Industrialisation du logiciel / Git – Gérer le versioning

2 jours (14h00) | 1250 € HT | Évaluation qualitative de fin de stage | Formation délivrée en présentiel ou distanciel

Document mis à jour le 15/05/2023

Objectifs pédagogiques :

- Comprendre les principes de l'industrialisation du logiciel
- Maîtriser les concepts de Git et du versioning
- Savoir utiliser Git pour gérer le versioning de manière efficace
- Comprendre les workflows Git et savoir comment les appliquer à différents projets
- Savoir résoudre les conflits Git
- Mettre en place une stratégie de gestion de versioning adaptée à l'entreprise

Modalités et moyens pédagogiques

Que ce soit en présentiel ou distanciel

Le formateur alterne entre méthode démonstrative, interrogative et active (via des travaux pratiques et/ou des mises en situation).
– Ordinateurs Mac/PC, connexion internet fibre, tableau ou paperboard, vidéoprojecteur – Environnements de formation installés sur les postes de travail ou en ligne
– Supports de cours et exercices

Niveau requis

Pratique et/ou connaissance du développement informatique

Public concerné

Architectes, chefs de projet, consultants, développeurs et ingénieurs.

Programme

1. Introduction à l'industrialisation du logiciel

Présentation des principes de l'industrialisation du logiciel
Compréhension des avantages de l'industrialisation du logiciel pour les entreprises
Définition des termes clés liés à l'industrialisation du logiciel : versioning, dépôt, commit, branche, merge, pull request, etc.

2. Comprendre Git et le versioning

Présentation des concepts de Git et du versioning
Compréhension de la structure des dépôts Git
Utilisation des commandes Git de base pour le versioning
Utilisation des outils graphiques pour Git : GitKraken, Sourcetree, etc.

3. Utilisation de Git pour gérer le versioning de manière efficace

Compréhension des fonctionnalités avancées de Git pour le versioning
Utilisation des branches pour travailler sur des fonctionnalités différentes en parallèle
Utilisation des tags pour marquer des versions importantes
Gestion des conflits Git

4. Comprendre les workflows Git

Présentation des différents workflows Git tels que le modèle de flux de travail centralisé, le flux de travail de fonctionnalité et le flux de travail Gitflow
Compréhension des avantages et des inconvénients de chaque workflow
Savoir comment appliquer chaque workflow à différents projets
Gestion des branches à travers les différents workflows

5. Stratégies de gestion de versioning

Compréhension des différents types de stratégies de versioning : Continuous Integration, Continuous Delivery, Continuous Deployment, Release Management, etc.
Compréhension des avantages et des inconvénients de chaque stratégie
Mise en place d'une stratégie de gestion de versioning adaptée à l'entreprise

6. Gestion des versions dans un environnement multi-projets

Compréhension des problématiques liées à la gestion de versions dans un environnement multi-projets
Utilisation de Git submodules et Git subtree pour gérer les dépendances entre les projets

7. Résoudre les conflits Git

Compréhension des différents types de conflits Git
Utilisation des outils Git pour résoudre les conflits : Git mergetool, Git difftool, etc.
Compréhension des bonnes pratiques pour éviter les conflits Git

8. Sécuriser les dépôts Git

Comprendre les risques de sécurité liés à l'utilisation de Git et des dépôts Git
Mettre en place des politiques de sécurité pour les dépôts Git : restrictions d'accès, authentification, chiffrement, etc.
Utiliser des outils pour surveiller et détecter les activités malveillantes dans les dépôts Git

9. Automatisation de la gestion de versioning

Comprendre les avantages de l'automatisation de la gestion de versioning
Utilisation de Git hooks pour automatiser certaines tâches
Comprendre les outils d'intégration continue tels que Jenkins, GitLab CI/CD, etc.

10. Bonnes pratiques pour une gestion efficace de versioning

Comprendre les bonnes pratiques pour une gestion efficace de versioning
Savoir comment documenter les commits et les changements apportés au code
Savoir comment organiser les branches Git
Comprendre l'importance de l'interaction entre les développeurs pour éviter les conflits Git

11. Études de cas pratiques

Réalisation d'études de cas pratiques pour appliquer les concepts et les outils appris dans la formation
Discussion de cas pratiques liés à l'entreprise des participants pour appliquer les concepts appris à des situations réelles

12. Conclusion et évaluation

Récapitulation des points clés de la formation
Évaluation de la satisfaction des participants

Modalités d'évaluation des acquis

– Exercices + qcm entre chaque chapitre