

Formation GitHub Copilot avancé : personnaliser, gouverner et industrialiser pour l'équipe

Réf. SF34749

Faites de GitHub Copilot un outil d'équipe standardisé, sécurisé et mesurable, de l'IDE jusqu'à la pull request et à la CI/CD.

1 j Durée	800 € HT Inter / participant	0 Sessions à venir
--------------	---------------------------------	-----------------------

Objectifs pédagogiques

- ✓ Concevoir une configuration Copilot partagée par l'équipe (instructions ciblées, prompt files, custom agents, skills, serveurs MCP)
- ✓ Appliquer des techniques de context engineering pour fiabiliser les résultats de Copilot sur des tâches complexes
- ✓ Automatiser des tâches de développement avec l'agent cloud et GitHub Actions, de l'issue à la pull request revue
- ✓ Configurer les garde-fous de sécurité et de gouvernance de Copilot à l'échelle d'une organisation
- ✓ Évaluer l'impact de Copilot à partir des métriques d'usage et d'indicateurs de qualité

Public visé & prérequis

Public visé Architectes logiciels, Développeurs, Ingénieurs DevOps

Prérequis- Avoir suivi la formation « GitHub Copilot : accélérer et fiabiliser le développement avec l'IA » ou avoir une pratique régulière équivalente (Copilot Chat, agent mode, instructions personnalisées) - Pratique courante de Git et de GitHub (branches, pull requests, revues) - Notions de CI/CD, idéalement avec GitHub Actions

Programme détaillé

Context engineering : fiabiliser Copilot sur les tâches complexes

- Panorama des agents Copilot : IDE, CLI, agent cloud et code review, quel agent pour quelle tâche
- Choix du modèle : adapter le modèle à la tâche (raisonnement, rapidité) et maîtriser la consommation de requêtes premium
- Budget de contexte : ce que Copilot reçoit réellement, références ciblées plutôt que contexte surchargé
- Décomposition des tâches complexes : plan préalable, étapes vérifiables et critères d'acceptation fournis à l'agent
- Prompting expert : exemples (few-shot), raisonnement pas à pas et contraintes explicites pour les algorithmes et le débogage en profondeur
- Hallucinations et dérives : les repérer, recadrer la session ou repartir d'un contexte propre

Activités pratiques :

- Résolution d'un bug complexe en pilotant le contexte, avec comparaison de deux modèles
- Découpage d'une évolution en plan, puis exécution étape par étape avec vérification par les tests

Personnaliser Copilot pour l'équipe

- Instructions du dépôt : encoder les conventions communes (style, nommage, architecture, principes Clean Code)
- Instructions ciblées par chemin : règles propres à une couche, un langage ou un dossier ('applyTo')
- AGENTS.md : des instructions lisibles par Copilot comme par les autres agents de code
- Prompt files : transformer les tâches récurrentes (revue, génération de tests, migration) en modèles réutilisables
- Custom agents : profils spécialisés (revue sécurité, documentation, migration) dotés de leurs instructions et de leurs outils
- Agent skills : capitaliser un savoir-faire (instructions, scripts, ressources) que Copilot mobilise quand la tâche s'y prête
- Partage et versionnement : configuration dans le dépôt, agents d'organisation, revue des changements de configuration comme du code

Activités pratiques :

- Équipement du dépôt fil rouge en instructions communes et ciblées par dossier
- Création d'un custom agent de revue et d'un skill d'équipe

Étendre Copilot avec MCP

- MCP (Model Context Protocol) : le standard qui ouvre Copilot aux outils et données externes, en remplacement des anciennes extensions
- Choix des serveurs : GitHub, outils de qualité et de sécurité (analyse statique, scanners de dépendances), documentation, outils internes
- Configuration : au niveau du poste ou du dépôt, authentification et gestion des secrets
- Maîtrise des risques : limiter les outils exposés, injection de prompt via les sources externes, liste d'autorisation de l'organisation

Activités pratiques :

- Connexion d'un serveur MCP d'analyse de code et exploitation de ses résultats dans une session agent

Déléguer à l'agent cloud et automatiser dans GitHub Actions

- Agent cloud : confier une issue, suivre la session, faire itérer l'agent par commentaires sur la pull request
- Environnement de l'agent : préparer dépendances, outils et services (`copilot-setup-steps`), accès réseau et configuration MCP
- Issues déléguables : périmètre, critères d'acceptation et fichiers concernés pour un résultat exploitable
- Review Copilot automatique : activation par les règles du dépôt et instructions de review propres à l'équipe
- CLI Copilot en mode programmatique dans GitHub Actions : tâches non interactives (triage, documentation, rapports, scripts DevOps), outils autorisés au plus juste
- Refactoring de code legacy à l'échelle : découper en tâches déléguées, relire et fusionner le code généré, résoudre les conflits

Activités pratiques :

- Délégation d'un refactoring legacy à l'agent cloud, de l'issue à la pull request revue
- Workflow GitHub Actions qui lance le CLI Copilot pour produire un rapport sur chaque pull request

Sécurité et gouvernance de Copilot

- Chaîne de contrôle du code généré : code scanning, Autofix, analyse des dépendances et scanners tiers en CI, indépendants du prompt
- Garde-fous des agents : permissions d'outils et hooks qui contrôlent les actions avant et après exécution
- Protection des données : exclusion de contenu, filtre de code public, vigilance RGPD dans les environnements sensibles
- Politiques d'organisation : modèles autorisés, fonctionnalités activées (agent cloud, MCP), licences et requêtes premium
- Traçabilité et responsabilité : journaux d'audit, identification des contributions de l'agent, validation humaine avant fusion

Activités pratiques :

- Audit d'un code généré par Copilot : détection et correction de vulnérabilités (OWASP) avec la chaîne de contrôle
- Démonstration des politiques d'organisation et de l'exclusion de contenu (console d'administration)

Mesurer l'impact et améliorer en continu

- Métriques d'usage Copilot : tableau de bord et API (adoption, acceptation des suggestions, usage des agents)
- Indicateurs d'impact : débit et délais de pull requests, croisés avec la qualité (couverture de tests, défauts relevés en revue)
- Limites de la mesure : ce que les métriques ne disent pas, pièges des indicateurs de productivité
- Boucle d'amélioration continue : ajuster instructions, agents et skills à partir des constats et des retours de review

Activités pratiques :

- Lecture d'un tableau de métriques et élaboration d'un plan d'amélioration de la configuration d'équipe

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

4,9/5 — 165 avis

Selim B. — Très bonne formation, très instructif même après quelques années de Python dans les bras. Une très petite équipe de stagiaires qui facilite les échanges. Je redoutais le mode...

Alexis C. — Anthony explique bien les concepts attendus et prend le temps de répondre aux questions et aux différents problèmes rencontrés. Le support présenté créé par lui-même est différent des...

Ils nous font confiance

Cette année notre filière Développement et Ingénierie de Logiciels a accompagné plus de 459 entreprises dont :

SOPRA STERIA RENAULT SAS ORANGE CAMPUS TECH ORANGE INNOVATION SCHOOL STMICROELECTRONICS ARPEGE GRANT THORNTON FRANCE ORANGE BUSINESS RTE SII ILE DE FRANCE +449 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com