

## Formation Usine Logicielle pour développeurs PHP

<b>Durée :</b>	5 jours
<b>Public :</b>	Administrateurs systèmes - DevOps - Développeurs confirmés
<b>Pré-requis :</b>	Notions d'administration systèmes
<b>Objectifs :</b>	Comprendre les principes DevOps - Mettre en place une solution de configuration logicielle basée sur Git - Gérer les versions des projets du dépôt de données - Mettre en œuvre et exploiter un serveur d'intégration continue - Gérer les interconnexions avec un système de build et de tests
<b>Sanction :</b>	Attestation de fin de stage mentionnant le résultat des acquis
<b>Taux de retour à l'emploi:</b>	Aucune donnée disponible
<b>Référence:</b>	PHP101039-F
<b>Note de satisfaction des participants:</b>	Pas de données disponibles

### Appréhender la culture agile

Le manifeste Agile  
De la vision à la livraison, la chronologie du projet  
Le système sensoriel pour suivre l'évolution du projet  
La planification et la gestion de l'inconnu  
Le rythme : travailler de façon itérative  
La mutation : amélioration progressive et continue du projet et des process  
Quelques principes : KISS, YAGNI, empirisme, transparence, ...

### Apprendre les pratiques agiles

Le Lean Management : objectif, principes  
Kanban : principe, avantage, cycle de vie d'une étiquette  
Pratiques XP (eXtreme Programming)  
Le cadre Scrum, distribution des rôles  
Time boxes : Sprint planning, Sprint Review, Sprint Retrospective, Daily Scrum  
Artéfacts : Product Backlog, Sprint Backlog, Burndown chart

### Mettre en place une solution de gestion de version Git

Principes de gestion de contrôle de source (SCM)  
Historique, contrôle local, centralisé et distribué  
Fonctionnement des instantanées, comparaison avec les différences  
Installation (Linux, MacOS, Windows)  
Accès au manuel : man / help  
Configuration initiale de Git : préférences, profil utilisateur

Initialisation d'un dépôt local

## **Atelier : Installation de Git - Création d'un projet**

### **Exploiter le dépôt local et son cycle de vie**

Concepts, de répertoire de travail, index et dépôt

Vérifier l'état de la copie de travail : status

Indexer ses modifications : add

Ignorer des fichiers : .gitignore

Valider ses modifications : commit

Supprimer et déplacer des fichiers

### **Atelier : contributions et validations**

### **Visualiser l'historique**

Visualiser les modifications : log

Personnaliser le format : stat, pretty, ...

Filtrer par date, auteur, message de commit, contenu modifié, ...

Visualiser et exporter une différence (format natif, outil externe)

Étiqueter ses validations : étiquettes légères et annotées

Rechercher avec git-grep

### **Annuler des actions**

Réécrire la dernière validation

Désindexer un fichier

Réinitialiser un fichier

### **Travailler avec les branches**

Principe de branche, le pointeur HEAD

Créer une branche

Basculer entre les branches, le mode détaché

Fusionner les branches : avance-rapide, trois sources

Gérer les conflits de fusion

Outil de fusion externe : mergetool (emerge, vimdiff, meld, ...)

Visualiser les branches existantes, celles qui ont été fusionnées

Supprimer une branche

Stratégies de gestion de branches : branche longue, thématique, ...

### **Travailler avec un dépôt distant**

Dépôt distant, branches distantes, suivi de branche

Afficher et inspecter les dépôts distants

Ajouter, renommer, retirer ses dépôts distants

Tirer, pousser et supprimer une branche distante

### **Réécrire l'histoire, rebaser**

Mise en garde : les dangers de la réécriture

Rebaser une portion de branche

Quand rebaser et quand fusionner

## **Remiser et nettoyer**

Remiser son travail en cours  
Créer une branche depuis une remise  
Nettoyer son répertoire de travail

## **Personnaliser Git**

Configurer éditeur par défaut, exclusions automatiques, ...  
Création et utilisation d'alias  
Outils graphiques : Git-Gui, GitKraken, SmartGit, ...  
Créer des filtres : smudge et clean  
Crochets côté client : pre-commit, pre-rebase, post-rewrite...  
Crochets côté serveur : pre-receive, update, post-receive

## **Faire référence à un projet externe**

Principe des sous-modules  
Déclarer, tirer et mettre à jour un sous-module  
Modifier et gérer les conflits sur une bibliothèque externe  
Problèmes des sous-modules

## **Publier un dépôt Git sur un serveur**

Les protocoles : local, HTTP, SSH, Git  
Création d'un dépôt nu, comptes utilisateurs  
Utilisateur git unique, clés SSH et git-shell  
Démon Git

### **Atelier : Mise en place d'un serveur Git**

## **Appréhender Docker**

Les différentes formes de virtualisation et leur concept  
Présentation des avantages et des cas d'utilisation des conteneurs  
Présentation de Docker et de son architecture

## **Comprendre l'intégration continue**

Processus de développement, d'intégration et de déploiement  
Intégration continue : présentation, positionnement dans une démarche agile  
Gestion des environnements : développement, recette, production  
Panorama outils de gestion : versionning, build, tests, qualité  
Présentation d'outils d'intégration continue : Jenkins, GitLab-CI, Bamboo, ...

### **Atelier : Publier un projet PHP sur une plateforme d'intégration continue**

## **Mettre en place un pipeline d'intégration PHP avec Gitlab-CI**

Chargement d'une image Docker PHP CLI  
Mise en place du pipeline : les stages et les jobs

Exécution du pipeline et visualisation de la sortie  
Configurer les dépendances entre jobs  
Gérer le déclenchement de l'intégration suivant les branches ou tags

### **Atelier : Configurer et lancer l'intégration**

### **S'équiper pour l'assurance qualité**

Outils de qualité, types et intérêts  
Analyse à froid : php\_cs\_fixer, PHP\_CodeSniffer  
Panorama des types tests : unitaires, fonctionnels, e2e  
Stratégies des tests, TDD, BDD, StoryBDD, non-régression  
Ce qu'il faut et ne faut pas tester

### **Atelier : Mettre en place un contrôle de convention d'écriture**

### **Tester son application**

Présentation de PHPUnit  
Classes et méthodes de tests, assertion  
Provisionner en données : fixtures  
Les doublures : bouchons, mock  
Les résultats : succès, échec, erreur, risqué, incomplet

### **Atelier : Mettre en place des tests automatisés, contrôler les scénarios**

### **Passer de l'intégration au déploiement : le mouvement DevOps**

Présentation du mouvement DevOps  
Valeurs DevOps: Culture, Automatisation, Lean, Mesure, Partage  
Culture DevOps : présentation, caractéristiques, mise en place  
Principe d'infrastructure as code  
Présentation d'Ansible