DEVOPS & DÉPLOIEMENT

CI/CD avec GitHub Actions

Apprenez à automatiser les tests et les déploiements de vos projets avec GitHub Actions : workflows YAML, cache et matrices, secrets et environnements, images Docker, déploiement continu et protection des branches.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
3j (21h)	**1 275 € / participant** Inter-entreprise. Intra : sur devis, au plus 1 275 € / participant	**Intermédiaire**	**Français**	**1-6**	**Formation** Présentiel ou à distance

1 PUBLIC VISÉ

- Développeurs web (Node.js, PHP, Python) qui utilisent Git et GitHub et veulent automatiser leurs tests et leurs déploiements
- Équipes de développement et freelances qui veulent remplacer leurs déploiements manuels par un pipeline

2 PRÉREQUIS

- Créer une branche, ouvrir une pull request et la fusionner sur GitHub, et lancer en ligne de commande les tests d’un projet web (Node.js, PHP ou Python)
- Aucune expérience de GitHub Actions ni de Docker n’est requise

3 OBJECTIFS PÉDAGOGIQUES

- Écrire des workflows GitHub Actions en YAML : déclencheurs, jobs, étapes, runners
- Automatiser le lint et les tests sur chaque push et pull request, pour un projet Node.js, PHP ou Python
- Réduire la durée d’un pipeline avec le cache des dépendances, les jobs parallèles et les matrices
- Stocker secrets et variables par environnement, et soumettre le déploiement en production à une approbation
- Construire une image Docker, la publier sur ghcr.io et la faire analyser par Trivy
- Déployer automatiquement sur un VPS par SSH ou sur Render ou Railway
- Bloquer la fusion d’une pull request si les tests, le lint ou le seuil de couverture de code échouent
- Regrouper les étapes communes dans une action composite ou un workflow réutilisable

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Premiers workflows et tests automatisés

Module 1 : Écrire ses premiers workflows GitHub Actions (3h30)

La CI/CD et l’architecture de GitHub Actions (1h30)

- Intégration continue, livraison continue, déploiement continu : ce que chacune automatise
- L’écosystème : GitHub Actions, GitLab CI, CircleCI, Jenkins
- Les composants : workflow, event, job, step, action, runner ; runners hébergés par GitHub ou auto-hébergés

- Le répertoire .github/workflows/ : un fichier par workflow
- La syntaxe YAML : indentation stricte, types de valeurs, chaînes multilignes
- L'onglet Actions : visualiser le pipeline, lire les logs ; les minutes gratuites et leur consommation
- Exercice d'entraînement : créer un premier workflow hello.yaml déclenché sur push et le suivre dans l'onglet Actions

Déclencheurs, jobs, étapes et contextes (2h)

- Les événements : push (branches, paths-ignore), pull_request, workflow_dispatch avec inputs, schedule (cron), release ; combiner plusieurs événements
- Déclarer un job : runs-on, steps, needs
- Les actions officielles : actions/checkout, setup-node, setup-python, setup-php
- Commandes shell dans les steps : run sur une ou plusieurs lignes
- Contextes github, env, steps, runner ; passer des données entre steps : outputs, $GITHUB_ENV
- Conditions : if, success(), failure(), github.ref
- TP évalué : pipeline à deux jobs, lint puis build, déclenché sur push et sur pull request, le job build ne démarrant que si le lint réussit

Module 2 : Automatiser et accélérer les tests (3h30)

Pipelines de tests Node.js, PHP et Python (1h30)

- Node.js : setup-node, npm ci, tests avec Jest ou Vitest, ESLint, Prettier en mode --check
- PHP : setup-php, Composer, PHPUnit, PHPStan
- Python : setup-python, pip, pytest, ruff
- Dépendances de test : base de données de test, variables d'environnement ; lancer PostgreSQL ou MySQL comme service
- Publier les résultats de tests : rapport JUnit avec actions/upload-artifact
- Exercice d'entraînement : pipeline complet sur une API Express : checkout, Node.js en version LTS, npm ci, ESLint, Jest

Cache, matrices et jobs parallèles (2h)

- Cache des dépendances : actions/cache pour node_modules, pip, Composer
- La stratégie matrix : tester plusieurs versions en parallèle ; fail-fast
- Jobs parallèles : lint, tests unitaires et tests d'intégration en même temps
- Artifacts : rapports de couverture, captures d'écran des tests
- TP évalué : dans son langage (Node.js, PHP ou Python), pipeline qui lance le lint et les tests sur chaque push et pull request, avec cache des dépendances, matrice de deux versions et jobs parallèles, et mesure du temps gagné

Jour 2 (7h) : Secrets, images Docker et déploiement continu

Module 3 : Secrets, environnements et images Docker (3h30)

Secrets et environnements (1h30)

- Pourquoi ne jamais stocker de secrets dans le code ni dans les logs
- Secrets de dépôt, d'organisation et d'environnement ; ${{ secrets.MON_SECRET }} ; masquage dans les logs
- Les variables (vars) pour les valeurs non sensibles
- Environnements staging et production ; règle d'approbation manuelle avant la production ; secrets par environnement ; environment: production dans un job
- L'historique des déploiements
- Exercice d'entraînement : deux environnements, staging automatique et production avec approbation, chacun avec son secret

Construire, publier et analyser une image Docker (2h)

- docker/build-push-action ; s'authentifier sur Docker Hub ou GitHub Container Registry
- Tagger l'image avec le SHA du commit et latest ; cache des couches : cache-from, cache-to
- Pourquoi analyser : failles connues dans les images et les dépendances
- Analyser les dépendances : npm audit, pip-audit, Dependabot
- Analyser l'image avec Trivy, action épinglée à un SHA de commit : seuil de sévérité, rapport SARIF publié dans l'onglet Security
- TP évalué : pipeline qui construit l'image, la publie sur ghcr.io avec les tags SHA et latest, et s'arrête si Trivy trouve une faille critique

Module 4 : Déployer automatiquement (3h30)

Déployer sur un VPS par SSH (1h30)

- Une paire de clés SSH dédiée au déploiement : clé privée dans les secrets, clé publique sur le VPS
- Se connecter au VPS depuis le workflow avec appleboy/ssh-action
- Déployer : docker compose pull puis docker compose up -d
- Vérifier : test de santé HTTP après redémarrage ; revenir à l'image précédente en cas d'échec
- Exercice d'entraînement : se connecter au VPS depuis un workflow et y lancer une commande

Déployer sur Render ou Railway (2h)

- Render : déploiement d'une image Docker par l'API Render
- Railway : déploiement depuis le dépôt GitHub ou depuis une image
- Synchroniser les variables d'environnement de la plateforme avec les secrets GitHub
- TP évalué : pipeline complet : push sur main, tests, image, déploiement automatique en staging (VPS par SSH ou Render), test de santé, puis production après approbation, avec des secrets par environnement

Jour 3 (7h) : Protection des branches, réutilisation et projet de synthèse

Module 5 : Protéger la branche principale et factoriser les workflows (3h)

Protection de branche et couverture de code (1h30)

- Protéger main avec un ruleset (ensemble de règles) : pas de push direct, pull request obligatoire
- Vérifications obligatoires : pas de fusion si l'intégration continue échoue ; relectures obligatoires
- Rapport de couverture : couverture intégrée de Jest ou Vitest pour Node.js, coverage.py pour Python, Xdebug pour PHP ; publication en commentaire de la pull request
- Annotations de lint dans les diffs de pull request
- Exercice d'entraînement : configurer la protection de main et vérifier qu'un push direct est refusé

Actions composites et workflows réutilisables (1h30)

- Créer une action composite dans .github/actions/ : action.yaml, inputs, outputs
- Workflows réutilisables avec workflow_call : inputs et secrets
- Évaluer une action du Marketplace ; l'épingler à un SHA de commit ; limiter les droits du GITHUB_TOKEN avec permissions
- Concurrency groups : annuler une exécution en cours quand une nouvelle démarre
- TP évalué : regrouper setup, cache et installation dans une action composite locale appelée par deux jobs, et bloquer la fusion d'une pull request si la couverture passe sous 70 %

Module 6 : Projet de synthèse, évalué (4h)

Conception et réalisation du pipeline (2h30)

- Étape 1 : choisir le projet (application apportée ou projet fourni), dessiner le pipeline : événements, jobs, environnements, secrets, cible de déploiement
- Étape 2 : workflow d'intégration continue sur pull request : tests, lint, couverture
- Étape 3 : workflow de déploiement continu sur fusion dans main : image Docker, registre, staging automatique, production avec approbation
- Étape 4 : protection de main, cache, jobs parallèles, concurrency group

Présentation et retour (1h30)

- Présentation du pipeline : un push qui déclenche l'intégration puis le déploiement ; revue collective des choix
- Retour individuel du formateur sur la structure des workflows, la gestion des secrets et la stratégie de déploiement

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

5 COMPÉTENCES VISÉES

- Écrire des workflows GitHub Actions qui testent automatiquement le code
- Accélérer et factoriser un pipeline : cache, matrices, jobs parallèles, actions composites
- Construire, publier et déployer une application depuis un pipeline : image Docker, VPS ou plateforme cloud
- Sécuriser la livraison : secrets par environnement, approbation de la production, fusion bloquée si l'intégration continue échoue

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou à distance, en classe virtuelle. Le formateur alterne entre méthode démonstrative, méthode interrogative (analyse des logs d'échec et diagnostic collectif) et méthode active (exercices d'entraînement, TP évalués et projet de synthèse).

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition
- Fichiers de workflows YAML annotés avec exercices et corrections par module
- Environnement : compte GitHub + compte Render gratuit + VS Code + accès à un VPS Ubuntu pour les déploiements
- Pour le distanciel : visioconférence (Microsoft Teams ou équivalent), partage d'écran, chat en direct
- Supports remis par e-mail et sur la plateforme pédagogique LaPolaris (ressources, émargement)

En formation intra-entreprise 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

- Workflow placé dans .github/workflows, déclenché par les bons événements, jobs enchaînés avec needs, exécution réussie visible dans l'onglet Actions
- Lint et tests lancés automatiquement sur chaque push et pull request ; un test en échec fait échouer le pipeline
- Cache des dépendances actif, matrice d'au moins deux versions, jobs indépendants lancés en parallèle ; durée mesurée avant et après
- Secrets stockés dans GitHub, absents du code et des logs ; environnements staging et production, approbation requise pour la production
- Image Docker construite, publiée sur ghcr.io avec les tags SHA et latest, analysée par Trivy avec arrêt sur faille critique
- Déploiement automatique en staging sur un VPS par SSH ou sur Render ou Railway, vérifié par un test de santé
- Fusion d'une pull request bloquée quand les tests, le lint ou le seuil de couverture échouent
- Étapes communes regroupées dans une action composite ou un workflow réutilisable, appelés par au moins deux jobs ou workflows

10 INDICATEURS DE RÉSULTATS

Les indicateurs de résultats de cette formation (taux de satisfaction et taux de réussite) seront publiés sur sa fiche en ligne à l'issue des premières sessions réalisées.

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 électronique signé par chaque participant et par le formateur à chaque demi-journée, en présentiel comme à distance
- Traçabilité des activités pédagogiques réalisées

- Attestation d’assiduité délivrée en fin de formation
- Suivi individuel via les QCM de fin de journée, les exercices corrigés et les TP évalués

13 CONDITIONS D'ACCÈS

Formation accessible sans prérequis administratif particulier. 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