L’importance du ci/cd dans un projet de dév

Définition :

En génie logiciel, **CI/CD** (parfois écrit CICD) est la combinaison des pratiques d'intégration continue et de livraison continue ou de déploiement continu.

Les plateforrmes de CI/CD

Gittlab,github(qui intègre le ci/cd dans les repo) et jenkins,travis CI, Drone circleci (qui sont des plateformes fait que pour le CI/CD.

Implémentation dans GitLab :

Quelque définition :

Runner : GitLab Runner est une application qui fonctionne avec GitLab CI/CD pour exécuter des tâches dans un pipeline (ça peut être soit le serveur runner fournit par gitlab qui exécute nos taches ou bien une machine personnelle)

Pipeline : est une combinaison de tâches qui seront exécuter par le runner après chaque commit.

Artifacts : les fichiers produit par le job et qu’on veut garder après

Comment définir le pipeline :

Il faudra ajouté un fichier .gitlab-ci.yml en racine du projet qui définit une liste des stages(étapes) et un ensemble de jobs(commandes) qui sont inclus dans un des stages défini juste avant.

Mots clés peuvent être utilisé dans un job

-Image : image docker utilisée pour exécuter le job (on peut choisir que notre jobs soit exécuter sur diffèrent type d’exécutors “page 10” dons un container docker pour cella il suffit de rajouté le mot clé image avec l’image docker sur la quelle vous voulez exécuté votre projet par exemple pour un projet node sur arm ça serai image: balenalib/armv7hf-node)

- script: commande shell à appliquer

- variables: variables d’environnement

- artifacts: artifacts produits par le job à garder

(Quand un job se déclenche, le repo git est cloné dans le répertoire du runner, et le script démarre)

Pourquoi ?

Pouvoir automatiser les tests, augmenter la fréquence de distribution

Conclusion

L'importance du CI/CD

Qu'est-ce que c'est ?

Le CI/CD : introduire un haut degré d'automatisation et de surveillance continues dans le processus de développement des applications.

À quoi ça sert ?

Intégrer rapidement tout nouveau segment de code : aujourd'hui nous sommes de plus en plus nombreux à collaborer sur des projets que nous devons livrer de plus en plus vite !

Intégrer fréquemment tout nouveau segment de code : éviter les conflits en mergeant plus souvent des plus petits morceaux de codes (horreur : faire merger 10 dev, le même jour (jour du déploiement) avec chacun des centaines de lignes en commun...).

Surveillance en continu du code, garantir une qualité de service : critère de qualité (performance, taux de couverture, temps de réponse...).

Disposer d'un code toujours prêt à être déployé en production : déploiement depuis n'importe quel endroit, n'importe quand.

(pour les gens qui veulent vraiment savoir comment ça marche)

Petit exemple d'utilisation (projet angular) :

Pour un projet angular on voudrait qu'après chaque commit notre Ci/CD exécute quelque test et après déployer notre app web sur une rasp

Il suffit de crée 3 stages avec 4 jobs

Le premier stages test : Avec un job qui exécute nos test

Le 2 eme stage et le build :

> Avec un job angular_build av (on imagine qu'on est sur un executors de type Shell) on aura de dans juste une commande ng build (commande pour build d' un projet anguar en version dist)

Et le 3eme stage deploy :

Avec un job deploy_angular (de type shell aussi sur le quelle on utilisera un pscp qui nous permet de transférer des fichiers en ssh) avec une command

> pscp.exe -P 22 -l pi -pw raspberry path/dit/angular pi@192.168.x.x:/var/html/www