Mini-projet Test et Qualité logicielle

Présenté par : Gargouri Faten

Année universitaire 2021/2022

Plan

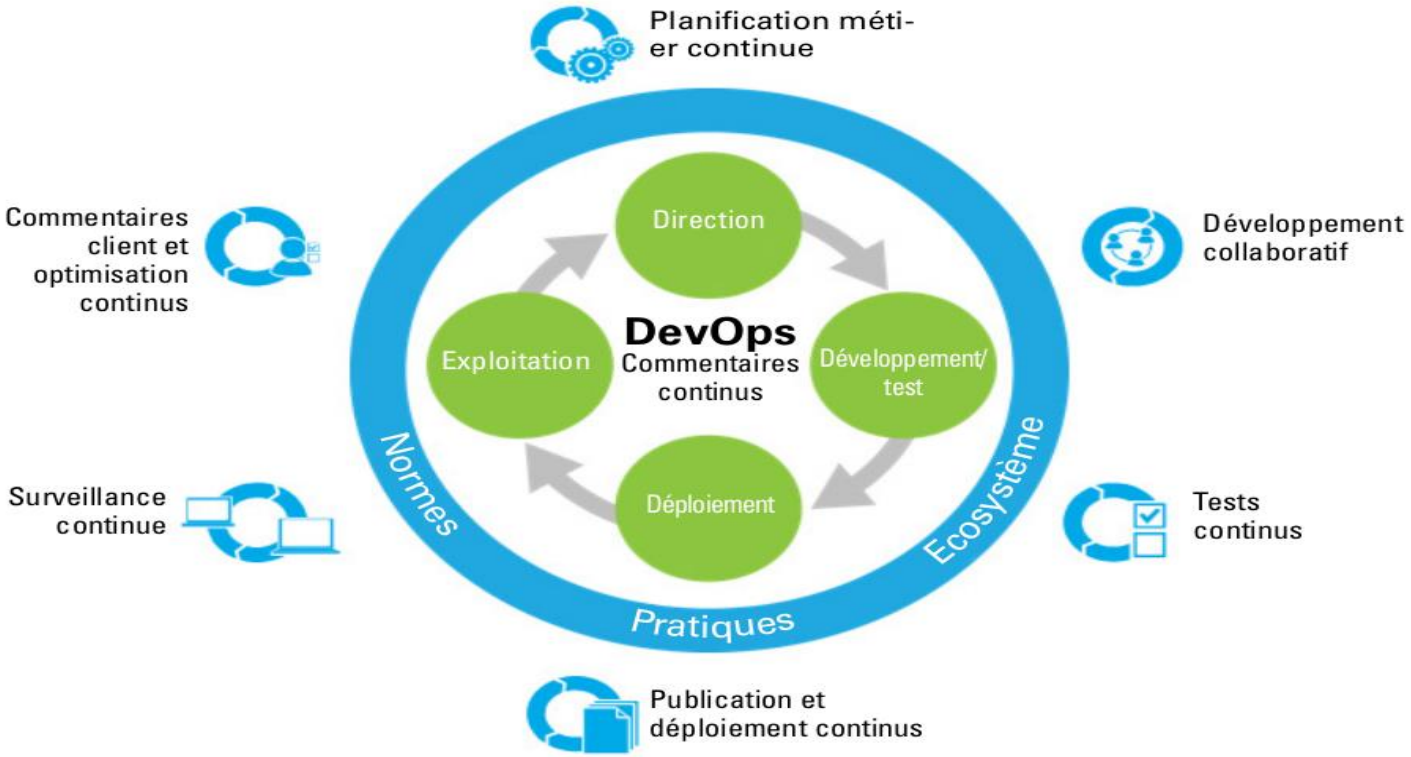
*C’est quoi chaine DevOps.
*Les pratiques proposés .
*Différentes étapes du cycle de vie
*Les outils DevOps .
*Les outils de test proposés.
*Les outils pour la qualité de code proposés.

C’est quoi chaine Devops ?

Une chaîne Devops :est un ensemble d’outils de programmation utilisés pour réaliser une tâche complexe en développement de logiciel .

Les pratiques proposés

Les pratiques DevOps améliorent en continu et automatisent les processus.



-Développement continu: **Cette pratique couvre les phases de planification et de codage dans le cycle de vie DevOps et peut inclure des mécanismes de contrôle des versions.**

-Tests continus: **Cette pratique prévoit des tests automatisés, planifiés et continus lors de l'écriture ou de la mise à jour du code de l'application qui accérelent la livraison du code en production.**

-Intégration continue: **Cette pratique rassemble des outils de gestion de la configuration, de test et de développement pour assurer le suivi de la mise en production des différentes portions du code. Elle implique une collaboration étroite entre les équipes responsables des tests et du développement pour identifier et résoudre rapidement les problèmes de code.**

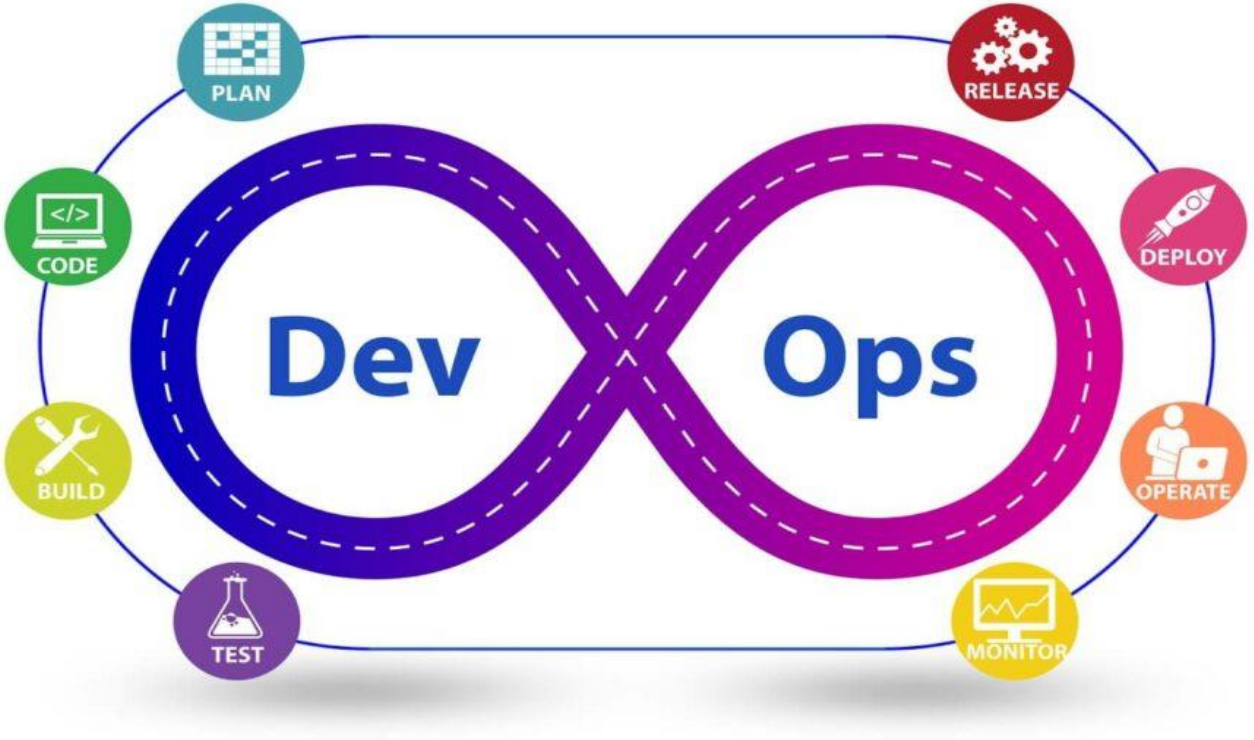
-Livraison continue : Cette pratique automatise la publication des modifications du code après la phase de test, dans un environnement intermédiaire ou de préproduction.

-Déploiement continu : Cette pratique automatise la publication d'un code nouveau ou modifié dans l'environnement de production,

-Surveillance continue: Cette pratique prévoit une surveillance continue du code exécuté et de l'infrastructure sous-jacente.

-Infrastruce as code : Cette pratique peut être suivie dans plusieurs phases DevOps pour automatiser le provisionnement de l'infrastructure requise pour une version logicielle.

Différentes étapes du cycle de vie DevOps



1- Planification : Cette phase permet de définir la valeur commerciale et les exigences. Jira et Git peuvent être utilisés pour le suivi des problèmes connus et la gestion des projets.

2- Code : Cette phase inclut la conception logicielle et la création du code logiciel à l'aide des logiciels GitHub, GitLab, Bitbucket ou Stash, par exemple.

3- Création : Cette phase consiste à gérer les versions logicielles et à exploiter des outils automatisés pour compiler et intégrer le code en vue de sa mise en production. Des référentiels de code source ou de package « empaquettent » aussi l'infrastructure requise pour la livraison du produit à l'aide des logiciels Docker, Ansible, Puppet, Chef, Gradle, Maven ou JFrog Artifactory, par exemple.

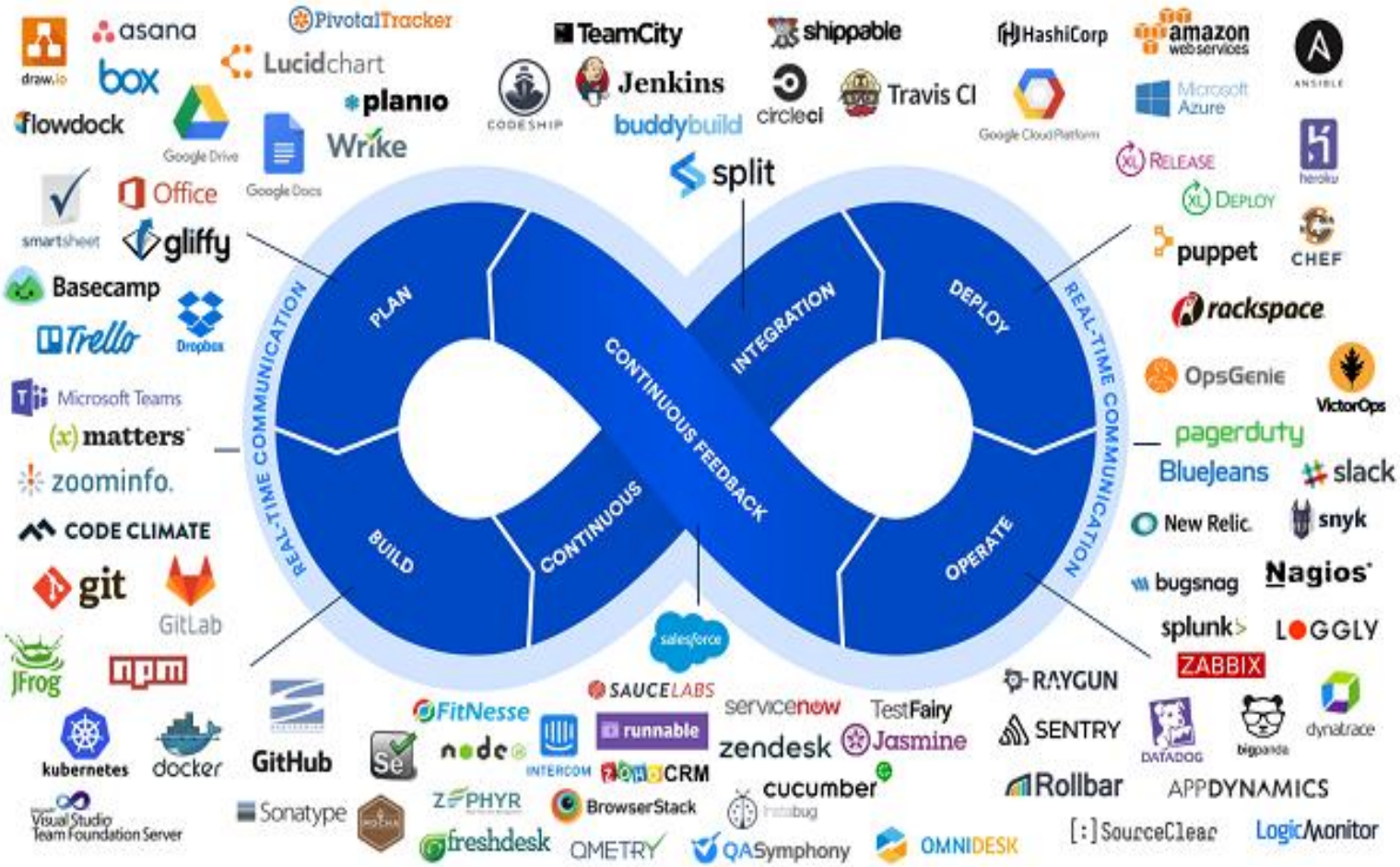
4- Test : Cette phase comprend des tests continus, qu'ils soient manuels ou automatisés, et vise à assurer une qualité de code optimale à l'aide des logiciels JUnit, Codeception, Selenium, Vagrant, TestNG ou BlazeMeter, par exemple.

5- Déploiement : Cette phase peut inclure des outils de gestion, de coordination, de planification et d'automatisation de la mise en production des produits, avec Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker ou Jira, par exemple.

6- Exploitation : Cette phase permet de gérer les logiciels en production à l'aide des logiciels Ansible, Puppet, PowerShell, Chef, Salt ou Otter, par exemple.

7- Supervision: Cette phase permet d'identifier les problèmes affectant une version logicielle en production et de collecter les informations correspondantes à l'aide des logiciels New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios ou Slack, par exemple.

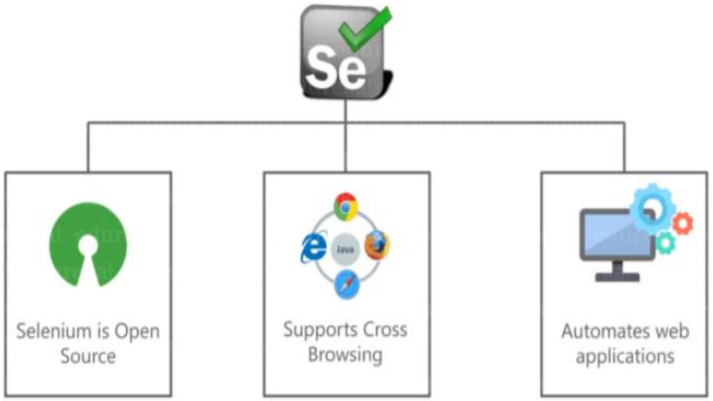
Les outils devOps



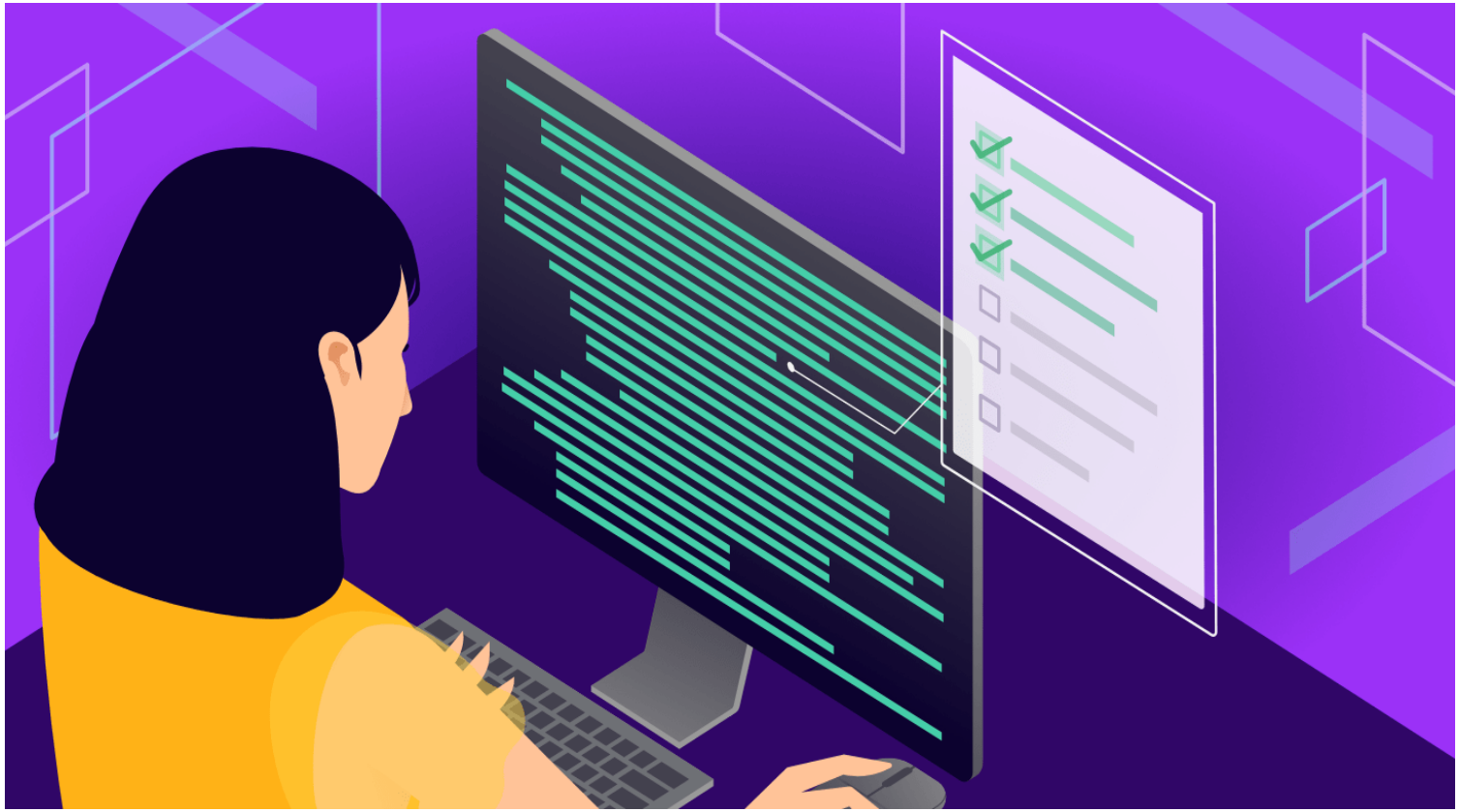
Les outils de test proposés.

1. **Junit:** Un framework de test unitaire pour le langage de programmation java, créé par Kent Beck et Erich Gamma . JUnit définit deux types de fichiers de tests : TestCase et TestSuite .

2. **Sélénuim:** C'est un automate de navigateurs. Plus précisément, il propose une suite d'outils pour interagir avec des navigateurs et planifier les exécutions des scripts.



Les outils pour la qualité de code proposés



1- Checkstyle : est un outil d'analyse statique pour Java. A l'origine conçu pour faire respecter un ensemble de normes de codage hautement configurable, Checkstyle permet aussi maintenant de vérifier les mauvaises pratiques de codage, ainsi que le code trop complexe ou dupliqué.

2-**PMD/CPD :** Il se focalise sur les problèmes potentiels de codage comme le code non utilisé ou sous-optimisé, la taille et la complexité du code, et les bonnes pratiques de codage.

3-**FindBugs :** est un puissant outil d'analyse de code qui vérifie le byte code de votre application afin de trouver des bogues potentiels, des problèmes de performances, ou des mauvaises habitudes de codage.

4-CodeNarc : est un outil d'analyse statique de code Groovy, similaire à PMD pour Java. Il vérifie le code source Groovy afin de trouver des défauts potentiels, des mauvais styles et pratiques de codage, du code trop complexe, et ainsi de suite.

Merci pour votre attention