Torvalds en 2005. Aujourd'hui...

Par Fatima Zahra



**24 Avr • GIT**  
**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...


Par Brice Bastly

## Laissez une réponse


Vous devez vous connecter pour publier un commentaire.


Ce site utilise Akismet pour réduire les indésirables. [En savoir plus](#) sur comment les données de vos commentaires sont utilisées.


Excellent



Sur la base de **173 avis**

  
Super STAFF et super Hébergeur  
Le service client est super réactif et amical  
J'ai recommandé ! Un grand merci à...  
Julien Masure  
Nos avis 4 et 5 étoiles

  
18 octobre  
En passant de OVH à Hostinger  
En passant de OVH à Hostinger, notre site a gagné en temps de réponse, et nous é...  
Carl

  
1 octobre  
Une équipe VRAIMENT  
Haut la Vach