SIMPLON.CO

Référentiel de certification et d'évaluation

Appliquer les méthodes et outils de développement d'application dans une organisation DevOps

Juin 2024

REFERENTIEL DE COMPETENCES *Identifie les compétences et les connaissances, y compris transversales, qui découlent du référentiel d'activités*	REFERENTIEL D'EVALUATION *Définit les critères et les modalités d'évaluation des acquis*	
	MODALITES D'EVALUATION	**CRITÈRES D'ÉVALUATION**
C1. Créer un environnement de développement en s'appuyant sur une plateforme de conteneurisation* afin de réduire la complexité de la mise en place et de la maintenance des environnements de développement et de test.	**E1. Projet professionnel (C1, C2, C3, C4)** L'évaluation se fait dans un contexte de projet de développement d'une application, réelle ou fictive, avec une organisation DevOps*. Le candidat est partie prenante du projet de transformation des pratiques de développement, avec un rôle d'expert technique garant de la méthodologie à suivre. L'évaluation porte sur les étapes de définition des environnements jusqu'à la configuration des automatisations.	- Les choix techniques sont adaptés aux caractéristiques techniques du projet existant, - Chaque service est isolé dans un conteneur (api, client, reverse-proxy, base de données...), - Les ressources allouées à chaque conteneur sont définies et sont adaptées au service, - Les liens de dépendances et interconnexions entre les différents conteneurs fonctionnent et permettent à l'application de fonctionner, - L'application fonctionne dans

	Livrable : - rapport professionnel individuel, Évaluation : - correction du rapport professionnel, - soutenance orale individuelle.	l'environnement de développement créé, - Les configurations visent à faciliter le travail du développeur : accès aux sorties en console en temps réel, accès aux journaux utiles facilités, synchronisation des systèmes de fichiers, etc., - L'environnement est documenté dans un livrable pour permettre sa réplication par d'autres développeurs.
C2. Définir un flux de travail Git* en identifiant les fonctionnalités et les pratiques Git adaptées à la taille de l'équipe, aux modes d'organisation et au planning de livraison de l'application afin d'améliorer l'efficacité du travail d'équipe, de l'intégration continue et des livraisons.		- Une approche générale de versionnement (*git workflow**) est choisie et appliquée (*git flow, Github flow, ...)*, - Le cas échéant, une extension git-flow est choisie pour appuyer l'application du *git workflow*, - Les stratégies de *merge** sont précisées pour chaque étape prévue par le *git workflow*, - Un convention pour la rédaction des

		messages des *commits** est suivie (par exemple : Jira, Conventional Commits, etc.), - La gestion des secrets est définie, les fichiers à ignorer sont configurés correctement.
C3. Intégrer des tests statiques et dynamiques à un projet en configurant un environnement de test à partir de l'environnement de développement et en s'appuyant sur une plateforme de conteneurisation pour permettre la validation automatique du bon fonctionnement du périmètre de l'application testé.		- Le choix des outils est argumenté et l'intérêt de leur utilisation et/ou l'adaptation de ses outils à la situation est démontré, - Au moins 2 suites de tests unitaires sont intégrées, - Au moins 1 *linter** est configuré au projet, - La couverture du code par les tests unitaires est correctement évaluée, - Les tests statiques et dynamiques sont bien exécutés, - Le rapport des tests est interprété sans erreur.

C4. Créer une chaîne d'intégration continue*, adaptée au flux de travail Git, à l'organisation de l'équipe et aux livraisons de l'application, en définissant toutes les étapes de la chaîne et en configurant les déclencheurs automatiques des *builds** et des tests pour renforcer la stabilité de l'application.		- Les étapes, les flux et les actions de la chaîne d'intégration continue sont définis dans un diagramme, - Les environnements d'exécution de la chaîne sont adaptés, reproduisent l'environnement cible et donnent accès aux secrets (clés, token, mot de passe…) de façon sécurisée, - La configuration de la chaîne est produite et respecte le format de la plateforme utilisée (Gitlab, Github…) et les déclencheurs et actions préalablement définis dans le diagramme, - Lorsqu'un événement Git de la chaîne est déclenché, les actions prévues à cette étape (*build*, test…) s'exécutent correctement, - Les rapports de la chaîne sont interprétés sans erreur.

Glossaire :

- **Build** : Dans le domaine du développement de logiciels, est nommé “build” le processus de conversion des fichiers de code source en artefact(s) logiciel(s) autonome(s) pouvant être exécuté(s) sur un ordinateur, ou le résultat de ce processus.
- **Conteneurisation** : Un conteneur est une enveloppe virtuelle qui permet de distribuer une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers sources, environnement d'exécution, librairies et outils. Ils forment un ensemble cohérent et prêt à être déployé sur un serveur. Contrairement à la virtualisation de serveurs et à une machine virtuelle, le conteneur n’intègre pas de noyau; il s’appuie directement sur le noyau de l'ordinateur sur lequel il est déployé - il est donc plus léger et plus facilement portable qu’une machine virtuelle.
- **DevOps** : Le DevOps est un mouvement en ingénierie informatique et un ensemble de pratiques techniques et de modes d’organisation du travail visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops).
- **Git** : Git est un système de contrôle de version distribué qui suit les versions des fichiers. Il est souvent utilisé pour contrôler le code source par des programmeurs qui développent des logiciels en collaboration
- **Git workflow** : un workflow Git est une une recommandation expliquant comment utiliser Git pour accomplir une tâche de façon cohérente et productive. Les workflows Git encouragent les développeurs à exploiter Git de façon efficace et cohérente. Git procure une grande flexibilité, il n'existe aucun processus standardisé pour interagir avec lui, il est alors important de s’assurer que l'équipe est sur la même longueur d'onde quant à la méthode utilisée pour appliquer les changements.
- **Lint/linting/linter** : Au départ *lint* était un logiciel qui faisait une analyse statique de code C pour en détecter les erreurs récurrentes, les erreurs d'indentation et de syntaxe. Il a ensuite été cloné dans tous les langages de programmation. Le terme fait donc référence à un outil de test statique de la qualité d’une base de code face à un standard.

- **Merge (Git)** : Incorpore les modifications des commits nommés (depuis le moment où leurs historiques ont divergé de la branche courante) dans la branche courante.
- **Commit (Git)** : Enregistre les modifications dans le dépôt des sources et crée un nouveau “commit” contenant le contenu actuel de l'index et le message de log donné décrivant les changements.
- **Intégration continue (CI)** : L'intégration continue est une méthode de développement logiciel qui permet de vérifier fréquemment que le code fonctionne correctement après chaque modification.