

Git

“Version Control”

Why Version Control ?

- Keep track of changes made to files
- Automated backups
- Ability to rollback to any previous state (permanently or temporary)
- Allowing multiple contributors for the same set of files at the same time
- Maintain different versions of the same project simultaneously
- Share an issue with other developers

Local Computer

Checkout

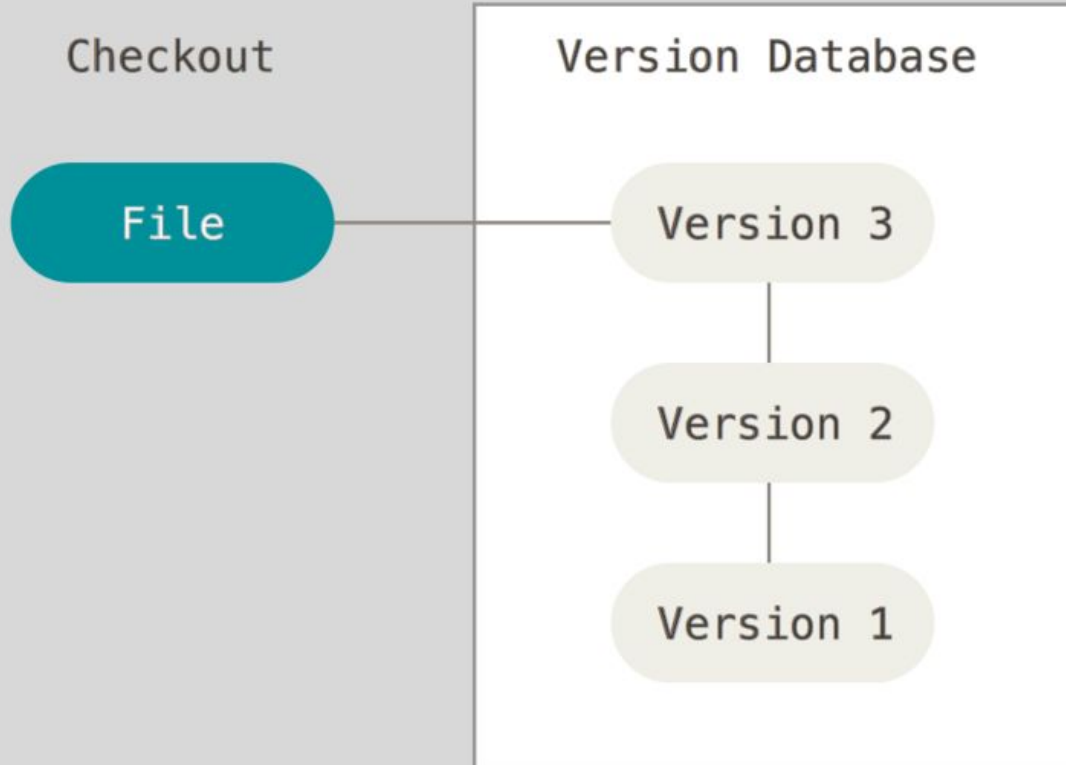
File

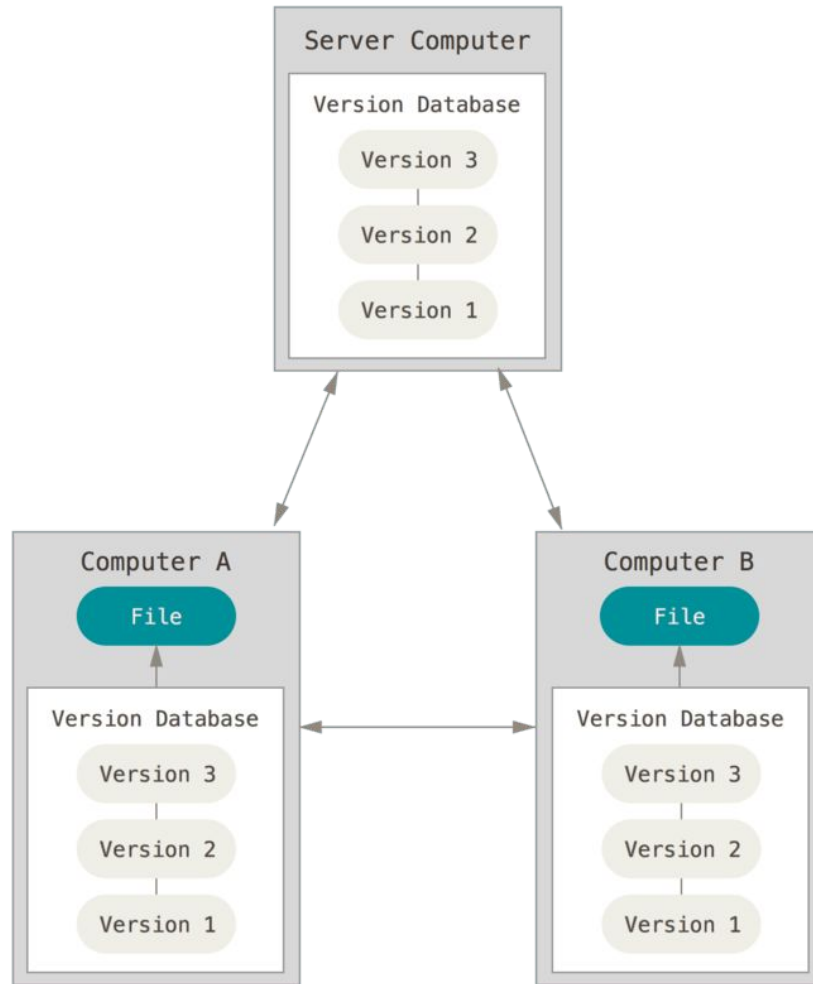
Version Database

Version 3

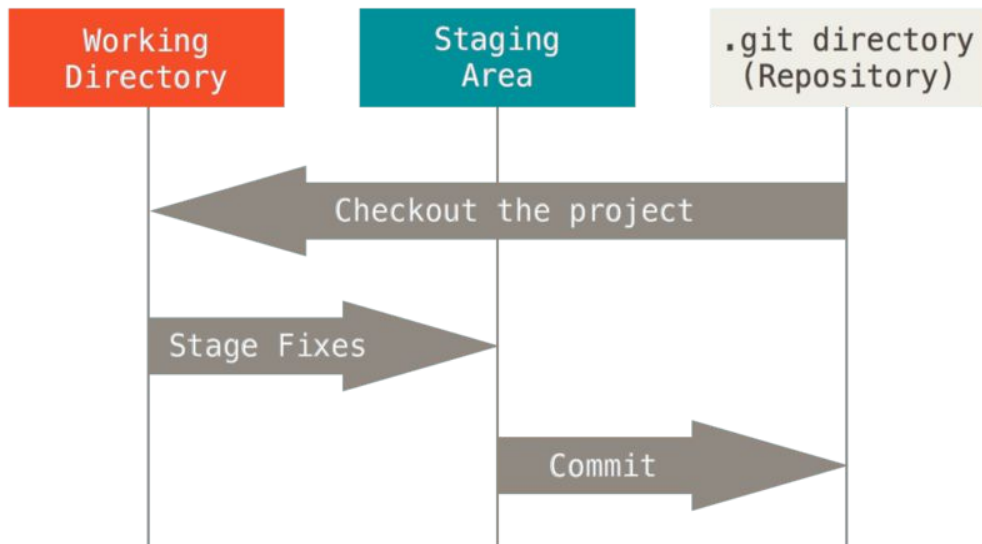
Version 2

Version 1





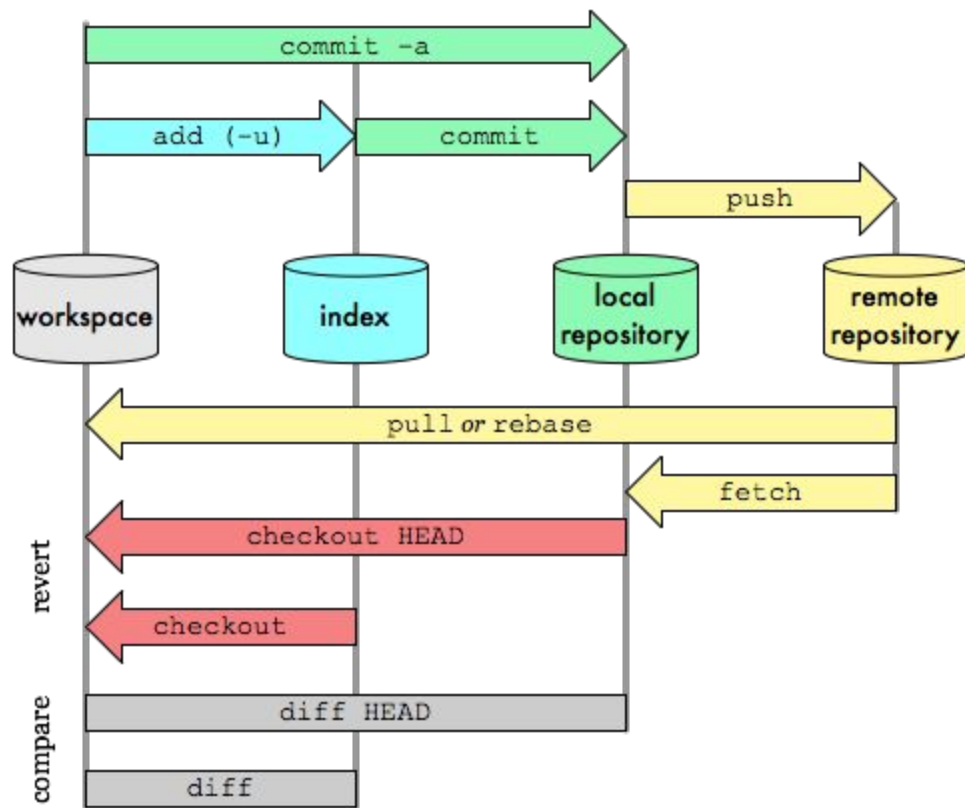
The Local Process



1. You modify files in your working directory.
2. You stage the files, adding snapshots of them to your staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

Git Data Transport Commands

<http://osteele.com>



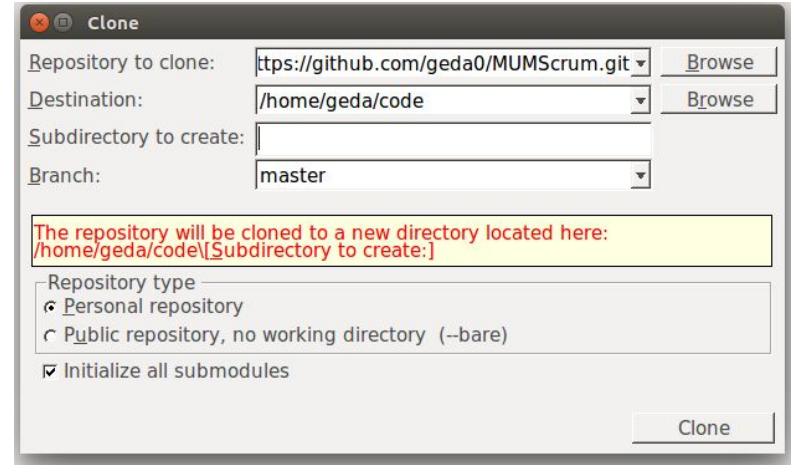
Getting Started

- GitExtensions (or any git client)
- <https://gitextensions.github.io/>

- KDiff3 (very useful for merging files)
- <https://sourceforge.net/projects/kdiff3/files/kdiff3/0.9.98/>
 - If your git client doesn't have a build-in diff tool

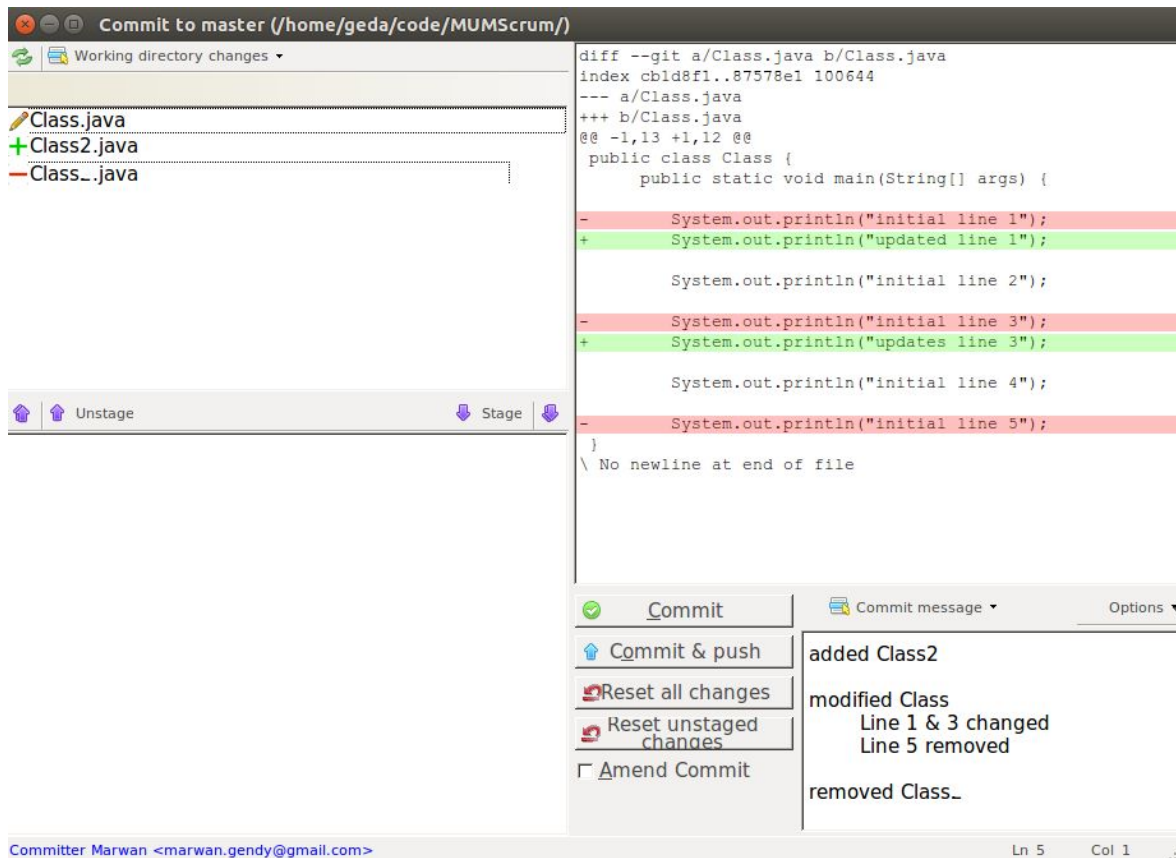
Getting Started - the easiest method

1. Create a new repository on github.com (or any other place)
2. Clone the repository to your computer
3. Start your project in the repository folder, or add your existing files (it is safe to move the repository folder as a whole within your computer)



Commit

1. Check every file and its contents
2. Stage files
3. Write commit message
4. Commit
5. Push (optional)



Conflicts - (mergetool & difftool)

A (Base): geda/code/MUMScrum2/Class.java.BASE Encoding: UTF-8 Line end style: Unix

```
public class Class {  
    ....public static void main(String[] args) {  
        ....System.out.println("updated.line.1");  
        ....System.out.println("initial.line.2");  
        ....System.out.println("updates.line.3");  
        .....System.out.println("initial.line.4");  
    }  
}
```

B: /home/geda/code/MUMScrum2/Class.java.LOCAL Encoding: UTF-8 Line end style: Unix

```
public class Class {  
    ....public static void main(String[] args) {  
        ....System.out.println("updated.line.1");  
        ....System.out.println("initial.line.2");  
        ....System.out.println("updates.line.3");  
        .....System.out.println("line-conflict-in.line.4.from.dev.2");  
        .....System.out.println("added.line.5.dev2");  
    }  
}
```

C: /home/geda/code/MUMScrum2/Class.java.REMOTE Encoding: UTF-8 Line end style: Unix

```
public class Class {  
    ....public static void main(String[] args) {  
        ....System.out.println("updated.line.1");  
        ....System.out.println("initial.line.2");  
        ....System.out.println("updates.line.3");  
        .....System.out.println("dev.1.changes.line.4");  
        .....System.out.println("added.a.different.line.5.from.dev1");  
    }  
}
```

Output: /home/geda/code/MUMScrum2/Class.java Encoding for saving: Codec from C: UTF-8 Line end style: Unix (A, B, C)

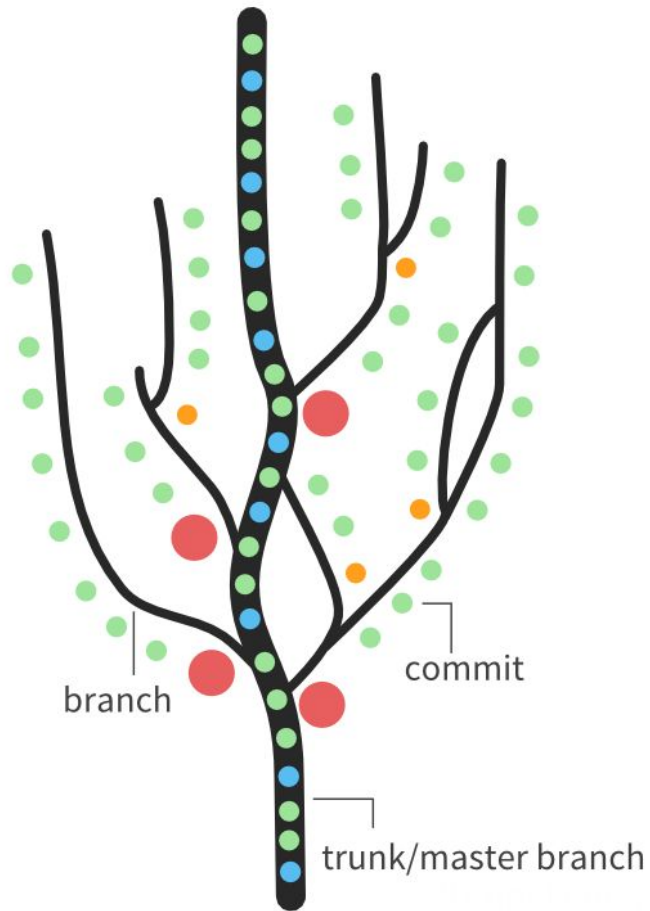
```
public class Class {  
    ....public static void main(String[] args) {  
        ....System.out.println("updated.line.1");  
        ....System.out.println("initial.line.2");  
        ....System.out.println("updates.line.3");  
        .....System.out.println("initial.line.4");  
        .....System.out.println("line-conflict-in.line.4.from.dev.2");  
        .....System.out.println("added.line.5.dev2");  
    }  
}
```

? <Merge Conflict>
C
? <Merge Conflict>
}

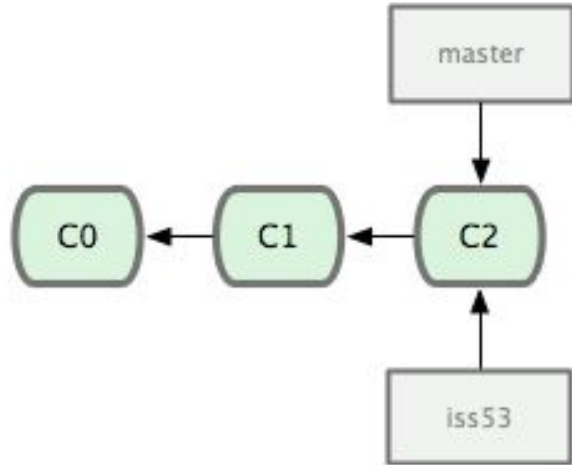
.gitignore

- Contains files to be ignored while committing your work
- Temp files, Executables, etc..
- Examples
 - *.class
 - settings/
- .gitignore is a file like any other file, you need to commit it, and push it for others to use

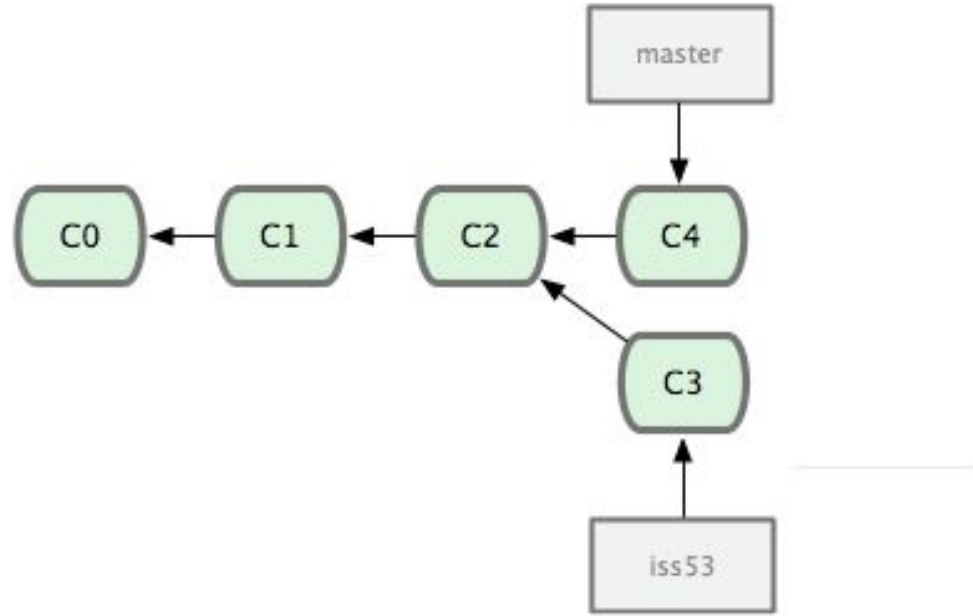
Branches



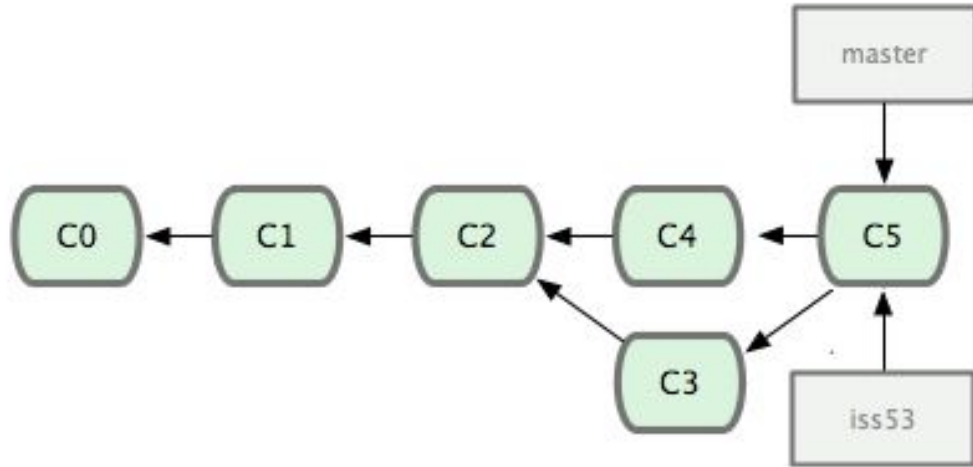
Creating a Branch



Committing and Pushing to a Branch



Merging a Branch



Before commit

- Check where you stand

Before Push and Pull

- Check your current branch

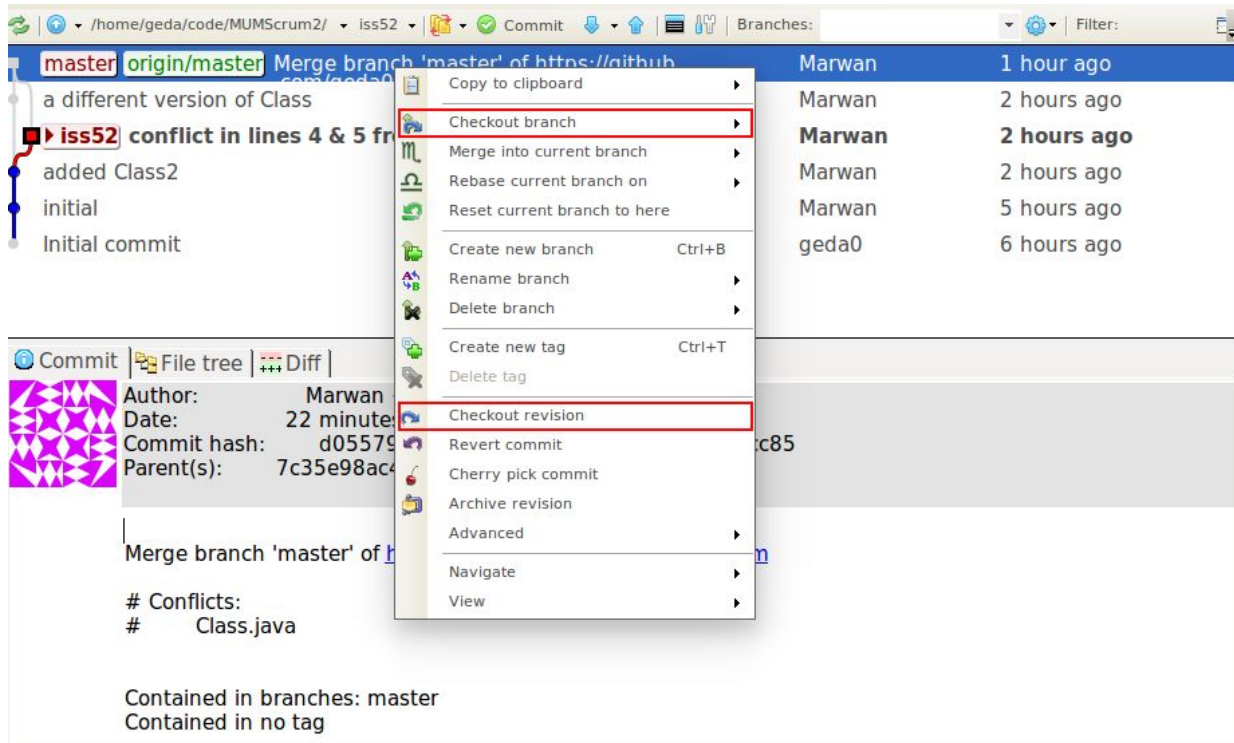
The screenshot shows the VS Code interface with a merge conflict. The top menu bar includes: Start, Repository, Navigate, View, Commands, Github, Plugins, Tools, Help. The toolbar shows icons for commit, pull, push, and branches. The main area displays a merge conflict between 'master' and 'origin/master' branches. The conflict is in the 'Class' file, lines 4 & 5. The commit list on the left shows 'initial' and 'Initial commit'.

Branch	Author	Time
master	Marwan	22 minutes ago
origin/master	Marwan	1 hour ago
iss52	Marwan	1 hour ago
added Class2	Marwan	1 hour ago
initial	Marwan	4 hours ago
Initial commit	geda0	5 hours ago

Checkout

To switch between branches we use “checkout”

Checking out a branch sets it branch as your current branch, you are now working on this branch now



Merging Branches - best practices

- To merge a Branch “test” into “master” branch
- Checkout master
- Pull (to update your local “master” branch)
- Pull from “test” (should merge the two branches locally)
- If everything works well, Push!

Reference

<https://git-scm.com/book/en/v2>