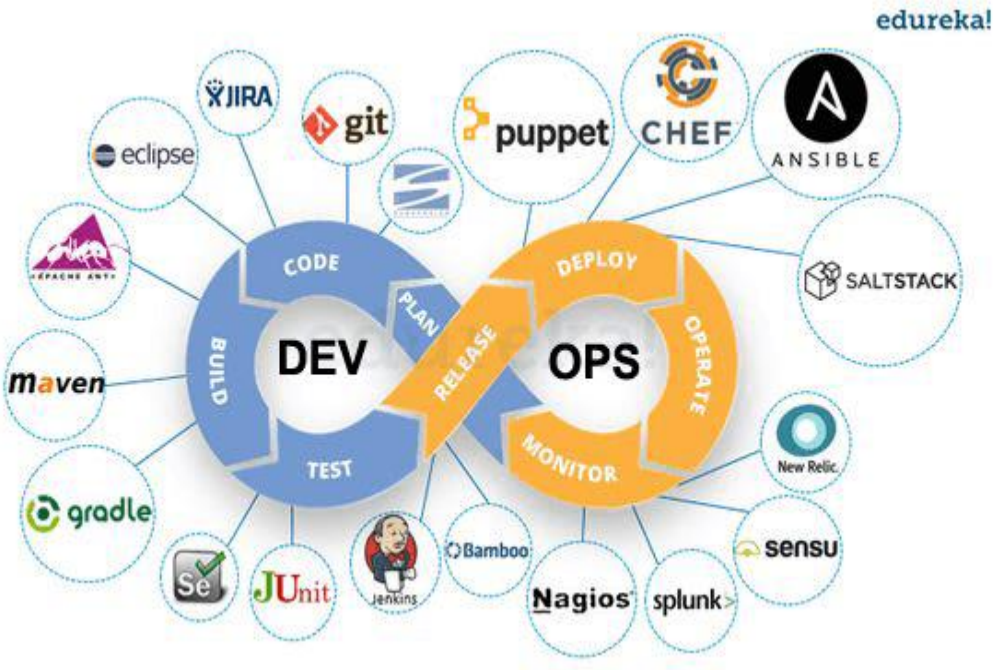
Table des matières

Qu’est-ce que Dev OPS ?

Le Dev OPS est définie comme une démarche qui consiste à faire collaborer étroitement les équipes de développement & études avec les équipes des opérations et d’exploitation. Cette collaboration a pour but d’automatiser et d’intégrer les processus entre ces deux équipes, dans une responsabilité commune.

Les outils DevOPS.



Les outils de monitoring DevOps fournissent des capacités d'automatisation et de mesure tout au long du cycle de développement. Ils sont indispensables pour bénéficier d'une vue d'ensemble temps réel de l'environnement de production, face à la haute fréquence des changements de code.

Qu'est-ce qu'un Système de contrôle de version ?

Egalement appelé **contrôle de source,** le contrôle de version désigne la pratique consistant à suivre et à gérer les changements apportes au code d'un logiciel, d'une application ou d'une page web. Ils en existent deux types

Centraliser VCS	VS	Distribuer VCS
-Utilise un serveur centraliser pour stocker toutes les version de fichiers.		- Améliore la capacité de travail hors ligne et ne dépend pas d'un emplacement pour les sauvegarde
-Aucun développeur ne dispose d'une copie de tous les fichier en / locale.		-Chaque développeur dispose d'une copie de toutes les version en locale.
-S'il arrive que le serveur central tombe en panne, toutes les données du projet seront perdues.		Il n'y a aucune menace dans le cas où le serveur tomberait en panne.

Qu'est-ce que Git ?

Git est un logiciel de gestion de version décentraliser. C'est un logiciel libre et gratuit, créer en 2005 par Linus Torvalds, auteur du noyau Linux, et distribue selon les termes de la Licence publique générale **GNU** version 2.

A quoi sert Git ?

Git permet de conserver un historique des modifications effectuées sur un projet afin de pouvoir rapidement identifier les changements effectués et de revenir à une ancienne version en cas de problème.

Parmi les logiciels de gestion de versions, Git est le leader incontesté et il est donc indispensable pour tout développeur de savoir utiliser Git.

Les Fonctionnalités Git.

- Historique de suivi. C'est probablement la fonctionnalité la plus importante de git car elle garde une trace de tous les changements qui se produisent.
- Gratuit et Open source. Git est un outil open source et es publié sous la licence GPL [General Public Licence].
- Evolutif. Git est évolutif, ce qui signifie que lorsque le

nombre d'utilisateur augmente, git gère facilement une telle situation.

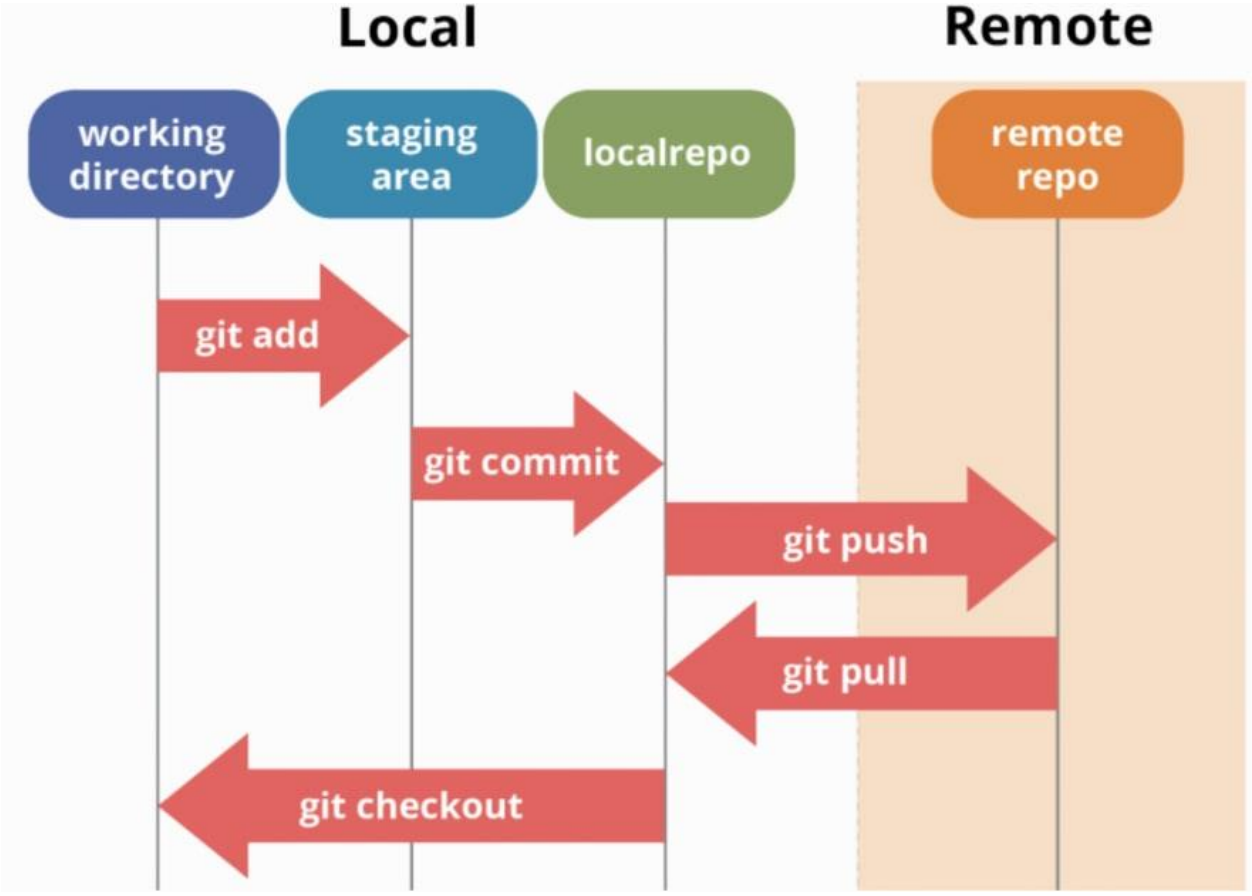
- Sécurité. Git es sécuriser, il utilise le SHA1[fonction sécurisée] pour nommer et identifier les objets dans son référentiel{repository}.
- Speed.
- Branching et Merging.

Flux de travail de Git.

Le flux de travail typique d'un projet git :

- Premièrement obtenez une copie locale di projet en clonant un dépôt, ou repo. Si vous collaborez, vous devez d'abord « forker » {Un fork est une copie séparée d'un dépôt git que vous pouvez gérer et modifier sans affecter le projet original.} le repo {référentiel}.
- Ensuite créez une branche avec un nom représentatif de la fonctionnalité sur laquelle vous travailler.
- Modifier le projet.
- Validez les modifications sur votre machine locale.
- Poussez les modifications vers repo distant.
- Créez un pull request vers le repo d'origine.

- Fusionnez et résolvez les conflits dans la branche principale du repo d'origine.



Les 3 zones différentes pour votre code Git :

- Répertoire de travail [Directory] : Zone dans laquelle vous allez travailler (création, modification, suppression et organisation des fichiers).
- Zone de transit [Zone d'index] : La zone dans laquelle vous listerez les modifications apportées au répertoire de travail.

- Référentiel [Dépôt local] : Ou git stocke en permanence les modifications que vous avez apportées sous différentes versions.

Les Branches :

La branche par défaut dans Git s'appelle **master**. Une branche est une version parallèle d'un dépôt qui permet de travailler sur des fonctionnalités ou des corrections sans affecter la branche principale (ou master). Les branches sont utilisées pour ;

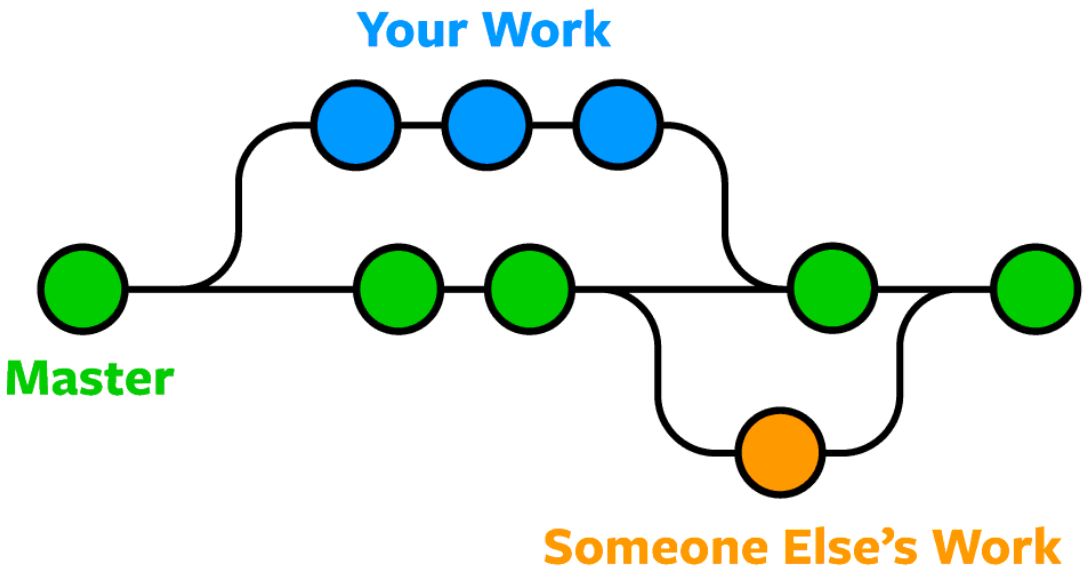
- La maintenance d'anciennes versions du logiciel (sur les branches) tout en continuant le développement des futures versions (sur le tronc) ;
- Le développement parallèle de plusieurs fonctionnalités volumineuses sans bloquer le travail quotidien sur les autres fonctionnalités.

Git Branche : La commande git branche peut-être utilisée pour répertorier, créer ou supprimer des branches. Pour répertorier toutes les branches présentes dans dépôt, utiliser :

```
git branch <nom-branche>.
```

Pour supprimer une branche :

```
git branch   -d <nom-branche>.
```



COMMANDES GIT DE BASE

Git config : Git config es l'une des commandes git les plus utilisées. On l'utilise pour configurer les préférences de l'utilisateur.

```
git config –-global user .email
```

Git init : Cette commande est utilisée pour créer un nouveau dépôt git.

```
git init
```

Git add : Cette commande est utilisée pour ajouter des fichiers à l'index.

```
git add temp.txt
```

Git clone : La commande git clone es utilisée pour cloner un dépôt distant vers la machine locale.

```
git clone
```

Git commit : La commande git commit permet de valider les modifications apportées au HEAD. Notez que tout commit ne se fera pas dans le dépôt distant.

```
git commit –m ‘’Description du commit‘’
```

Git status : La commande git status affiche la liste des fichiers modifies ainsi que les fichiers qui doivent encore être ajoutes ou valides.

```
git status
```

Git checkout : La commande git checkout peut être utilisée pour créer des branches ou pour basculer entre elle.

```
git checkout
```

Git remote : La commande remote permet à l’utilisateur de connecter le dépôt local a un serveur distant.

```
git remote add origin
```

Cette commande permet également à un utilisateur de se connecter à un dépôt distant

```
git remote -v
```

Git pull : Git pull utiliser pour fusionner toutes les modification présentes sur un dépôt distant dans le répertoire de travail local.

```
git pull
```

Partie2.

Qu'est-ce que GITHUB ?

GitHub est un site web et un service de cloud [service d'hébergement en ligne] qui aide les développeurs à stocker et à gérer leur code, ainsi qu'a suivre et contrôler les modifications qui lui sont apportées. A l'instar de GitHub ils existent plusieurs autres hébergeurs telle que : Bitbucket, Gitlab, Fogbugz etc.

Les Fonctionnalités GitHub :

GitHub est un lieu où les programmeurs et les concepteurs travaillent ensemble. Ils collaborent, contribuent et corrigent les bugs ensemble. Certaines de ses caractéristiques importantes sont les suivantes :

- Collaboration.

- Gestion de projet.
- Conversation.
- Suivi intègre des problèmes et de bugs.
- Représentation graphique des branches.
- Hébergement de codes.

GIT VS GITHUB

- Git es installer localement sur le système.	- GitHub est sur le cloud.
- Git peut être utiliser hors ligne et n'a pas besoin d'une connexion internet.	- GitHub ne peut pas être utiliser hors ligne et nécessite une connexion internet.
- Git est indépendant et peut-être utiliser sans GitHub.	- GitHub n'est pas indépendant et ne peut pas être utilise sans git.
- Les modifications telles que les commits, merge etc. sont effectuées à l'aide de l'invite de commande.	- Tout se fait via une interface web.

Conflits de Merge Git.

Un conflit survient lorsque vous essayer la merge [fusion] des branches qui ont des commmits [commentaire] concurrentes et git à besoin d'une intervention manuel pour décider quelles modifications intégrer dans la merge [fusion] final.

Les deux principaux points ou un merge [fusion]peut entrer dans un état conflictuel es : Aux démarrage du processus de merge [fusion], Pendant le processus de fusion.

- Aux démarrage du processus de fusion [merge], Sil y'a des changements dans le répertoire de travail de la zone de transit [zone de préparation] du projet en cours, la fusion ne démarrera pas. Dans ce cas, le conflit survient en raison des modifications en attente qui doivent être stabiliser à l'aide d'une autre commande git
- Pendant le processus de fusion [merge], l'echec lors du processus de fusion [merge] indique qu'il existe un conflit entre la branche locale et la branche en cours de fusion. Dans ce cas, git résout autant que possible mais il y a des choses qui doivent être résolues manuellement dans les fichiers en conflit.

Comment Résoudre un Merge Conflits

Le moyen le plus simple de résoudre un fichier en conflit consiste premièrement à ouvrir le fichier en question et y apporter les modifications manuellement requises.

Après avoir éditer le fichier, nous allons utiliser la commande : git add pour mettre en zone transitoire le nouveau contenu de merge[fusion]. La dernière étape consiste à créer un nouveau commit avec l'aide de la commande : git commit. A la suite de ça git va créer un nouveau merge commit pour finaliser la fusion [merge].

Commandes Git Pour résoudre un conflit

1- Git log ––merge : La commande git log ––merge est utilisée pour lister les commits qui cause le conflit.

2- Git diff : La commande git diff est utilisée pour trouver la difference

3- Git checkout : La commande git checkout est utilisée pour annuler les modifications apportées au fichier ou pour modifier les branches.

Git reset --mixed : La commande git reset --mixed est utilisée pour annuler les modifications apportées au répertoire de travail et à la zone de transit.

4- Git merge --abort : La commande git merge --abort aide à quitter le processus de fusion et à revenir

à l'état avant le début de la fusion.

5- Git reset : La commande git reset est utilisée au moment d'un conflit de fusion pour réinitialiser les fichiers en conflit à leur état d'origine.

FIN... Merci d’avoir suivi jusqu’à la fin.

Summary :

SHA1 est un algorithme de Hachage prenant un entrée jusqu’a 2^64 bits, et retourne une suite unique de 40 caractères Hexadécimaux : se procéder et utiliser par git pour connaitre les objets qu’il a sauvegarder.