

# Git Hooks et du Hook Pre-commit

Date de modification	auteur	N°Ticket
14/01/2025	ZEA	TDEV700-37

<b>Git Hooks et du Hook Pre-commit</b>	<b>1</b>
Introduction aux Git Hooks	2
Hooks côté client courants	2
Configuration d'un Hook Pre-commit	2
Création du script du hook Pre-commit dans le fichier hooks/pre-commit	2
Configuration d'un Hook Commit-msg	2
Création du script du hook Pre-commit dans le fichier hooks/commit-msg	3
Explication du Script	3
Exemple de message de commit attendu	3
Cas d'erreur	4
Automatisation de la configuration des hooks avec le Makefile	4
Configuration des hooks après un clonage	4
Utilisation du Hook Pre-commit	5
Utilisation du Hook Commit-msg	5

## Introduction aux Git Hooks

Les Git hooks sont des scripts qui sont déclenchés par des actions spécifiques de Git. Ces scripts peuvent automatiser des tâches, imposer des normes de codage ou effectuer des vérifications avant de permettre certaines actions comme les commits ou les pushes.

### Hooks côté client courants

- **pre-commit** : S'exécute avant qu'un commit ne soit créé. Il est souvent utilisé pour vérifier la qualité du code, exécuter des tests ou s'assurer que le formatage est correct.
- **commit-msg** : S'exécute après la création du message de commit, mais avant que le commit ne soit finalisé. Il peut être utilisé pour imposer des normes sur le message de commit.
- **post-commit** : S'exécute après la fin du commit. Il peut être utilisé pour des notifications ou une automatisation supplémentaire.

### Configuration d'un Hook Pre-commit

Le hook pre-commit est couramment utilisé pour empêcher les commits qui ne respectent pas certaines normes de qualité, comme des erreurs de linting.

#### Création du script du hook Pre-commit dans le fichier `hooks/pre-commit`

Ce script :

1. Se déplacer dans les répertoires de projet.
2. Exécute la commande `npm run lint` pour vérifier les problèmes de linting dans le frontend.
3. Si le processus de linting échoue (code de sortie différent de 0), il affiche un message et bloque le commit.

Sauvegardez le script dans le répertoire `hooks` de votre projet, par exemple `hooks/pre-commit`.

Rendez le script exécutable :

```
chmod +x hooks/pre-commit
```

### Configuration d'un Hook Commit-msg

Le hook `commit-msg` est utilisé pour vérifier que les messages de commit suivent un format prédéfini. Cela peut être important pour garantir que les messages sont clairs, cohérents, et qu'ils contiennent toutes les informations nécessaires, comme les références aux tickets, les types de changements et les catégories.

### Création du script du hook Pre-commit dans le fichier `hooks/commit-msg`

Le script suivant est exécuté chaque fois qu'un commit est effectué, avant que le commit ne soit finalisé. Il vérifie si le message de commit correspond à une expression régulière spécifique. Si le message ne correspond pas, le commit est bloqué, et un message d'erreur est affiché. Sinon, le commit est autorisé à se poursuivre.

#### Explication du Script

1. **Lecture du message de commit** : Le script commence par lire le message de commit à partir du fichier passé en argument. Ce fichier contient le message que vous avez saisi lors de l'exécution de `git commit`.

**Expression régulière (REGEX)** : L'expression régulière définie (`REGEX`) permet de valider le format du message de commit. Elle exige que le message commence par un numéro de ticket (`TDEV###-###`), suivi d'un emoji, d'un type de modification (`fix`, `feat`, `clean`, `config`), d'une catégorie entre parenthèses (`frontend`, `backend`, `project`), puis d'une description. Exemple de format valide :

`TDEV700-37 🚀 feat (frontend): ajouter une nouvelle fonctionnalité`

2. **Vérification du message de commit** : Si le message ne correspond pas à l'expression régulière, un message d'erreur est affiché, précisant le format attendu et donnant des exemples de ce qui est autorisé. Le commit est alors annulé avec `exit 1`.
3. **Validation** : Si le message est valide, le script termine avec `exit 0`, ce qui permet au commit de continuer normalement.

#### Exemple de message de commit attendu

Voici un exemple d'un message de commit qui correspond à ce format :

`TDEV700-37 🚀 feat (frontend): ajouter une nouvelle fonctionnalité`

## Cas d'erreur

Si le message de commit ne respecte pas le format, vous verrez un message d'erreur comme celui-ci :

```
ERR!: Commit message does not match the required format!
```

Expected format:

Ticket number + emoji + fix/feat/clean/config +  
(frontend/backend/project): description

Emoji Examples: 🐛 (fix) 🧹 (clean) 🚀 (feat) 🛠️ (config)

Commit message example: TDEV700-37 🚀 feat (frontend): add new feature

## Automatisation de la configuration des hooks avec le Makefile

Vous pouvez utiliser un Makefile pour automatiser la configuration des hooks pre-commit et commit-msg. Ajoutez la cible suivante à votre Makefile :

```
prepare:
```

```
    cp hooks/pre-commit .git/hooks/pre-commit && chmod +x  
    .git/hooks/pre-commit
```

```
    cp hooks/commit-msg .git/hooks/commit-msg && chmod +x  
    .git/hooks/commit-msg
```

## Configuration des hooks après un clonage

Après avoir cloné le projet, les hooks doivent être configurés manuellement, car Git ne clone pas automatiquement les hooks. Pour configurer les hooks pre-commit et commit-msg, exécutez la commande suivante :

```
make prepare
```

Cette commande :

1. Copie le script `hooks/pre-commit` dans `.git/hooks/pre-commit`.
2. Copie le script `hooks/commit-msg` dans `.git/hooks/commit-msg`.
3. Rend les scripts exécutables.


## Utilisation du Hook Pre-commit

Une fois le hook pre-commit configuré :

1. Chaque fois que vous tentez de faire un commit, le hook exécutera la vérification du linting.
2. Si le linting passe, le commit continue.
3. Si le linting échoue, le commit est bloqué et vous recevrez un message d'erreur vous invitant à corriger les problèmes avant de faire le commit.

## Utilisation du Hook Commit-msg

Le hook commit-msg garantit que les messages de commit suivent un format ou un modèle spécifique. Il s'exécute après la création du message de commit :

- Des messages de commit conventionnels.
- Une structure spécifique (par exemple: TDEV700-37  config (frontend): description).