Documentation des tests automatisés et de la livraison continue

Documentation : Partie Intégration Continue (CI)

1. Installation de l’environnement de test

L'installation de l'environnement de test est automatisée via un workflow CI utilisant GitHub Actions.

- **Récupération du code source** : Le code est récupéré à partir du dépôt grâce à une action GitHub spécifique qui permet de télécharger la version du code à chaque déclenchement du workflow.
- **Installation des dépendances** : Les dépendances du projet sont installées à partir d'un fichier nommé `requirements.txt`, qui liste toutes les bibliothèques nécessaires. Cette étape garantit que tous les paquets requis pour le projet sont bien installés.
- **Mise en cache des dépendances** : Pour accélérer les prochaines exécutions du workflow, les dépendances Python sont mises en cache. Cela évite de réinstaller tous les paquets à chaque fois.

2. Dépendances installées

- **Version de Python** : L'environnement utilise Python 3.10, spécifié lors de la configuration pour s'assurer que le projet tourne avec la bonne version du langage.
- **Analyse de sécurité** : L'outil Bandit est installé pour analyser le code et détecter des éventuelles vulnérabilités de sécurité.
- **Tests et couverture** : Pytest est utilisé pour exécuter les tests unitaires. Un seuil de couverture des tests est défini à 80%, ce qui signifie que si la couverture est inférieure à ce seuil, le workflow échoue.

3. Exécution des tests

- **Test de sécurité** : Un scan de sécurité est effectué sur l'ensemble du code pour identifier les vulnérabilités potentielles. Cette étape permet de détecter des problèmes de sécurité avant de déployer le code.
- **Vérification de la syntaxe** : Le code est également soumis à un contrôle de formatage pour s'assurer qu'il respecte les standards définis dans le projet.
- **Tests unitaires** : Des tests sont exécutés automatiquement sur les différentes parties du projet, et la couverture de code est mesurée pour garantir que les tests couvrent une grande partie du code.

Documentation : Partie Chaîne de Livraison Continue (CD)

1. Installation et Configuration

Le processus de livraison continue (CD) utilise un workflow dédié, également configuré via GitHub Actions. Il repose sur des fichiers YAML qui spécifient les étapes à suivre pour déployer le projet. Ce workflow se déclenche après la réussite des tests dans la phase d'intégration continue.

- **Création de l'image Docker** : Le code source est utilisé pour construire une nouvelle image Docker. Cette image contient toutes les instructions nécessaires pour faire tourner l'application.
- **Connexion à Azure Container Registry** : Une fois l'image Docker créée, elle est envoyée vers le registre de conteneurs d'Azure où elle est stockée. Les informations d'authentification pour Azure sont sécurisées à l'aide des credentials Azure stockés de manière sécurisée.

2. Déploiement et Test de la Chaîne

- **Suppression des instances précédentes** : Si une instance de l'application est déjà en cours d'exécution sur Azure, elle est supprimée pour permettre le déploiement de la nouvelle version.
- **Création d'une nouvelle instance** : Une nouvelle instance de conteneur est créée sur Azure avec l'image Docker mise à jour. Les ressources (comme le CPU, la mémoire et les ports) sont définies pour garantir que l'application s'exécute dans un environnement adapté.
- **Surveillance et validation** : Le processus inclut des étapes pour surveiller le déploiement et valider que tout s'est bien passé. Si des erreurs surviennent, elles sont automatiquement signalées.

3. Sécurisation des informations sensibles

Les informations sensibles comme les identifiants Azure et les clés de connexion sont stockées de manière sécurisée via les credentials d’Azure. Cela garantit que les informations ne sont pas exposées directement dans le code, et que seules les personnes autorisées peuvent y accéder.