DEVOPS & DÉPLOIEMENT

Git, GitHub Actions et Cyber Resilience Act : pipeline sécurisé pour développement .NET

Apprenez à travailler en équipe avec Git et GitHub, à automatiser le build, les tests et les contrôles de sécurité d'une solution .NET avec GitHub Actions, et à préparer le Cyber Resilience Act.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
7j (49h)	**Au plus 2 635 € / participant** Intra-entreprise, sur devis, dégressif selon la taille du groupe	**Intermédiaire**	**Français**	**4-6**	**Formation** Présentiel

1 PUBLIC VISÉ

- Développeurs et automaticiens qui travaillent sur des applications desktop .NET (C#, VB.NET, WPF) et veulent structurer leur travail en équipe avec Git et GitHub
- Responsables techniques qui préparent la conformité de leurs logiciels au Cyber Resilience Act

2 PRÉREQUIS

- Développer en C# ou en VB.NET sous Visual Studio 2022 : ouvrir une solution, la compiler et la déboguer
- Aucune expérience de Git, de GitHub Actions ni du Cyber Resilience Act n'est requise

3 OBJECTIFS PÉDAGOGIQUES

- Enregistrer et consulter son travail avec les commandes Git essentielles, en passant par les trois zones, et rattraper une erreur avec restore, amend, stash ou reflog
- Créer, naviguer et fusionner des branches selon une stratégie adaptée au contexte de l'équipe
- Identifier, lire et résoudre un conflit de fusion sans perte de données
- Ouvrir une pull request et conduire une revue de code structurée sur GitHub
- Écrire et exécuter des tests automatisés xUnit sur une solution .NET
- Écrire un workflow GitHub Actions qui compile et teste une solution .NET à chaque pull request
- Publier automatiquement les binaires d'une solution .NET comme artefacts en sortie de pipeline, à la place d'une livraison manuelle
- Configurer les outils de sécurité automatisés : Dependabot, CodeQL, secret scanning
- Générer dans le pipeline un SBOM CycloneDX pour une solution .NET et y repérer les dépendances directes et transitives
- Classer un produit logiciel au regard du Cyber Resilience Act : périmètre, catégorie, échéances
- Rédiger une notification de vulnérabilité reprenant les éléments et les délais de l'article 14 du CRA

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Git collaboratif : reconstruire des fondations solides

Module 1 : Enregistrer son travail avec Git et rattraper ses erreurs (7h)

Comprendre Git en profondeur (1h30)

- Pourquoi Git n'est pas un outil de sauvegarde mais un graphe de snapshots
- Les objets fondamentaux : blob, tree, commit et leurs relations
- Le pointeur HEAD et le mécanisme des références
- Les trois zones : répertoire de travail, index, dépôt local
- Exercice d'entraînement : observer les objets d'un dépôt de démonstration et le déplacement de HEAD

Manipulation au quotidien (2h)

- Cycle de vie d'un fichier : untracked, modified, staged, committed
- Commandes essentielles : status, add, restore, commit, log, diff
- Conventions de messages de commit : standard Conventional Commits
- Le fichier .gitignore pour les projets Visual Studio (.vs, bin, obj, packages)
- Exercice d'entraînement : initialiser un dépôt, premiers commits, navigation dans l'historique

Récupération en cas d'erreur (3h30)

- Annuler une modification non commitée
- Modifier le dernier commit avec amend
- Mettre de côté son travail avec stash
- Retrouver des commits perdus avec reflog
- Exercice d'entraînement : retrouver un commit perdu avec reflog
- TP évalué : sur un projet Visual Studio fourni, écrire le .gitignore, enregistrer le travail en commits au format Conventional Commits, puis rattraper trois erreurs (fichier modifié par erreur, dernier commit incomplet, commit perdu) avec restore, amend, stash et reflog

Jour 2 (7h) : Stratégies de branches et résolution de conflits

Module 2 : Branches, fusions et conflits (7h)

Travailler avec les branches (1h30)

- Création, navigation et suppression de branches locales
- Comparaison des stratégies : Git Flow, GitHub Flow, Trunk-Based
- Recommandation pour un contexte logiciel desktop industriel
- Conventions de nommage des branches en équipe
- Exercice d'entraînement : créer, changer et supprimer des branches nommées selon une convention d'équipe

Merge et rebase (2h)

- Mécanisme du merge : commit de fusion et fast-forward
- Option --no-ff et préservation de la traçabilité des fonctionnalités
- Le rebase : réécriture d'historique et linéarisation
- Règle d'or : ne jamais rebaser une branche partagée
- Quand utiliser merge, quand utiliser rebase
- Exercice d'entraînement : fusionner une branche avec et sans --no-ff, puis rebaser une branche locale

Résolution de conflits en pratique (3h30)

- Comprendre pourquoi un conflit apparaît
- Lecture et interprétation des marqueurs de conflit
- Résolution avec l'outil intégré à Visual Studio 2022
- Exercice d'entraînement, en binôme : créer volontairement un conflit puis le résoudre proprement
- Cas concrets de conflits récurrents et comment les éviter
- TP évalué : sur un dépôt fourni, créer une branche par fonctionnalité selon la stratégie retenue pour l'équipe, les fusionner dans la branche principale et résoudre les conflits avec l'outil de Visual Studio 2022, sans perdre de modification

Jour 3 (7h) : GitHub : plateforme et collaboration

Module 3 : GitHub : organisation, pull requests et revue de code (7h)

Organisation GitHub et authentification (2h)

- Comptes individuels GitHub Free : les limites pour le travail professionnel
- Création d'une organisation GitHub Team : structure, membres, équipes
- Risques juridiques du stockage de code professionnel sur comptes personnels
- Limites de l'authentification par mot de passe ; Personal Access Token : génération, portée et durée ; clé SSH ed25519
- Git Credential Manager pour Windows ; configuration de Visual Studio 2022 avec authentification moderne
- Exercice d'entraînement : création d'une organisation et transfert d'un dépôt

Anatomie d'une pull request (1h30)

- Cycle de vie complet d'une pull request : ouverture, revue, itération, fusion
- Modèle de pull request en .github/pull_request_template.md
- Stratégies de fusion : merge commit, squash, rebase ; recommandation pour préserver la traçabilité des fonctionnalités
- Exercice d'entraînement : ouvrir une pull request avec le modèle de l'équipe

Revue de code, protection des branches et issues (3h30)

- Les sept dimensions d'une revue : fonctionnel, architectural, sécurité, performance, lisibilité, tests, style
- Conduite constructive : critiquer le code et non la personne ; niveaux d'importance : blocker, suggestion, nit, question ; suggestions applicables en un clic
- Règles de protection des branches main et develop : approbations, vérifications CI, résolution des conversations ; le fichier CODEOWNERS
- Issues : titre, description, labels, milestones ; liaison automatique avec les pull requests (Closes #123) ; labels pour un projet industriel
- Conventional Commits en équipe : changelog automatique, versionnage sémantique
- TP évalué : revue croisée en binôme d'une pull request contenant des défauts plantés (commentaires classés blocker, suggestion, nit ou question, suggestions applicables), puis ouverture de sa propre pull request liée à une issue, sur une branche protégée

Jour 4 (7h) : Tests automatisés et GitHub Actions

Module 4 : Tests automatisés et anatomie de GitHub Actions (7h)

Tests automatisés avec xUnit (2h30)

- Limites des tests manuels ; ce que les tests automatisés apportent : régression, rapidité, confiance
- Pyramide des tests : unitaires, intégration, end-to-end ; code testable et conception orientée tests

- Création d’un projet de tests dans une solution .NET ; anatomie d’un test xUnit : Fact, Theory, assertions
- Exécution des tests depuis Visual Studio et en ligne de commande ; couverture de code avec coverlet
- Exercice d’entraînement : premiers tests sur une classe métier de démonstration

Concepts et structure d’un workflow YAML (4h30)

- Workflow, event, job, step, action, runner : la hiérarchie
- Choix du runner : windows-latest pour WPF, ubuntu-latest pour les analyses ; coût comparé
- La marketplace des actions et la sécurité des actions tierces
- Déclencheurs : push, pull_request, schedule, workflow_dispatch ; jobs et steps ; variables d’environnement et contextes
- Lecture commentée d’un workflow CI minimal
- Exercice d’entraînement : modifier un workflow minimal pour ajouter un déclencheur et un job
- TP évalué : écrire des tests xUnit (Fact et Theory) pour une classe métier, mesurer leur couverture avec coverlet, puis écrire un workflow qui les lance à chaque pull request

Jour 5 (7h) : Pipeline CI pour solution .NET

Module 5 : Pipeline d’intégration continue pour une solution .NET (7h)

Construction d’un workflow de build (3h30)

- Checkout du code ; SDK .NET avec actions/setup-dotnet
- Cache des packages NuGet pour accélérer les builds
- Restore, build et test d’une solution .NET en ligne de commande
- Publication des résultats de tests dans l’interface GitHub
- Exercice d’entraînement : workflow de build et de tests sur le projet .NET de démonstration

Gestion des artefacts (1h30)

- Production des binaires en sortie de pipeline
- Upload d’artefacts avec actions/upload-artifact
- Durée de rétention et bonnes pratiques
- Exercice d’entraînement : publier les binaires comme artefact et régler leur rétention

Diagnostic et TP (2h)

- Lire les logs et diagnostiquer un échec de build courant
- TP évalué : sur un projet .NET fourni, workflow complet à chaque pull request : restore, build, tests avec publication des résultats, cache NuGet et binaires publiés comme artefacts ; diagnostic et correction d’un échec de build provoqué

Jour 6 (7h) : Sécurité du pipeline et Cyber Resilience Act

Module 6 : Sécurité du pipeline et SBOM (4h30)

Menaces, Dependabot, CodeQL et secret scanning (3h30)

- Périmètre de la chaîne logicielle d’un logiciel desktop .NET ; catégories de menaces : vulnérabilités du code, dépendances vulnérables ou malveillantes, compromission du pipeline, falsification des artefacts, mises à jour piégées
- Cas concrets : Log4Shell, SolarWinds, compromissions npm et PyPI
- Dependabot pour NuGet : alertes, fichier .github/dependabot.yml, lecture et qualification d’une alerte ; audit avec dotnet list package --vulnerable --include-transitive
- CodeQL : fonctionnement, workflow pour csharp, suites de requêtes, exemples de vulnérabilités en C# (injection SQL, path traversal, désérialisation)

- Secret scanning et push protection : démonstration d'un push de secret bloqué, refaite par chaque participant
- Exercice d'entraînement : configurer Dependabot pour NuGet et un workflow CodeQL pour C# sur le dépôt de démonstration

SBOM CycloneDX (1h)

- Qu'est-ce qu'un SBOM et pourquoi le CRA l'exige ; formats CycloneDX et SPDX
- Génération avec dotnet CycloneDX ; lecture : dépendances directes et transitives
- Intégration dans le pipeline et archivage avec chaque version livrée
- TP évalué : dans le pipeline du jour 5, activer secret scanning et push protection, générer le SBOM CycloneDX à chaque build, l'archiver avec les artefacts et y repérer la dépendance transitive signalée par une alerte Dependabot, qualifiée

Module 7 : Le Cyber Resilience Act appliqué (2h30)

Le règlement et les obligations des fabricants (1h30)

- Règlement (UE) 2024/2847 : calendrier ; périmètre : produits comportant des éléments numériques mis sur le marché de l'UE
- Échéances clés : 11 septembre 2026 (signalement) et 11 décembre 2027 (conformité)
- Classification des produits : par défaut, important de classe I et II, critique ; sanctions
- Annexe I : exigences essentielles relatives au produit et à la gestion des vulnérabilités ; documentation technique (Annexe VII) et conservation 10 ans
- Notification 24 h, 72 h, 14 jours via la plateforme de l'ENISA ; divulgation coordonnée et fichier security.txt
- Exercice d'entraînement : classer trois produits logiciels fictifs

Simulation : classement et notification (1h)

- TP évalué : classer le logiciel desktop industriel de l'équipe (périmètre, catégorie, échéances, questions à soumettre à un juriste), puis rédiger la notification initiale à 24 h d'une vulnérabilité fictive sur une dépendance NuGet
- Correction : cartographie des outils vus en formation et des exigences du CRA qu'ils couvrent

Jour 7 (7h) : Application sur projets réels et plan d'action

Module 8 : Projet de synthèse, évalué (7h)

Mise en place du pipeline sur un projet de l'équipe (3h30)

- Sélection d'un projet .NET réel de l'équipe ; chaque participant configure son propre dépôt
- Organisation GitHub Team et migration du dépôt
- Règles de protection des branches et CODEOWNERS adaptés à l'équipe
- Workflow CI avec build, tests et publication d'artefacts
- CodeQL et Dependabot pour les packages NuGet
- Premier SBOM CycloneDX et archivage

Charte Git d'équipe et ébauche de plan d'action CRA (2h)

- Charte : stratégie de branches et conventions de nommage, format des messages de commit, règles de pull request, modalités de revue de code et délais de validation
- Rédaction en groupe, en séance, d'une première version de la charte, à partir du modèle fourni ; l'équipe la reprend ensuite sous sa responsabilité
- Ébauche de plan d'action CRA rédigée en séance à partir du modèle fourni, exercice de formation : cartographie des produits et questions de classification à soumettre à un juriste, écarts entre pratiques actuelles et exigences du CRA, premières étapes jusqu'à décembre 2027, rôles internes (référent sécurité, référent vulnérabilités)
- Travail de groupe, en dehors de l'évaluation des acquis

Revue, retour individuel et bilan (1h30)

- Revue collective des pipelines mis en place
- Retour individuel du formateur
- Bilan : synthèse des acquis et points à approfondir, ressources pour continuer, questionnaire d'appréciation et tour de table

Le contenu peut être ajusté selon le niveau, les prérequis et les besoins des participants.

5 COMPÉTENCES VISÉES

- Enregistrer et organiser son travail dans Git : commits, branches, stratégie d'équipe
- Résoudre les conflits et faire relire le code par pull request sur GitHub
- Tester une solution .NET et automatiser son build avec GitHub Actions
- Livrer automatiquement les binaires d'une solution .NET avec leur SBOM
- Intégrer les contrôles de sécurité automatisés dans un pipeline
- Classer un produit logiciel au regard du Cyber Resilience Act et notifier une vulnérabilité dans les délais

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel intra-entreprise, sur le site du client. Le formateur alterne entre méthode démonstrative, méthode interrogative (questionnement actif des stagiaires) et méthode active (exercices d'entraînement, TP évalués et projet de synthèse sur un projet réel de l'équipe).
La formation présente le règlement et une méthode pour s'y préparer. Elle ne constitue pas un avis juridique : la qualification des produits et les décisions de conformité relèvent de l'entreprise et de son conseil.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique remis aux participants au format PDF
- Fiche de référence des commandes Git essentielles
- Modèles de workflows GitHub Actions prêts à l'emploi pour projets .NET
- Checklist des exigences du Cyber Resilience Act
- Glossaire technique et réglementaire
- Dépôt GitHub de démonstration avec projet .NET et vulnérabilités plantées pour les ateliers
- Templates : pull request, CODEOWNERS, dependabot.yml, security.txt
- Modèles de notification de vulnérabilité suivant l'article 14 du CRA
- Environnement de travail de chaque participant : Visual Studio 2022, accès à Internet et compte GitHub
- Supports remis par e-mail et sur la plateforme pédagogique LaPolaris (ressources, émargement)

La formation se déroulant sur le site du client, celui-ci s'assure et s'engage à disposer de toutes les ressources matérielles nécessaires à son bon déroulement (salle, postes de travail, logiciels, accès à Internet), conformément aux prérequis indiqués dans le programme et aux moyens prévus dans la convention.

8 MODALITÉS D'ÉVALUATION

- Questionnaire de positionnement individuel renseigné par chaque participant avant le démarrage de la formation, analysé par le formateur et donnant lieu, le cas échéant, à un ajustement du déroulé
- Exercices d’entraînement réalisés par chaque participant sur son poste, corrigés en séance
- Évaluation des acquis par les TP évalués, réalisés individuellement sur poste : un par module, puis le projet de synthèse décrit dans le dernier module du programme ; chaque objectif est évalué dans au moins un TP
- Grille de suivi renseignée par le formateur pour chaque participant et pour chacun des objectifs de l’action, au regard des critères d’évaluation ci-dessous
- Questionnaire d’appréciation renseigné par les participants en clôture de formation, et enquête à froid dans un délai de trois à six mois

9 CRITÈRES D'ÉVALUATION

- Commits atomiques au format Conventional Commits, .gitignore adapté à Visual Studio (.vs, bin, obj, packages), erreurs rattrapées sans perte de travail avec restore, amend, stash ou reflog
- Une branche par fonctionnalité, nommée selon la convention de l’équipe, fusionnée selon la stratégie retenue (merge --no-ff, ou rebase d’une branche non partagée)
- Conflit résolu avec l’outil de Visual Studio 2022 : marqueurs retirés, modifications des deux côtés conservées, résolution validée par un commit
- Pull request ouverte avec le modèle de l’équipe et liée à une issue ; revue d’une autre pull request avec des commentaires classés (blocker, suggestion, nit, question) et des suggestions applicables
- Projet de tests xUnit ajouté à la solution, tests Fact et Theory qui passent dans Visual Studio et en ligne de commande, couverture mesurée avec coverlet
- Workflow déclenché à chaque pull request, sur le runner adapté, qui restaure, compile et teste la solution avec cache NuGet ; résultats des tests publiés dans GitHub
- Binaires publiés comme artefacts à chaque exécution du pipeline, avec une durée de rétention définie
- Dependabot configuré pour NuGet, workflow CodeQL pour C# actif, secret scanning et push protection activés ; une alerte lue et qualifiée
- SBOM CycloneDX généré à chaque build et archivé avec les artefacts ; dépendances directes et transitives repérées
- Produit classé : périmètre, catégorie et échéances justifiés, questions à soumettre à un juriste listées
- Notification initiale rédigée avec les éléments et les délais de l’article 14 (24 h, 72 h, 14 jours)

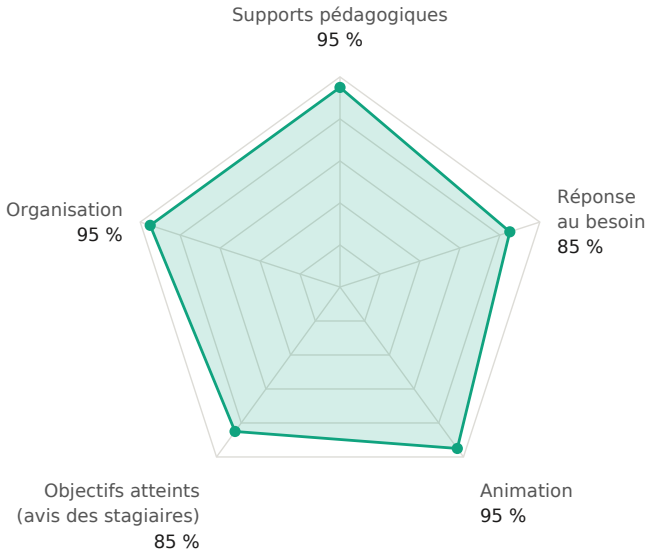
10 INDICATEURS DE RÉSULTATS

Satisfaction globale des stagiaires : 85 %

Méthode : satisfaction globale = moyenne des réponses à la question « Globalement, êtes-vous satisfait de cette formation ? », notée de 1 à 5 et ramenée en pourcentage.

Taux de réussite : 100 %

Méthode : dans la grille d’évaluation de fin de formation, chaque objectif vaut 1 s’il est atteint, 0,5 s’il l’est partiellement et 0 sinon ; un stagiaire réussit si la moyenne de ses objectifs atteint 70 %. Taux de réussite = stagiaires qui réussissent ÷ stagiaires évalués.



Chaque critère du graphique est la moyenne des réponses des stagiaires à sa question, notée de 1 à 5 et ramenée en pourcentage.

11 MODALITÉS DE VALIDATION

Une attestation de fin de formation est remise à chaque participant à l'issue du parcours. Elle mentionne **l'intitulé** et la **nature de l'action**, sa **durée** en heures, sa **période de réalisation**, **les objectifs de l'action** et le **niveau d'atteinte** de chacun d'eux, exprimé en trois valeurs : **atteint**, **partiellement atteint**, **non atteint**. **Une attestation d'assiduité**, établie à partir des émargements, est délivrée dans les mêmes conditions.

Formation non certifiante : elle ne délivre pas de certification enregistrée au RNCP ou au Répertoire spécifique.

12 SUIVI ET ACCOMPAGNEMENT

- Émargement signé par chaque participant et par le formateur à chaque demi-journée
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les QCM, les exercices corrigés et les TP évalués
- Échange possible en fin de journée pour ajuster le contenu du jour suivant

13 CONDITIONS D'ACCÈS

Formation organisée en intra-entreprise, sur devis. Le financement est pris en charge par l'entreprise sur son propre budget. Cette formation n'est pas éligible aux financements publics ou mutualisés, notamment au compte personnel de formation et à une prise en charge par un opérateur de compétences.

14 DÉLAIS D'ACCÈS

Inscription possible jusqu'à **15 jours ouvrés** avant le début de la session pour permettre la préparation logistique et, si besoin, l'adaptation du contenu au contexte client. Pour toute demande spécifique, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations peuvent être accessibles aux personnes en situation de handicap. La question d'un besoin d'aménagement (matériel, temporel ou pédagogique) est posée lors de l'analyse du besoin et du questionnaire de positionnement ; vous pouvez aussi nous le signaler à tout moment, y compris pendant la formation, afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798
EMAIL contact@lapolaris.fr
WEB lapolaris.fr