Industrialisation CI/CD pour l'application "MicroFlow"

Contexte :

Vous êtes responsable de l'industrialisation de l'intégration continue (CI) et du déploiement continu (CD) pour un projet international utilisant une architecture microservices, **"MicroFlow"**. Ce projet est développé par trois équipes réparties entre la France et l'Inde. L'application est mise à jour régulièrement pour intégrer des correctifs et des évolutions. Vous devez mettre en place la stratégie DevOps et automatiser le pipeline CI/CD, en intégrant les meilleures pratiques pour la gestion des branches, des tests, de la qualité du code et du déploiement des microservices.

L'objectif est de configurer un processus DevOps complet en utilisant GitHub pour la gestion du code et des versions, en assurant l'intégration de tests automatisés, la gestion des artefacts Docker, ainsi que la notification des équipes sur les changements majeurs.

Objectifs :

1. **Définition de la stratégie DevOps :**
    - **Stratégie CI/CD :** Définir une stratégie pour la mise en place d'un pipeline CI/CD adapté à une architecture microservices (avec mise en production fréquente).
    - **Backlog des activités :** Créer un backlog des tâches DevOps à réaliser. Cela comprend la gestion des branches, la configuration des tests, la mise en place du pipeline CI/CD, et la gestion des artefacts Docker.
    - **Outils à utiliser :** Sélectionner et justifier les outils nécessaires, notamment GitHub pour la gestion du code source, GitHub Actions ou Jenkins pour le pipeline, Docker pour les conteneurs, SonarQube pour l'analyse de la qualité du code, et un système de notification (Slack, Teams ou autre).
2. **Mise en place du pipeline CI/CD :**
    - **Automatisation des étapes :** Créer un pipeline CI/CD automatisé qui :
        - Compile et construit les microservices à chaque commit sur GitHub.
        - Exécute les tests unitaires, d'API et End-to-End.
        - Analyse la qualité du code avec **SonarQube**.

        - Déploie les microservices dans un environnement de staging ou de production.
        - Pousse les images Docker dans un registre Docker (ex : Docker Hub).
        - Utilise des stratégies de branchement, de **code review** et de gestion des releases.

3. **Gestion des branches et des releases :**
    - **Gestion des branches :** Mettre en place une stratégie de branches (par exemple, **Git Flow** ou **GitHub Flow**) pour une gestion efficace du code et des versions.
    - **Code reviews :** Implémenter un processus de **code review** sur les pull requests avant la fusion dans la branche principale (master/main).
    - **Releases :** Organiser les releases et déploiements selon un cycle précis (par exemple, une mise à jour majeure chaque trimestre).
4. **Gestion des notifications et collaboration entre équipes :**
    - **Notifications :** Mettre en place un système de notifications pour informer les équipes des événements importants (tests échoués, déploiement réussi, nouvelles versions, etc.).
    - **Collaboration :** Assurer une collaboration efficace entre les équipes en France et en Inde via des outils comme **Slack**, **Jira**, ou **Trello** pour la gestion des tâches DevOps et des priorités.
5. **Suivi de la qualité et des bonnes pratiques DevOps :**
    - **Analyse de la qualité du code :** S'assurer que la qualité du code est régulièrement analysée avec **SonarQube** et que les problèmes de qualité sont corrigés.
    - **Automatisation des tests :** Veiller à ce que les tests soient exécutés automatiquement à chaque commit et que les erreurs soient remontées immédiatement.

Livrables attendus :

1. **Repository GitHub privé :**
    - Code source du pipeline CI/CD
      (fichier .github/workflows/ci.yml ou Jenkinsfile selon l'outil choisi).

- Fichiers de configuration Docker pour la gestion des microservices et du push dans le Docker Hub.
- Backlog détaillé des tâches DevOps dans un fichier **README.md**, expliquant la stratégie CI/CD, les outils utilisés, la gestion des branches et des releases, la gestion des tests et de la qualité du code, ainsi que la configuration des notifications.

2. **Pipeline CI/CD complet :**
   - Le pipeline CI/CD doit automatiser les étapes de construction, de test, de déploiement et d'analyse de la qualité du code, et inclure la gestion des artefacts Docker.
3. **Gestion des notifications :**
   - Configuration de notifications pour les événements importants du pipeline (échec des tests, succès des déploiements, analyse de la qualité du code, etc.).
4. **Documentation détaillée :**
   - Documentation expliquant les choix d'outils, les bonnes pratiques adoptées, la structure du pipeline, et la stratégie de gestion des branches et des releases.
   - Capture d'écran ou rapport détaillant le fonctionnement du pipeline avec les notifications mises en place.

Grille de notation (100 points)

Critère	Description	Points
1. Stratégie DevOps et choix des outils (15 points)	Définition claire et justifiée de la stratégie CI/CD, choix d'outils appropriés (GitHub, GitHub Actions ou Jenkins, Docker, SonarQube, Slack/Teams), et création d'un backlog des tâches DevOps.	15
2. Mise en place du pipeline CI/CD (35 points)	Configuration complète du pipeline CI/CD automatisant la construction, les tests unitaires, les tests d'API, End-to-End, l'analyse de la qualité avec SonarQube, le déploiement des microservices et la gestion des artefacts Docker.	35

Critère	Description	Points
3. Gestion des branches et des releases (15 points)	Mise en place d'une stratégie de branches (Git Flow ou GitHub Flow) et d'une procédure de code review avec des releases bien organisées.	15
4. Gestion des notifications (10 points)	Mise en place d'un système de notifications efficace pour informer les équipes des événements majeurs du pipeline (tests échoués, déploiement réussi, nouvelles versions).	10
5. Gestion des artefacts Docker (10 points)	Création et gestion des images Docker pour les microservices, configuration du push dans Docker Hub, intégration dans le pipeline.	10
6. Documentation et présentation (10 points)	Clarté et exhaustivité de la documentation du projet dans le **README.md**. Explication détaillée de la stratégie CI/CD, des outils utilisés, de la gestion des tests, des notifications et de la gestion des branches.	10
7. Suivi de la qualité du code (5 points)	Analyse de la qualité du code avec SonarQube et mise en place d'une stratégie pour corriger les problèmes identifiés.	5

Détails supplémentaires :

- **Format de rendu :** Vous devez soumettre un repository GitHub privé avec l'ensemble de votre travail, y compris le code source, les configurations du pipeline CI/CD, les configurations Docker, et la documentation.
- **Outils suggérés :**
    - **GitHub** pour la gestion du code source et des versions.
    - **GitHub Actions** ou **Jenkins** pour la gestion du pipeline.
    - **Docker** pour la gestion des microservices en conteneur.
    - **SonarQube** pour l'analyse de la qualité du code.
    - **Slack** ou **Teams** pour la gestion des notifications.
    - **Jira** ou **Trello** pour la gestion du backlog des tâches DevOps.