

Chapitre 2: Sauvegarde collaborative et continue des projets **DESS** en Technologie de l'Information

Dr.-Ing. Austin Waffo Kouhoué, PhD

UNITECH-DESS TI- 2025 - austin.waffo@gmail.com

Introduction

Contexte

Aujourd'hui, la majorité des projets dépendent de fichiers numériques (code, scripts, données, documents). Or :

- Un clic peut tout effacer
- Un bug peut corrompre des fichiers essentiels
- Une mauvaise version peut détruire des jours de travail.
- Un collaborateur peut écraser le travail d'un autre.

Les risques informatiques courants

Risque 1 : Suppression accidentelle

- **Conséquences** : Perte de données
- **Exemple** : `rm -rf` mal exécuté

Les risques informatiques courants

Risque 1 : Suppression accidentelle

- **Conséquences** : Perte de données
- **Exemple** : `rm -rf` mal exécuté

Risque 2 : Mauvaise version

- **Conséquence** : Résultats faussés
- **Exemple** : Script mal versionné

Risque 3 : Conflit de collaboration

- **Conséquence** : Code brisé, chaos
- **Exemple** : Deux personnes modifient le même fichier

Les risques informatiques courant

Risque 3 : Conflit de collaboration

- **Conséquence** : Code brisé, chaos
- **Exemple** : Deux personnes modifient le même fichier

Risque 4 : Absence de sauvegarde

- **Conséquence** : Impossible de restaurer
- **Exemple** : Ordinateur volé ou crashé

Git et GitHub

- de garder une trace de chaque modification
- de revenir dans le temps à n'importe quelle version
- de travailler à plusieurs sans se marcher dessus
- de sauvegarder en ligne automatiquement
- de **réduire drastiquement les risques humains et techniques**

Plan

- ① Quelques commandes usuelles
- ② Installation
 - Installer GIT
 - Créer un compte GitHub
- ③ Sauvegarde

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers câchés)

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers câchés) **dir /a :h** (uniquement les fichiers câchés)

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers câchés) **dir /a :h** (uniquement les fichiers câchés)
- **ls -h** liste les fichiers suivi des tailles respectives

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers c ch s) **dir /a :h** (uniquement les fichiers c ch s)
- **ls -h** liste les fichiers suivi des tailles respectives
- **ls -l** liste les fichiers et dossiers du r pertoire courant avec les d tails (permissions, propri taire, taille, etc.)

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers c ch s) **dir /a :h** (uniquement les fichiers c ch s)
- **ls -h** liste les fichiers suivi des tailles respectives
- **ls -l** liste les fichiers et dossiers du r pertoire courant avec les d tails (permissions, propri taire, taille, etc.)
- **ls -R** Affiche r cursivement le contenu des sous-dossiers

Quelques commandes usuelles

Lister le contenu d'un repertoire (dir) et ls

- **dir /a** (tous le contenu y compris les fichiers câchés) **dir /a :h** (uniquement les fichiers câchés)
- **ls -h** liste les fichiers suivi des tailles respectives
- **ls -l** liste les fichiers et dossiers du répertoire courant avec les détails (permissions, propriétaire, taille, etc.)
- **ls -R** Affiche récursivement le contenu des sous-dossiers
- **ls -a** Affiche tous les fichiers, même ceux cachés (.)
- **ls -t** Trie par date de modification (plus récent en premier)
- **ls -r** Trie en ordre inverse
- **ls -S** Trie par taille décroissante
- **ls -d */** Affiche uniquement les dossiers du répertoire courant

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant
- **cd ..** Revenir au dossier parent

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant
- **cd ..** Revenir au dossier parent
- **cd ../..**

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant
- **cd ..** Revenir au dossier parent
- **cd ../..** Revenir deux niveaux plus haut

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant
- **cd ..** Revenir au dossier parent
- **cd ../..** Revenir deux niveaux plus haut **cd /** : Aller à la racine du système

Quelques commandes usuelles

Créer un dossier : **mkdir**

- **mkdir**

Naviguer dans un repertoire : **cd**

- **cd dossier** : Aller dans un dossier du répertoire courant
- **cd ..** Revenir au dossier parent
- **cd ../..** Revenir deux niveaux plus haut **cd /** : Aller à la racine du système

créer un fichier : **touch** et **echo**

- **touch nomFichier.extension** (sous linux)
- **echo > nomFichier.extension** (sous Windows)

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt
- **rm fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt
- **rm fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt
- **rm fichier1 fichier2** supprime plusieurs fichiers

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt
- **rm fichier1.txt** : Supprime un fichier qui a pour nom fichier1.txt
- **rm fichier1 fichier2** supprime plusieurs fichiers
- **rm -i fichier.txt**

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1 fichier2** supprime plusieurs fichiers
- **rm -i fichier.txt** Demande confirmation avant suppression

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1 fichier2** supprime plusieurs fichiers
- **rm -i fichier.txt** Demande confirmation avant suppression **rm -f fichier.txt** : Force la suppression sans confirmation
- **rm -r dossier/** supprime le dossier avec tout son contenu

Quelques commandes usuelles

Supprimer un repertoire : **rm** et **del**

- **del fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1.txt** : Supprime un fichier qui a pour nom **fichier1.txt**
- **rm fichier1 fichier2** supprime plusieurs fichiers
- **rm -i fichier.txt** Demande confirmation avant suppression **rm -f fichier.txt** : Force la suppression sans confirmation
- **rm -r dossier/** supprime le dossier avec tout son contenu

créer un fichier : **touch** et **echo**

- **touch nomFichier.extension** (sous linux)
- **echo > nomFichier.extension** (sous Windows)

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**
- Exemple pratique :
 - ❶ créer deux répertoires nommés `seance5_1` et `seance5_2` dans le répertoire `seance5`
 - ❷ Créer deux fichiers nommés `info1.txt` et `info.txt` dans le répertoire `seance5`
 - ❸ A travers la commande **cp**, copier ces deux fichiers du répertoire `seance5` vers le répertoire `seance5_1`
 - ❹ A travers la commande **cp** toujours, copier ces deux fichiers du répertoire `seance5_1` vers répertoire `seance5_2` sans se positionner dans le répertoire `seance5_1`
- **-a** copie récursive en préservant les liens symboliques, les permissions, les timestamps, etc

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**
- Exemple pratique :
 - ❶ créer deux répertoires nommés `seance5_1` et `seance5_2` dans le répertoire `seance5`
 - ❷ Créer deux fichiers nommés `info1.txt` et `info.txt` dans le répertoire `seance5`
 - ❸ A travers la commande **cp**, copier ces deux fichiers du répertoire `seance5` vers le répertoire `seance5_1`
 - ❹ A travers la commande **cp** toujours, copier ces deux fichiers du répertoire `seance5_1` vers répertoire `seance5_2` sans se positionner dans le répertoire `seance5_1`
- **-a** copie récursive en préservant les liens symboliques, les permissions, les timestamps, etc
- **-R ou -r**

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**
- Exemple pratique :
 - ❶ créer deux répertoires nommés `seance5_1` et `seance5_2` dans le répertoire `seance5`
 - ❷ Créer deux fichiers nommés `info1.txt` et `info.txt` dans le répertoire `seance5`
 - ❸ A travers la commande **cp**, copier ces deux fichiers du répertoire `seance5` vers le répertoire `seance5_1`
 - ❹ A travers la commande **cp** toujours, copier ces deux fichiers du répertoire `seance5_1` vers répertoire `seance5_2` sans se positionner dans le répertoire `seance5_1`
- **-a** copie récursive en préservant les liens symboliques, les permissions, les timestamps, etc
- **-R ou -r** copie les répertoires de manière récursive

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**
- Exemple pratique :
 - ❶ créer deux répertoires nommés `seance5_1` et `seance5_2` dans le répertoire `seance5`
 - ❷ Créer deux fichiers nommés `info1.txt` et `info.txt` dans le répertoire `seance5`
 - ❸ A travers la commande **cp**, copier ces deux fichiers du répertoire `seance5` vers le répertoire `seance5_1`
 - ❹ A travers la commande **cp** toujours, copier ces deux fichiers du répertoire `seance5_1` vers répertoire `seance5_2` sans se positionner dans le répertoire `seance5_1`
- **-a** copie récursive en préservant les liens symboliques, les permissions, les timestamps, etc
- **-R ou -r** copie les répertoires de manière récursive **-u** :

Quelques commandes usuelles

Copie des fichiers et des répertoires : **cp**

- **cp source destination :**
- Exemple pratique :
 - ❶ créer deux répertoires nommés `seance5_1` et `seance5_2` dans le répertoire `seance5`
 - ❷ Créer deux fichiers nommés `info1.txt` et `info.txt` dans le répertoire `seance5`
 - ❸ A travers la commande **cp**, copier ces deux fichiers du répertoire `seance5` vers le répertoire `seance5_1`
 - ❹ A travers la commande **cp** toujours, copier ces deux fichiers du répertoire `seance5_1` vers répertoire `seance5_2` sans se positionner dans le répertoire `seance5_1`
- **-a** copie récursive en préservant les liens symboliques, les permissions, les timestamps, etc
- **-R** ou **-r** copie les répertoires de manière récursive **-u** :

Quelques commandes usuelles

Déplacer ou renommer des fichiers/dossiers : **mv** et **rename**

- **rename source destination**

Quelques commandes usuelles

Déplacer ou renommer des fichiers/dossiers : **mv** et **rename**

- **rename source destination**
- **mv source destination**

Quelques commandes usuelles

Déplacer ou renommer des fichiers/dossiers : **mv** et **rename**

- **rename source destination**
- **mv source destination**
- **mv fichier.txt dossier/**

Quelques commandes usuelles

Déplacer ou renommer des fichiers/dossiers : **mv** et **rename**

- **rename source destination**
- **mv source destination**
- **mv fichier.txt dossier/**
- **mv ancien_nom.txt nouveau_nom.txt**

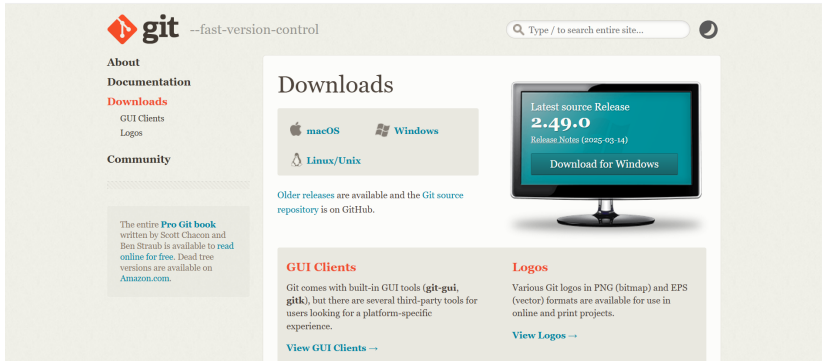
Quelques commandes usuelles

Déplacer ou renommer des fichiers/dossiers : **mv** et **rename**

- `rename source destination`
- `mv source destination`
- `mv fichier.txt dossier/`
- `mv ancien_nom.txt nouveau_nom.txt`
`mv fichier1.txt fichier2.txt dossier/`

Installation de Git

Télécharger git à ce lien <https://git-scm.com/downloads>



The screenshot shows the Git website's Downloads page. The header features the Git logo with the tagline "--fast-version-control", a search bar, and a moon icon. The left sidebar contains navigation links: About, Documentation, Downloads (highlighted), GUI Clients, Logos, and Community. The main content area is titled "Downloads" and includes a section for "Older releases" with links to macOS, Windows, and Linux/Unix. A monitor graphic displays the "Latest source Release 2.49.0" with a "Download for Windows" button. Below this, there are sections for "GUI Clients" and "Logos".

git --fast-version-control

Search: Type / to search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

Downloads

Older releases are available and the Git source repository is on GitHub.

GUI Clients
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.
[View GUI Clients →](#)

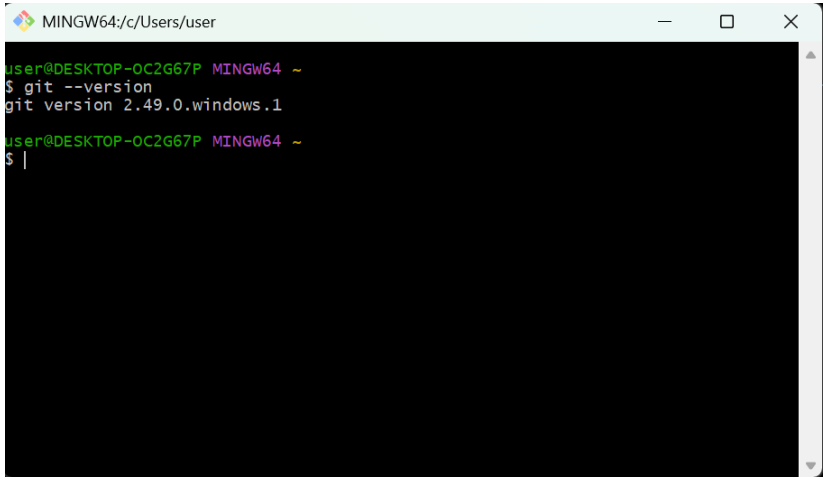
Logos
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.
[View Logos →](#)

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Latest source Release
2.49.0
Release Notes (2025-03-14)
[Download for Windows](#)

Tester l'installation de Git

Lancer **git bash** et exécuter la commande **git --version**



```
MINGW64:/c/Users/user  
user@DESKTOP-OC2G67P MINGW64 ~  
$ git --version  
git version 2.49.0.windows.1  
user@DESKTOP-OC2G67P MINGW64 ~  
$ |
```

Création de compte GitHub

Compte GitHub personnel

- Création de compte GitHub <https://github.com/>

Sign up to GitHub

Email*

Password*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

⚠ Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your country*

For compliance reasons, we're required to collect country information to send

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire Projet1

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir** **Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire Projet1 **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire Projet1 **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls**

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**
- créée par un fichier nommé **rapport1.txt** dans ce répertoire

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**
- créée par un fichier nommé **rapport1.txt** dans ce répertoire **touch rapport1.txt**

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**
- crée par un fichier nommé **rapport1.txt** dans ce répertoire **touch rapport1.txt** ouvrir le fichier **rapport1.txt** et ajouter le texte **Bonne après-midi à tous nano rapport1.txt**
- intiatiliser le répertoire pour le rendre versionnable **git init**

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**
- crée par un fichier nommé **rapport1.txt** dans ce répertoire **touch rapport1.txt** ouvrir le fichier **rapport1.txt** et ajouter le texte **Bonne après-midi à tous nano rapport1.txt**
- intiatiliser le répertoire pour le rendre versionnable **git init**
- réafficher le contenu du répertoire

Sauvegarde

Création et initialisation du repertoire

- Lancer votre **git bash**, saisir la commande permettant de créer un répertoire **Projet1** **mkdir Projet1**
- rassurer que le repertoire a été bien créée **dir** ou **ls**
- se positionner dans ce répertoire **cd projet1**
- afficher le contenu **ls** afficher le contenu complet (avec les fichiers cachés) **ls -a**
- crée par un fichier nommé **rapport1.txt** dans ce répertoire **touch rapport1.txt** ouvrir le fichier **rapport1.txt** et ajouter le texte **Bonne après-midi à tous nano rapport1.txt**
- intiatiliser le répertoire pour le rendre versionnable **git init**
- réafficher le contenu du répertoire
- exécuter la commande nommée **git status**

Indexation et historisation

- cela signifie que notre fichier n'est pas encore présent dans la zone d'indexation

Indexation et historisation

- cela signifie que notre fichier n'est pas encore présent dans la zone d'indexation
- avant de continuer crée encore un fichier nommé **rapport2.txt** et ajouter le texte **Ce cours consiste à effectuer de la sauvegarde sécurisée de nos données**

Indexation et historisation

- cela signifie que notre fichier n'est pas encore présent dans la zone d'indexation
- avant de continuer créez encore un fichier nommé **rapport2.txt** et ajoutez le texte **Ce cours consiste à effectuer de la sauvegarde sécurisée de nos données**
- exécuter **git status** de nouveau

Indexation et historisation

- cela signifie que notre fichier n'est pas encore présent dans la zone d'indexation
- avant de continuer crée encore un fichier nommé **rapport2.txt** et ajouter le texte **Ce cours consiste à effectuer de la sauvegarde sécurisée de nos données**
- exécuter **git status** de nouveau
- pour ajouter dans la zone d'indexation, exécuter la commande **git add nom__du__fichier.extension**

Indexation et historisation

- cela signifie que notre fichier n'est pas encore présent dans la zone d'indexation
- avant de continuer crée encore un fichier nommé **rapport2.txt** et ajouter le texte **Ce cours consiste à effectuer de la sauvegarde sécurisée de nos données**
- exécuter **git status** de nouveau
- pour ajouter dans la zone d'indexation, exécuter la commande **git add nom_du_fichier.extension**
- vérifier avec quelle commande **git status**

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande `git commit -m "message" -a`.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande `git commit -m "message" -a`.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE
- renommer le fichier **rapport1.txt** en **rapport11.txt** et mettre le texte suivant à l'intérieur : **Bonne soirée à tous également**

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande **git commit -m "message" -a**.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE
- renommer le fichier **rapport1.txt** en **rapport11.txt** et mettre le texte suivant à l'intérieur : **Bonne soirée à tous également** renommer le fichier **rapport2.txt** en **rapport21.txt** et mettre le texte suivant **le cours a une durée de 3 heures** à l'intérieur

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande **git commit -m "message" -a**.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE
- renommer le fichier **rapport1.txt** en **rapport11.txt** et mettre le texte suivant à l'intérieur : **Bonne soirée à tous également** renommer le fichier **rapport2.txt** en **rapport21.txt** et mettre le texte suivant **le cours a une durée de 3 heures** à l'intérieur
- exécuter **git status**

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande **git commit -m "message" -a**.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE
- renommer le fichier **rapport1.txt** en **rapport11.txt** et mettre le texte suivant à l'intérieur : **Bonne soirée à tous également** renommer le fichier **rapport2.txt** en **rapport21.txt** et mettre le texte suivant **le cours a une durée de 3 heures** à l'intérieur
- exécuter **git status**
- exécuter **git commit -m "ajouter des fichiers de la deuxième version -a**

Indexation et historisation

- pour mettre les fichiers dans la zone d'historisation, il faut exécuter la commande **git commit -m "message" -a**.
A CHAQUE FOIS QUE VOUS FAITES UN COMMIT METTEZ TOUJOURS LE MESSAGE
- renommer le fichier **rapport1.txt** en **rapport11.txt** et mettre le texte suivant à l'intérieur : **Bonne soirée à tous également** renommer le fichier **rapport2.txt** en **rapport21.txt** et mettre le texte suivant **le cours a une durée de 3 heures** à l'intérieur
- exécuter **git status**
- exécuter **git commit -m "ajouter des fichiers de la deuxième version -a**

Sauvegarde

Configurer son compte

Configurer son compte, en cas de nécessité :

- `git config --global user.name "Ton Nom"`
- `git config --global user.email "ton.email@example.com"`

Récupérer une version spécifique

Nous souhaitons revenir à une étape spécifique

- 1 Afficher l'historique des modifications **git log**

Sauvegarde

Configurer son compte

Configurer son compte, en cas de nécessité :

- `git config --global user.name "Ton Nom"`
- `git config --global user.email "ton.email@example.com"`

Récupérer une version spécifique

Nous souhaitons revenir à une étape spécifique

- 1 Afficher l'historique des modifications **git log**
- 2 exécuter un **git checkout identifiant_historique**

Sauvegarde à distance : GitHUB

Pour commencer :

- 1 créer un compte sur github

Sauvegarde à distance : GitHub

Pour commencer :

- 1 créer un compte sur github
- 2 créer le répertoire de votre projet

Sauvegarde à distance : GitHub

Pour commencer :

- 1 créer un compte sur github
- 2 créer le répertoire de votre projet mettez **Test1** comme nom de répertoire

Sauvegarde à distance : GitHUB

Pour commencer :

- 1 créer un compte sur github
- 2 créer le répertoire de votre projet mettez **Test1** comme nom de répertoire
- 3 cloner le répertoire principal du projet

Sauvegarde à distance : GitHub

Pour commencer :

- 1 créer un compte sur github
- 2 créer le répertoire de votre projet mettez **Test1** comme nom de répertoire
- 3 cloner le répertoire principal du projet **git clone**
`https://github.com/utilisateur/
nom-du-repertoire.git`

Faites vos opérations

- création de fichiers
- modification de fichiers
- git add
- git status
- git commit

Sauvegarde à distance : GitHub

Pousser à GitHub

- `git push origin main` ou `git push origin master`

Sauvegarde à distance : GitHub

Pousser à GitHub

- **git push origin main** ou **git push origin master**
- Aller actualiser la fenêtre GitHub pour se rassurer de la mise à jour

Authentication

Créer un fine-grained token

- 1 cliquer sur **profile**

Créer un fine-grained token

- 1 cliquer sur **profile**
- 2 **setting**

Créer un fine-grained token

- 1 cliquer sur **profile**
- 2 **setting**
- 3 **Developer setting**

Créer un fine-grained token

- 1 cliquer sur **profile**
- 2 **setting**
- 3 **Developer setting**
- 4 **Personal access tokens**

Créer un fine-grained token

- 1 cliquer sur **profile**
- 2 **setting**
- 3 **Developer setting**
- 4 **Personal access tokens**
- 5 **Fine-grained tokens**

Créer un fine-grained token

- 1 cliquer sur **profile**
- 2 **setting**
- 3 **Developer setting**
- 4 **Personal access tokens**
- 5 **Fine-grained tokens**
- 6 **Generate new token**

Authentication

New fine-grained personal access token


Create a fine-grained, repository-scoped token suitable for personal API use and for using Git over HTTPS.

Token name *

A unique name for this token. May be visible to resource owners or users with possession of the token.

Description

Resource owner

 Austin-FE ▾

The token will only be able to make changes to resources owned by the selected resource owner. Tokens can always read all public repositories

Authentication

Repository access

☒ **Public repositories**

Read-only access to public repositories.

☐ **All repositories**

This applies to all current and future repositories you own. Also includes public repositories (read-only).

☐ **Only select repositories**

Select at least one repository. Max 50 repositories. Also includes public repositories (read-only).

Permissions

Choose the minimal permissions necessary for your needs. [Learn more about permissions.](#)

Account permissions

User permissions permit access to resources under your personal GitHub account.


Generate token

Cancel





Authentication

Copier le token généré, et mettre dans un fichier texte

 **token1** Delete

Never used • Expires on Mon, Apr 28 2025

 **Make sure to copy your personal access token now as you will not be able to see this again.**

`github_pat_11BRWATFA02c7AVgUhmaSY_19D1sz4bAh7L3EB0j5ndWTfCnoKMAog8dRNOGgsaiz1EA573S73s1Mij` 

Afficher les branches existantes

- Afficher les branches locales :

Afficher les branches existantes

- Afficher les branches locales : **git branch**

Afficher les branches existantes

- Afficher les branches locales : **git branch**
- Afficher les branches distantes :

Afficher les branches existantes

- Afficher les branches locales : **git branch**
- Afficher les branches distantes : **git branch -r**

Afficher les branches existantes

- Afficher les branches locales : **git branch**
- Afficher les branches distantes : **git branch -r**
- Afficher les branches locales et distantes :

Afficher les branches existantes

- Afficher les branches locales : **git branch**
- Afficher les branches distantes : **git branch -r**
- Afficher les branches locales et distantes : **git branch -a**

Afficher les branches existantes

- Afficher les branches locales : **git branch**
- Afficher les branches distantes : **git branch -r**
- Afficher les branches locales et distantes : **git branch -a**