📖

GitIgnore

Le fichier `.gitignore` est un fichier essentiel pour indiquer à Git quels fichiers ou répertoires ne doivent pas être suivis (et donc ignorés). Voici tout ce qu'il faut savoir pour le créer et l'utiliser efficacement.

1. Créer un `.gitignore`

Pour créer un fichier `.gitignore`, utiliser simplement une commande dans le terminal :

```
touch .gitignore
```

Ensuite, l'ouvrir avec un éditeur de texte :

```
nano .gitignore
```

2. Contenu d'un `.gitignore`

Voici quelques exemples de ce que l'on peut y inclure :

2.1. Ignorer des fichiers spécifiques

```
# Ignorer des fichiers spécifiques
secret-key.txt
config.env
```

2.2. Ignorer des répertoires

```
# Ignorer le dossier "node_modules"
node_modules/

# Ignorer tout le contenu du dossier "build"
build/
```

2.3. Ignorer des fichiers temporaires ou spécifiques au système

```
# Fichiers générés par les IDE ou éditeurs
*.swp
*.idea/
.vscode/

# Fichiers de sauvegarde
*.bak
*.tmp

# Fichiers système
.DS_Store
Thumbs.db
```

2.4. Ignorer tout sauf certains fichiers

```
# Ignorer tout sauf un fichier
*
!.gitignore
!README.md
```

3. Utiliser des modèles prédéfinis

Si on ne veut pas écrire le `.gitignore` à la main, on peut utiliser des modèles spécifiques au projet. GitHub propose des modèles prêts à l'emploi dans son dépôt `.gitignore`.

Exemple : Si on travaille sur un projet Node.js, on peut récupérer un modèle avec cette commande :

```
curl -o .gitignore https://raw.githubusercontent.com/github/gitignore/main/Node.gitign
ore
```

4. Vérifier si un fichier est ignoré

Pour voir si un fichier ou dossier est ignoré par Git :

```
git check-ignore -v fichier_ou_dossier
```

5. Les fichiers déjà suivis ne sont pas affectés

Si un fichier est déjà suivi par Git (ajouté et versionné), ajouter son nom dans le `.gitignore` ne le supprimera pas automatiquement du suivi.

Solution : Supprimer un fichier du suivi

Si on veut vraiment qu'il soit ignoré :

```
git rm --cached fichier_a_ignorer
```

- **Explication** : Cette commande retire le fichier de l'index (suivi par Git) sans le supprimer physiquement.
- Puis, ajouter le fichier au `.gitignore` :

  ```
  echo "fichier_a_ignorer" >> .gitignore
  ```

5. Ajouter le `.gitignore` au repository (faire un commit)

Après avoir configuré le fichier `.gitignore`, l'ajouter et le pousser vers le repository :

```
git add .gitignore
git commit -m "Ajout du fichier .gitignore"
git push
```

6. Hiérarchie des `.gitignore`

- **Dans chaque répertoire** : nous pouvons avoir plusieurs `.gitignore` à différents niveaux d'un projet. Les règles dans un sous-dossier prévalent sur celles des dossiers parents.

Exemple :

Si le projet a cette structure :

```
.
├── .gitignore
├── src/
│   ├── .gitignore
│   └── main.py
```

- Les fichiers ignorés dans `src/.gitignore` ne seront pas affectés par le `.gitignore` global.

7. Le fichier `.git/info/exclude` (ignores locaux)

Si nous voulons ignorer des fichiers **localement** (sur un ordinateur uniquement, sans affecter les autres contributeurs), on peut utiliser le fichier `.git/info/exclude`.

Exemple d'utilisation :

- Ouvrir ou modifier ce fichier :

  ```
  nano .git/info/exclude
  ```

- Ajouter les règles pour ignorer des fichiers ou dossiers spécifiques à la machine.

8. Wildcards (caractères génériques) dans `.gitignore`

Le `.gitignore` prend en charge des **caractères génériques** puissants pour ignorer des groupes de fichiers.

Quelques exemples :

- **Ignorer tous les fichiers avec une extension spécifique :**

  ```
  *.log
  ```

  Ignorera tous les fichiers `.log`.

- **Ignorer tous les fichiers d'un type spécifique dans tous les sous-dossiers :**

  ```
  **/*.bak
  ```

  Ignorera tous les fichiers `.bak` dans n'importe quel sous-dossier.

- **Ignorer tous les fichiers sauf un :**

  ```
  *.txt
  !important.txt
  ```

  Ignorera tous les fichiers `.txt` sauf `important.txt`.

9. L'ordre des règles compte

- Les règles sont interprétées dans l'ordre, de haut en bas. Si un fichier correspond à plusieurs règles, la dernière règle appliquée prévaut.

Exemple :

```
# Ignorer tous les fichiers .log
*.log

# Mais suivre "important.log"
!important.log
```

10. Bonnes pratiques pour `.gitignore`

1. **Planifier à l'avance :** Mettre en place un `.gitignore` dès le début du projet pour éviter d'avoir des fichiers indésirables versionnés.
2. **Ne jamais mettre de secrets dans le dépôt :** Prévoir d'utiliser le `.gitignore` pour exclure les fichiers contenant des secrets (comme des clés API) ou des mots de passe.
3. **Utiliser des modèles existants :** Comme mentionné, GitHub propose des modèles optimisés pour différents langages/frameworks. Lien vers les modèles `.gitignore`.

11. Déboguer un `.gitignore`

Si un fichier n'est pas ignoré comme prévu, voici quelques étapes pour diagnostiquer :

1. **Vérifie si le fichier est suivi par Git :**

   ```
   git ls-files fichier_en_question
   ```

   Si le fichier apparaît, il est suivi par Git. Le retirer avec `git rm --cached`.

2. **Teste les règles d'ignorance :**

   ```
   git check-ignore -v fichier_en_question
   ```

   Cette commande affiche quelle règle (et quel `.gitignore`) cause l'ignorance.