

GitHub Copilot

GitHub Copilot est un assistant de programmation basé sur l'IA (développé par GitHub et OpenAI) qui analyse ton code en temps réel et propose des suggestions pour coder plus vite. Il est intégré directement dans PhpStorm via un plugin officiel.

Prerequis

- Un compte GitHub actif avec un abonnement GitHub Copilot (individuel, Business ou Enterprise)
- PhpStorm version 2023.1 ou supérieure (compatible avec le plugin officiel)

Installation

1. Aller dans **File > Settings > Plugins** (ou `Ctrl+Alt+S` puis « Plugins »)
2. Chercher **GitHub Copilot** dans l'onglet *Marketplace*
3. Cliquer sur **Install** puis redémarrer PhpStorm
4. Se connecter à GitHub via la notification qui apparaît en bas à droite

Complétion de code automatique

Dès que tu commences à taper, Copilot propose des suggestions en gris directement dans l'éditeur.

Raccourci	Action
<code>Tab</code>	Accepter la suggestion
<code>Échap</code>	Rejeter la suggestion
<code>Alt +]</code> / <code>Alt + [</code>	Suggestion suivante / précédente
<code>Ctrl + Entrée</code>	Ouvrir le panneau de toutes les suggestions
<code>Alt + Shift + \</code>	Déclencher une suggestion manuellement

Astuce : écrire un commentaire avant ton code améliore considérablement la pertinence des suggestions. Exemple :

```
// Fonction qui valide un email et retourne true si valide
```

```
function validateEmail(string $email): bool {
```

Copilot Chat

Le panneau Chat permet d'interagir avec son code en langage naturel. Il est accessible via **Tools > GitHub Copilot > Open Chat**, ou en cliquant sur l'icône ci-dessous dédié dans la barre latérale droite de PhpStorm.



Les modes du Chat

Un menu déroulant en bas de la fenêtre permet de choisir entre quatre modes.

Le mode **Ask** est le plus simple : on pose une question, Copilot répond sans modifier le code. Idéal pour comprendre un bout de code, explorer une approche ou débloquer une erreur.

Le mode **Edit** permet de décrire une modification en langage naturel sur un ou plusieurs fichiers. Copilot applique les changements et affiche un diff à valider. Utile quand on sait ce qu'on veut mais qu'on ne veut pas l'écrire soi-même.

Le mode **Plan** génère un plan détaillé des étapes à suivre pour accomplir une tâche, sans toucher au code. Pratique pour cadrer une fonctionnalité complexe avant de se lancer.

Le mode **Agent** est le plus autonome : Copilot analyse le projet, choisit les fichiers à modifier et itère jusqu'à compléter la tâche. À réserver aux tâches complexes bien définies — il consomme plus de requêtes premium que les autres modes.

Choix du modèle

Le sélecteur de modèle est disponible en haut de la fenêtre Chat. Plusieurs modèles sont disponibles selon l'abonnement : **GPT-4o** est le modèle par défaut (bon équilibre qualité/vitesse), **GPT-4.1** est plus performant sur les tâches complexes, **Claude Sonnet / Opus** excellent en compréhension de contexte et refactoring, **Gemini 2.5 Pro** est particulièrement efficace sur les données.

En cas de doute, le mode **Auto** laisse Copilot choisir le modèle le plus adapté à chaque requête.

Commandes slash

Dans le Chat, des commandes slash permettent de déclencher des actions rapidement sur le code sélectionné :

- `/explain` — explique le code sélectionné

- `/fix` — corrige les problèmes détectés
- `/tests` — génère des tests unitaires
- `/doc` — génère la documentation PHPDoc
- `/optimize` — propose des optimisations

Générer un message de commit avec Copilot

Dans la fenêtre de commit de PhpStorm, l'icône Copilot située au-dessus du champ de message permet de générer automatiquement un message en analysant le diff des fichiers stagés. Il ne reste plus qu'à relire, ajuster si besoin, et valider.

Personnaliser le format des messages générés

Par défaut, Copilot génère des messages dans son propre format. Il est possible de lui fournir des instructions globales pour qu'il respecte les conventions du projet.

Pour cela, aller dans **Settings > Tools > GitHub Copilot > Customizations**, puis dans la section **Git Commit Instructions**, cliquer sur **Global**. Cela génère et ouvre le fichier `global-git-commit-instructions.md` directement dans l'éditeur, prêt à être renseigné.

Ci-dessous les instructions configurées pour que Copilot génère des messages de commit conformes aux conventions de l'équipe tech, intégrant l'utilisation de [Gitmoji](#).

```
# GitHub Copilot Instructions
```

```
## Commit Messages
```

```
Always generate commit messages following the format below and using the appropriate emoji for the type of change being made. This helps maintain a clear and consistent commit history, making it easier for everyone to understand the purpose of each commit at a glance.
```

```
### Format
```

```
...
```

```
<emoji> [<scope>] <short description>
```

```
[optional body]
```

```
[optional footer]
```

```
...
```

```
### Types et emojis associés (liste officielle Gitmoji)
```

Emoji	Code	Type à utiliser	Description
----- ----- ----- -----			
🎉	<code>sparkles`</code>	<code>`feat`</code>	Introduce new features
🐛	<code>bug:`</code>	<code>`fix`</code>	Fix a bug
🚑	<code>ambulance:`</code>	<code>`fix`</code>	Critical hotfix
🩹	<code>adhesive_bandage:`</code>	<code>`fix`</code>	Simple fix for a non-critical issue
♻️	<code>recycle:`</code>	<code>`refactor`</code>	Refactor code
🏗️	<code>art:`</code>	<code>`style`</code>	Improve structure / format of the code
💄	<code>lipstick:`</code>	<code>`style`</code>	Add or update the UI and style files
📝	<code>memo:`</code>	<code>`docs`</code>	Add or update documentation
💡	<code>bulb:`</code>	<code>`docs`</code>	Add or update comments in source code
✅	<code>white_check_mark:`</code>	<code>`test`</code>	Add, update, or pass tests
🧪	<code>test_tube:`</code>	<code>`test`</code>	Add a failing test
🔧	<code>wrench:`</code>	<code>`chore`</code>	Add or update configuration files
🔨	<code>hammer:`</code>	<code>`chore`</code>	Add or update development scripts
⬆️	<code>arrow_up:`</code>	<code>`chore`</code>	Upgrade dependencies
⬇️	<code>arrow_down:`</code>	<code>`chore`</code>	Downgrade dependencies
➕	<code>heavy_plus_sign:`</code>	<code>`chore`</code>	Add a dependency
➖	<code>heavy_minus_sign:`</code>	<code>`chore`</code>	Remove a dependency
📌	<code>pushpin:`</code>	<code>`chore`</code>	Pin dependencies to specific versions
📦	<code>package:`</code>	<code>`chore`</code>	Add or update compiled files or packages
🚫	<code>see_no_evil:`</code>	<code>`chore`</code>	Add or update a .gitignore file
⚡	<code>zap:`</code>	<code>`perf`</code>	Improve performance
👷	<code>construction_worker:`</code>	<code>`ci`</code>	Add or update CI build system
🟢	<code>green_heart:`</code>	<code>`ci`</code>	Fix CI Build
⏮️	<code>rewind:`</code>	<code>`revert`</code>	Revert changes
🔥	<code>fire:`</code>	<code>`chore`</code>	Remove code or files
🚀	<code>rocket:`</code>	<code>`chore`</code>	Deploy stuff
🎉	<code>tada:`</code>	<code>`chore`</code>	Begin a project
🔒	<code>lock:`</code>	<code>`fix`</code>	Fix security or privacy issues
🔑	<code>closed_lock_with_key:`</code>	<code>`chore`</code>	Add or update secrets
🔖	<code>bookmark:`</code>	<code>`chore`</code>	Release / Version tags
💡	<code>rotating_light:`</code>	<code>`fix`</code>	Fix compiler / linter warnings
🏗️	<code>construction:`</code>	<code>`chore`</code>	Work in progress
📈	<code>chart_with_upwards_trend:`</code>	<code>`feat`</code>	Add or update analytics or track code
🌐	<code>globe_with_meridians:`</code>	<code>`feat`</code>	Internationalization and localization
✎️	<code>pencil2:`</code>	<code>`fix`</code>	Fix typos
⏮️	<code>rewind:`</code>	<code>`revert`</code>	Revert changes

| 🗄 | `:twisted_rightwards_arrows:` | `chore` | Merge branches |

| 🗄 | `:alien:` | `fix` | Update code due to external API changes |

| 🗄 | `:truck:` | `refactor` | Move or rename resources |

| 🗄 | `:page_facing_up:` | `chore` | Add or update license |

| 🗄 | `:boom:` | `feat` | Introduce breaking changes |

| 🗄 | `:bento:` | `chore` | Add or update assets |

| & | `:wheelchair:` | `feat` | Improve accessibility |

| 🗄 | `:speech_balloon:` | `feat` | Add or update text and literals |

| 🗄 | `:card_file_box:` | `feat` | Perform database related changes |

| 🗄 | `:loud_sound:` | `chore` | Add or update logs |

| 🗄 | `:mute:` | `chore` | Remove logs |

| 🗄 | `:children_crossing:` | `feat` | Improve user experience / usability |

| 🗄 | `:building_construction:` | `refactor` | Make architectural changes |

| 🗄 | `:iphone:` | `feat` | Work on responsive design |

| 🗄 | `:goal_net:` | `fix` | Catch errors |

| 🗄 | `:dizzy:` | `feat` | Add or update animations and transitions |

| 🗄 | `:wastebasket:` | `refactor` | Deprecate code that needs to be cleaned up |

| 🗄 | `:passport_control:` | `feat` | Work on authorization, roles and permissions |

| 🗄 | `:monocle_face:` | `chore` | Data exploration/inspection |

| ⚖ | `:coffin:` | `refactor` | Remove dead code |

| 🗄 | `:necktie:` | `feat` | Add or update business logic |

| 🗄 | `:stethoscope:` | `chore` | Add or update healthcheck |

| 🗄 | `:bricks:` | `chore` | Infrastructure related changes |

| 🗄🗄🗄 | `:technologist:` | `chore` | Improve developer experience |

| 🗄 | `:safety_vest:` | `feat` | Add or update code related to validation |

| 🗄 | `:label:` | `chore` | Add or update types |

| 🗄 | `:seedling:` | `chore` | Add or update seed files |

| 🗄 | `:triangular_flag_on_post:` | `feat` | Add, update, or remove feature flags |

| 🗄 | `:mag:` | `chore` | Improve SEO |

| 🧪 | `:alembic:` | `chore` | Perform experiments |

| 🗄 | `:camera_flash:` | `test` | Add or update snapshots |

| 🗄 | `:thread:` | `feat` | Add or update code related to multithreading |

| 🗄 | `:money_with_wings:` | `feat` | Add sponsorships or money related infrastructure |

| 🗄 | `:t-rex:` | `refactor` | Code that adds backwards compatibility |

| ✈ | `:airplane:` | `feat` | Improve offline support |

| 🗄 | `:clown_face:` | `test` | Mock things |

| 🗄 | `:egg:` | `chore` | Add or update an easter egg |

| 🗄 | `:busts_in_silhouette:` | `chore` | Add or update contributor(s) |

Règles

- L'**emoji** est obligatoire et doit correspondre au type de modification
- Le **scope** est obligatoire et doit être entre crochets, indiquant la partie du code affectée (ex: `[API]`, `[UI]`, `[Auth]`)
- La **description** doit être courte (50 caractères max), en minuscules, sans point final
- Le **body** est optionnel, séparé par une ligne vide, pour expliquer le "pourquoi"
- Utiliser `BREAKING CHANGE:` dans le footer si la modification est non rétrocompatible

Exemples

...

- ☐ [Home] add new account number display
- ☐ [API] handle null response on user fetch
- ♻️ [Service] extract payment logic into PaymentService
- ☐ [Deps] update symfony to 6.4.1
- ☐ [Auth] add unit tests for JWT expiration
- ☐ [API] remove deprecated v1 endpoints

BREAKING CHANGE: v1 endpoints have been removed

...

<emoji> [<scope>] <short description>

Bonnes pratiques

- **Nommer ses variables clairement** : Copilot s'appuie sur le contexte sémantique
- **Toujours relire le code généré** : les suggestions sont un point de départ, pas une vérité
- **Ne jamais écrire de secrets dans le code** lors de l'utilisation de Copilot (clés API, mots de passe...)

Liens utiles : [Documentation officielle](#) | [Plugin JetBrains](#)

Révision #8

Créé 4 mars 2026 09:25:43 par Romain Lacits

Mis à jour 10 mars 2026 17:14:19 par Romain Lacits