# AUTOMATISATION

## Ansible (check)

## Terraform

## Cron

# PLATEFORMES DE COMMUNICATION

## Kubernetes

## GitLab

<https://openclassrooms.com/fr/courses/2035736-mettez-en-place-lintegration-et-la-livraison-continues-avec-la-demarche-devops>

## Git/ Git\_Hub

<https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github>

### Partie 1 - Faites vos premiers pas avec Git

#### 1. Tirez le maximum de ce cours !

#### 2. Découvrez la magie du contrôle de versions

Un gestionnaire de versions permet aux développeurs de conserver un historique des modifications et des versions de tous leurs fichiers.

Git est un gestionnaire de versions tandis que GitHub est un service en ligne qui héberge les dépôts Git. On parle alors de dépôt distant.

Cet outil a donc trois grandes fonctionnalités :

* Revenir à une version précédente de votre code en cas de problème.
* Suivre l’évolution de votre code étape par étape.
* Travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs.

#### 3. Saisissez l'utilité des dépôts distants sur GitHub

|  |  |
| --- | --- |
| Dépôt local | Dépôt distant |
| Un dépôt local est un entrepôt virtuel de votre projet. Il vous permet d'enregistrer les versions de votre code et d'y accéder au besoin. | Le dépôt distant est un peu différent. Il permet de stocker les différentes versions de votre code afin de garder un historique délocalisé, c'est-à-dire un historique hébergé sur Internet ou sur un réseau. Vous pouvez avoir plusieurs dépôts distants avec des droits différents (lecture seule, écriture, etc.). |

* **Alors, pourquoi créer une copie locale ?**

Tout simplement car votre dépôt local est un clone de votre dépôt distant. C’est sur votre dépôt local que vous ferez toutes vos modifications de code.

* **Quelle plateforme utiliser pour héberger votre code ?**

Il existe plusieurs outils intéressants (GitHub, GitLab, Bitbucket), et nous allons donc voir les principaux avantages et inconvénients de chacun.

* **GitLab**

GitLab est la principale alternative à GitHub depuis le rachat de GitHub par Microsoft ! Les anti-Microsoft ont même lancé l’hashtag #MovingToGitLab ! GitLab propose une version gratuite hebergée par ses soins ou sur vos propres serveurs. Il existe aussi des versions payantes avec plus d’options.

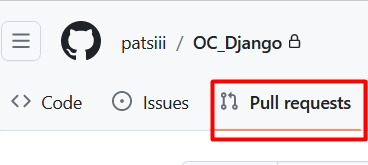
#### 4. Démarrez votre projet avec GitHub

<https://github.com/>

[paterneee@yahoo.fr](mailto:paterneee@yahoo.fr)

so\*\*\*

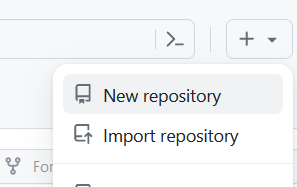
* L’onglet **Pull requests**, quant à lui, permet de faire des demandes de modifications réalisées sur le code.



Les pull requests (ou demandes de pull), vous permettent d'informer les autres utilisateurs des modifications que vous avez appliquées à une branche d'un repository sur GitHub, et que vous voulez fusionner avec le code principal.

* Via **Explore**, vous pourrez trouver de nouveaux projets open source sur lesquels travailler, en parcourant les projets recommandés, en vous connectant à la communauté GitHub et en recherchant des repositories par sujet ou par libellé.
* Pour mettre votre projet sur GitHub, vous devez créer un repository.

Créer son propre **dépôt**



#### 5. Installez Git sur votre ordinateur

<https://git-scm.com/downloads>

* Initialisez Git
* Configure credentials

$ git config --global user.name "patsiii" && git config --global user.email [paterneee@yahoo.fr](mailto:paterneee@yahoo.fr)

* git config --list
* Le dépôt local
* Créer un dépôt local

$ cd My\ Drive/STUDY/

$ git init

Initialized empty Git repository in G:/My Drive/STUDY/.git/

OU

* Cloner un dépôt distant

#### Quiz : Installer et configurer Git et GitHub

### Partie 2 - Utilisez les commandes de base de Git

#### 1. Travaillez depuis votre dépôt local Git

Depuis le 1er octobre 2020, tous les nouveaux dépôts créés sur GitHub sont appelés “**main**”, au lieu de “master” comme c’était le cas historiquement. Ce changement de terme a été opéré par la société GitHub afin d’adopter des références plus inclusives. Les dépôts existants, eux, ne changent pas de nom. Vous verrez donc les deux termes dans vos projets.

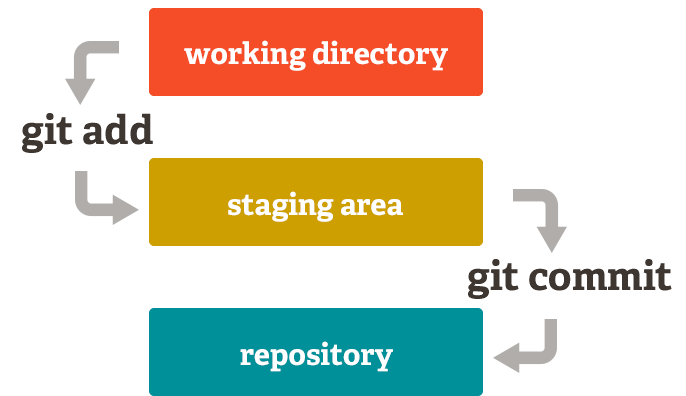
* Indexer les fichiers

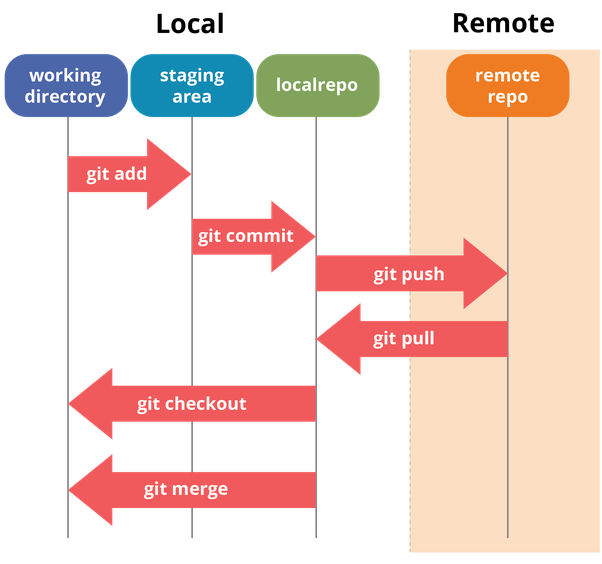
$ git add [fichier1.extensionX] [fichier2.extensionY]

* Archiver le projet en l’état

$ git commit -m "Ajout des fichier1 et fichier2"

Vos modifications sont maintenant enregistrées avec la description “Ajout des fichier1 et fichier2”.





* Relier « Local repository » ⬄ « Remote repository »

|  |  |
| --- | --- |
|  |  |
| Create en new “Remote repository” | Or use an existing one |
| And copy the link | |

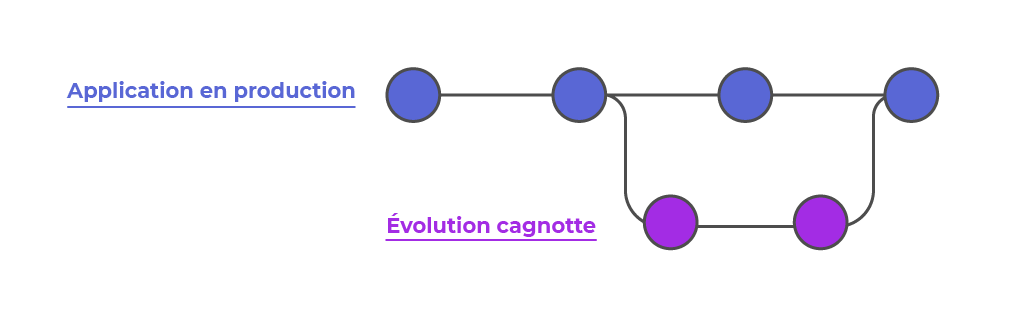
$ git remote add origin <https://github.com/patsiii/STUDY.git>

$ git branch -M main

* Envoyer des commits « Local repo » => « Remote repo »

$ git push -u origin main

#### 2. Appréhendez le système de branches



* Les opérations rattachées aux branches
* Lister les branches

$ git branch

* Créer une branche pour une modification

$ git branch modifiable\_branch

* Basculer vers une autre branche

$ git checkout modifiable\_branch

* Vous pouvez désormais réaliser votre évolution sans toucher à la branche main qui abrite votre code principal fonctionnel.
* Vous pouvez re-basculer si besoin à tout moment sur la branche main, sans impacter les modifications de la branche modifiable\_branch.
* Faire un commit des changements sur « modifiable\_branch »
* Réaliser un commit & push via la nouvelle branche

$ git add NOTE\_CISCO\_2023.docx

$ git commit -m "sur modifiable\_branch"

$ git branch -M modifiable\_branch

$ git push -u origin modifiable\_branch

* Fusionner le travail vers la branche principale

$ git checkout main

$ git merge modifiable\_branch

#### 3. Travaillez avec un dépôt distant

#### Quiz : Utiliser les commandes de base de Git

### Partie 3 - Pratiquez et corrigez les erreurs courantes

#### 1. Pratiquez et corrigez vos erreurs sur un dépôt local

#### 2. Corrigez vos erreurs sur votre dépôt distant

#### 3. Utilisez git reset

#### 4. Corrigez un commit raté

#### 5. Récapitulez ce que vous avez appris

#### Quiz : Corriger les erreurs courantes

#### Certificat de réussite (voir un exemple)

## Jenkins

Orchestration tools

<https://openclassrooms.com/fr/courses/1733551-gerez-votre-serveur-linux-et-ses-services>

# SCRIPTING

## Bash

# CLOUD COMPUTING

(Check)