Pratiques Devops

Partie 1 – Introduction à DevOps (origine, principes, enjeux)

1.

Le Devops est ensemble de pratiques qui met l'emphase sur l'automatisation des processus entre les équipes de développement et celle en charge du maintien en condition opérationnelle de l'application développée. Il chercher principalement les problèmes. Il vise principalement à supprimer le fossé historique entre les équipes de développement (Dev), axées sur l'innovation et la rapidité, et les équipes de production/exploitation (Ops), garantes de la stabilité et de la sécurité.

2. Situation ou le Devops peut aider

- Prenons le cas d'étudiants qui travaillent sur une application, pour tester l'application il doivent pousser leur travaux sur une branche Github pour les fusionner et faire une demo lors de la présentation, malheureusement le jour-j l'application marche chez l'un et pas chez l'autre a cause des versions d'outils différents. Le Devops ici va permettre grâce à Docker d'eviter les "ça marche sur mon PC" et automatise les tests avec GitHub Actions.
- La banque veut sortir une nouvelle fonctionnalité de "virement instantané". Les développeurs finissent

le code en 2 semaines, mais l'équipe "Ops" (production) met 1 mois à préparer les serveurs car ils n'étaient pas au courant des besoins techniques. Lors du lancement, l'app plante sous le poids des utilisateurs, et il faut 3 jours pour revenir à l'ancienne version. Le Devops ici Réduit le délai de mise en production et sécurise les mises à jour via des déploiements automatisés (CI/CD)

3. Réponses aux questions

Ce qui m'a le plus surpris, c'est que le DevOps mets aussi en valeur les relations humaines, permettant ainsi de créer des liens entre Dev et Ops afin d'avoir un bon rendu au niveau de la production. Il s'agit ici pour moi d'une sorte de tradition, la communication entre les Dev et les Ops a une très grande place au seins d'une structure, car permet au 2 équipes d'être au même niveau sur la réalisation d'un projet tout en créant des liens.

En mémé temps ce qui me semble le plus difficile a mettre en place dans un projet, c'est la **Communication**, car elle est un élément clé entre 2 équipes projet.

Partie 2 – Culture DevOps (collaboration, silo, communication)

1.

Pour moi la culture DevOps est moyen permettant de changer les mentalité, axer plus sur la communication entre les équipes Dev et Ops au sein structure pour réaliser un bien un projet.

2.

Les 3 moyens pour moi pour qu'on parle de DevOps au sien d'une Structure son :

- Il peuvent participer a certaines activités (déjeuner ensemble, programmer des sorties de groupes entre les 2 équipes)
- Les Dev peuvent inviter les Ops a leur Daily Meetings et vice-versa afin que chaque équipes puissent être au courant des différentes problématique au sein des 2 équipes, et ces problématiques seront incluses dans les différents sprints pour qu'ils soient résolues.
- Les 2 équipes doivent partager leur idées sur le projet histoire pour ceux-ci de diversifier les idées.

3.

Personnellement, j'applique ces mêmes idées, pour moi la réalisation d'un bon projet est base sur la communication, ici il faut savoir créer des liens avec les autres membres du groupe, partager les idées, s'écouter les un et autres. Et surtout, je priorise l'atmosphère au sein du groupe, elle doit être bonne pour un bon travail.

Je me reconnais sur un certain point qui est la communication, car je m'intègre assez facilement en communicant avec les membre du groupe par rapport au projet. Malgré tout je dois m'améliorer concernant le fait de poser ses problèmes, car j'ai du mal poser mes difficulté par peur de jugement.

Partie 3 - Le modèle C.A.L.M.S. (Culture, Automation, Lean, Measurement, Sharing)

1.

Le CALMS est un référence définissant la culture DevOps à travers cinq piliers : **la Collaboration**, **l'Automatisation**, **la Lean**, **la Mesure** et **le Sharing**.

2.

Voici un exemple de projet qui illustre le CALMS : une application mobile de météo).

- **C (Culture)** : Les développeurs et les administrateurs système se réunissent chaque matin pour discuter des objectifs communs, au lieu de travailler chacun de leur côté.
- **A (Automation)** : Dès qu'un développeur valide son code, l'automatisation des tests se fait pour vérifier qu'il n'y a pas de bugs.
- **L (Lean)** : L'équipe décide de livrer une seule petite fonctionnalité par semaine plutôt que d'attendre 3 mois pour une grosse mise à jour risquée.
- **M (Mesure)** : Un tableau de bord (Grafana) permettra d'afficher en temps réel le nombre d'utilisateurs qui ont des crashs sur l'application pour réagir immédiatement.
- **S (Sharing)**) : Si un bug grave survient, l'équipe rédige un document partagé expliquant la cause et la solution pour que tout le monde apprenne de cette erreur.

3.

Je me sens plus alaise avec la lettre **C**, car j'aime bien créer des liens dans un groupe. Malheureusement je me sentirai mal a l'aise avec la lettre **A**, l'automatisation me sera peut-être difficile.

Partie 4 - Automatisation, Intégration Continue et Livraison Continue

1.

La CI **(Intégration Continue)** automatise la validation du code (tests, build) à chaque ajout pour détecter les bugs tôt. La CD **(Livraison/Déploiement Continu)** automatise la mise en production de ce code validé pour qu'il soit accessible aux utilisateurs sans intervention manuelle.

2.

Gestionnaire de tâches CLI

Lancement : Automatique à chaque git push sur la branche main ou lors d'une Pull Request.

Étapes automatiques :

- **Checkout** : Récupération du code source.
- **Linting** : Vérification automatique de la syntaxe.
- **Tests** : Vérification que les fonctions (ajouter/supprimer tâche) marchent.
- **Build** : Création de l'exécutable ou de l'image Docker.

3.

Git et Branches : Le pipeline surveille la branche main. Les tests se lancent sur les branches secondaires pour valider le travail.

Pull Requests (PR) : Le pipeline sert de "garde-fou", il interdit le Merge si les tests échouent (croix rouge).

Scripts Shell : Ils sont le moteur du pipeline. Chaque étape (test, build) est lancée par une commande Bash dans le fichier de configuration (YAML).

Partie 5 - Bonnes pratiques et pièges DevOps)

1.

Les 3 bonnes pratique DevOps sont :
- Équipe d'évangéliste DevOps
- Automatiser les tests (CI)
- Favoriser le Lean

Ces pratiques sont importantes car permettent de creer une culture DevOps et favorise la realisation de bon projet.

2.

Les 2 mauvaise pratique DevOps sont :
- Silo Dev et Silo Ops
- Automatisation sans connaitre les commandes

3.

La mauvaise pratique que je risque de faire moi même c'est **l'automatisation sans connaitre les commandes**

et pour remédier, je vais les apprendre avant d'attaquer la partie concernant l'automatisation.