Sujet de Projet : Développement Système et Cloud - Gestion de Projet, Déploiement et Intégration Continue

Contexte

Ce projet vous permettra de développer des compétences en développement système et cloud, en couvrant des aspects clés tels que la gestion de projet agile, la configuration d'environnements de développement, et la mise en place d'un processus d'intégration continue. Le projet est structuré en plusieurs phases, chacune nécessitant des livrables spécifiques et la rédaction d'une documentation destinée à faciliter la prise en main du projet par un nouveau contributeur. Vous allez également développer une application exemple sous la forme d'un micro-service (API) orientée IT, qui servira de support pratique tout au long du projet.

Certifications DevOps

Avant de débuter le projet, une phase de formation est requise pour mener à bien celui-ci. Les deux ressources ci-dessous sont à étudier tout au long du projet.

1. **IBM Cloud - Introduction to DevOps**

- **Plateforme** : IBM Skills Network
- **Description** : Ce cours couvre les principes de base de DevOps, y compris l'automatisation, le CI/CD, la gestion de configuration et les pratiques de surveillance.
- **Certification** : Gratuit avec un certificat de fin de cours.
- **Lien** : IBM Cloud DevOps

2. **AWS Educate**

- **Plateforme** : Amazon Web Services (AWS)
- **Description** : AWS propose un programme éducatif pour les étudiants qui donne accès à des cours et des laboratoires pratiques sur des sujets DevOps comme l'automatisation, les conteneurs, et le CI/CD.
- **Certification** : AWS Educate Badge est gratuit pour les étudiants.
- **Lien** : AWS Educate

Phase 1 : Gestion de Projet Agile et documentation

Objectifs :

- **Structurer le projet selon une méthodologie agile** : Planifier le travail en sprints, définir un backlog, et suivre l'avancement des tâches.
- **Utiliser un outil de gestion de projet** : Mettre en place un tableau de suivi (Trello, Jira, ou GitLab) et y consigner les tâches, les responsables, et les délais.
- **Rédiger une documentation claire et structurée** : Créer un document au format markdown (Document As Code) détaillant chaque commande, outil, et processus mis en place, pour qu'un collaborateur puisse reproduire l'environnement.

Étapes Intermédiaires :

1. **Créer un backlog produit** : Lister toutes les tâches nécessaires pour réaliser le projet, les prioriser, et les diviser en sprints de 2 semaines.
2. **Définir les rôles et responsabilités** : Assigner les tâches aux membres de l'équipe, en définissant clairement les objectifs de chacun pour chaque sprint.
3. **Mettre en place un tableau de suivi** : Choisir un outil de gestion de projet (Trello, Jira, GitLab) et y créer un tableau Kanban pour suivre l'avancement des tâches.
4. **Documentation initiale** : Rédiger une première version du document expliquant comment cloner le dépôt Git, installer les dépendances, et démarrer le projet.

Livrables :

- **Un backlog produit** avec les tâches priorisées.
- **Un tableau de gestion de projet** mis à jour régulièrement.
- **Un document au format markdown** documentant le projet, mis à jour après chaque phase.

Ressources :

- Introduction à la méthodologie Agile
- Guide de gestion de projet avec Trello
- Documentation Markdown

Évaluation :

- Cohérence et exhaustivité du backlog.
- Adhérence à la méthodologie agile (suivi des sprints, ajustements).
- Qualité et clarté de la documentation.

Phase 2 : Développement de l'Application Exemple et Mise en place de l'Environnement de Développement

Objectifs :

- **Développer une application exemple sous la forme d'un micro-service (API) orientée IT** : Cette API gérera un inventaire de machines virtuelles, permettant de consulter, créer, mettre à jour, et supprimer des entrées via des endpoints RESTful.
- **Créer un environnement de développement Python** en suivant les bonnes pratiques, incluant la gestion de versions avec Git et l'intégration d'outils de tests.
- **Configurer un système de gestion de versions avec Git** pour assurer un suivi rigoureux des modifications de code.
- **Mettre en place des outils de vérification de code et d'automatisation des tests** avec `pre-commit` et d'autres outils de linting et de formatage.
- **Utiliser Vagrant pour créer des machines virtuelles** afin de tester le déploiement des applications en environnement isolé.

Étapes Intermédiaires :

1. **Définition des spécifications de l'API** : Identifier les endpoints à créer pour la gestion de l'inventaire des machines virtuelles (GET, POST, PUT, DELETE).
2. **Développement de l'API** : Créer une API en Python (utilisant un framework comme Flask ou FastAPI) avec des endpoints fonctionnels.
3. **Initialisation de l'environnement Python** : Installation de Python, création d'un environnement virtuel, et gestion des dépendances avec `pip`.
4. **Configuration de Git** : Création du dépôt, configuration des branches, écriture des premiers commits.
5. **Mise en place de pre-commit** : Installation de `pre-commit`, configuration des hooks pour inclure des outils comme `flake8` et `black`.
6. **Création de scripts Vagrant** : Écriture de scripts pour provisionner des VM qui imitent l'environnement de production.

Livrables :

- Une **API fonctionnelle** avec documentation des endpoints.
- Un **environnement de développement Python** opérationnel.
- Un **dépôt Git** bien structuré avec un historique de commits.
- Une **configuration pre-commit** intégrant les outils nécessaires.
- Des **scripts Vagrant** permettant de créer des VM pour les tests.
- **Un document au format markdown** documentant le projet, mis à jour après chaque phase.

Ressources :

- Documentation Flask
- FastAPI - Documentation
- Guide Git pour les débutants
- Documentation Pre-commit
- Vagrant - Guide de démarrage

Évaluation :

- Fonctionnalité et robustesse de l'API développée.
- Qualité de l'environnement de développement (cohérence, efficacité).
- Utilisation correcte de Git et respect des conventions de commit.
- Intégration et fonctionnalité des outils de vérification de code.
- Fonctionnalité et reproductibilité des scripts Vagrant.

Documentation :

- **Rédiger un guide complet en markdown** expliquant chaque étape du processus, depuis l'installation des outils jusqu'à la configuration des scripts, afin qu'un contributeur puisse recréer l'environnement de développement et déployer l'API.

Phase 3 : Déploiement et Intégration Continue

Objectifs :

- **Configurer une machine virtuelle partagée** pour l'équipe de développement, intégrant une registry Docker privée.
- **Conteneuriser l'API** développée en créant une image Docker.
- **Automatiser le processus de build et de push d'images Docker** vers la registry.
- **Déployer et exécuter les conteneurs Docker** sur la VM partagée.

Étapes Intermédiaires :

1. **Création de la VM partagée** : Utilisation d'Ansible ou d'un autre outil pour configurer une VM accessible à toute l'équipe.
2. **Installation de Docker et de la registry Docker** : Mise en place d'une registry Docker privée sur la VM et configuration pour permettre l'accès sécurisé.
3. **Création du Dockerfile pour l'API** : Conteneuriser l'API en utilisant Docker, en spécifiant toutes les dépendances nécessaires dans un Dockerfile.
4. **Automatisation du pipeline CI/CD** : Utilisation d'un outil comme GitLab CI pour automatiser le build, le test, et le push des images Docker.
5. **Test de déploiement** : Déploiement de l'API conteneurisée depuis la registry et exécution sur la VM pour validation.

Livrables :

- Une **VM partagée** avec Docker et une registry privée configurée.
- Un **Dockerfile** fonctionnel pour l'API.
- Un **pipeline CI/CD** automatisé pour le build et le déploiement d'images Docker.
- Documentation complète du processus de déploiement.

Ressources :

- Docker - Guide de démarrage
- Configurer une registry Docker privée
- Introduction à l'intégration continue avec Jenkins
- Ansible pour l'automatisation de déploiement

Évaluation :

- Fonctionnalité de l'API conteneurisée et déployée.
- Fonctionnalité et sécurité de la registry Docker.
- Efficacité et automatisation du pipeline CI/CD.
- Documentation claire et détaillée.

Documentation :

- **Mettre à jour le document** pour inclure les instructions de déploiement, la configuration du pipeline, et l'utilisation de la registry Docker.

Conclusion

Ce projet vous permettra de maîtriser le cycle de vie complet du développement d'une application, depuis la gestion de projet jusqu'au déploiement en environnement cloud. Vous serez également en charge de

rédiger une documentation claire pour faciliter la prise en main du projet par un autre membre de l'équipe. L'application exemple que vous développerez sera un micro-service (API) orientée IT pour gérer un inventaire de machines virtuelles, vous permettant d'appliquer concrètement les compétences acquises.