

Sessions garanties disponibles

Formation Git : maîtriser le versioning et le travail collaboratif

Réf. SF33282

Versionner, brancher, fusionner, collaborer : prenez en main Git pour gérer vos projets, seul ou en équipe, du dépôt local à la merge request !

2 j Durée	1 600 € HT Inter / participant	3 300 € HT Intra (≤12 pers.)	44 Sessions à venir
--------------	-----------------------------------	---------------------------------	------------------------

Objectifs pédagogiques

- ✓ Décrire le fonctionnement d'un système de contrôle de versions distribué et le modèle objet de Git
- ✓ Installer et configurer Git sur son poste de travail
- ✓ Utiliser les commandes Git du quotidien pour gérer ses fichiers en local (add, commit, status, log, stash, ignore)
- ✓ Manipuler les branches et résoudre les conflits de fusion (merge, rebase)
- ✓ Synchroniser un dépôt local avec un dépôt distant (push, pull, fetch, branches de suivi)
- ✓ Collaborer en équipe via GitHub ou GitLab (pull/merge request, revue de code)
- ✓ Choisir un workflow d'équipe adapté à son projet (GitHub flow, Gitflow, trunk-based)
- ✓ Appliquer les bonnes pratiques de qualité (conventions de commit, hooks pré-commit)

Public visé & prérequis

Public visé Administrateurs systèmes et réseaux, Architectes logiciels, Architectes techniques, Chefs de projet, Développeurs

Prérequis Notions de base du développement logiciel (fichiers source, projet, éditeur de code).

Programme détaillé

Fondamentaux du contrôle de versions distribué

- Contrôle de versions : rôle, enjeux et différence entre approche centralisée et distribuée
- Histoire et positionnement de Git : pourquoi Git s'est imposé face à SVN, CVS, Mercurial
- Écosystème Git : panorama des plateformes (GitHub, GitLab, BitBucket) et place dans la chaîne CI/CD
- Modèle objet de Git : blob, tree, commit, tag — comprendre comment Git stocke réellement l'historique
- Intégrité par hash SHA : pourquoi un commit est immuable et comment Git détecte la corruption
- Les trois zones de travail : working directory, index (staging area), dépôt local

Activités pratiques :

- Démonstration commentée du contenu interne d'un dépôt `.git``
- Lecture d'un graphe de commits pour identifier blobs, trees et références

Installation, configuration et prise en main

- Installation multi-OS : Git for Windows, macOS, Linux — choix du client et options par défaut
- Configuration utilisateur : ``git config`` global vs local, identité, éditeur, fin de ligne
- Personnalisation de l'environnement : alias de commande, outil de diff/merge, coloration
- Intégration éditeur : VSCode + GitLens, indicateurs visuels et commandes embarquées
- Aide intégrée : ``git help``, manpages, ressources de référence
- Création et initialisation d'un dépôt local (``git init``) — premier commit

Activités pratiques :

- Installation et configuration complète d'un poste Git
- Mise en place d'alias utiles et d'un outil de diff/merge visuel

- Initialisation d'un dépôt et premier commit

Workflow local au quotidien

- Cycle de vie d'un fichier : non suivi, suivi, modifié, indexé, validé
- Commandes essentielles : ``status``, ``add``, ``commit``, ``diff``, ``log``
- Sélection fine : ``git add -p`` pour découper ses changements en commits cohérents
- Fichiers ignorés : ``.gitignore``, patterns courants, ``.gitignore`` global
- Mise de côté du travail en cours : ``git stash`` (save, list, pop, drop)
- Annulations : ``git restore``, ``git reset`` (soft/mixed/hard), ``git revert``, ``git reflog``
- Conventions de commit : structure d'un bon message, Conventional Commits en mention
- Exploration de l'historique : filtres, recherche, formatage des logs

Activités pratiques :

- Production d'une série de commits propres à partir de modifications désordonnées
- Récupération d'un fichier supprimé et retour à un état antérieur
- Utilisation de ``stash`` pour basculer rapidement de contexte
- Rédaction de messages de commit alignés sur les conventions

Branches, fusion et résolution de conflits

- Branches comme pointeurs : modèle interne et coût quasi nul de création
- Manipulation : ``git branch``, ``git switch``, ``git checkout``, suppression
- Fusion : fast-forward vs three-way merge, différences et cas d'usage
- Conflits de fusion : lecture des marqueurs, stratégies de résolution, outils visuels
- Rebase : réécrire l'historique pour le rendre linéaire — différence avec merge
- Rebase interactif : ``squash``, ``fixup``, ``reword``, ``drop`` pour nettoyer ses commits
- Bonnes pratiques : quand rebase, quand merger, règle d'or de l'historique partagé

Activités pratiques :

- Création de branches, navigation et fusion sur un dépôt préparé
- Résolution guidée de conflits de fusion réalistes
- Rebase interactif pour réorganiser une série de commits avant intégration

Synchronisation avec un dépôt GitHub distant

- Notion de remote : ``origin``, ajout/suppression de remotes
- Clonage : ``git clone``, configuration automatique du remote
- Synchronisation : ``fetch``, ``pull`` (et son option `–rebase``), ``push``
- Branches de suivi (tracking branches) : lien local / distant, mise en place
- Gestion des échecs de push : non-fast-forward, conflits distants, stratégies de récupération
- Tags : tags légers vs annotés, publication des tags sur le distant
- Authentification : HTTPS avec PAT (Personal Access Token) ou SSH avec clés

Activités pratiques :

- Mise en place d'un dépôt distant et configuration des clés SSH
- Simulation d'un travail à plusieurs : push concurrent, gestion du rejet
- Récupération d'une branche distante et synchronisation continue

Collaboration en équipe sur GitHub et GitLab

- Plateformes : GitHub, GitLab, BitBucket — points communs et différences
- Pull Request / Merge Request : ouverture, description, conventions de revue
- Revue de code : commentaires, suggestions, approbations, intégration finale
- Stratégies d'intégration : merge commit, squash and merge, rebase and merge
- Workflows d'équipe : Gitflow (releases longues), GitHub flow (continu), trunk-based development — critères de choix selon le rythme de livraison
- Gestion des droits : rôles, protections de branches, règles d'approbation
- Hooks Git côté local : ``pre-commit``, ``commit-msg`` — automatiser les vérifications avant push
- Panorama des intégrations en aval : CI/CD déclenchée par push ou pull request

Activités pratiques :

- Création d'un projet d'équipe sur GitHub ou GitLab avec protection de branche

- Cycle complet : branche de feature, commits, ouverture d'une pull/merge request, revue, fusion
- Choix raisonné d'un workflow pour un cas projet donné
- Mise en place d'un hook `pre-commit` pour bloquer un commit en cas d'erreur de lint

Prochaines sessions

DATE	LIEU	STATUT
28/10/2026	à distance	Sur confirmation
28/10/2026	Lyon	Sur confirmation
28/10/2026	Aix-en-Provence	Sur confirmation
28/10/2026	Grenoble	Sur confirmation
28/10/2026	Toulouse	Sur confirmation
02/11/2026	Paris	Sur confirmation

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 — 21 avis

Dorian B. — Formateur à l'écoute, curieux et qui livre des retours d'expériences intéressants.

Patrice B. — Formation très intéressante et intense. Formateur très compétent et à l'écoute.

Votre formateur

Eric D. — Formateur

Eric intervient notamment sur les langages et écosystèmes backend, ainsi que sur le développement web et fullstack. Son activité de formation remonte à 2009, soit plus de 15 ans d'expérience.

Ils nous font confiance

Cette année notre filière CI/CD & Usine Logicielle a accompagné plus de 96 entreprises dont :

INETUM SII ILE DE FRANCE LA FRANÇAISE DES JEUX AMADEUS SAS HARDIS GROUP SILAE AMADEUS DATA PROCESSING GMBH EDF EPSILON FRANCE FDJ-UNITED +86 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