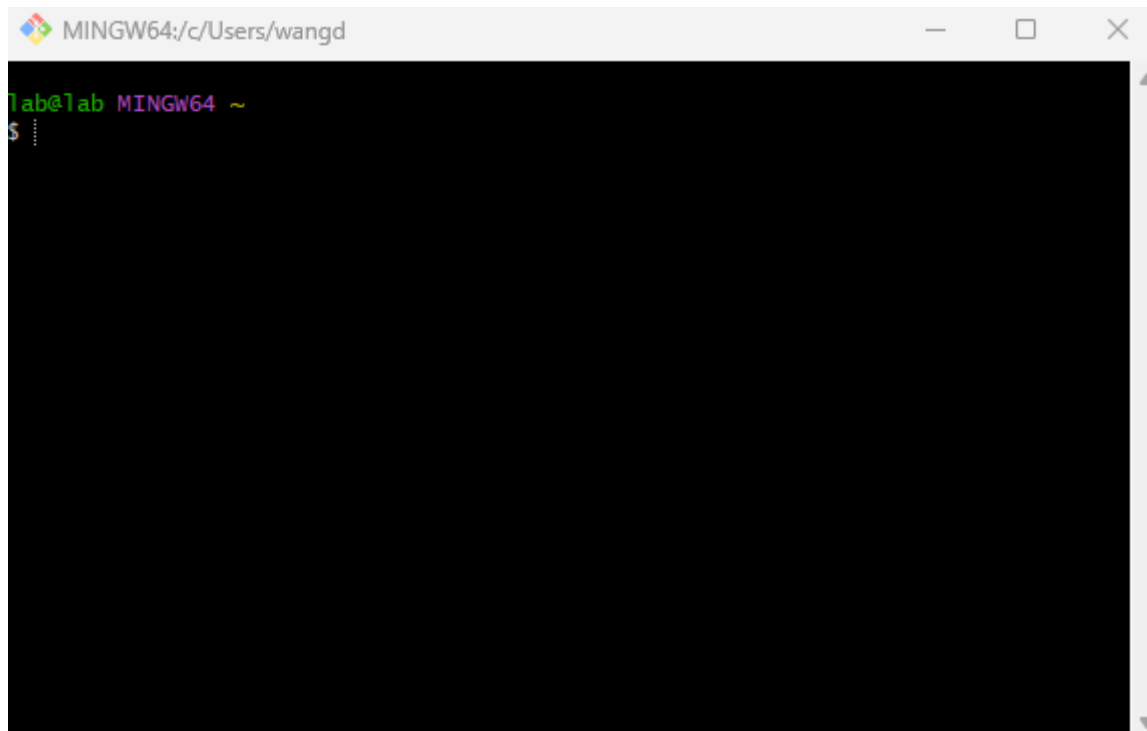


# Introduction to Git

This lab tutorial is about how to use git to manage our projects.

## Open a terminal

After successfully installing git from <https://git-scm.com>, you should be able to start a "git bash" from your start up menu if you are using windows:



If you are using linux then you only need to open a terminal.

## Create a project

Now create a project with `git init`:

```
git init
```

After that the ".git" directory should be created:

```
MINGW64:/c/Users/wangd/Desktop/hello
lab@lab MINGW64 ~
$ cd Desktop && mkdir hello && cd hello

lab@lab MINGW64 ~/Desktop/hello
$ git init
Initialized empty Git repository in C:/Users/wangd/Desktop/hello/.git/

lab@lab MINGW64 ~/Desktop/hello (master)
$ ls

lab@lab MINGW64 ~/Desktop/hello (master)
$ ls -al
total 8
drwxr-xr-x 1 lab 197121 0 Feb  7 11:01 ./
drwxr-xr-x 1 lab 197121 0 Feb  7 11:01 ../
drwxr-xr-x 1 lab 197121 0 Feb  7 11:01 .git/
```

The ".git" directory shows that this directory is a git project.

## Add and commit

Let's first create a new file in the project:

```
lab@lab MINGW64 ~/Desktop/hello (master)
$ ls -al
total 9
drwxr-xr-x 1 lab 197121  0 Feb  7 14:25 ./
drwxr-xr-x 1 lab 197121  0 Feb  7 11:01 ../
drwxr-xr-x 1 lab 197121  0 Feb  7 11:01 .git/
-rw-r--r-- 1 lab 197121 116 Feb  7 14:30 Hello.java

lab@lab MINGW64 ~/Desktop/hello (master)
$ cat Hello.java
public class Hello {
    public static void main( String[] args ) {
        System.out.println("Hello");
    }
}
```

Then you need to take a snapshot to the staging area:

```
git add .
```

You can use `git diff --cached` to see what is added to your project:

```
MINGW64:/c/Users/wangd/Desktop/hello

lab@lab MINGW64 ~/Desktop/hello (main)
$ git add .

lab@lab MINGW64 ~/Desktop/hello (main)
$ git diff --cached
diff --git a/Hello.java b/Hello.java
new file mode 100644
index 0000000..f027015
--- /dev/null
+++ b/Hello.java
@@ -0,0 +1,5 @@
+public class Hello {
+    public static void main( String[] args ) {
+        System.out.println("Hello");
+    }
+}
\ No newline at end of file
```

Note that the `git diff` command with or without the `--cached` argument is different. See git document <https://git-scm.com/docs/git-diff> for more details.

This is not over, you must commit the changes to finish this step with:

```
git commit
```

```
lab@lab MINGW64 ~/Desktop/hello (main)
$ git commit
[main (root-commit) 2973aed] Create Hello.java
1 file changed, 5 insertions(+)
create mode 100644 Hello.java
```

The terminal will prompt for you to enter the "commit message", it is better if you know how to use "vim".

You can also specify the commit message when doing the commit, so that you don't need to type the commit message in the next step:

```
MINGW64:/c/Users/wangd/Desktop/hello

lab@lab MINGW64 ~/Desktop/hello (main)
$ git add .

lab@lab MINGW64 ~/Desktop/hello (main)
$ git commit -m "modified Hello.java"
[main f649003] modified Hello.java
1 file changed, 1 insertion(+), 1 deletion(-)
```

## See log

Now we have done one or two commits. If there are many commits in the repository, you may want to see the log to know the history.

You can choose one of the following command or refer to <https://www.git-scm.com/docs/git-log> for a detailed demonstration.

```
git log
git log -p
git log --stat --summary
```

## Making branches

You can have multiple branches in your project.

```
lab@lab MINGW64 ~/Desktop/hello (main)
$ git branch b1

lab@lab MINGW64 ~/Desktop/hello (main)
$ git branch
b1
* main

lab@lab MINGW64 ~/Desktop/hello (main)
```


In the above example, we created a new branch named "b1" with command `git branch b1`. Then we use `git branch` to show the branches and we see two branches there: "b1" and "main". The "main" branch is the default one. We also see that the current selected branch is "main".

You can switch between branches:

```
lab@lab MINGW64 ~/Desktop/hello (main)
$ git switch b1
Switched to branch 'b1'

lab@lab MINGW64 ~/Desktop/hello (b1)
$ git branch
* b1
  main
```

Now let's do some changes and commit on branch "b1":

 Hello.java - Notepad	lab@lab MINGW64 ~/Desktop/hello (main)
File Edit View	\$ git switch b1
	Switched to branch 'b1'
public class Hello {	lab@lab MINGW64 ~/Desktop/hello (b1)
public static void main( String[] args ) {	\$ git branch
System.out.println("Hello");	* b1
}	main
}	lab@lab MINGW64 ~/Desktop/hello (b1)
// modify on branch b1	\$ git add . && git commit -m "add branch b1 comment in Hello.java"
	[b1 f9e8e10] add branch b1 comment in Hello.java
	1 file changed, 1 insertion(+)
	lab@lab MINGW64 ~/Desktop/hello (b1)

Go back to "main" and make some changes:

```
Hello.java - Notepad
File Edit View

public class Hello {
    public static void main( String[] args ) {
        System.out.println("Hello");
    }
}
// modify on branch main
```

```
lab@lab MINGW64 ~/Desktop/hello (b1)
$ git switch main
Switched to branch 'main'

lab@lab MINGW64 ~/Desktop/hello (main)
$ git add . && git commit -m "add branch main comment in Hello.java"
[main aac76a4] add branch main comment in Hello.java
1 file changed, 1 insertion(+)

lab@lab MINGW64 ~/Desktop/hello (main)
$
```

Merge branch "b1" from branch "main" with `git merge b1` . If there is a conflict, as follows:

```
Hello.java - Notepad
File Edit View

public class Hello {
    public static void main( String[] args ) {
        System.out.println("Hello");
    }
}
<<<<<< HEAD
// modify on branch main
=====
// modify on branch b1
>>>>>> b1
```

```
MINGW64:/c/Users/wangd/Desktop/hello

lab@lab MINGW64 ~/Desktop/hello (main)
$ git switch b1
Switched to branch 'b1'

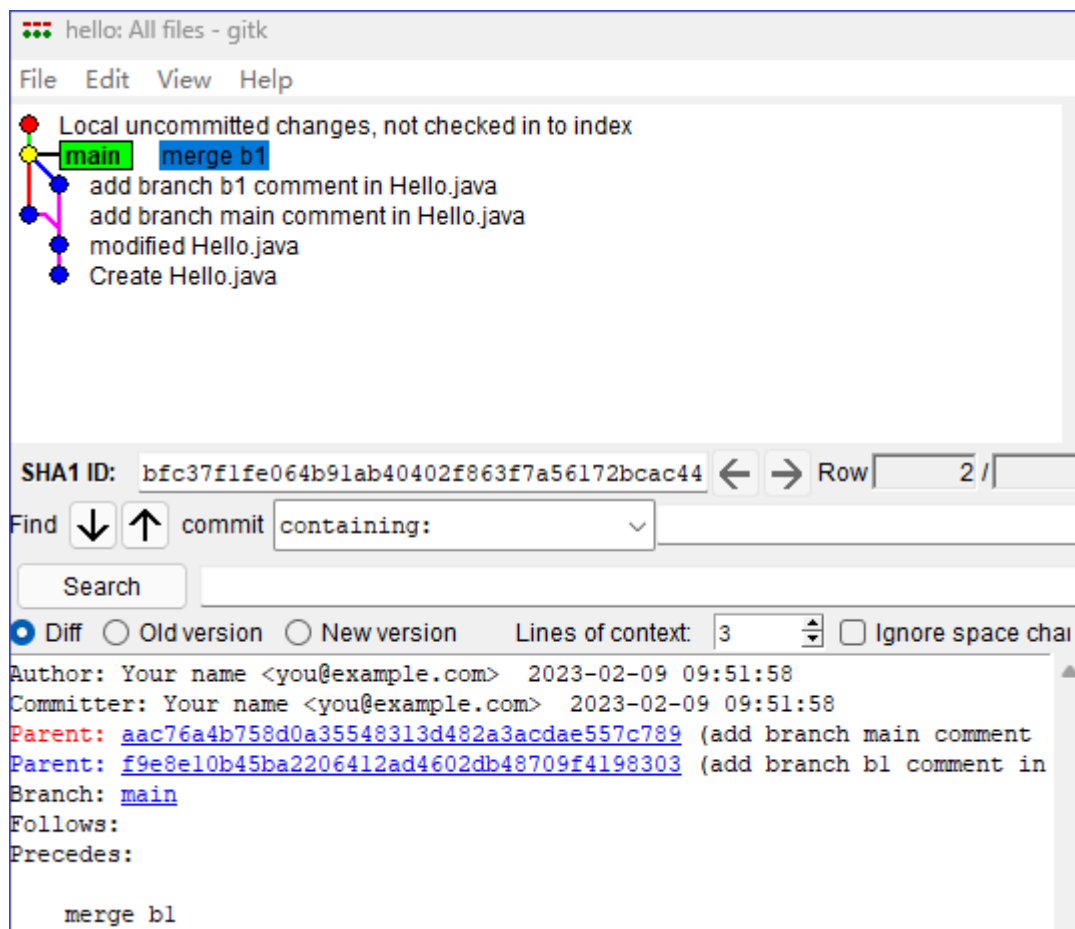
lab@lab MINGW64 ~/Desktop/hello (b1)
$ git switch main
Switched to branch 'main'

lab@lab MINGW64 ~/Desktop/hello (main)
$ git merge b1
Auto-merging Hello.java
CONFLICT (content): Merge conflict in Hello.java
Automatic merge failed; fix conflicts and then commit the result.

lab@lab MINGW64 ~/Desktop/hello (main|MERGING)
$
```

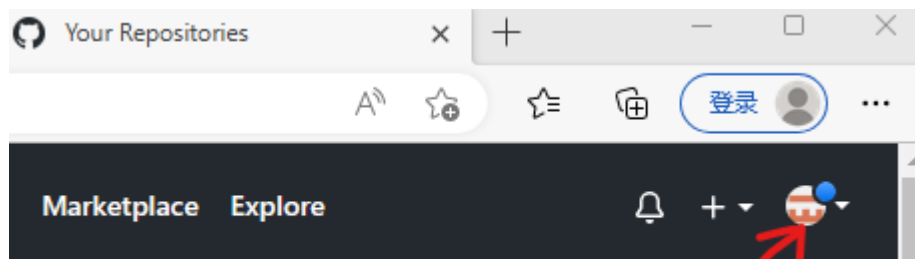
you should manually solve the conflict and do a commit. Then everything is fine and you can remove branch "b1" if it is not useful anymore with `git branch -d b1` . Argument "-d" and "-D" are different, refer to git document for more details.

Now everything is fine and you can execute `gitk` to see a graphical view of the change log:



## Using github

Go to <https://github.com> and register an account if you haven't already done it.



Click the button above, click "repositories", "new" to create a new repository. When github ask you to set the initial configuration, you can choose to add "readme", "gitignore" and "licence" file to you repo:

### Initialize this repository with:

Skip this step if you're importing an existing repository.

#### ☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

#### Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Java ▾

#### Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

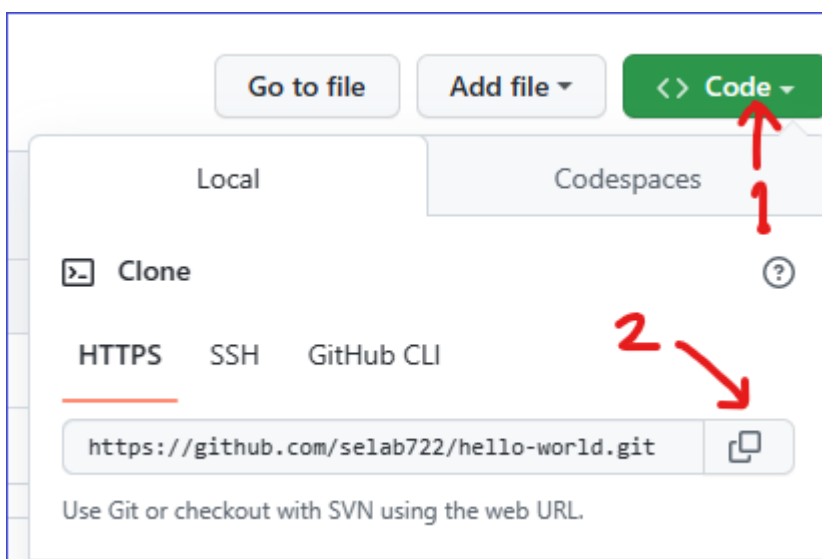
License: MIT License ▾

This will set  `main` as the default branch. Change the default name in your [settings](#).

Click "Create repository" and your repository will be created.

You can directly edit some files using github but sometimes you still want to edit them locally, specially when you need an IDE.

So click "code" and "clone" to copy the link (like <https://github.com/username/hello-world.git>) to the click board:



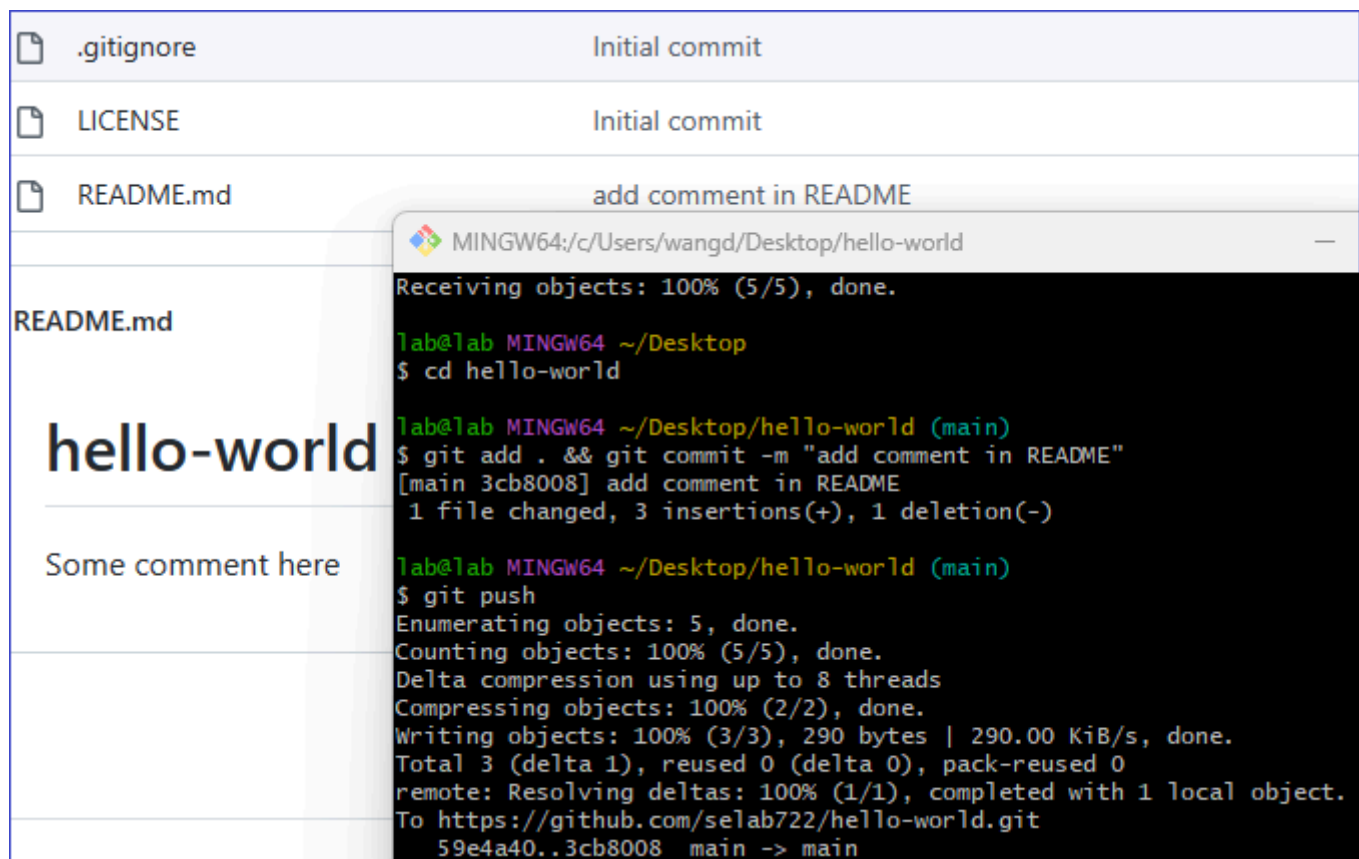
Then run `git clone` locally to clone the repository to your local disk:

```
MINGW64:/c/Users/wangd/Desktop/hello-world

lab@lab MINGW64 ~/Desktop
$ git clone https://github.com/selab722/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

lab@lab MINGW64 ~/Desktop
$ cd hello-world
```

You can now do any modification as a local project (remember to "cd" into the project directory). You can also use `git pull` to "pull" updates from the github repo to your local. You can use `git push` to "push" local updates to the github repo.



## References

- <https://git-scm.com/docs/gittutorial>
- <https://git-scm.com/docs/user-manual>