

Gestion des notes des étudiants

Introduction

Le but de ce TP est de créer des classes permettant de représenter des étudiants (classe `Student`), des notes (classe `Grade`), des résultats à une unité d'enseignement (classe `TeachingUnitResult`) et des promotions d'étudiants (classe `Cohort`).

L'objectif pour vous est de vous initier :

- à l'utilisation d'un IDE : IntelliJ IDEA de JetBrains est mis en avant mais n'importe quel IDE équivalent comme par exemple Eclipse ou Visual studio code pourra être utilisé (néanmoins on ne vous aidera pas sur l'utilisation des autres IDE),
- à la gestion de version grâce à Git,
- à l'utilisation des tests unitaires (ici déjà écrits) pour valider votre programme.

Vous allez sans doute découvrir un certain nombre de nouveaux outils durant ce TP. Le temps d'apprentissage de ces outils peut être au début frustrant mais ils vont vous faire gagner du temps par la suite.

Mise en place du TP

Introduction

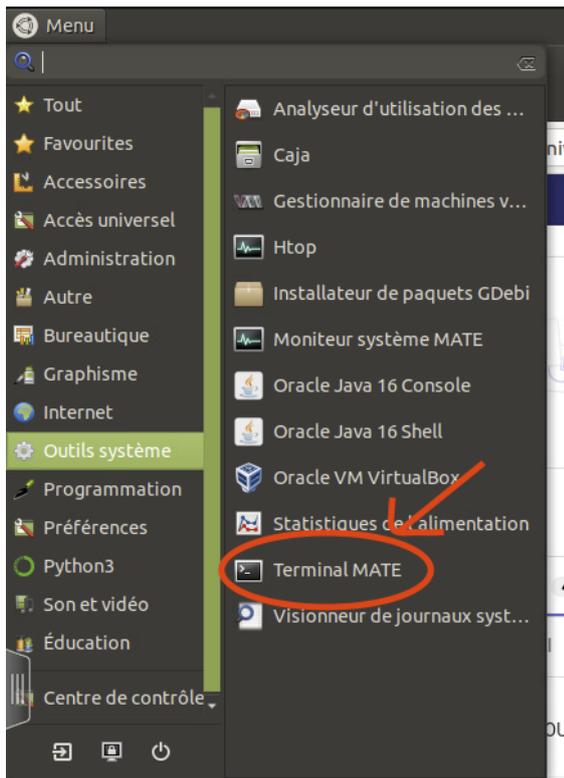
Ce TP est l'occasion d'apprendre à utiliser des outils de développement professionnel. En plus de l'IDE IntelliJ, vous aurez à utiliser un outil de gestion de versions, en l'occurrence `git`. Un tel outil permet d'enregistrer à distance votre projet et de permettre à plusieurs personnes de travailler simultanément, sans devoir échanger les fichiers par courriel, et sans se marcher sur les pieds (par exemple modification sans le savoir du même fichier).

Configuration clé ssh pour le gitlab AMU

La première étape pour utiliser la gestion de version est de vous connecter au gitlab de l'université qui est accessible à l'adresse suivante : <https://etulab.univ-amu.fr/>. L'identifiant et le mot de passe sont ceux de votre compte étudiant AMU.

Afin de pouvoir accéder à distance à vos dépôts git, vous allez configurer une clé SSH dans votre profil. Pour cela, il vous faut:

1. Ouvrir un terminal (par exemple terminal MATE de la vdi Linux qui est accessible dans le menu au sous-menu Outils systèmes). Si vous êtes sous windows et que vous avez installé Git pour windows, il est conseillé d'utiliser le terminal bash de git.



2. Générer une paire de clés privé/public pour votre compte. Pour cela, il vous faut entrer la commande suivante dans le terminal :

```
~$ ssh-keygen -t ed25519
```

Appuyer une fois sur Entrée pour confirmer que vous voulez sauvegarder vos clés dans le répertoire par défaut.

Ensuite, il vous est demandé de rentrer deux fois une “passphrase”. Vous pouvez entrer une phrase (plusieurs mots donc) qui servira de mot de passe pour l’accès. Puisque la sécurité de vos dépôt n’est pas critique, vous pouvez vous contenter de ne rien rentrer (pas de passphrase) et donc d’appuyer de nouveau sur Entrée deux fois.

Si tout s’est bien passé, votre couple de clés a été généré et vous devriez voir un affichage similaire à celui ci-dessous :

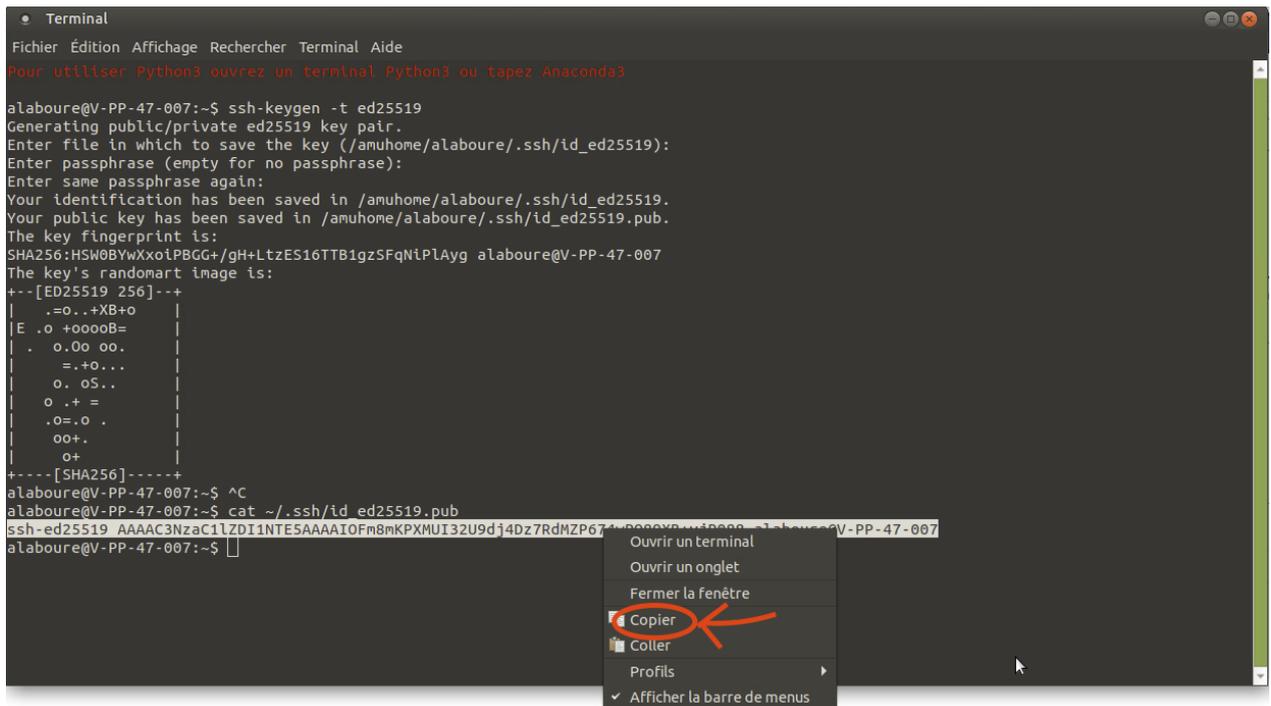
```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide

alaboure@V-PP-47-007:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/amuhome/alaboure/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /amuhome/alaboure/.ssh/id_ed25519.
Your public key has been saved in /amuhome/alaboure/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:HSW0BYwXxoiPBGG+/gH+LtzES16TTB1gzSFqNiPlAyg alaboure@V-PP-47-007
The key's randomart image is:
+--[ED25519 256]--+
|   .o..+XB+o   |
|E  .o +ooooB=  |
| .  o.Oo oo.   |
|   =.+o...     |
|   o. oS..     |
| o  .+ =       |
| .o=.o .       |
|   oo+.        |
|   o+          |
+-----[SHA256]-----+
```

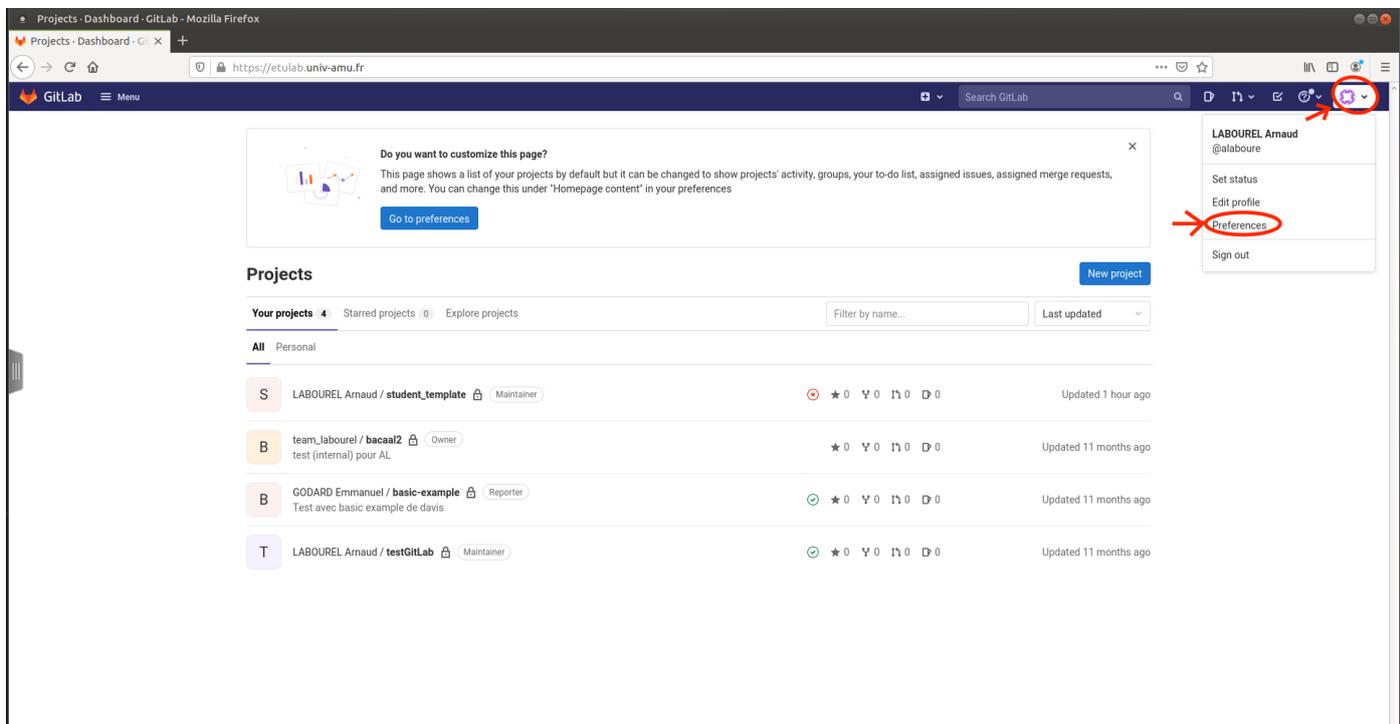
Afficher votre clé dans le terminal en entrant la commande suivante (sous powershell, il faut remplacer la commande `cat` par la commande `type`) :

```
~$ cat ~/.ssh/id_ed25519.pub
```

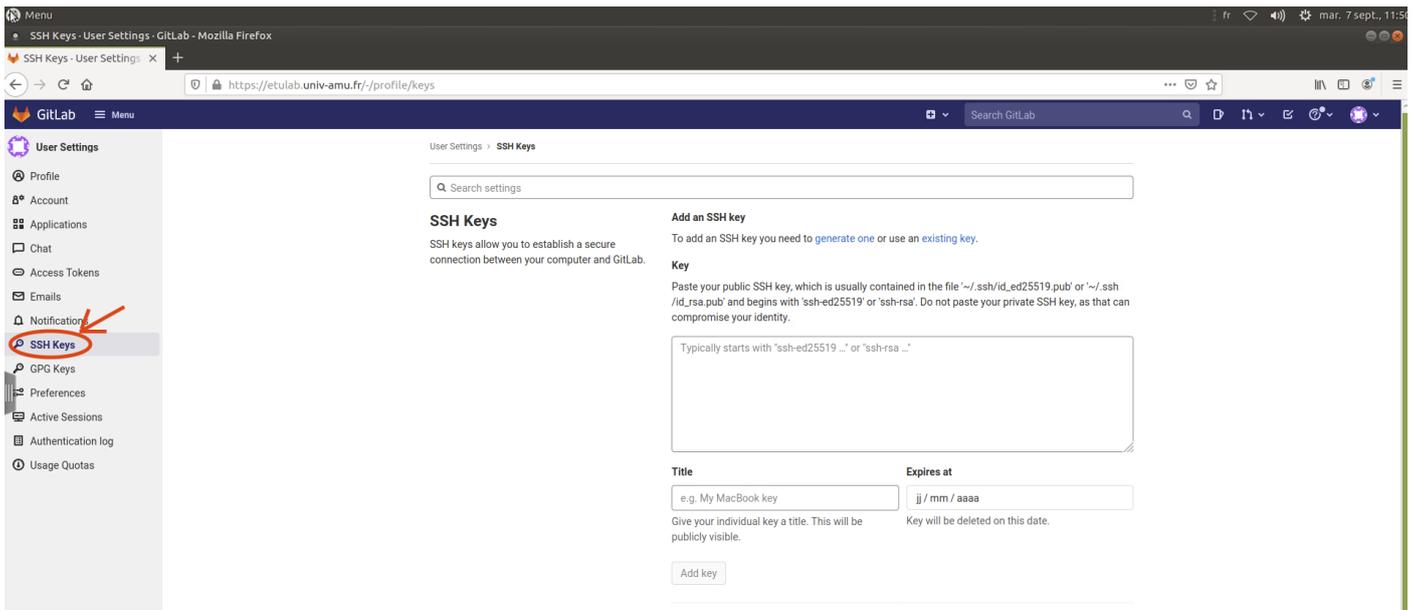
Sélectionner la ligne affichée par le terminal et copier là dans le presse-papier (sélection puis clic droit et copier)



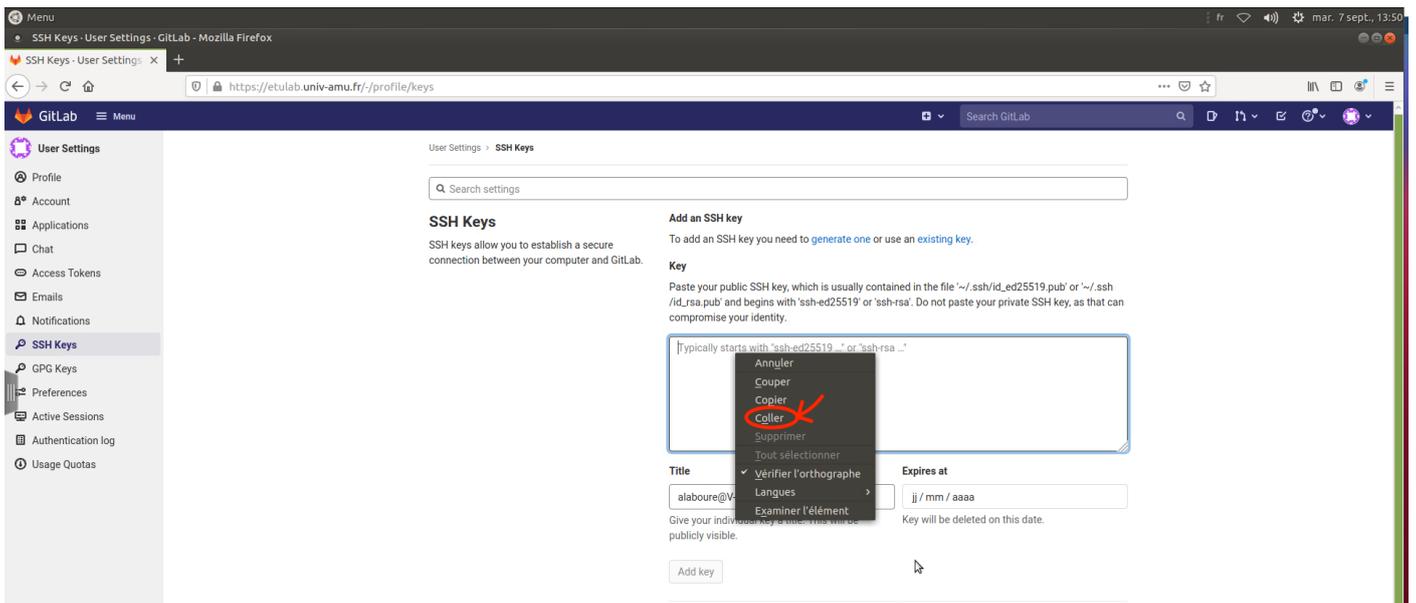
3. Configurer la clé dans votre compte gitlab. Pour cela, vous allez accéder aux préférences de votre compte gitlab. Il vous faut vous connecter à <https://etulab.univ-amu.fr/> puis cliquez sur votre avatar en haut à gauche de l'écran puis sur *preferences* dans le menu qui apparaît.



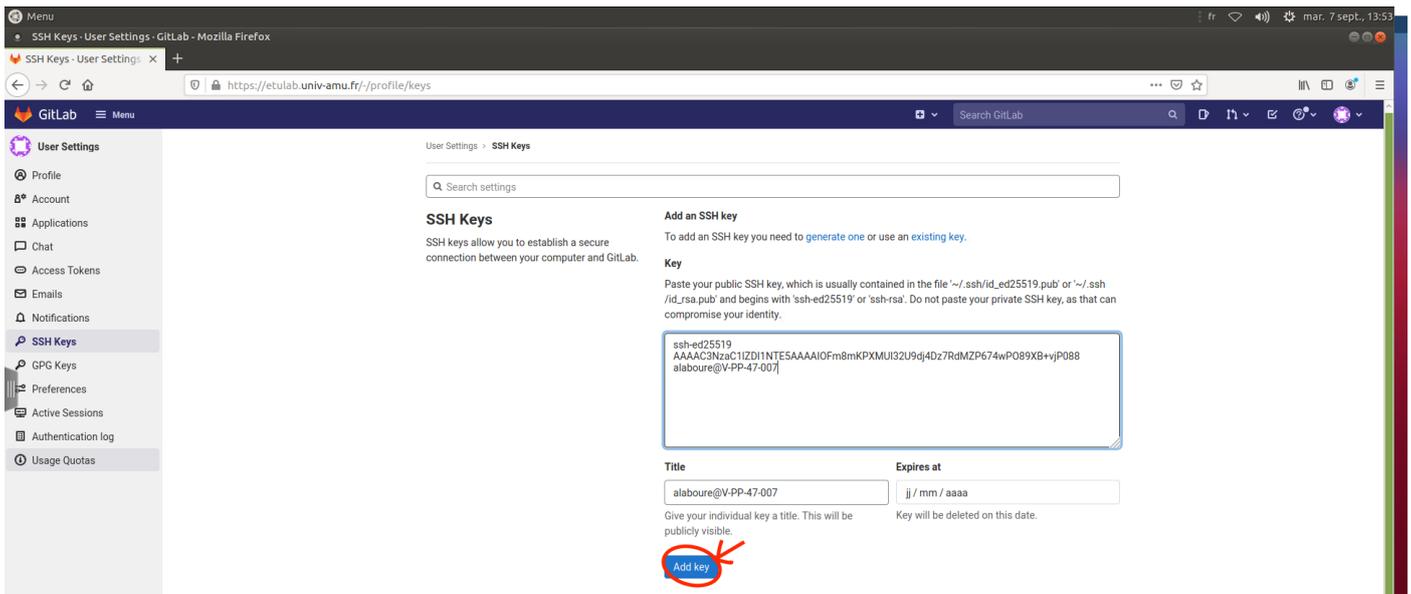
Allez au menu des clés SSH en cliquant sur le lien correspondant dans le menu de gauche.



4. Revenez sur la page de configuration des clés ssh sur votre navigateur. Coller votre clé (clic droit puis coller) dans l'espace prévu pour cela.



Ajouter la clé en cliquant sur le bouton *add*.

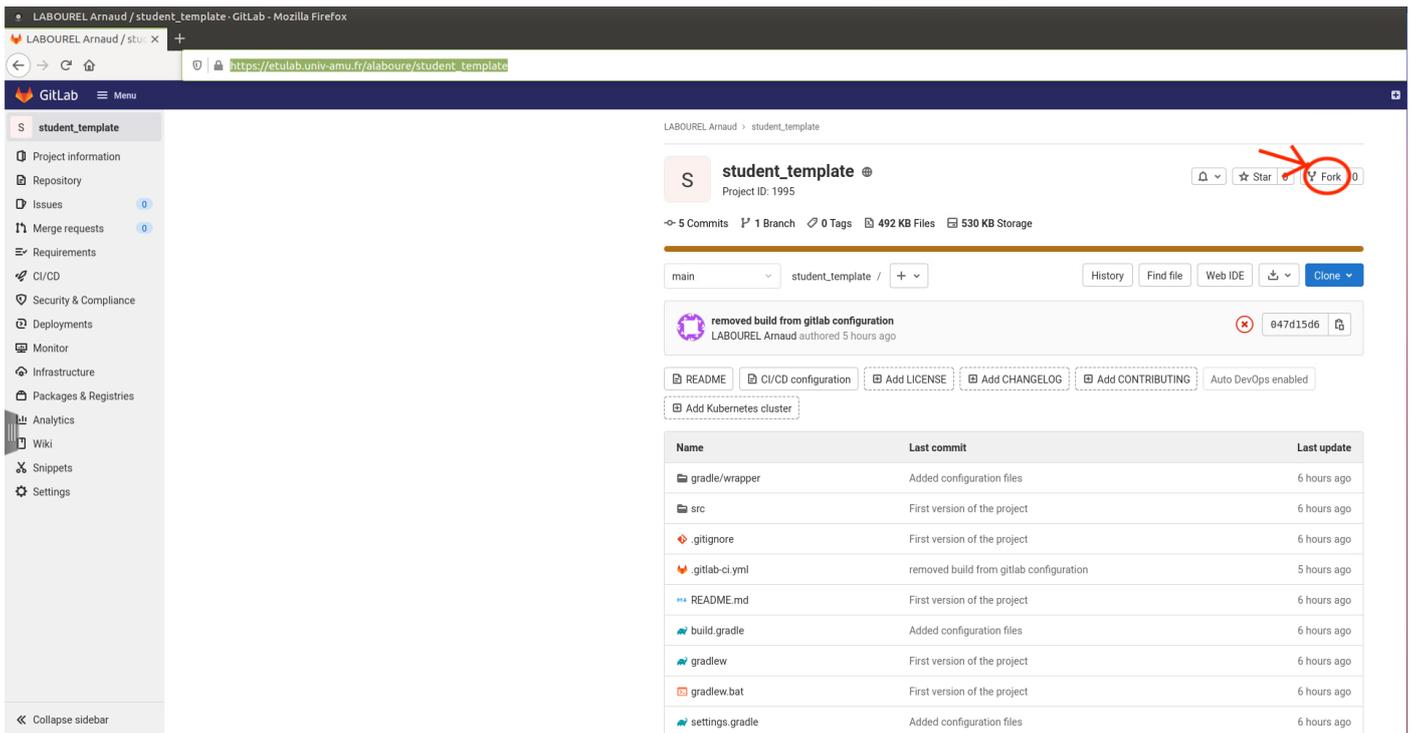


Vous devriez recevoir un mail sur votre mail étudiant confirmant que vous avez rajouté une clé. Vous pouvez rajouter autant de clés que vous voulez. Vous pouvez donc faire de même pour rajouter par exemple des clés supplémentaires si vous souhaitez accéder à votre dépôt depuis chez vous.

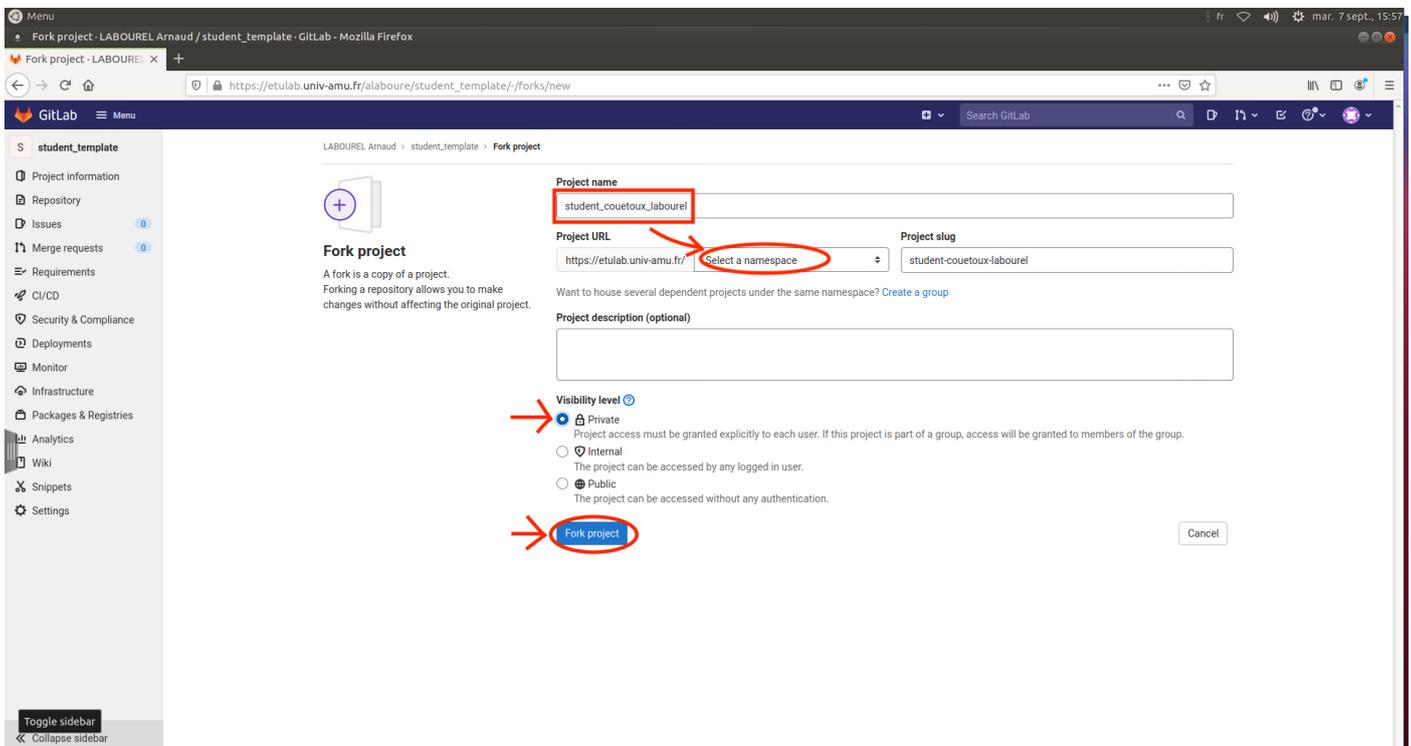
Fork d'un projet

Vous allez maintenant créer votre projet en utilisant un projet déjà existant. Pour cela, il faut :

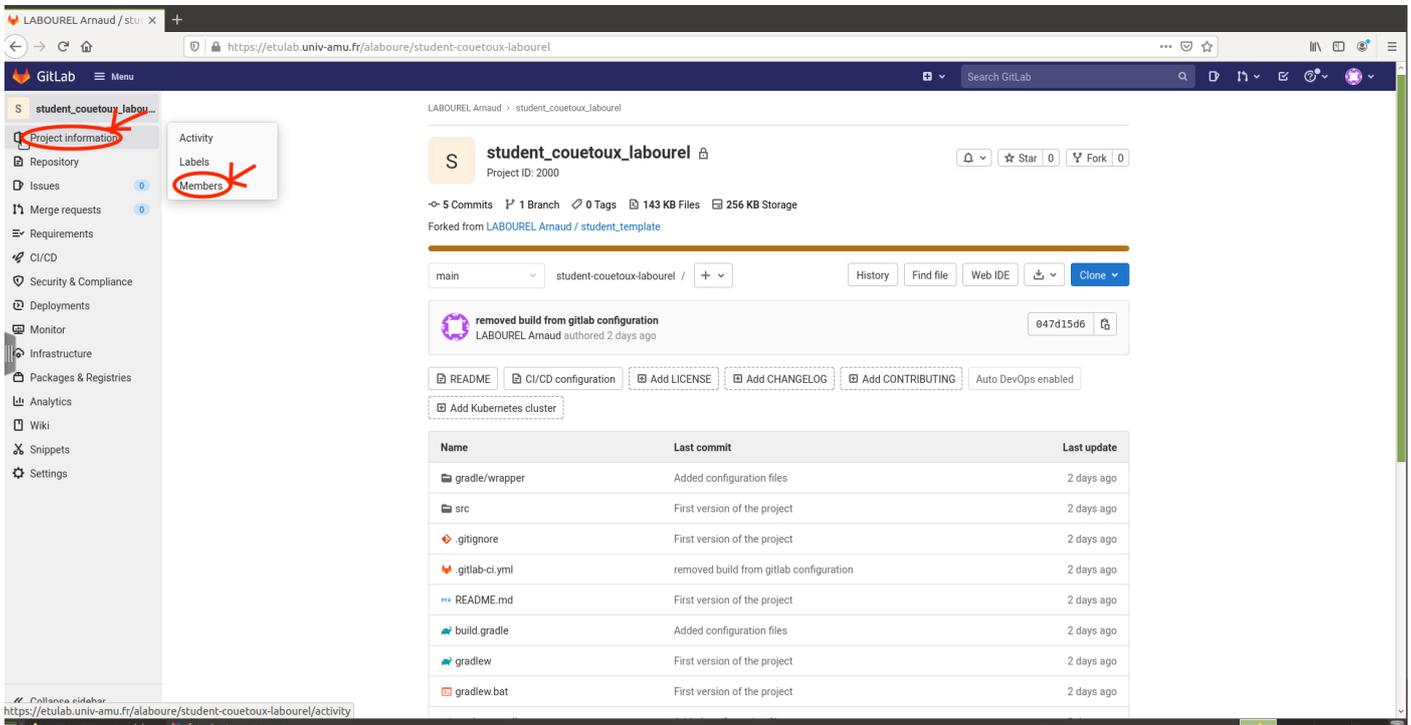
1. Aller sur le projet `student_template` qui servira de base pour ce premier TP en accédant à l'adresse suivante : https://etulab.univ-amu.fr/alaboure/student_template
2. Cliquer sur le bouton *fork*.



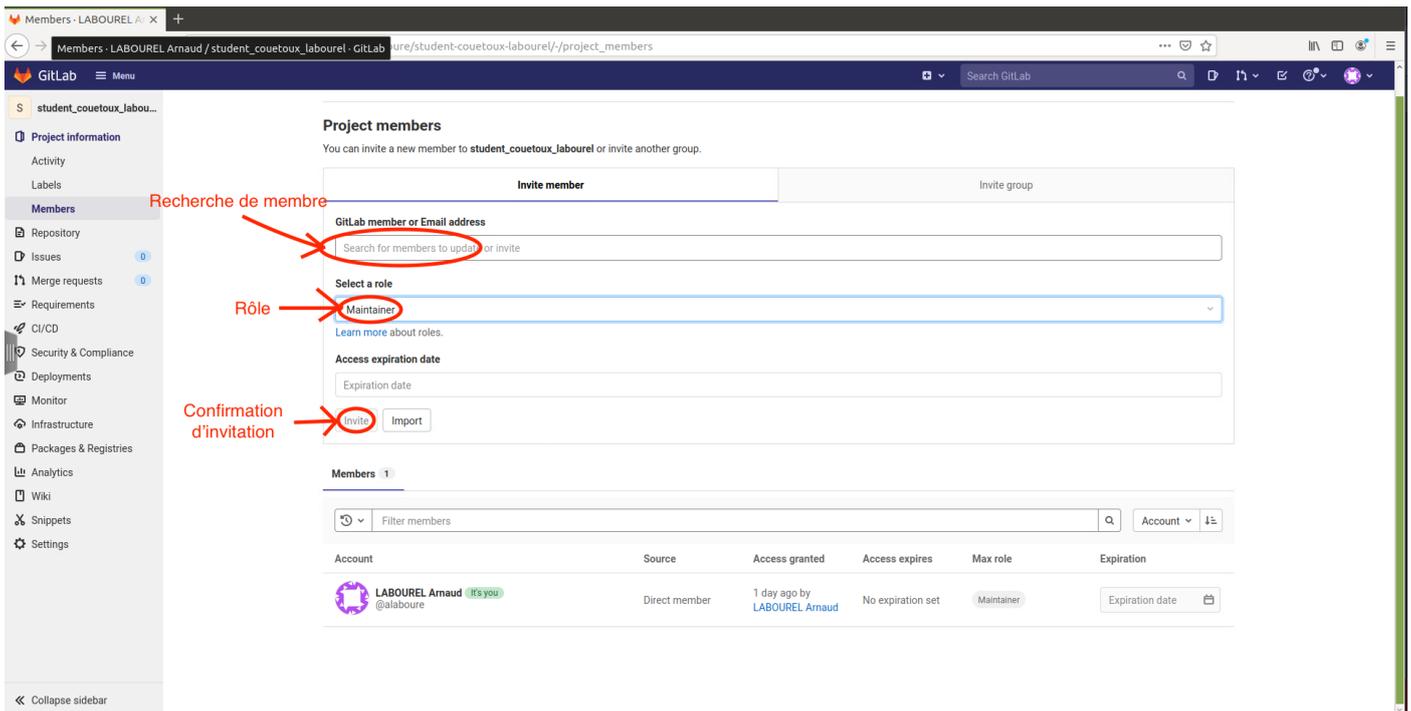
3. Changer le nom du projet en enlevant template et en mettant le nom du ou des étudiants sur le projet. Sélectionner comme espace de nom (*spacename*) votre propre compte. Mettez la visibilité du projet en private afin que vos camarades en dehors du projet n'y aient pas accès et validez le fork en cliquant sur le bouton fork project.



4. Une fois le projet créé, vous pouvez rajouter des membres (par exemple la personne en charge de l'enseignement Arnaud Labourel arnaud.labourel@univ-amu.fr, login `alabourel`) en cliquant sur **project information** dans le menu de gauche puis **members**.



5. Ensuite vous pouvez rechercher une personne dans la barre dédiée. Une fois celle-ci trouvée vous pouvez lui donner un rôle (au moins **reporter** pour donner l'accès en lecture du code ou bien **maintainer** pour l'accès en lecture et écriture), puis confirmer son invitation en cliquant sur le bouton **invite**.



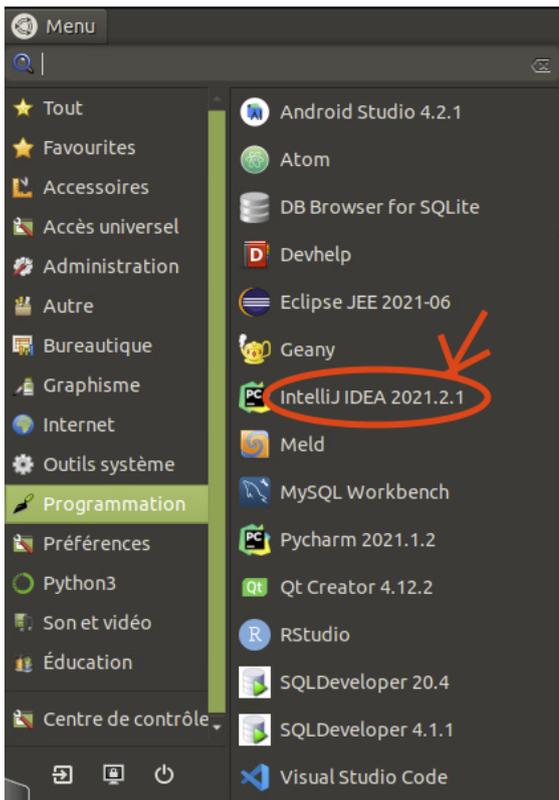
IDE IntelliJ

Environnement de développement (IDE)

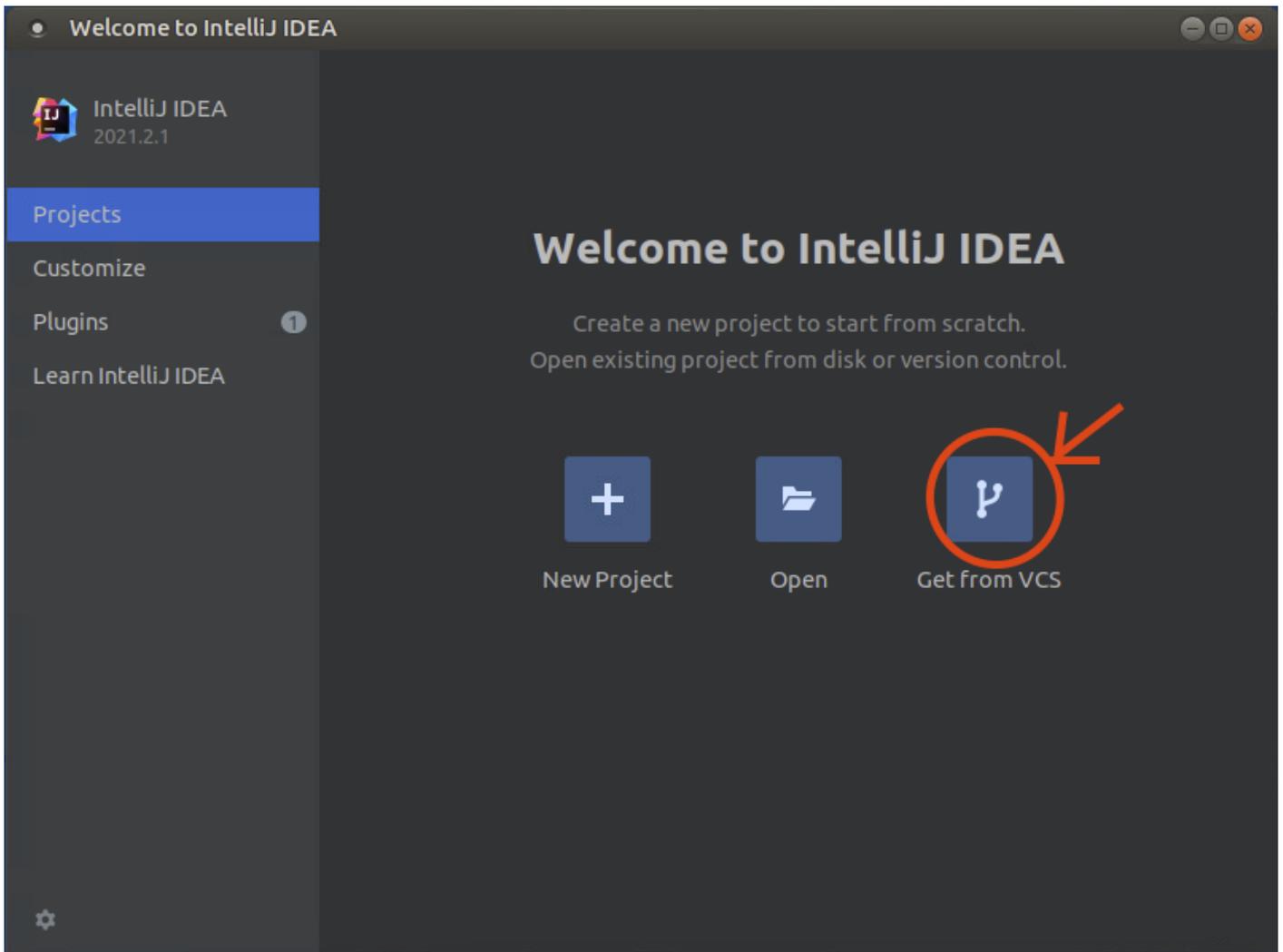
Afin de programmer dans ce cours, nous allons utiliser un environnement de développement (Integrated Development Environment : IDE). Il existe de nombreux IDE pour Java. Dans ce cours, nous vous conseillons d'utiliser IntelliJ IDEA de JetBrains mais vous pouvez aussi utiliser Eclipse, Visual Studio Code, NetBeans ou un autre IDE si vous êtes familier avec ceux-ci.

Lancement de l'IDE

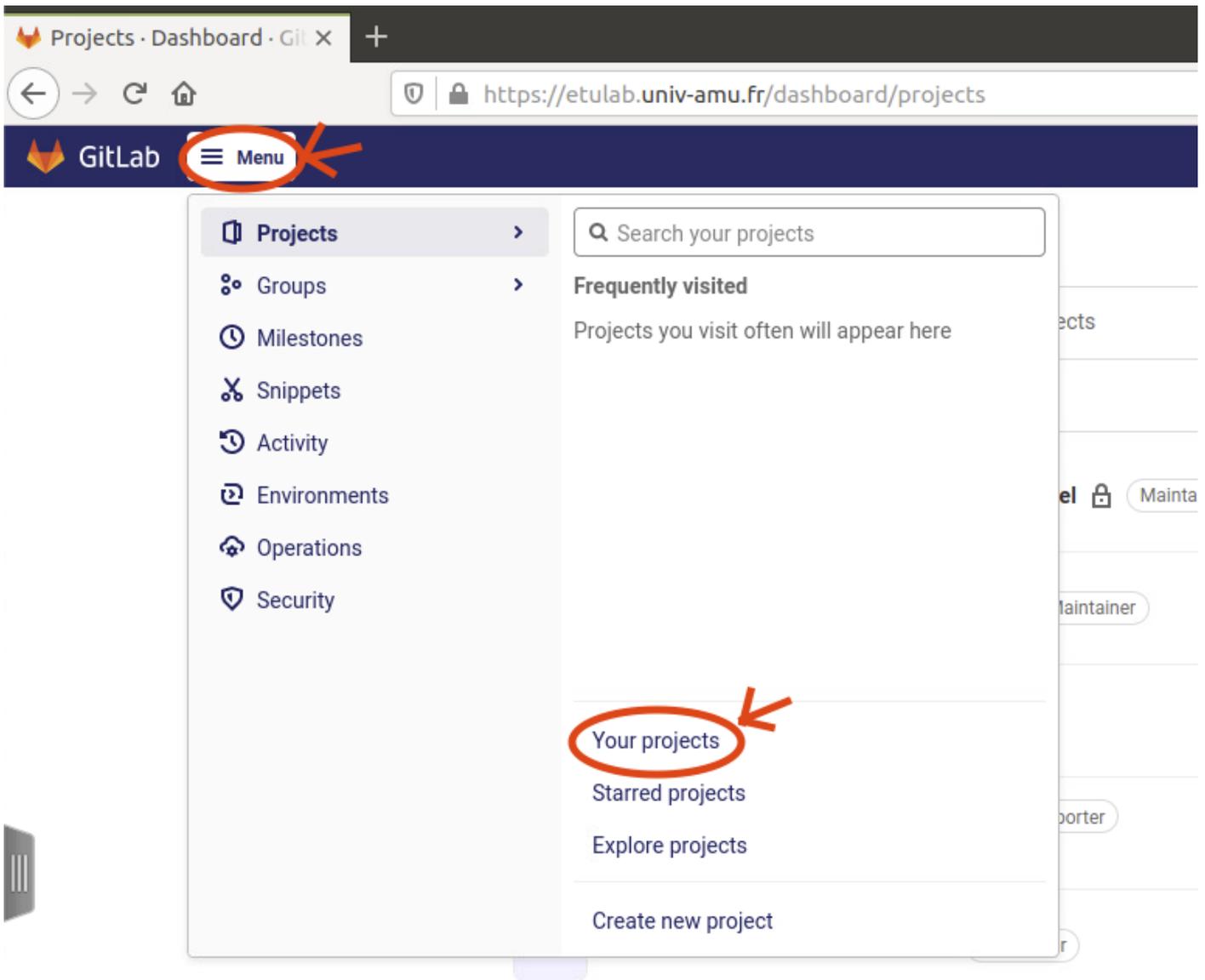
Pour lancer le logiciel sur les machines virtuelles de l'université (image LUNIX à [https://pedagogie-vdi.univ-amu.fr/]), il vous faut aller dans le Menu (en haut à gauche de l'écran) et cliquer sur 'IntelliJ IDEA 2022.2.3' dans la partie Programmation du menu.



1. Vous devriez accéder à l'écran d'accueil d'IntelliJ IDEA. Vous allez récupérer le contenu de votre dépôt. Pour cela, cliquez sur `Get from VCS` puis sur `git` dans le menu qui apparaît.



1. Vous allez maintenant récupérer votre dépôt (en fait le cloner, c'est-à-dire créer une copie du dépôt sur votre machine). Pour cela, connectez-vous à votre compte *etulab* dans un navigateur et accédez à votre projet. Vous pouvez accéder à la liste de vos projets.

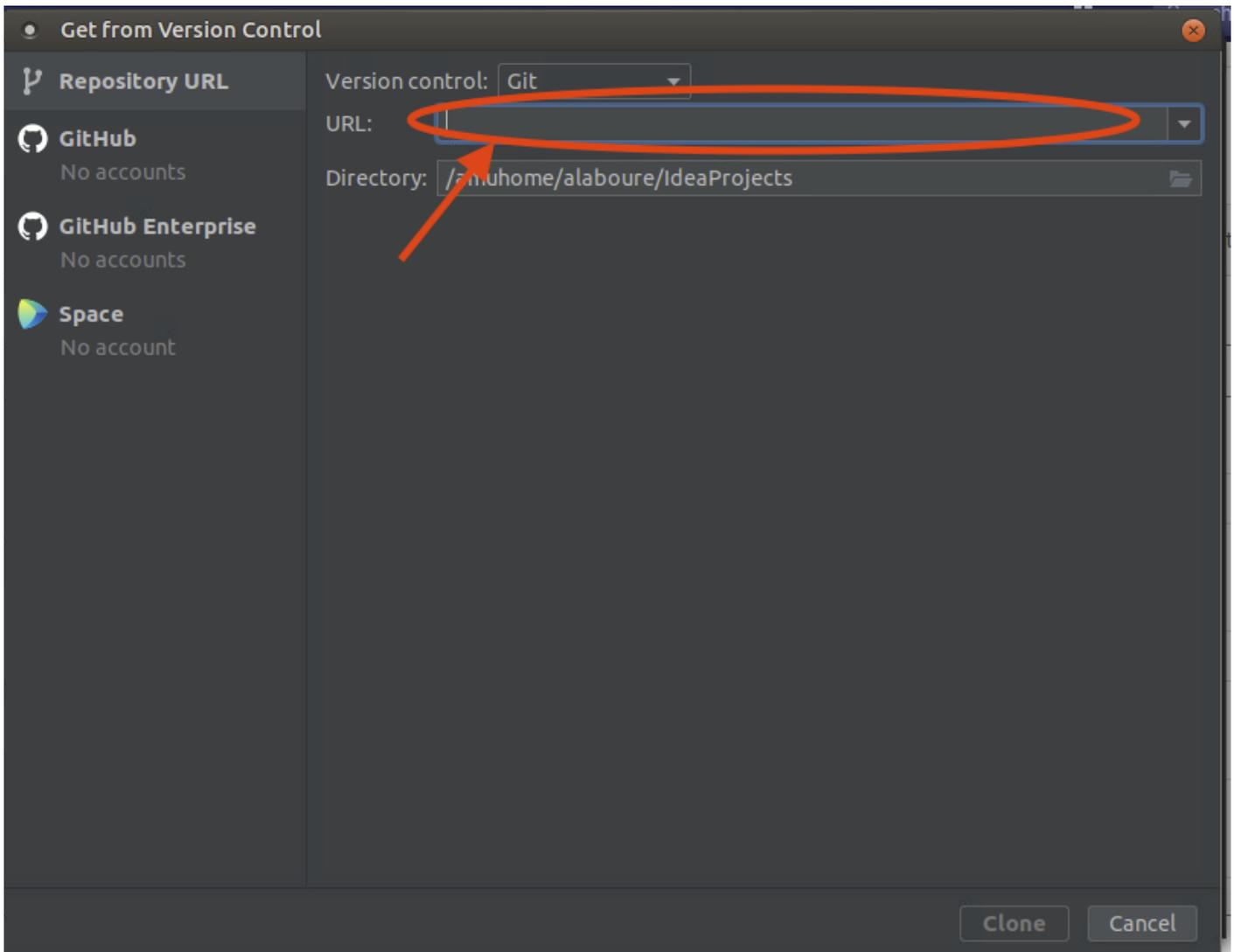


2. Copier l'adresse de votre dépôt sur la page web de votre dépôt en cliquant sur le bouton **clone** puis sur l'icône presse-papier.

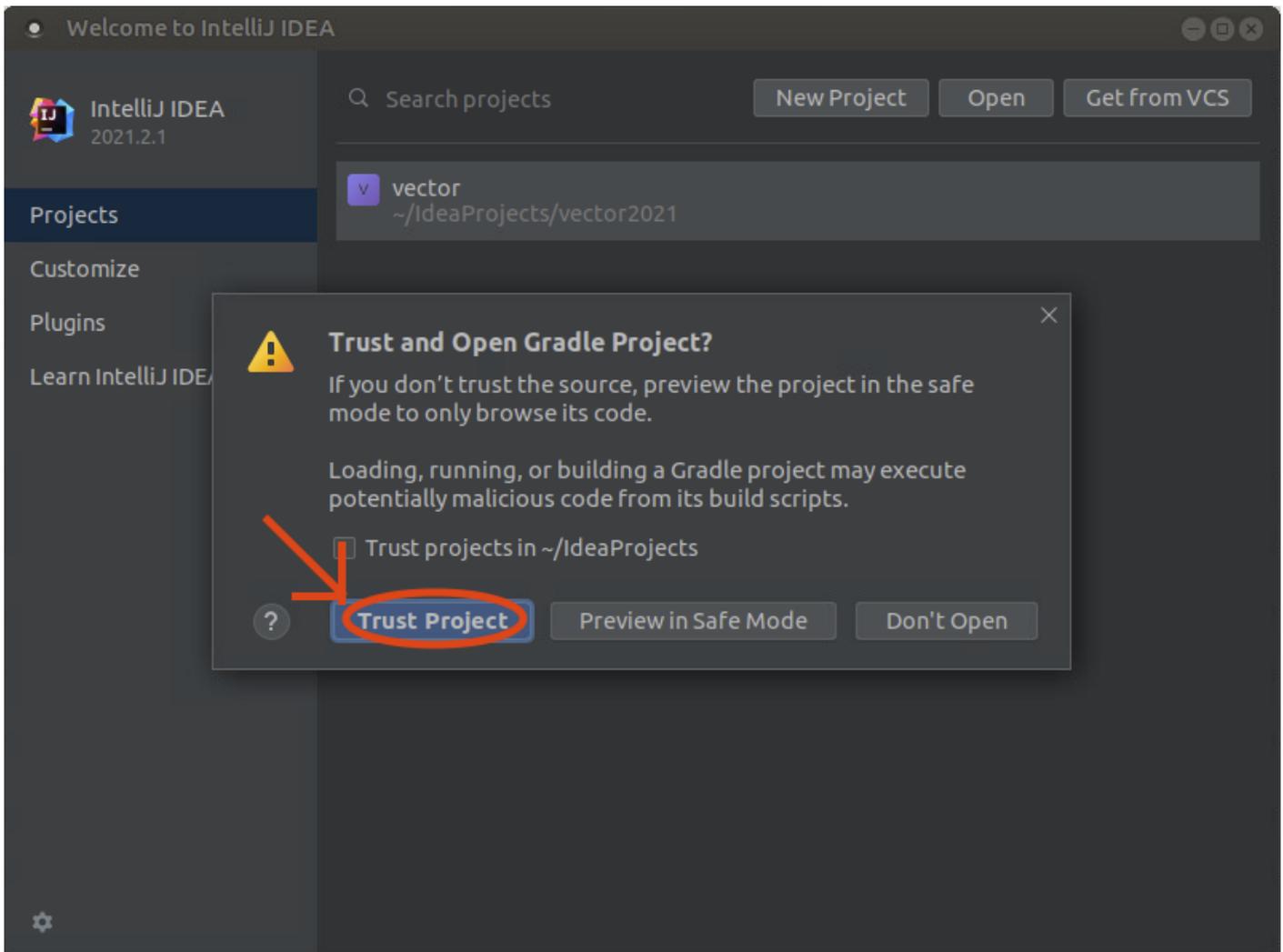
The screenshot shows the GitLab interface for a repository named 'student_couetoux_labourel'. The page includes a sidebar with navigation options, repository statistics (5 Commits, 1 Branch, 0 Tags, 143 KB Files, 256 KB Storage), and a file browser. A table lists the files in the repository, including 'gradle/wrapper', 'src', '.gitignore', '.gitlab-ci.yml', 'README.md', 'build.gradle', 'gradlew', and 'gradlew.bat'. A 'Clone' button is highlighted with a red circle and an arrow, and a copy icon next to the SSH URL is also highlighted with a red circle and an arrow.

| Name | Last commit |
|----------------|---|
| gradle/wrapper | Added configuration files |
| src | First version of the project 2 days ago |
| .gitignore | First version of the project 2 days ago |
| .gitlab-ci.yml | removed build from gitlab configuration 2 days ago |
| README.md | First version of the project 2 days ago |
| build.gradle | Added configuration files 2 days ago |
| gradlew | First version of the project 2 days ago |
| gradlew.bat | First version of the project 2 days ago |

1. Coller dans le champs URL d'IntelliJ, l'adresse de votre dépôt puis cliquez sur le bouton clone.



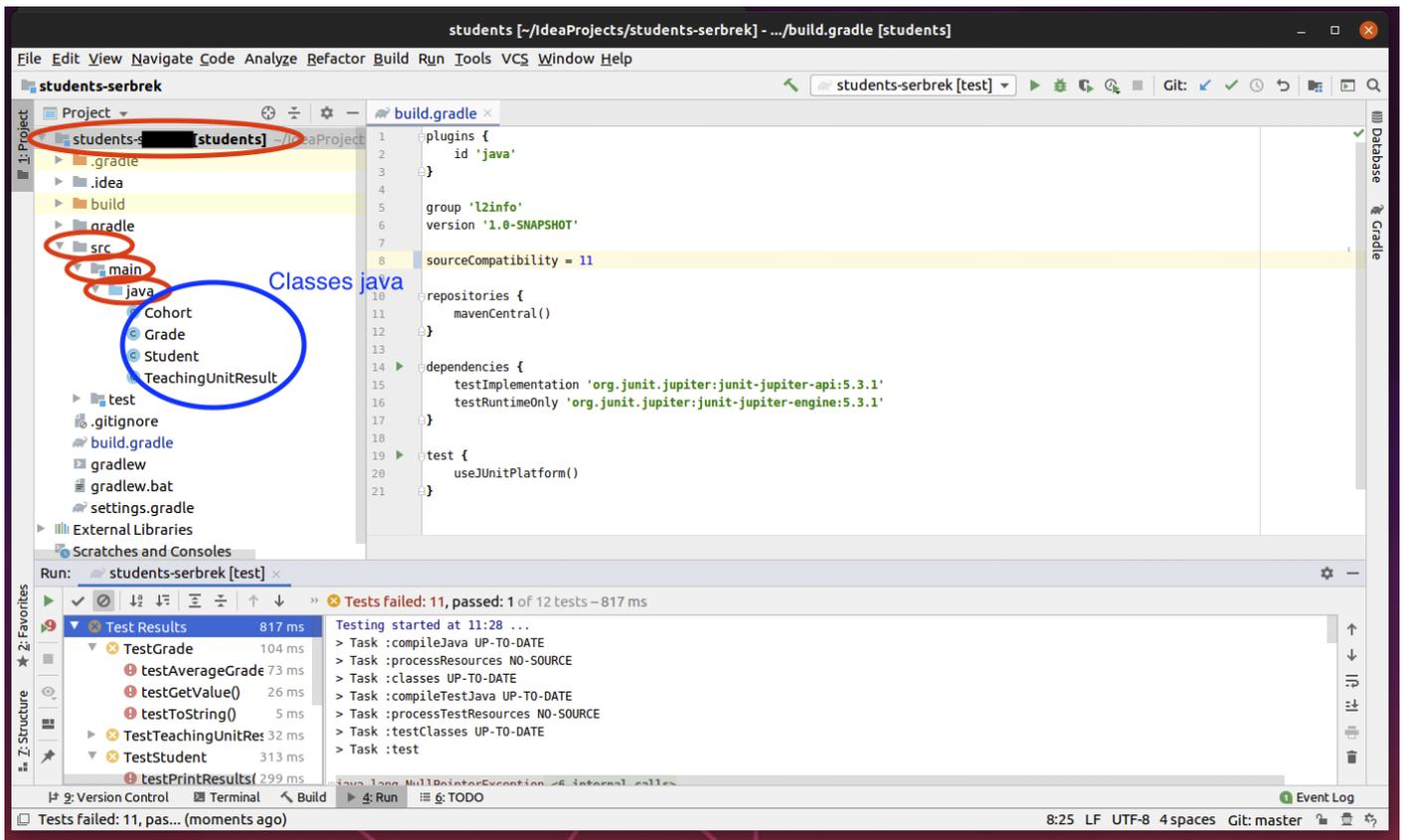
1. Après quelques instants de chargement, on vous demandera si vous faites confiance au projet. Cliquer **Trust Project** pour que le projet se lance.



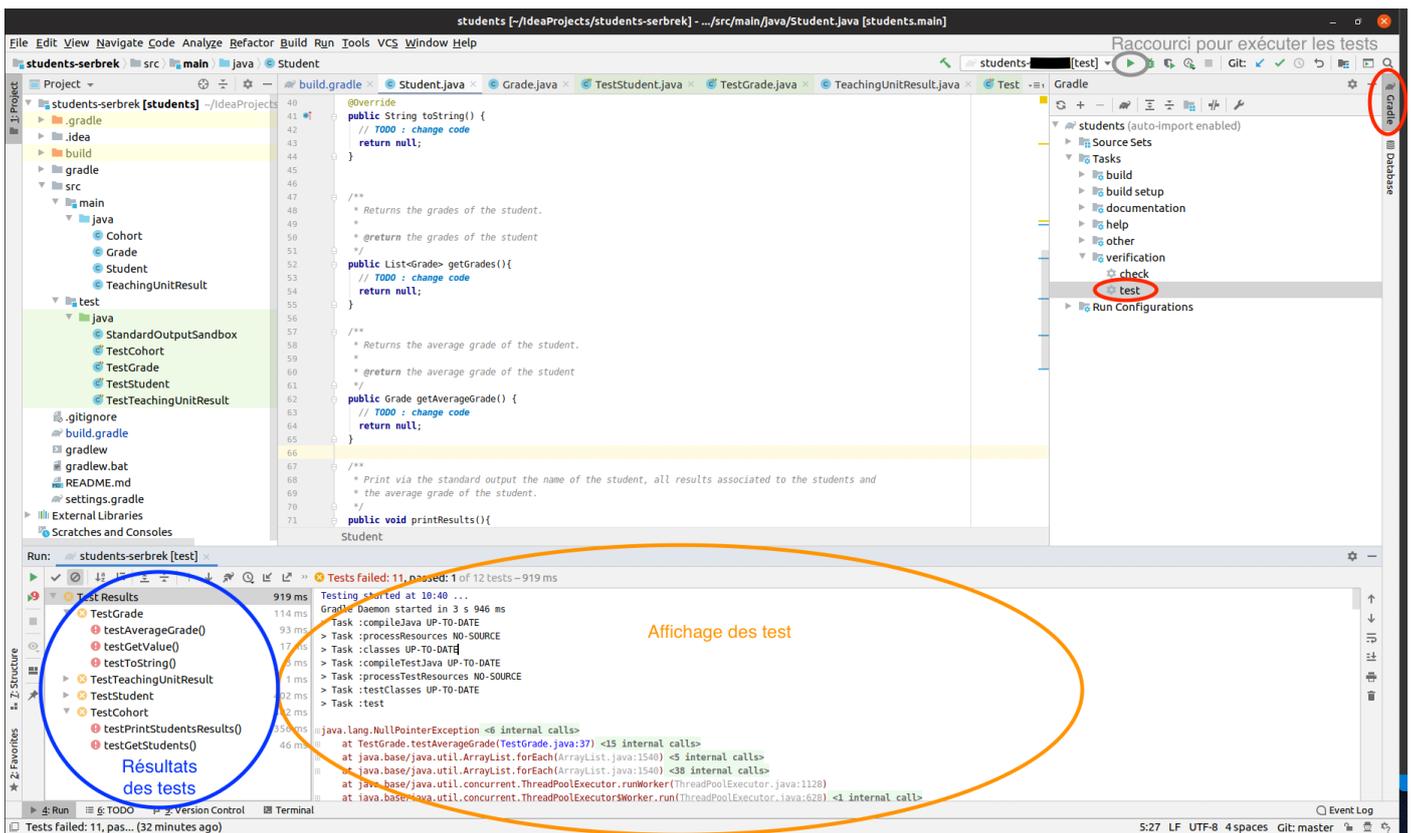
1. Votre projet devrait maintenant être utilisable sur votre machine.

Prise en main d'IntelliJ IDEA

1. Pour accéder aux fichiers `java` de votre dépôt il vous faut naviguer dans l'arborescence du projet. Les fichiers que vous devez modifier sont dans `students -> src -> main -> java` (il faut cliquer sur les triangles pour déplier l'arborescence).



1. Pour compiler et exécuter le programme, il faut passer par l'onglet gradle à droite et cliquer deux fois sur `students -> Tasks -> verification -> test`. Cela va compiler et exécuter les tests. Pour le moment, les tests ne passeront pas car certaines classes sont incomplètes.



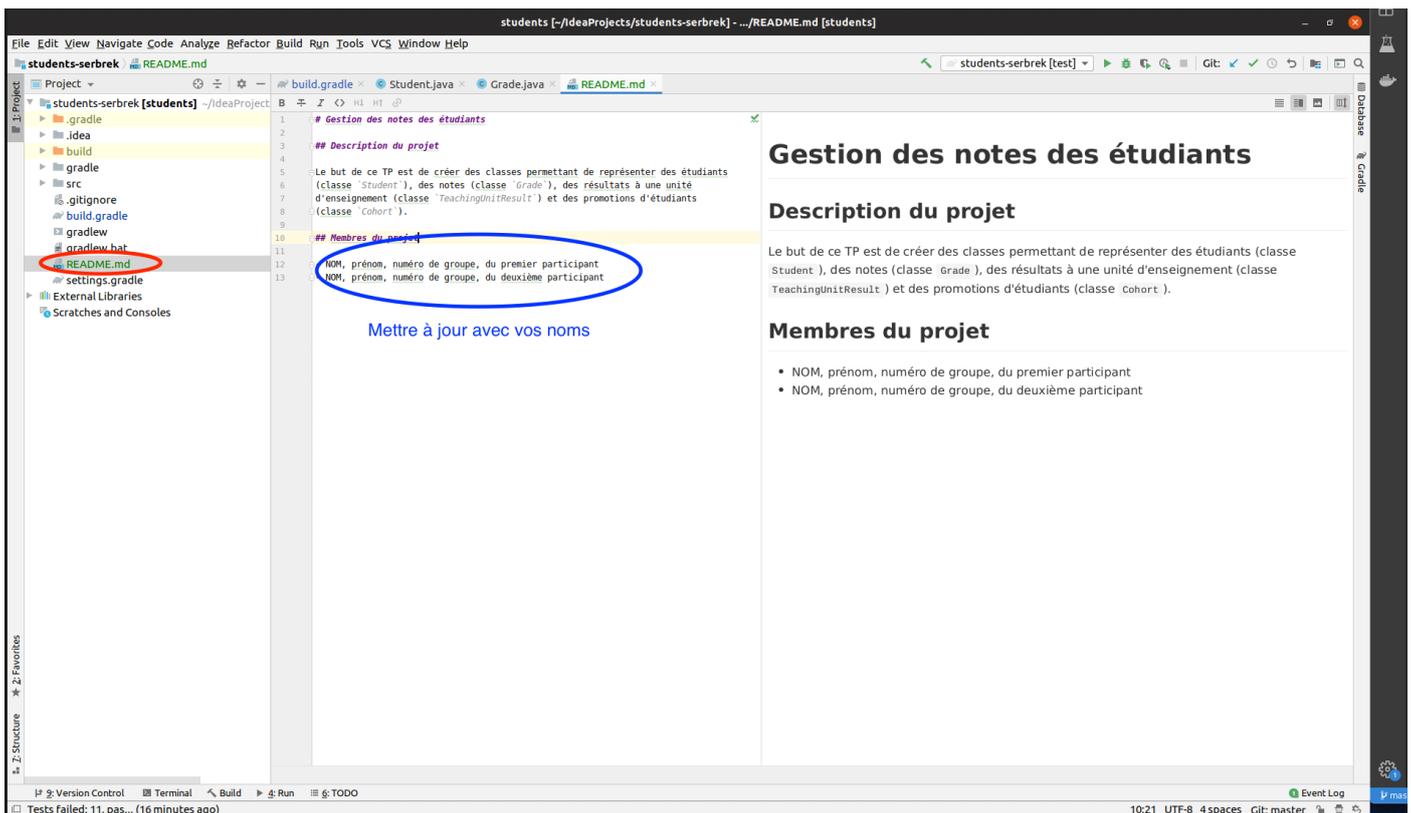
Git

Principe de git

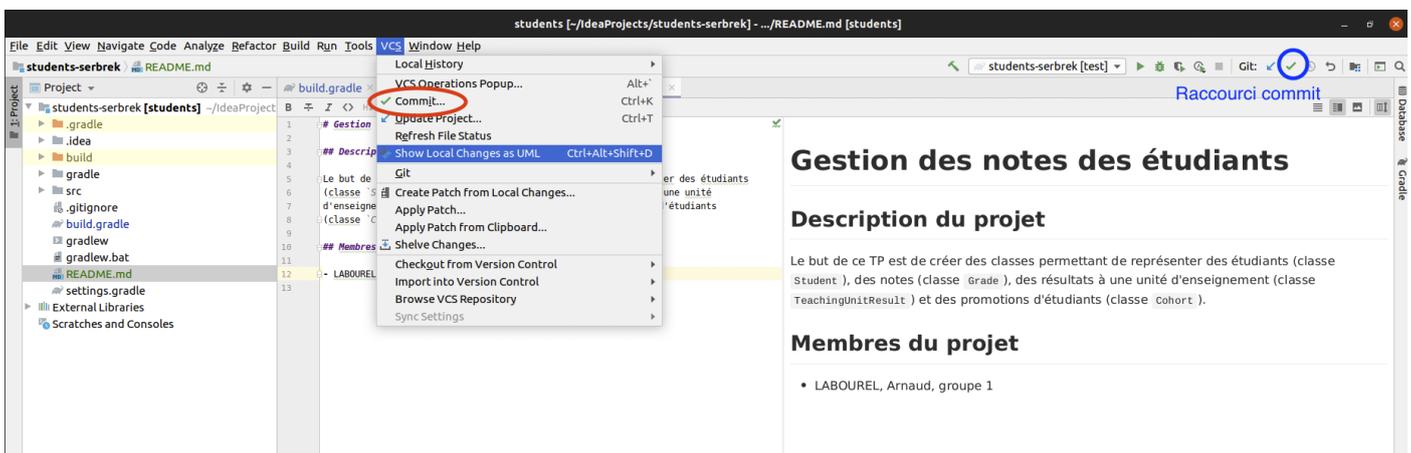
Le principe de git est d'avoir un *dépôt* distant : une version de votre projet stockée sur un serveur accessible par Internet (en l'occurrence hébergé par github). Vous disposez en plus de dépôts locaux sur les ordinateurs sur lesquels vous travaillez. Vous faites vos modifications sur votre ordinateur, et lorsque vous avez accompli une amélioration qui fonctionne bien, vous pouvez la faire enregistrer (commit) par git. Ces enregistrements sont locaux à votre ordinateur, et vous pouvez en faire autant que vous le souhaitez. Si vous voulez partager votre travail avec votre équipe, il vous faut l'envoyer vers le dépôt distant (push). À l'inverse, si vous souhaitez récupérer le travail fait par vos coéquipiers, il faut ramener ces modifications depuis le dépôt distant (pull). IntelliJ est capable de gérer git; vous trouverez dans le menu VCS l'option Commit, et l'option Git qui contient push et pull.

Première modification de votre dépôt

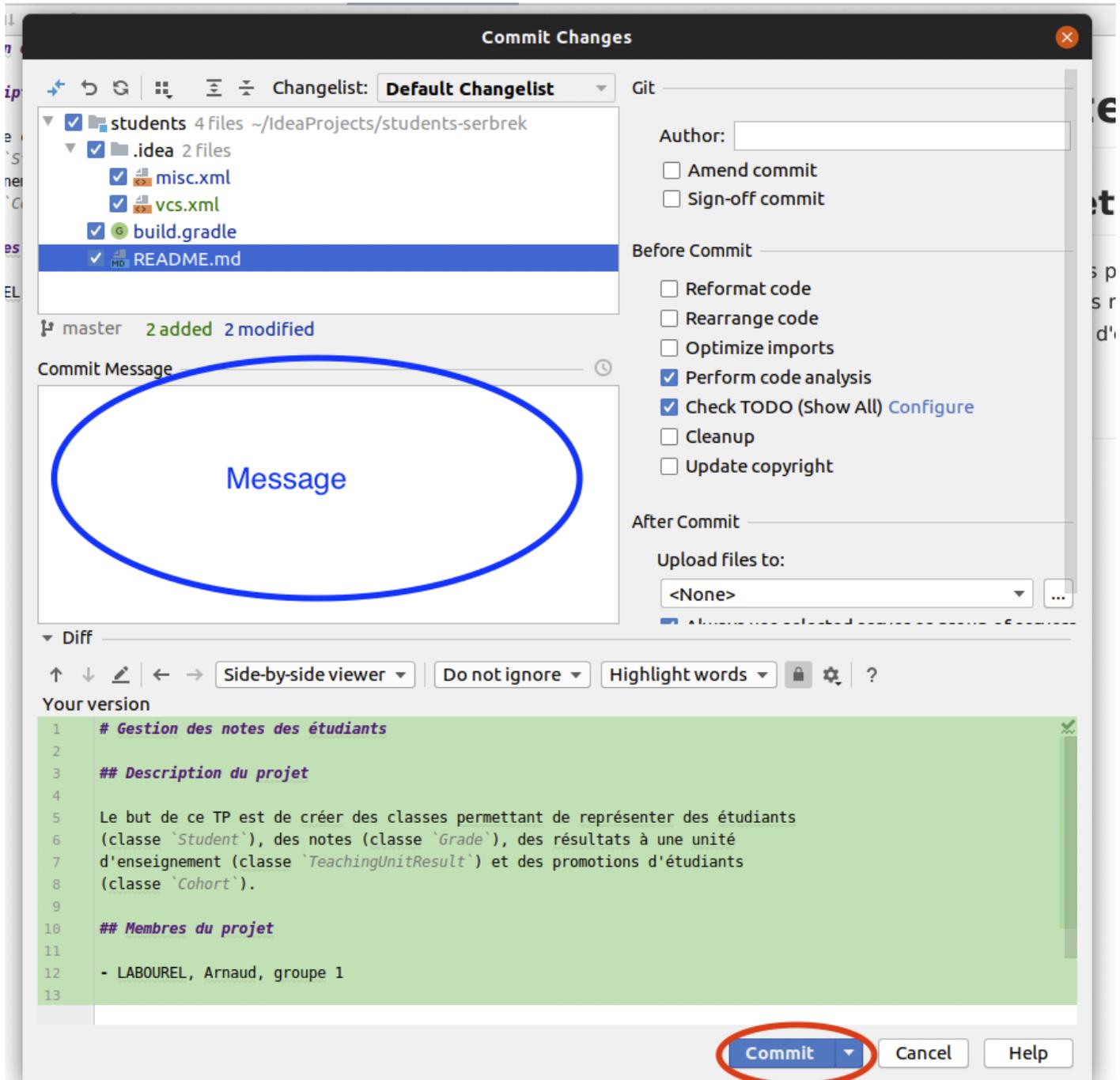
1. Modifiez le fichier README.md. Mettez votre nom.



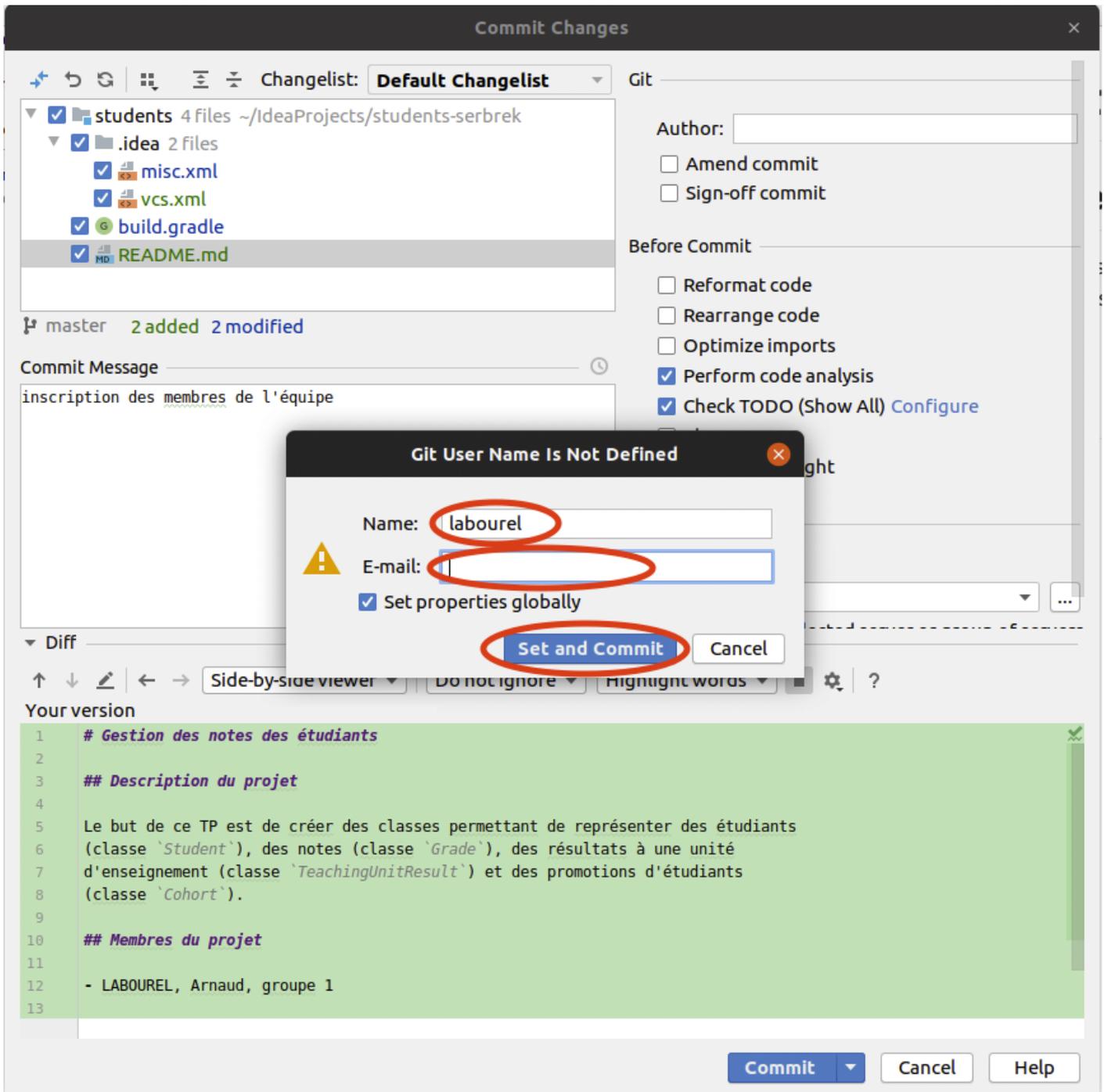
1. Vous allez maintenant mettre à jour votre dépôt local (celui sur votre machine) en effectuant un **commit**. Pour cela vous pouvez aller dans le menu **Git -> Commit** ou bien cliquer sur le raccourci en haut à droite de votre fenêtre.



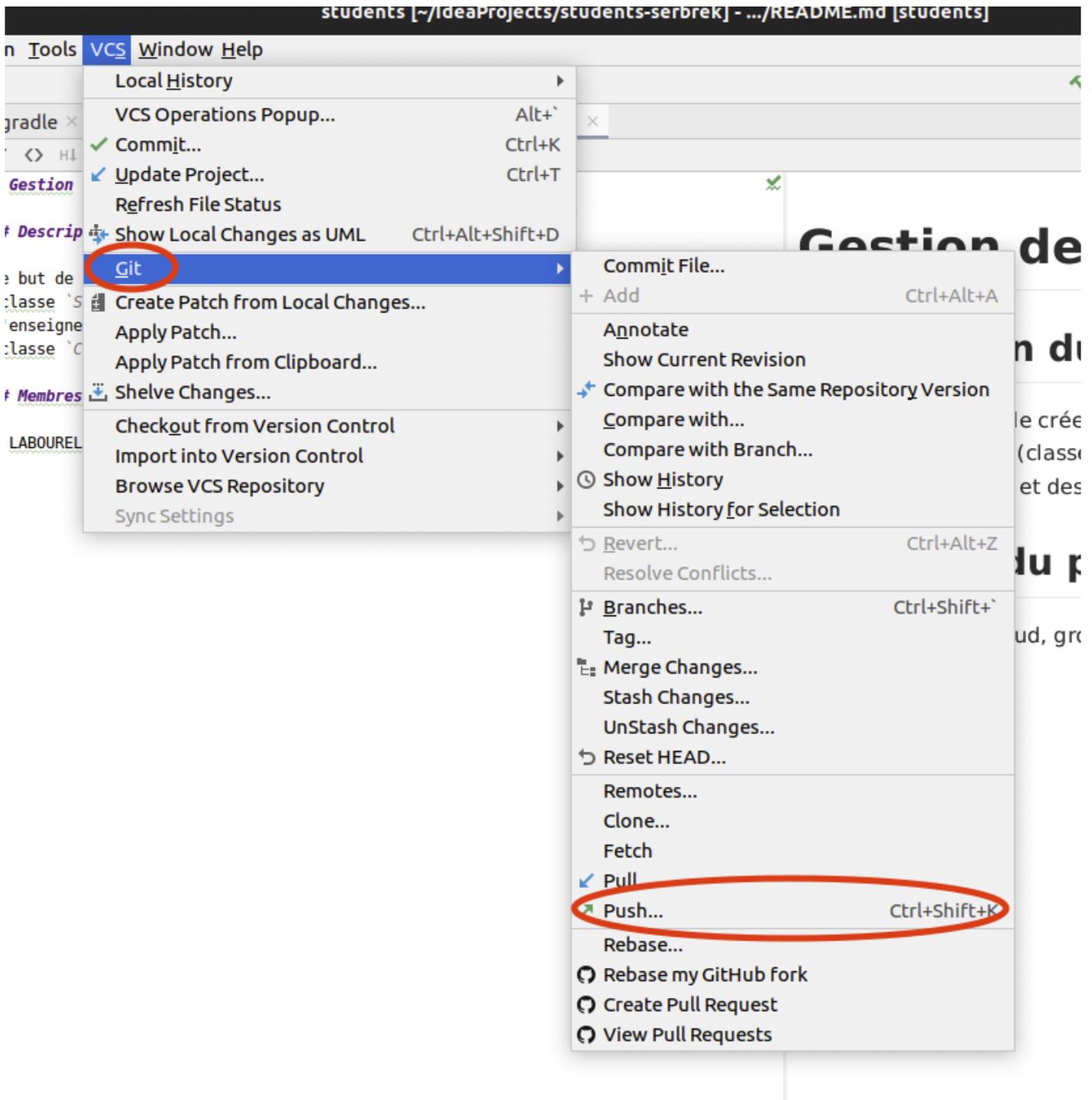
Faites un *commit* avec pour message "inscription des membres de l'équipe" en cliquant sur commit après avoir rempli le champ message.



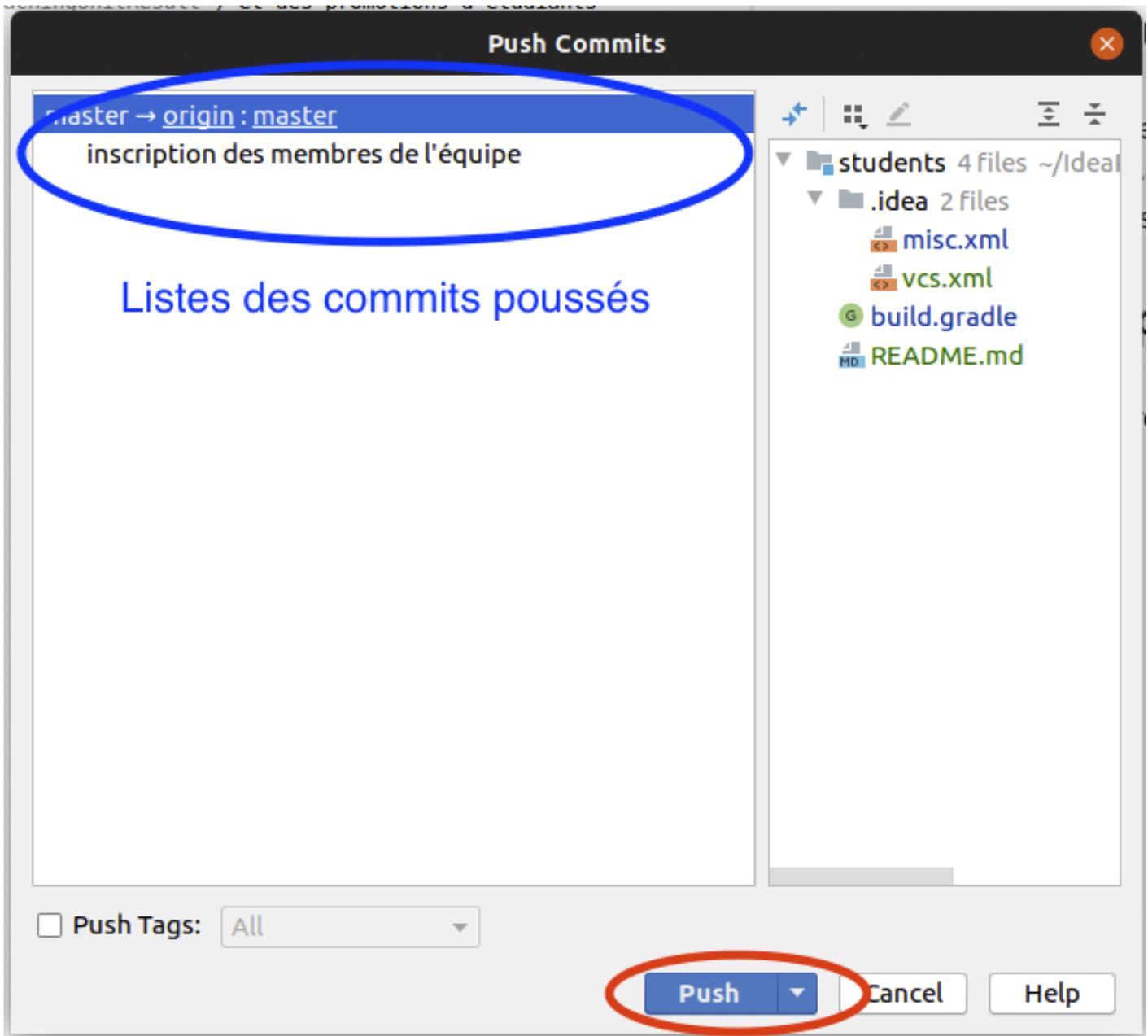
Préciser votre nom et email et cliquez sur le bouton Set and Commit.



1. Vous avez mis à jour votre dépôt local (sur votre machine) mais pas le dépôt distant (celui sur les serveurs de github). Pour cela, il faut faire un push. Allez dans le menu VCS -> git -> push.



Cliquez sur le bouton push de la fenêtre qui vient d'apparaître. Vous pouvez voir la liste des commits que vous vous apprêtez à pousser.



Si tout se passe bien un popup **push successful** devrait apparaître en bas à droite de votre fenêtre.

1. Le projet est configuré pour lancer des tests à chaque fois que vous effectués un *push*. Puisque vous n'avez pas encore modifié le code les tests ne passeront pas et vous devriez recevoir un mail indiquant cet échec.



✖ Pipeline #5463 has failed!

Project LABOUREL Arnaud / student_template

Branch  main

Commit  [8d6241aa](#)
updated tests with jassertj

Commit Author  LABOUREL Arnaud

Pipeline #5463 triggered by  LABOUREL Arnaud
had 1 failed job.

Failed jobs

✖ test

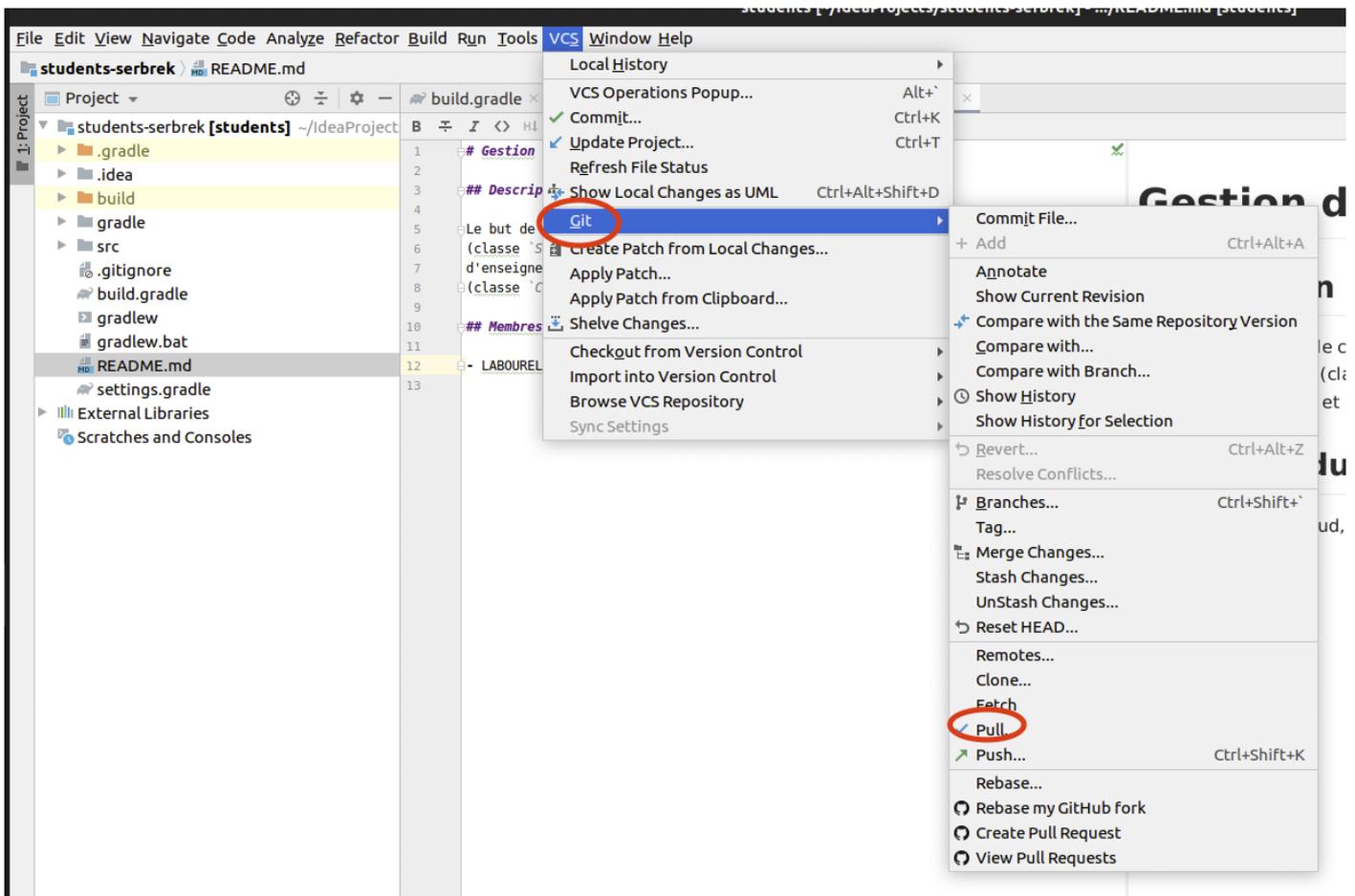
java



GitLab

You're receiving this email because of your account on etulab.univ-amu.fr. [Manage all notifications](#) · [Help](#)

1. Si jamais vous avez besoin de récupérer le projet sur le serveur (par exemple après un *push* depuis un autre ordinateur), il vous suffit de faire un pull. Allez dans le menu `Git` -> `pull`.



Correction du programme

Méthodologie pour le TP

Vous allez maintenant pouvoir attaquer la correction du programme java du dépôt. Pour cela vous allez devoir respecter les consignes suivantes :

- À chaque modification de programme, faites un **commit** en sélectionnant les fichiers modifiés. Chaque **commit** doit contenir un message précisant la nature des modifications effectuées.
- À chaque tâche terminée, faites un **push** de votre travail.
- Ceci est le minimum. Vous pouvez faire plus de **commit** et plus de **push**, ainsi que des **pull** si vous avez travaillé avec un autre ordinateur.
- Si vous avez un problème et souhaitez l'aide de votre instructeur en dehors des séances, un **push** lui permet de voir votre programme.

Classe Grade

[Lien documentation de la classe](#)

Cette classe va permettre de représenter une note obtenue par un étudiant. Une note est une valeur flottante

comprise entre 0 et 20.

Cette classe contient les éléments suivants qui sont corrects :

- `private static final int MAXIMUM_GRADE` : un attribut statique représentant la valeur de la note maximale qui est égal à 20.
- `private final double value` : la valeur de la note comprise entre 0 et `MAXIMUM_GRADE`.
- `public Grade(double value)` : constructeur évident.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux notes.

Tâche 1 : Votre but est de **compléter** les instructions des méthodes suivantes :

- `double getValue()` : retourne la valeur (`value`) de la note.
- `String toString()` : retourne une représentation de la note sous forme de chaîne de caractères. Pour une note ayant une valeur 12, cette méthode devra retourner la chaîne de caractères : "12.0/20".
- `static Grade averageGrade(List<Grade> grades)` : calcule et renvoie la moyenne d'une liste de notes.

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString`, `testGetValue` et `testAverageGrade` de `TestGrade` avec succès. Pour cela, il vous faut supprimer les annotations `@Disabled()` présentes avant chaque méthode de test afin que les tests soient exécutés par la commande `gradle test`. Le dépôt que vous avez forké est configuré pour lancer des tests à chaque mise à jour de celui-ci sur le serveur. Assurez-vous donc que votre code passe tous les tests avant de le pousser sur `etulab` car dans le cas contraire, vous recevrez un mail indiquant que la *pipeline* a échoué.

Classe `TeachingUnitResult`

[Lien documentation de la classe](#)

Cette classe va nous permettre de représenter un résultat obtenu par un étudiant, c'est-à-dire une note associée à une Unité d'Enseignement (UE).

Cette classe contient les éléments suivants qui sont corrects :

- `private final String teachingUnitName` : le nom de l'unité d'enseignement du résultat.
- `private final Grade grade` : la note du résultat.
- `public TeachingUnitResult(String teachingUnitName, Grade grade)` : constructeur évident.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux résultats.

Tâche 2 : Votre but est de **compléter** les instructions des méthodes suivantes :

- `Grade getGrade()` : retourne la note associée au résultat.
- `String toString()` : renvoie le nom de l'unité d'enseignement suivi de " : " suivi de la représentation en chaîne de caractère de la note. Par exemple, un résultat d'une UE de Programmation 2 avec une note de 20 devra renvoyer la chaîne de caractères suivante : "Programmation 2 : 20.0/20".

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString` et `testGetGrade` de `TestTeachingUnitResult` avec succès. Pour cela, il vous faut supprimer les annotations `@Disabled()` présentes avant chaque méthode de test afin que les tests soient exécutés par la commande `gradle test`.

Classe Student

[Lien documentation de la classe](#)

Cette classe va nous permettre de représenter un étudiant.

Cette classe contient les éléments suivants qui sont corrects :

- `private final String firstName` : le prénom de l'étudiant.
- `private final String lastName` : le nom de famille de l'étudiant.
- `private final List<TeachingUnitResult> results` : les résultats de l'étudiant.
- `public Student(String firstName, String lastName)` : constructeur initialisant le nom et prénom de l'étudiant avec les valeurs données et créant une liste vide pour les résultats.
- `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux étudiants.

Tâche 3 : Votre but est de compléter les instructions des méthodes suivantes :

- `void addResult(String teachingUnitName, Grade grade)` : ajoute un nouveau résultat à partir du nom de l'UE et d'une note.
- `List<Grade> getGrades()` : renvoie la liste des notes associées aux résultats de l'étudiant.
- `String toString()` : renvoie le nom de l'étudiant, c'est-à-dire son prénom, suivi d'un espace, suivi de son nom.
- `Grade averageGrade()` : renvoie la moyenne des notes associées aux résultats de l'étudiant.
- `void printResults()` : affiche les résultats de l'étudiant en sortie standard.

Un étudiant nommé Arnaud Labourel et ayant 20 en Programmation 2 et 20 en structures discrètes devra produire l'affichage suivant avec un appel à `printResults()` (avec un saut de ligne à la fin) :

```
Arnaud Labourel
Programmation 2 : 20.0/20
Structures discrètes : 20.0/20
Note moyenne : 20.0/20
```

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests `testToString`, `testGetGrades`, `testGetAverageGrade` et `testPrintResults` de `TestTeachingUnitResult` avec succès. Pour cela, il vous faut supprimer les annotations `@Disabled()` présentes avant chaque méthode de test afin que les tests soient exécutés par la commande `gradle test`.

Classe Cohort

[Lien documentation de la classe](#)

Cette classe va nous permettre de représenter une promotion d'étudiants. La classe `Cohort` contiendra les attributs, méthodes et constructeurs suivants :

Cette classe contient les éléments suivants qui sont corrects :

- `private final String name` : le nom de la promotion
- `private final List<Student> students` : les étudiants de la promotion
- `public Cohort(String name)` : constructeur à partir du nom de la promotion et initialisant à vide la liste des étudiants

Tâche 4 : Votre but est de compléter les instructions des méthodes suivantes :

- `public void addStudent(Student student)` : ajoute un étudiant à la promotion.
- `public List<Student> getStudents()` : renvoie la liste des étudiants de la promotion.
- `String toString()` : retourne une représentation de la promotion correspondant à son nom.
- `public void printStudentsResults()` : affiche les résultats de l'étudiant en sortie standard.

Une promotion ayant pour nom L2 informatique et deux étudiants devra produire un affichage similaire à l'affichage suivant (avec un saut de ligne à la fin) :

```
L2 informatique

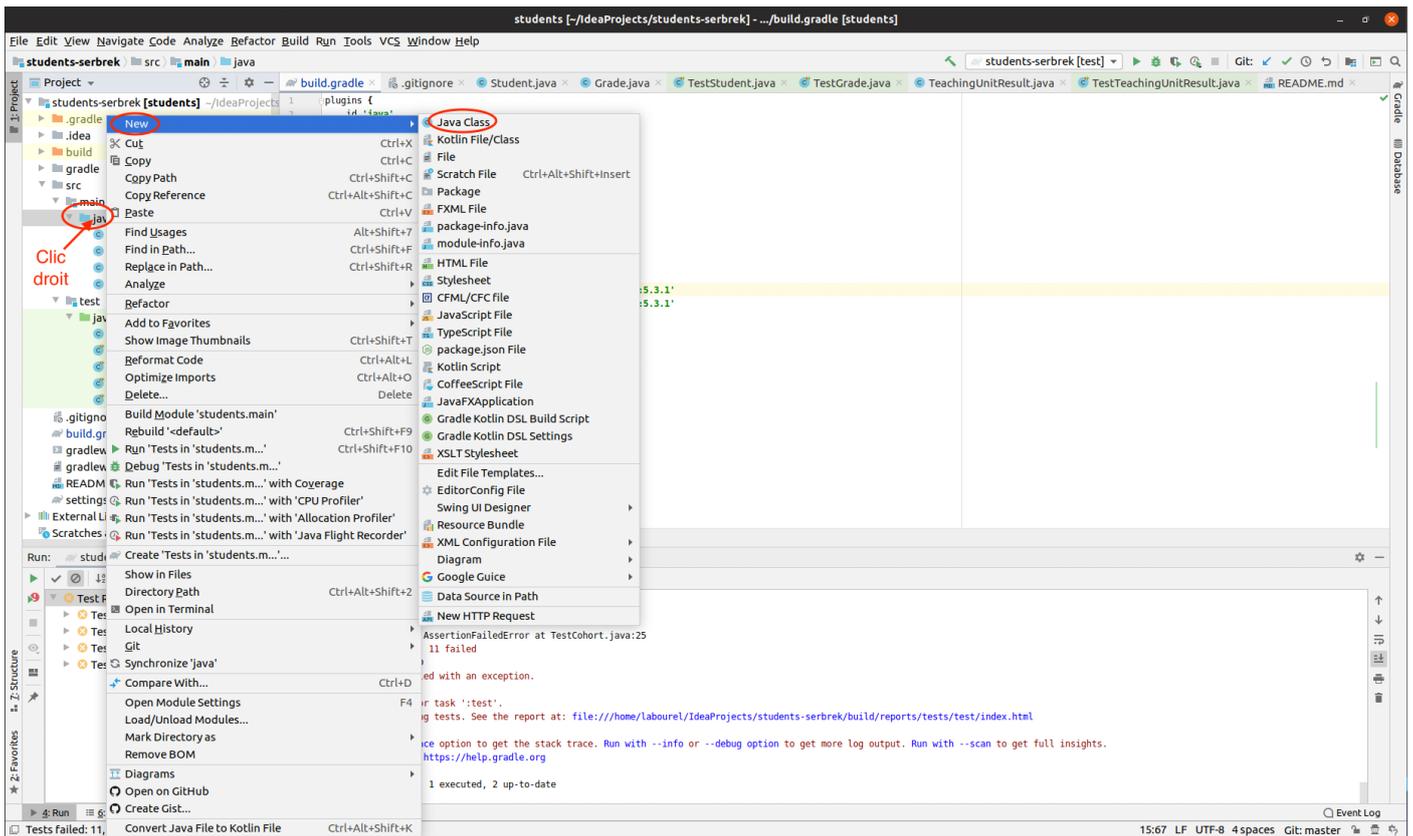
Jean-Michel Bruitage
Programmation 2 : 20.0/20
Structures discrètes : 20.0/20
Note moyenne : 20.0/20

David Goodenough
Programmation 2 : 0.0/20
Structures discrètes : 0.0/20
Note moyenne : 0.0/20
```

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests présents dans la classe `CohortTest`. Pour cela, il vous faut supprimer les annotations `@Disabled()` présentes avant chaque méthode de test afin que les tests soient exécutés par la commande `gradle test`.

Classe Main

Vous allez maintenant ajouter une classe `Main` au projet. Pour cela, il vous faut faire un clic droit sur le répertoire `src -> main -> java` et cliquer dans le menu `new -> Java Class`.



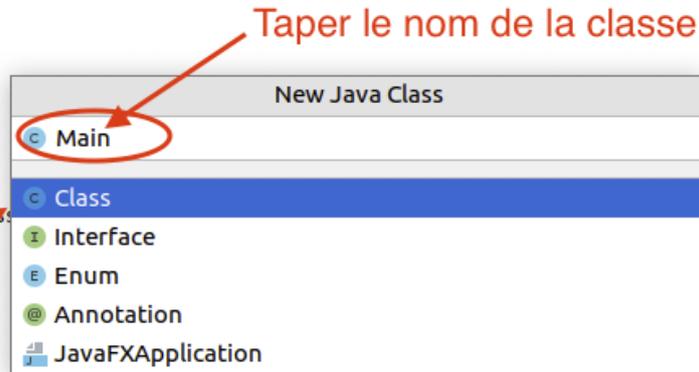
Tapez le nom de la classe, choisissez bien que vous souhaitez créer un classe et validez en appuyant sur entrée.

```
Implementation 'org.junit.jupiter:junit-jupiter-api:5.3.1'
runtimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.3.1'
```

```
Platform()
```

```
public: JavaExec() {
with sourceSets.main.runtimeClasspath {
"Main"
}
```

Choisir Class



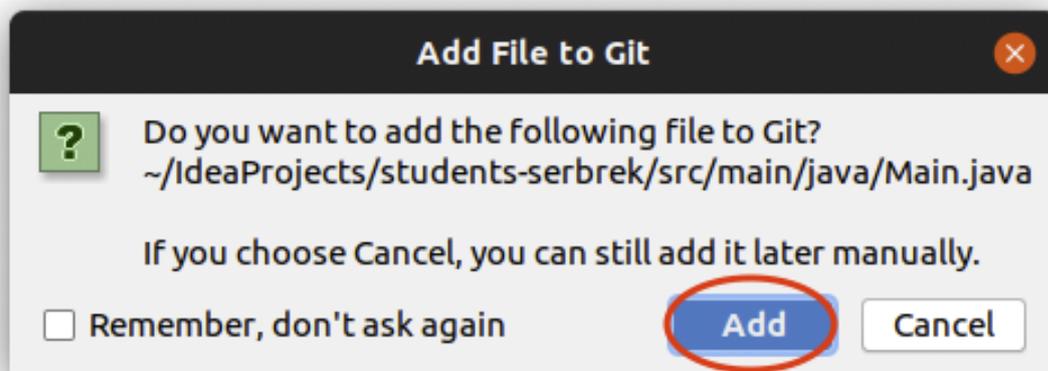
```
ies{}
```

11, passed: 1 of 12 tests – 747 ms

```
base/java.lang.Thread.run(Thread.java:834)
```

```
> testGetStudents() FAILED
ntest4j.AssertionFailedError at TestCohort.java:25
mpleted 11 failed
```

Normalement, on va vous demander si vous souhaitez ajouter ce nouveau fichier au dépôt git. Cliquez sur **add** pour l'ajouter afin qu'il soit mis à jour lors de votre prochain *commit*.

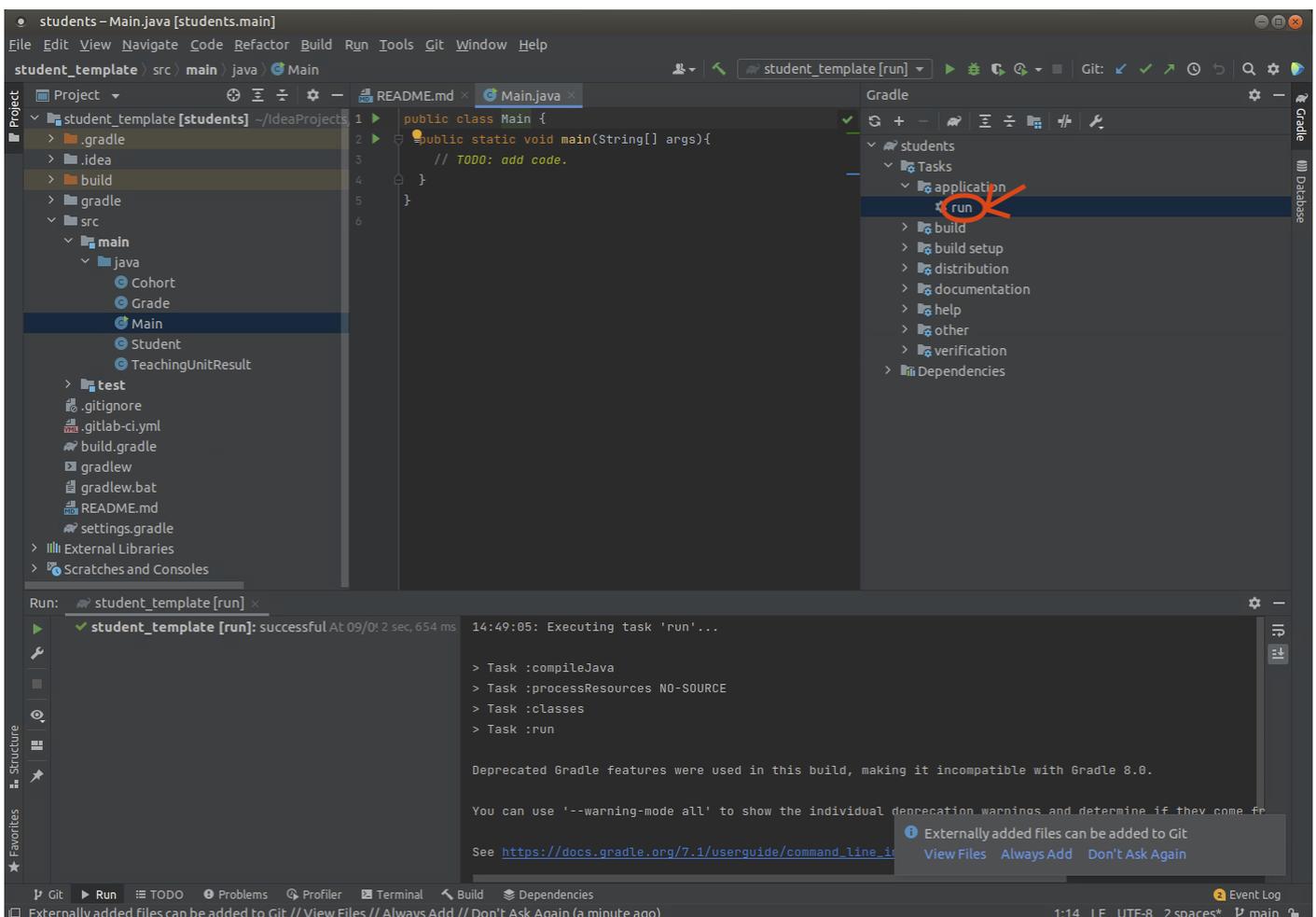


Tâche 5 : Ajoutez dans votre classe Main le code d'une méthode public static void

main(String[] args) qui :

1. crée des instances de **Student** ayant les nom et prénom du membre du projet plus un étudiant fictif,
2. ajoute à ces étudiants les notes en “Programmation 2” et “Structures discrètes” que vous aimeriez avoir,
3. crée une promotion (instance de **Cohort**) ayant nommée “L2 informatique”,
4. ajoute les étudiants créés à la promotion et
5. affiche les résultats de la promotion.

Pour compiler et exécuter le main, il faut passer par l’onglet gradle à droite et cliquer deux fois sur **students** -> **Application** -> **run**. Cela va compiler et exécuter votre méthode **main**.



Tâches optionnelles

Si vous avez fini les tâches précédentes, vous pouvez améliorer votre projet en rajoutant les fonctionnalités suivantes :

- Ajout d'une méthode comptant le nombre d'étudiants ayant validé leur année (moyenne supérieure ou égale à 10) dans une promotion.
- Changement de la classe `Grade` pour qu'elle permette de stocker des notes correspondant à une absence du résultat (affiché `ABS`).
- Calcul du nombre d'absents d'une promotion.
- Calcul de la note maximum et minimum (hors absence) d'une promotion.
- Création d'une classe `TeachingUnit` qui permet d'associer des crédits à une UE.
- Calcul pondéré de la moyenne de résultats en fonction du nombre de crédits des UE.
- Calcul de la moyenne (hors absence) d'une promotion.