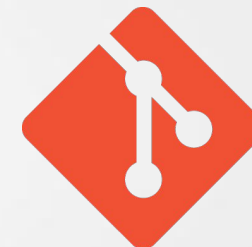




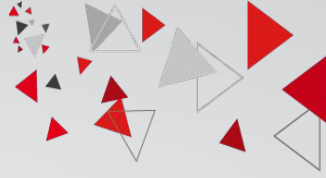
DevOps

Chapitre 4 : Git

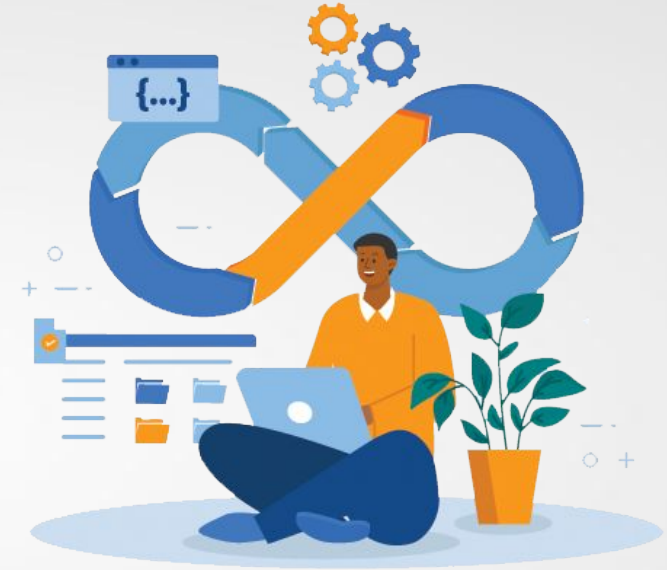


ESPRIT – UP ASI (Architecture des Systèmes d'Information)
Bureau E204

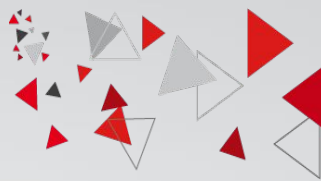
► Plan du cours



- Introduction
- Outils de Control de Version (CVS, SVN, **GIT**)
- Repositories Git (Local, Distant)
- Commandes GIT avec Git Bash
- Commandes Git avec un IDE
- **Travail à faire :**
 - Travail en équipe (Ajout / Récupération d'un projet depuis GitHub, gestion des branches, résolution des conflits, etc.)



► Problématiques



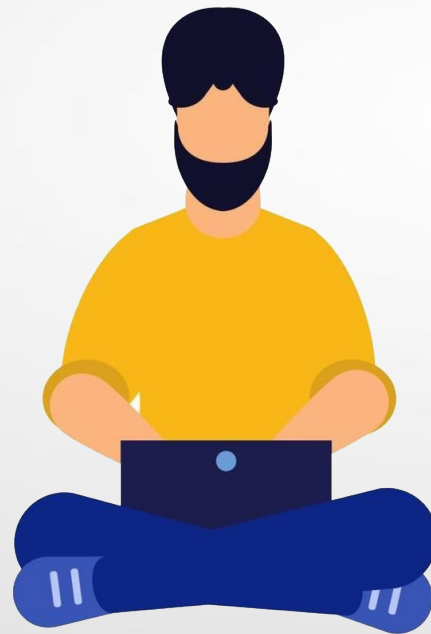
Collaboration ?

Historique des modifications ?

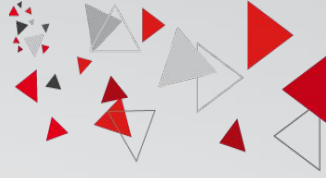
Travail à distance ?

Suivi des versions ?

Sauvegarde et sécurité ?



Introduction – SCM



SCM : Source Code Management

- **Historique complet** : Enregistre toutes les modifications du code.
- **Collaboration facilitée** : Permet à plusieurs développeurs de travailler ensemble sans conflits.
- **Sécurité des données** : Protège contre la perte de données en sauvegardant les versions.
- **Traçabilité des changements** : Indique qui a fait quoi, quand et pourquoi ?
- **Amélioration de la qualité** : Facilite les tests et l'amélioration continue du code.



► Outils de Contrôle de Version (CVS, SVN, GIT)



CVS (Concurrent Versions System)

- Ancien système de gestion de version.
- Gestion centralisée des fichiers, ce qui signifie que toutes les modifications sont stockées sur un serveur central.



SVN (Subversion)

- Système de gestion de version centralisé.
- Permet de suivre les modifications du code et gère les conflits de manière plus avancée que CVS.



Git

- Système de gestion de version **décentralisé**.
- Enregistre les modifications du code en local, favorisant la collaboration grâce à des *branches flexibles*.

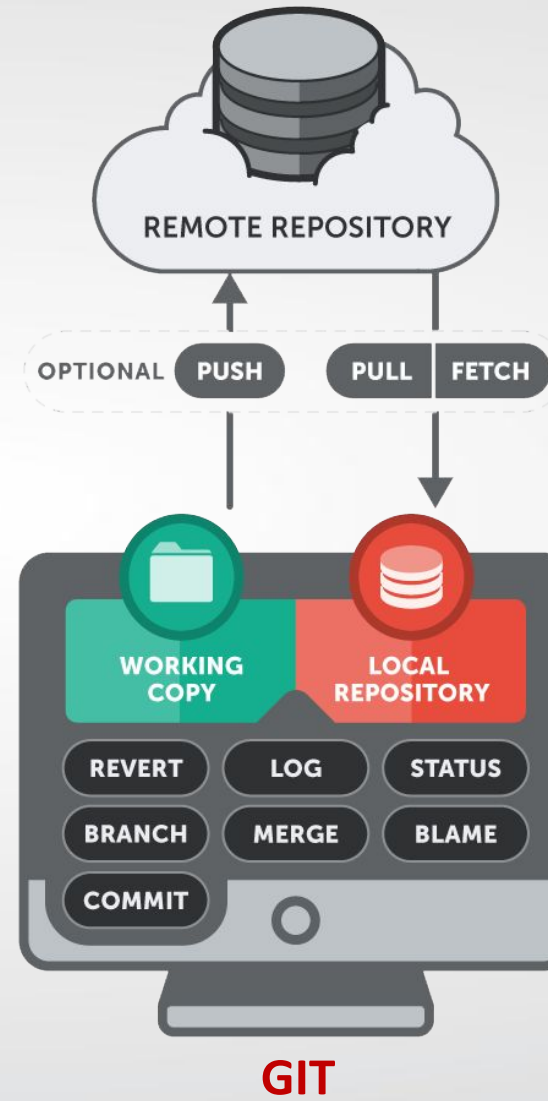
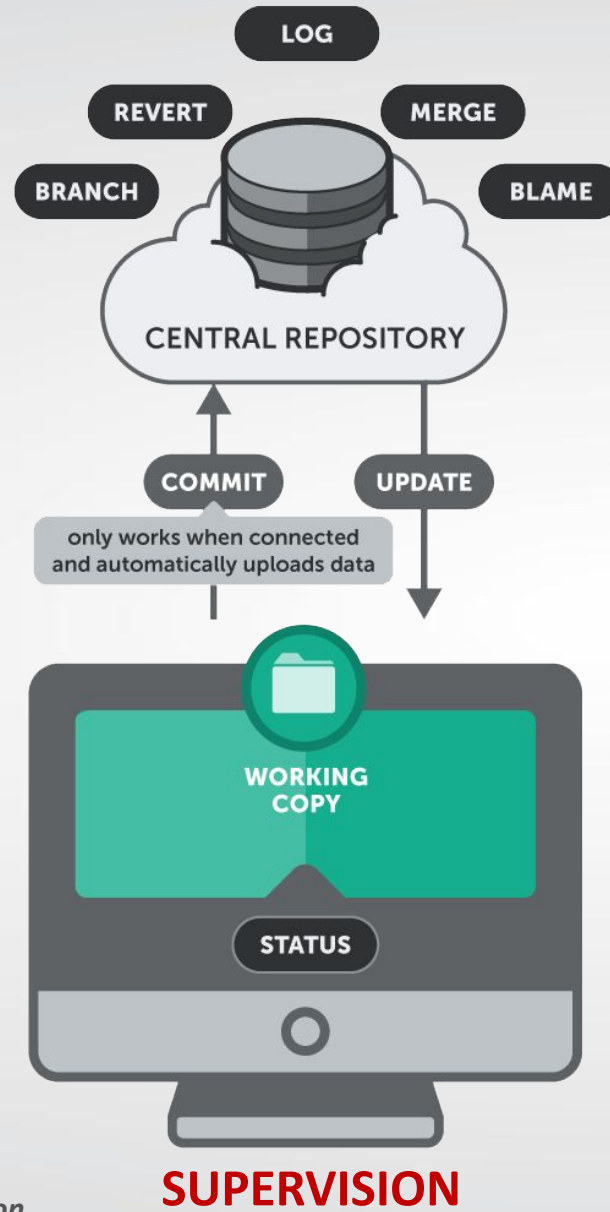
► Git - Définition



- Git est un système de gestion de version distribuée, [gratuit et open source](#), conçu pour gérer des projets de toutes tailles, de petits à très grands, avec rapidité et efficacité.



SVN vs Git



Git - Avantages



- **Gestion du code** : Git permet de suivre et de gérer les changements dans le code source.
- **Collaboration aisée** : Il facilite le travail en équipe en permettant à plusieurs développeurs de contribuer simultanément.
- **Historique complet** : Git enregistre toutes les versions du code, ce qui aide à comprendre l'évolution du projet.
- **Décentralisé** : Chaque développeur a une copie complète du code, ce qui permet de travailler hors ligne et de maintenir la flexibilité.



Git - Avantages



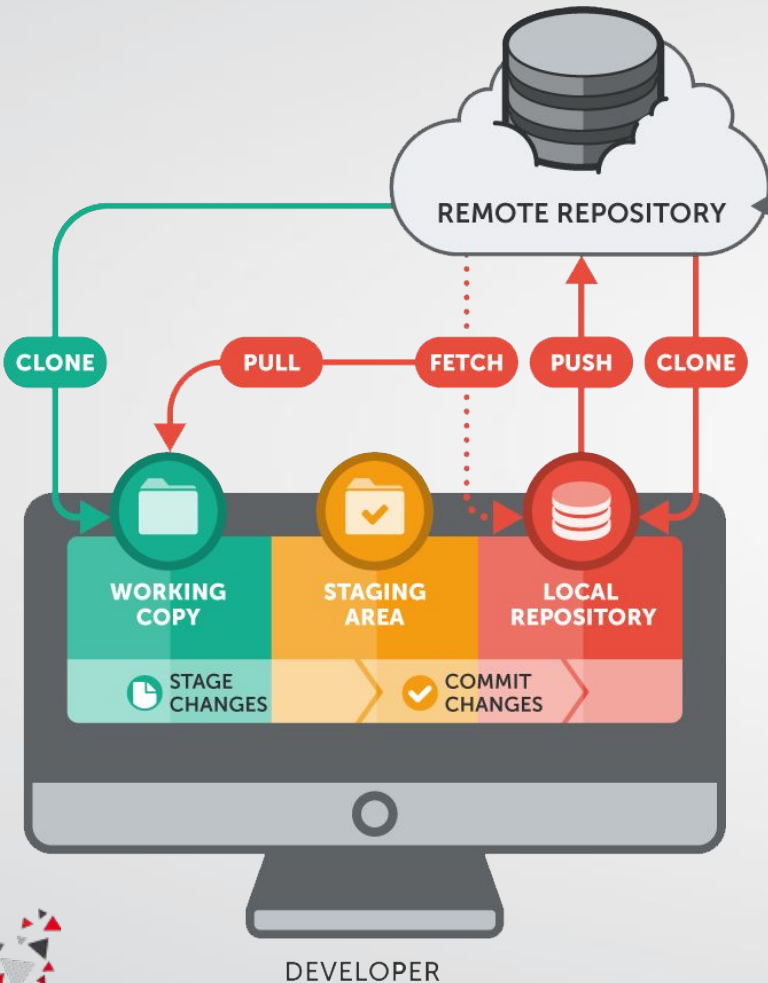
- **Sécurité des données** : Les données sont sauvegardées localement et sur des serveurs, ce qui réduit les risques de perte.
- **Grande communauté** : Git est largement utilisé et soutenu, avec une abondance de ressources et de plugins.
- **Intégration d'outils** : Il s'intègre facilement à d'autres outils de développement et de gestion de projet.
- **Performance élevée** : Git est rapide, ce qui permet de travailler efficacement même sur de grands projets.



▶ Git - Dépôts



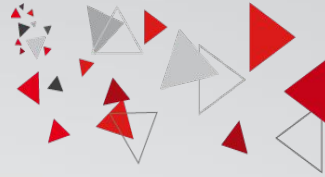
Git propose principalement **deux** types de **dépôts** (*repositories*) :



- **Dépôt Distant (Remote Repository)** : Une copie du code *sur un serveur distant* (comme GitHub, GitLab et Bitbucket), où plusieurs personnes peuvent collaborer et partager leurs modifications.
- **Dépôt Local (Local Repository)** : Une copie du code *sur votre propre ordinateur*, où vous travaillez et effectuez des modifications.



▶ Git - Dépôt Local

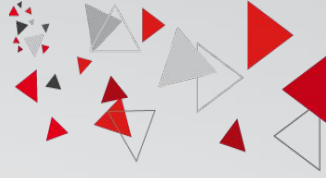


.git

C:\Work\workspace-sts\spring-jsf\timesheet\.git		
Nom		Mo
hooks		09/
info		09/
logs		09/
objects		09/
refs		09/
COMMIT_EDITMSG		09/
config		09/
description		09/
HEAD		09/
index		09/
ORIG_HEAD		09/



▶ Git - Dépôt Distant



Bit Bucket



GitHub



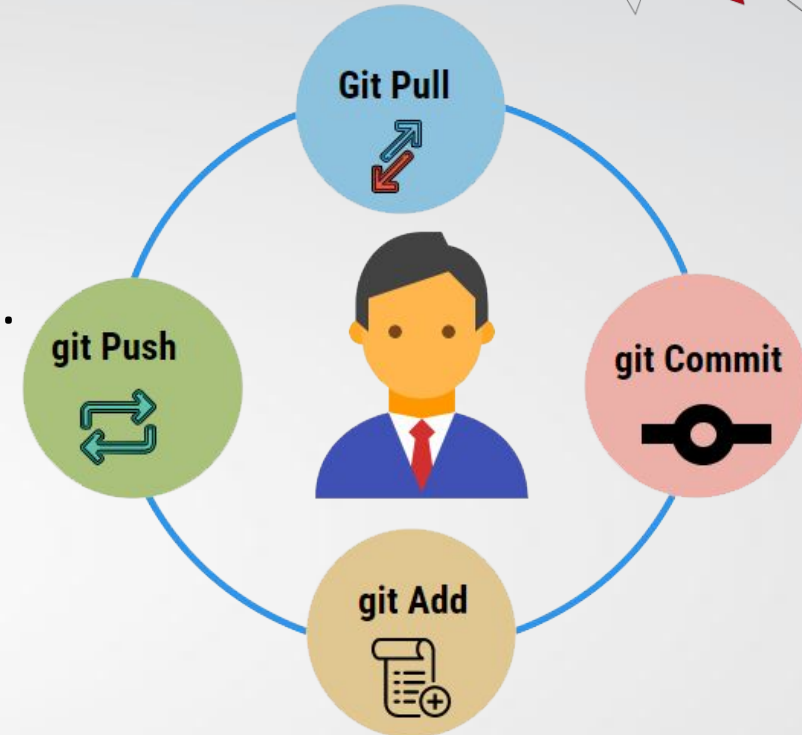
GitLab

- Ces services offrent des espaces de stockage sécurisés où les développeurs peuvent héberger leurs dépôts Git et collaborer sur des projets de manière transparente.

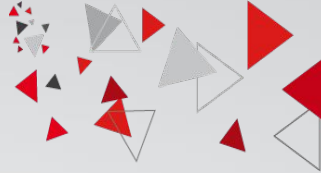


► Git - Terminologie

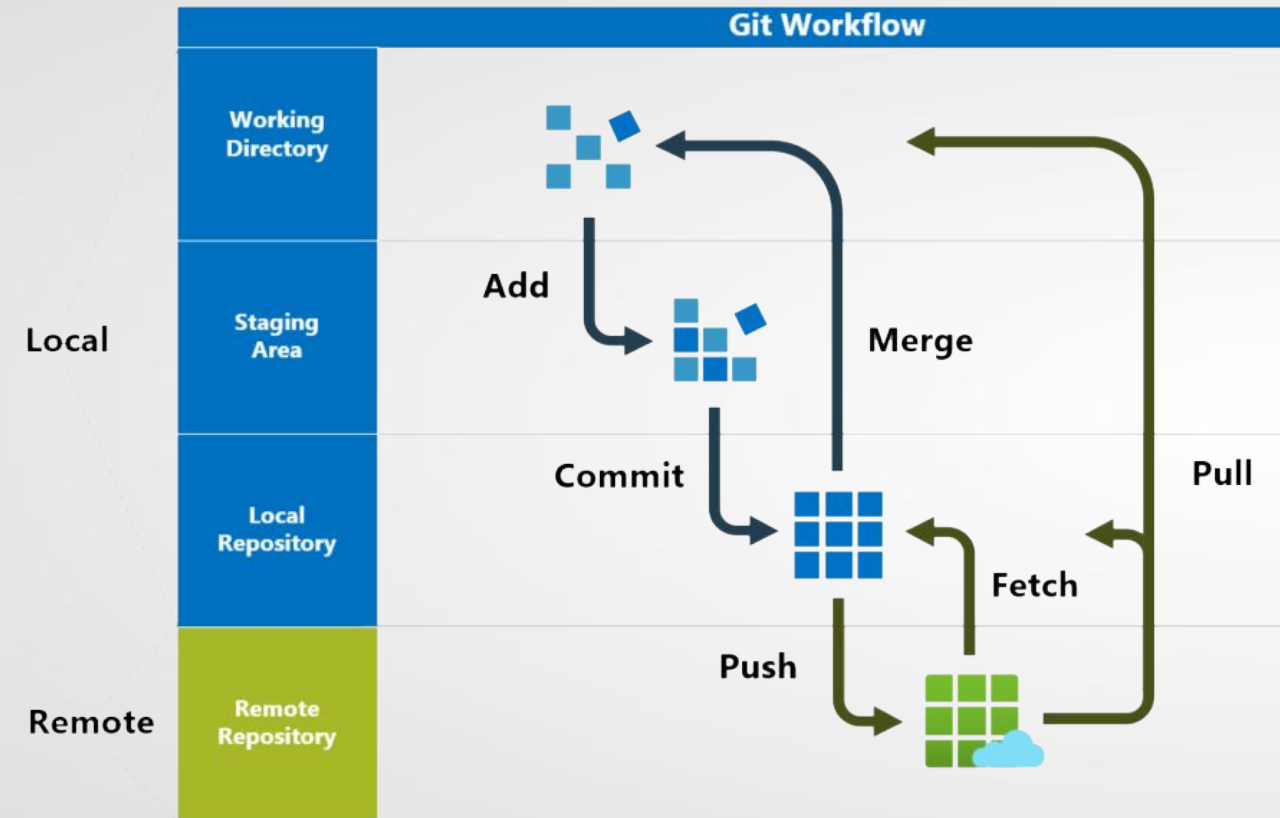
- **Commit** : Enregistre les changements faits dans le code.
- **Add** : Prépare des fichiers modifiés pour le prochain commit.
- **Push** : Envoie les changements vers un serveur distant.
- **Pull** : Récupère les changements depuis le serveur distant.
- **SHA (Secure Hash Algorithm)** : Un identifiant unique pour chaque commit.
- **Branche** : Une voie distincte pour travailler sans perturber le code principal.



Git - Zones



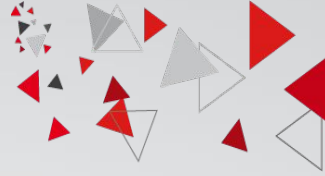
Git gère les versions de vos travaux locaux en utilisant trois zones principales :



- **Zone de travail (Working Directory) :**
L'endroit où vous travaillez sur vos fichiers.
- **Zone de préparation (Staging Area) :**
Un endroit pour sélectionner les modifications à enregistrer.
- **Dépôt local (Committed files) :**
L'endroit où Git stocke toutes les versions de votre projet.



▶ Git - Installation



- Téléchargez la dernière version depuis le site officiel de Git (<https://git-scm.com/download>) et suivez les instructions d'installation.

The screenshot shows the Git website's 'Downloads' page. At the top left is the Git logo and the tagline '--distributed-is-the-new-centralized'. A search bar is on the top right. The left sidebar contains links for 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. Below these is a box about the 'Pro Git book'. The main content area has a 'Downloads' heading, followed by download links for macOS, Windows, and Linux/Unix. A large monitor graphic displays the 'Latest source Release 2.42.0' and a 'Download for Windows' button. Below this, it mentions older releases are on GitHub. At the bottom, there are sections for 'GUI Clients' and 'Logos' with links to view them.

git --distributed-is-the-new-centralized

Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

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](#).

Downloads

macOS Windows Linux/Unix

Latest source Release
2.42.0
[Release Notes \(2023-08-21\)](#)
Download for Windows

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

GUI Clients

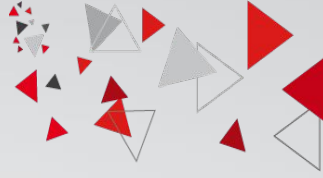
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.
[View GUI Clients →](#)

Logos

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



Git - Initialisation



- Pour vérifier que Git est correctement installé, ouvrez **Git Bash** et exécutez la commande :

```
$ git --version
```

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~  
$ git --version  
git version 2.42.0.windows.2
```



- Définissez votre nom d'utilisateur et votre adresse e-mail avec la commande `git config` pour éviter d'être invité à le faire à chaque action :

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~  
$ git config --global user.name "Ons BEN SALAH"  
  
onsbe@LAPTOP-R32K4R59 MINGW64 ~  
$ git config --global user.email "ons.bensalah@esprit.tn"
```

- `$ git config --list` : affiche toutes les configurations Git de votre système

▶ Git - Initialisation

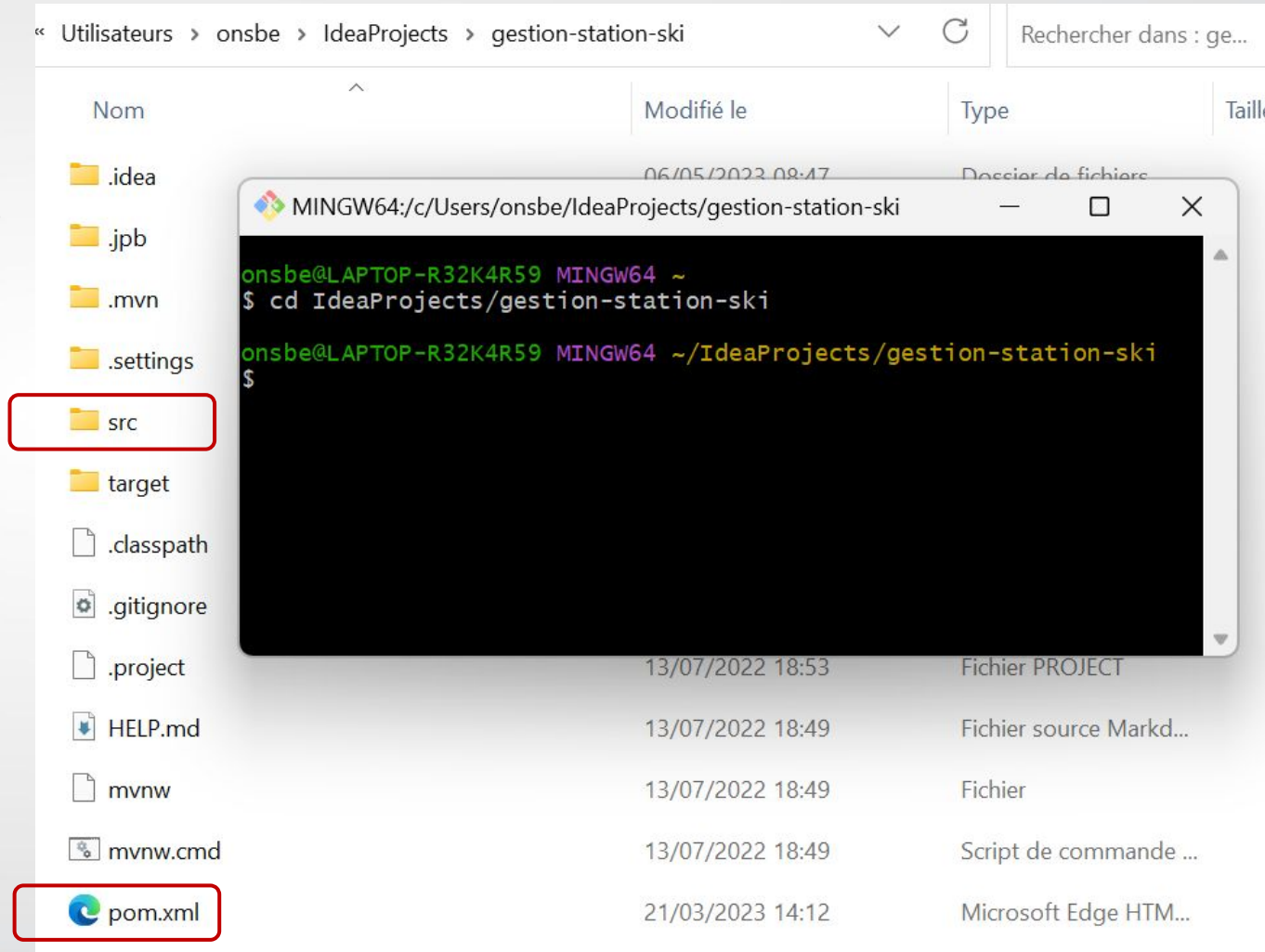


- Si vous souhaitez ajouter un projet déjà prêt à votre dépôt local sur votre compte Git, suivez ces étapes :

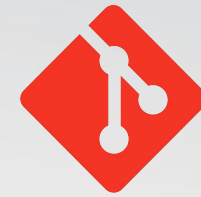
- Accédez au dossier du projet.
- Ouvrez l'invite de commande Git Bash.



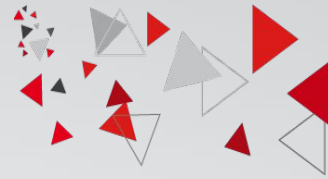
Seuls le dossier **src** et le fichier **pom.xml** seront gérés par Git



► Commandes Git - Initialisation



git



– Initialiser un dépôt Git :

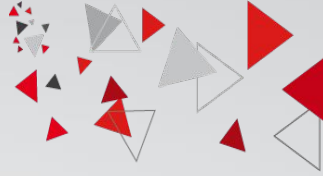
- Accédez au répertoire de votre projet en utilisant la commande `cd` pour naviguer jusqu'au bon dossier.
- Exécutez la commande `git init` pour initialiser un nouveau dépôt Git dans ce répertoire :

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski
$ git init
Initialized empty Git repository in C:/Users/onsbe/IdeaProjects/gestion-station-ski/.git/

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$
```



► Commandes Git - git status



– Vérifier l'état de dépôt Git :

- Exécutez la commande `git status` pour vérifier l'état actuel de votre dépôt Git.

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git status
On branch master

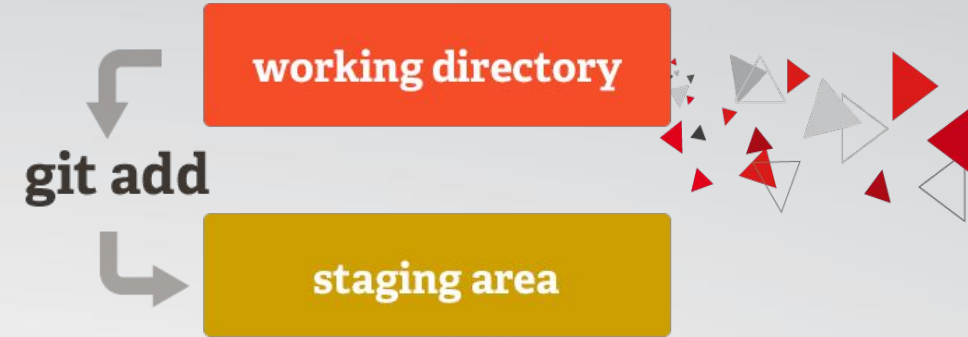
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .jpb/
        .mvn/
        mvnw
        mvnw.cmd
        pom.xml
        src/

nothing added to commit but untracked files present (use "git add" to track)
```



► Commandes Git - git add



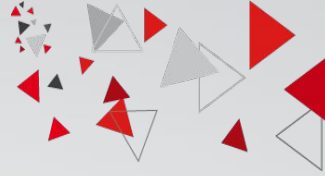
– Ajouter un ou des plusieurs fichiers :

- On va déplacer les fichiers de la zone de travail à la zone de préparation (staging Area).
- Pour ajouter des fichiers de manière individuelle ou par groupes, vous pouvez utiliser la commande : `$ git add <file1_name> <file2_name>`

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git add src/ pom.xml
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/tn/esprit/spring/GestionStationSkiApplication.java', LF
will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/resources/application.properties', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'src/test/java/tn/esprit/spring/GestionStationSkiApplicationTests.java'
, LF will be replaced by CRLF the next time Git touches it
```



► Commandes Git



– Ne pas inclure les fichiers :

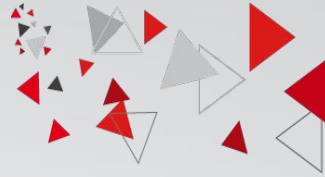
- .classpath
- .mvn/
- .project
- .settings/
- .springBeans
- HELP.md
- mvnw
- mvnw.cmd



→ Ces fichiers sont locaux et créés automatiquement, le développeur ne les changera jamais.



► Commandes Git - git status



- Vérifier l'état de dépôt Git après l'ajout de dossier src et de fichier pom.xml

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git status
On branch master

No commits yet

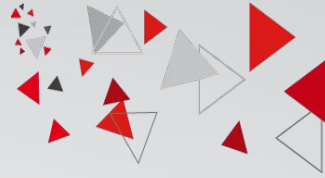
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   pom.xml
    new file:   src/main/java/tn/esprit/spring/GestionStationSkiApplication.java
    new file:   src/main/java/tn/esprit/spring/configs/OpenAPIConfig.java
    new file:   src/main/java/tn/esprit/spring/controllers/CourseRestController.java
    new file:   src/main/java/tn/esprit/spring/controllers/InstructorRestController.java
    new file:   src/main/java/tn/esprit/spring/controllers/PisteRestController.java
    new file:   src/main/java/tn/esprit/spring/controllers/RegistrationRestController.java
    new file:   src/main/java/tn/esprit/spring/controllers/SkierRestController.java
    new file:   src/main/java/tn/esprit/spring/controllers/SubscriptionRestController.java
    new file:   src/main/java/tn/esprit/spring/entities/Color.java
    new file:   src/main/java/tn/esprit/spring/entities/Course.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    .jpb/
    .mvn/
    mvnw
    mvnw.cmd

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$
```



► Commandes Git - git commit



– Faire un premier commit

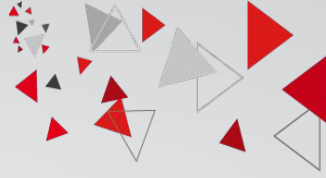
- La commande `git commit` permet d'enregistrer les fichiers de la zone de préparation (staging area) dans le dépôt local.



```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git commit -m "First commit of the projet SKI"
[master (root-commit) 4cba7dd] First commit of the projet SKI
39 files changed, 1372 insertions(+)
create mode 100644 pom.xml
create mode 100644 src/main/java/tn/esprit/spring/GestionStationSkiApplication.java
```



► Commandes Git - git commit



- Après avoir effectué des modifications dans un ou plusieurs fichiers, vous pouvez suivre ces étapes pour enregistrer vos modifications :

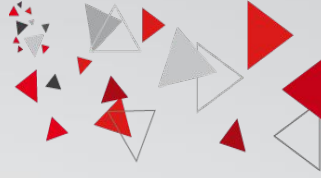


- Utilisez `git status` pour vérifier l'état de vos fichiers modifiés.
- Utilisez `git add` pour préparer les fichiers que vous souhaitez inclure dans le commit.
- Enfin, utilisez `git commit` pour enregistrer les modifications avec un commentaire approprié.

Assurez-vous de fournir un commentaire descriptif pour chaque commit afin de documenter les changements que vous avez effectués.



► Commandes Git - git push



– Configurer un dépôt distant (optionnel) :

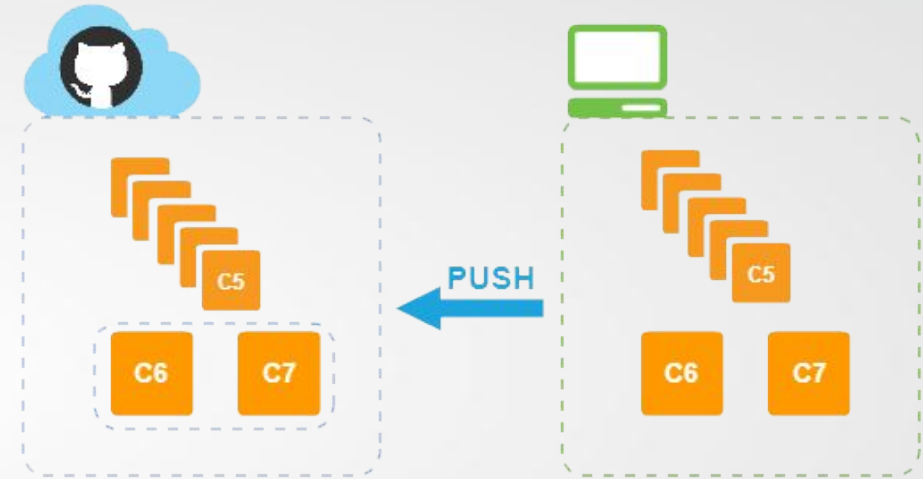
1. Connecter votre dépôt local à un dépôt distant (exple: GitHub) en utilisant la commande :

```
$ git remote add origin URL_du_dépôt
```

2. Transférer vos changements locaux vers le dépôt distant avec la commande :

```
$ git push -u origin master
```

- `$ git remote -v` : permet d'afficher les URL des dépôts distants associés.



Commandes Git - git push

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git remote -v

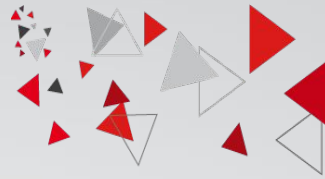
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git remote add origin https://github.com/OnsBENSALAH/ski.git

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git remote -v
origin https://github.com/OnsBENSALAH/ski.git (fetch)
origin https://github.com/OnsBENSALAH/ski.git (push)

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git push -u origin master
Enumerating objects: 58, done.
Counting objects: 100% (58/58), done.
Delta compression using up to 8 threads
Compressing objects: 100% (50/50), done.
Writing objects: 100% (58/58), 15.38 KiB | 2.20 MiB/s, done.
Total 58 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/OnsBENSALAH/ski/pull/new/master
remote:
To https://github.com/OnsBENSALAH/ski.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```



► Commandes Git - git log



- La commande `git log` permet d'afficher l'historique des commits

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git log
commit 4cba7ddf0773098a1170802e4950ccbea87d6901 (HEAD -> master, origin/master)
Author: Ons BEN SALAH <ons.bensalah@esprit.tn>
Date: Tue Oct 3 04:41:56 2023 +0100

    First commit of the projet SKI
```

- `$ git log --oneline` : Affiche chaque commit sur une seule ligne, avec son identifiant abrégé et son message.

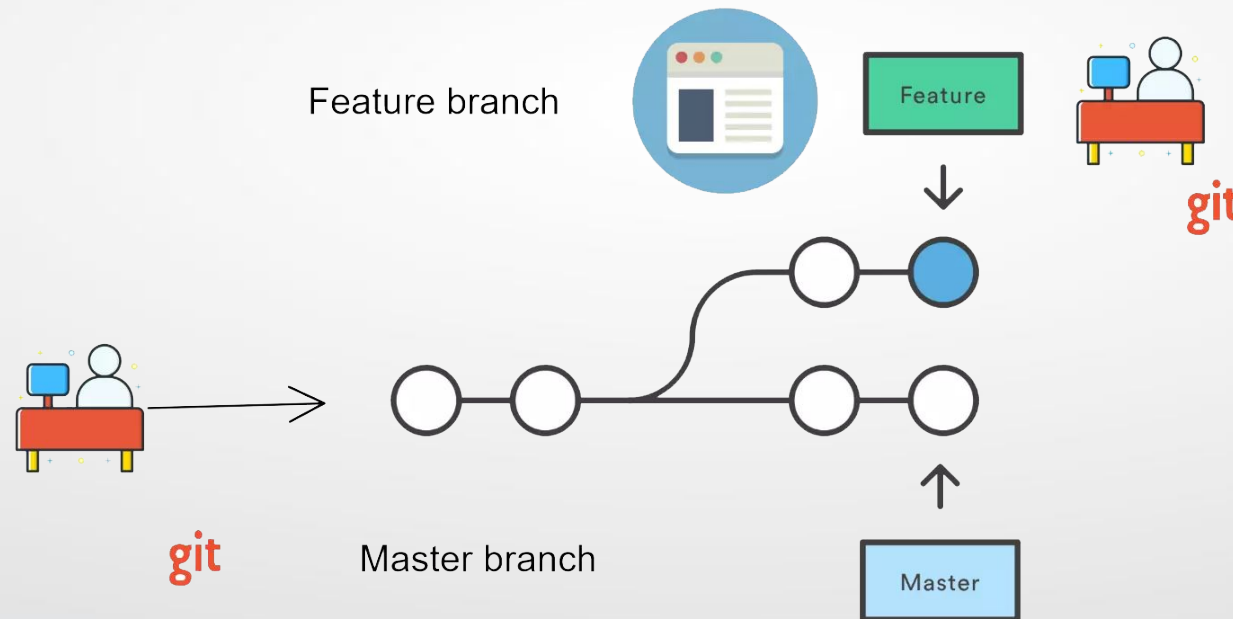
```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git log --oneline
4cba7dd (HEAD -> master, origin/master) First commit of the projet SKI
```



► Git - Les branches



- Les branches servent à séparer des modifications spécifiques pour répondre à des besoins particuliers, tels que des correctifs, des tests de fonctionnalités distinctes, etc.
- Cela permet une gestion plus efficace et organisée du développement logiciel.



► Commandes Git - git branch



- Pour vérifier les branches existantes, vous devez utiliser la commande :

```
$ git branch
```

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git branch
* master
```

* indique la branche actuelle

- Pour créer une nouvelle branche, utilisez la commande :

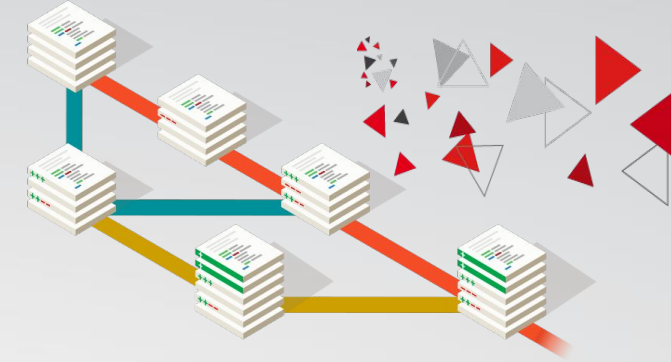
```
$ git branch <branch_name>
```

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git branch feature

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git branch
feature
* master
```



► Commandes Git - git checkout



- Pour basculer d'une branche à une autre utilisez la commande :

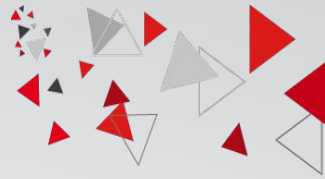
```
$ git checkout <branch_name>
```

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git checkout feature
Switched to branch 'feature'

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git branch
* feature
  master
```



► Commandes Git - les branches



- Effectuez une modification et enregistrez-la sur la branche avec un commit :

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git add src/main/java/tn/esprit/spring/entities/User.java

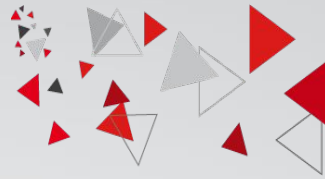
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   src/main/java/tn/esprit/spring/entities/User.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    .jpb/
    .mvn/
    mvnw
    mvnw.cmd

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git commit -m "new class User.java by feature branch"
[feature 3f11b7b] new class User.java by feature branch
1 file changed, 28 insertions(+)
create mode 100644 src/main/java/tn/esprit/spring/entities/User.java
```



► Commandes Git - les branches



- Visualisez l'historique des modifications avec la commande `git log`

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git log
commit 3f11b7b9ce48d1437c34d5b4dc62a9477ef01c99 (HEAD -> feature)
Author: Ons BEN SALAH <ons.bensalah@esprit.tn>
Date: Tue Oct 3 07:41:04 2023 +0100

    new class User.java by feature branch

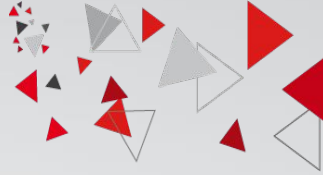
commit 4cba7ddf0773098a1170802e4950ccbea87d6901 (origin/master, master)
Author: Ons BEN SALAH <ons.bensalah@esprit.tn>
Date: Tue Oct 3 04:41:56 2023 +0100

    First commit of the projet SKI

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git log --oneline
3f11b7b (HEAD -> feature) new class User.java by feature branch
4cba7dd (origin/master, master) First commit of the projet SKI
```



► Commandes Git - git merge



- Pour fusionner le contenu de la branche `feature` dans la branche `master`, vous devez d'abord basculer vers la branche `master` et ensuite effectuer une fusion en utilisant la commande: `git merge`

```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (feature)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

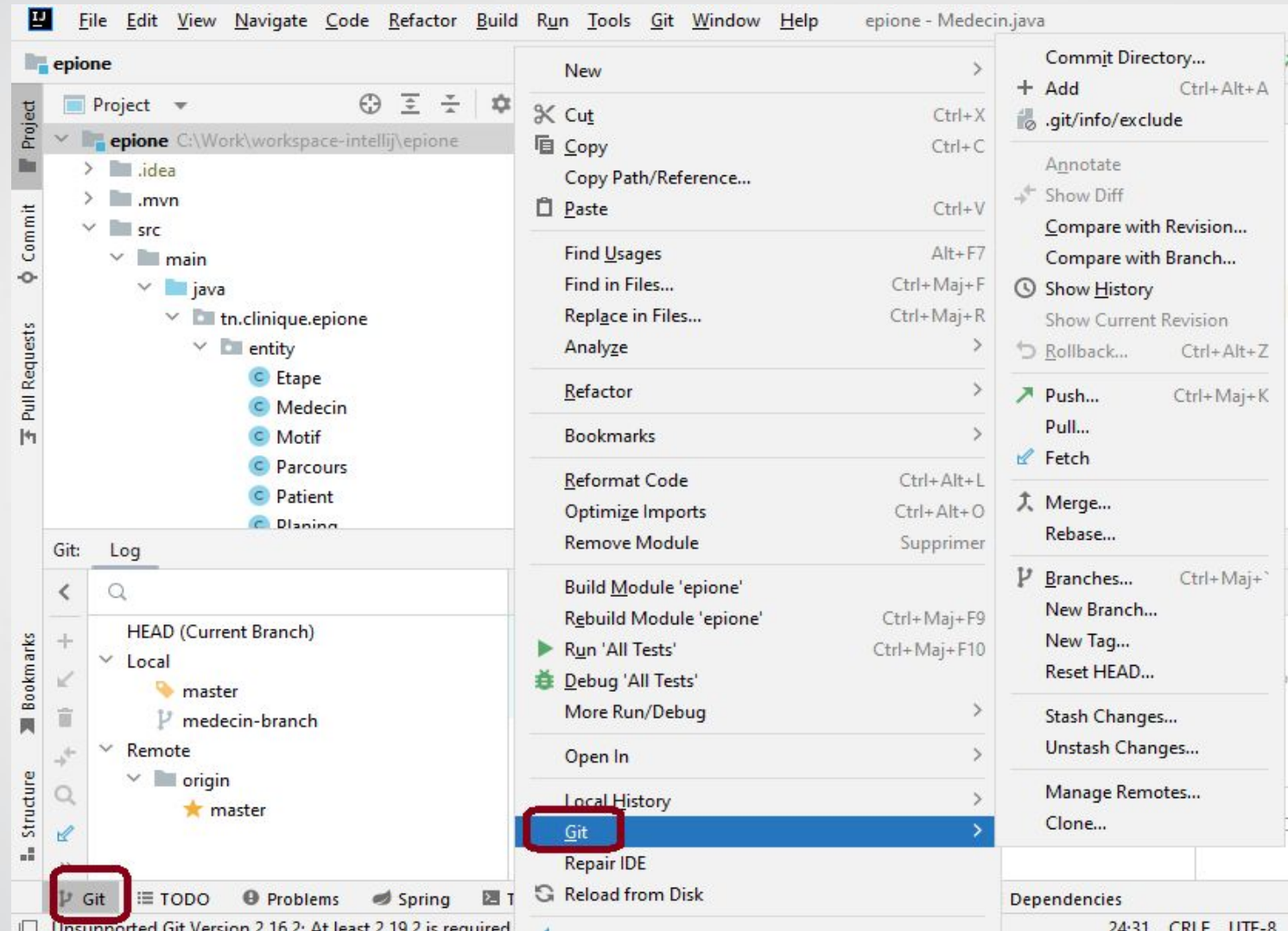
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git branch
  feature
* master

onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git merge feature
Updating 4cba7dd..3f11b7b
Fast-forward
 src/main/java/tn/esprit/spring/entities/User.java | 28 ++++++
 1 file changed, 28 insertions(+)
 create mode 100644 src/main/java/tn/esprit/spring/entities/User.java
```

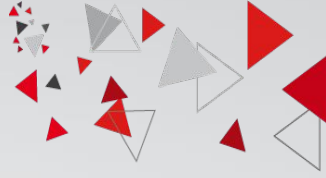


▶ Git avec un IDE

- Vous pouvez utiliser Git en ligne de commande ou sur IntelliJ :



Git - Collaborateurs

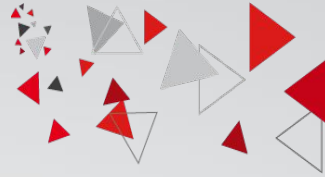


- En tant que master, je dois inviter les autres membres de l'équipe pour partager le code source de l'application.
- Sur GitHub, allez sur l'interface : **Settings -> Collaborators**

The screenshot shows the GitHub interface for a repository named 'ski' by user 'OnsBENSALAH'. The 'Settings' tab is active, and the 'Collaborators' sub-tab is selected. The 'Who has access' section indicates that the repository is public and visible to anyone, with 0 collaborators. The 'Manage access' section shows a message 'You haven't invited any collaborators yet' and a green 'Add people' button.



► Commandes Git - git clone

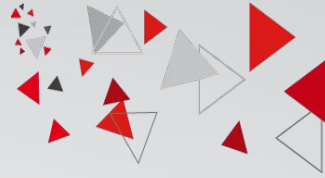


- Pour obtenir une copie locale d'un repo distant sur lequel travailler, vous devez taper la commande suivante : `$ git clone <https link>`

The screenshot shows a GitHub repository page for 'OnsBENSALAH' with the 'Code' dropdown menu open. The repository is on the 'master' branch, which is 1 commit ahead and 3 commits behind 'main'. The repository contains a 'src' folder and a 'pom.xml' file, both from the first commit. The 'Code' dropdown menu is open, showing options for 'Local' and 'Codespaces'. The 'Clone' option is selected, and the repository URL 'https://github.com/OnsBENSALAH/ski.git' is highlighted in a red box. Below the URL, it says 'Use Git or checkout with SVN using the web URL.' Other options in the menu include 'Open with GitHub Desktop' and 'Download ZIP'. The right sidebar shows repository statistics: 0 stars, 1 watching, and 3 forks. The 'Releases' section indicates no releases have been published.



► Commandes Git - git clone



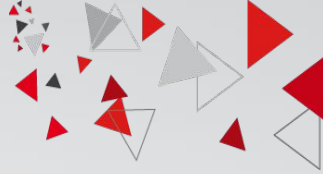
```
onsbe@LAPTOP-R32K4R59 MINGW64 ~/IdeaProjects/gestion-station-ski (master)
$ git clone https://github.com/OnsBENSALAH/ski.git
Cloning into 'ski'...
remote: Enumerating objects: 105, done.
remote: Counting objects: 100% (105/105), done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 105 (delta 30), reused 83 (delta 14), pack-reused 0
Receiving objects: 100% (105/105), 24.95 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (30/30), done.
```

\$ git clone c'est équivalent à :

- git init
- git remote add origin https://github.com/OnsBENSALAH/ski.git
- git pull origin master



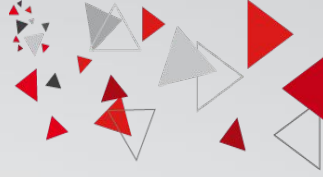
Git - Travail à faire (1/2)



- Vous avez tous effectué des opérations sur le projet exemple. Maintenant, vous allez collaborer en équipe sur votre projet final :
 1. Un membre de chaque équipe récupère le projet prêt à l'emploi depuis le dossier "code-source" du Drive, le décompresse dans son espace de travail local IntelliJ, puis le publie sur GitHub en suivant les étapes ci-dessus.
 2. Celui qui a publié le projet sur GitHub doit accorder l'accès à ses collègues (dans l'interface GitHub : Setting -> Collaborators).
 3. Les autres membres de chaque équipe récupèrent le projet depuis GitHub (et non depuis le Drive).
 4. Chacun crée sa propre branche et la pousse sur GitHub.




Git - Travail à faire (2/2)



5. Mettez à jour le fichier XLS partagé par votre enseignant sur le Classroom avec la liste des membres de votre groupe.
6. Invite votre enseignant à rejoindre le dépôt de l'équipe sur GitHub.com
7. Chacun crée un Job de type Pipeline sur son instance Jenkins avec le format "Prenom_NOM_CLASSE", comportant deux étapes :
 - **Récupération du code de sa propre branche**
 - **Lancement de la commande Maven qui nettoie le projet et le compile**





*"Apprendre par le projet, c'est découvrir
▶ par l'action, créer par la compréhension, et
réussir par la persévérance."*