Minilab 4 - Git

1. Configuration

The first step is to set up git information so people can see who created these changes.

```bash
git config user.name "evan stokdyk"
git config user.email "evan.stokdyk@gmail.com"

git config init.default branch main
```

2. Initialization

Next, you can create the initial repository.

```bash
git init
git status
```

```
Initialized empty Git repository in /home/focus/dev/uni/csci-062/labs/4/tmp/
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt
        file2.txt
        file3.txt
        hello.cpp
        myclass.cpp
        myclass.h

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

3. Adding Files

Files that are staged can be committed with a message.

```bash
git add hello.cpp
git commit -m "First commit"
```

```
[main (root-commit) 50e5bdb] First commit
 1 file changed, 11 insertions(+)
 create mode 100644 hello.cpp
```

It is easy to view the change records.

```bash
git log
```

```
commit 50e5bdb8b35b681f244091ed0b9aa474dc92bba6
Author: evan stokdyk <evan.stokdyk@gmail.com>
Date:   Thu May 30 15:05:19 2024 -0700

    First commit
```

4. Ignoring Files

To ignore files list them in a gitignore file.

```bash
touch .gitignore
git status
```

```
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file1.txt
        file2.txt
        file3.txt
        myclass.cpp
        myclass.h

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

Adding file names in this file causes them to be no longer be shown as unstaged files.

```bash
echo "myclass.h" >> .gitignore
git status
```

```
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file1.txt
        file2.txt
        file3.txt
        myclass.cpp

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

Files can also be ignored using globs. Comments begin with ’#’ and have no effect.

```bash
echo '*.txt' >> .gitignore
echo '# This is a comment' >> .gitignore
git status
```

```
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        myclass.cpp

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

This is what the gitignore file look like at this point.

```bash
cat .gitignore
```

```
myclass.h
*.txt
# This is a comment
```

Now the rest of the files can be added.

```bash
git add .
git commit -m "Add all files"
```

```
[main d8481c0] Add all files
 2 files changed, 15 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 myclass.cpp
```

5. Editing

Changes to file can be observed as diffs.

```bash
sed -i -e 's/x = x_;/x = 2 * x_;/' myclass.cpp
git diff
```

```
diff --git a/myclass.cpp b/myclass.cpp
index 5983a1c..041e45a 100644
--- a/myclass.cpp
+++ b/myclass.cpp
@@ -4,7 +4,7 @@ MyClass::MyClass() {
 }

 MyClass::MyClass(int x_) {
-  x = x_;
+  x = 2 * x_;
 }

 int MyClass::getX() {
```

6. More Adding

You can add and commit changes in one line!

```bash
echo "// important documentation" >> myclass.h
sed -i -e 's/Hello World/Yeah, Git!/' hello.cpp
git commit -a -m "description"
```

```
[main 76cc81f] description
 2 files changed, 2 insertions(+), 2 deletions(-)
```

7. Branches

7.1. Simple Branches

Branches help you create different versions of a file on one machine.

```bash
git branch mybranch
git switch mybranch
```

Files can be edited and commited on a branch.

```bash
sed -i -e 's/" << endl/" << "\\n"/' hello.cpp
git commit -a -m "remove useless buffer flush"
```

```
[mybranch a185446] remove useless buffer flush
 1 file changed, 1 insertion(+), 1 deletion(-)
```

The changes are local to that branch.

```bash
git log | head
```

```
commit a18544676b835febc6f16618d27de1fe31d0a21e
Author: evan stokdyk <evan.stokdyk@gmail.com>
Date:   Thu May 30 15:05:19 2024 -0700

    remove useless buffer flush

commit 76cc81f439f3a6dac1f2fee53d35adadddcca43d
Author: evan stokdyk <evan.stokdyk@gmail.com>
Date:   Thu May 30 15:05:19 2024 -0700
```

Branches can be merged to combine work from different sources.

```bash
git checkout main
git merge -m "message" mybranch
```

```
Updating 76cc81f..a185446
Fast-forward (no commit created; -m option ignored)
 hello.cpp | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

The changes from the other branch appear as if they were created on this branch now!

```bash
git log | head -n 12
```

```
commit a18544676b835febc6f16618d27de1fe31d0a21e
Author: evan stokdyk <evan.stokdyk@gmail.com>
Date:   Thu May 30 15:05:19 2024 -0700

    remove useless buffer flush

commit 76cc81f439f3a6dac1f2fee53d35adadddcca43d
Author: evan stokdyk <evan.stokdyk@gmail.com>
Date:   Thu May 30 15:05:19 2024 -0700

    description
```

7.2. Merge Conflicts

A merge conflict happense when many changes happen to a file that git cannot resolve. This usually happens when the same line is changed in two different branchs.

```bash
sed -i -e 's/5/4/' hello.cpp
git commit -a -m "lower X"
git checkout mybranch
sed -i -e 's/5/6/' hello.cpp
git commit -a -m "raise X"
git checkout main
```

```
[main f127eff] lower X
 1 file changed, 1 insertion(+), 1 deletion(-)
[mybranch fcf72c8] raise X
 1 file changed, 1 insertion(+), 1 deletion(-)
```

When trying to merge, it will fail putting our local files into a strange state to resolve the conflict (the additional syntax is used to show the error in this document).

```bash
git merge -m "message" mybranch 2>&1 || exit 0
```

```
Auto-merging hello.cpp
CONFLICT (content): Merge conflict in hello.cpp
Automatic merge failed; fix conflicts and then commit the result.
```

Resolving the conflict also requires removing the indicators git places in the files.

```bash
sed -i -e '/HEAD/d;/4/d;/===/d;/mybranch/d' hello.cpp
```

Once it is done the merge can be finished with a commmit to name it.

```bash
git commit -a -m "fix merge conflict"
git branch -d mybranch
```

```
[main 84d5b7c] fix merge conflict
Deleted branch mybranch (was fcf72c8).
```

8. Publishing

8.1. Creating a remote

To create a remote repository, use the ui at 'github.com' to create a new repository.

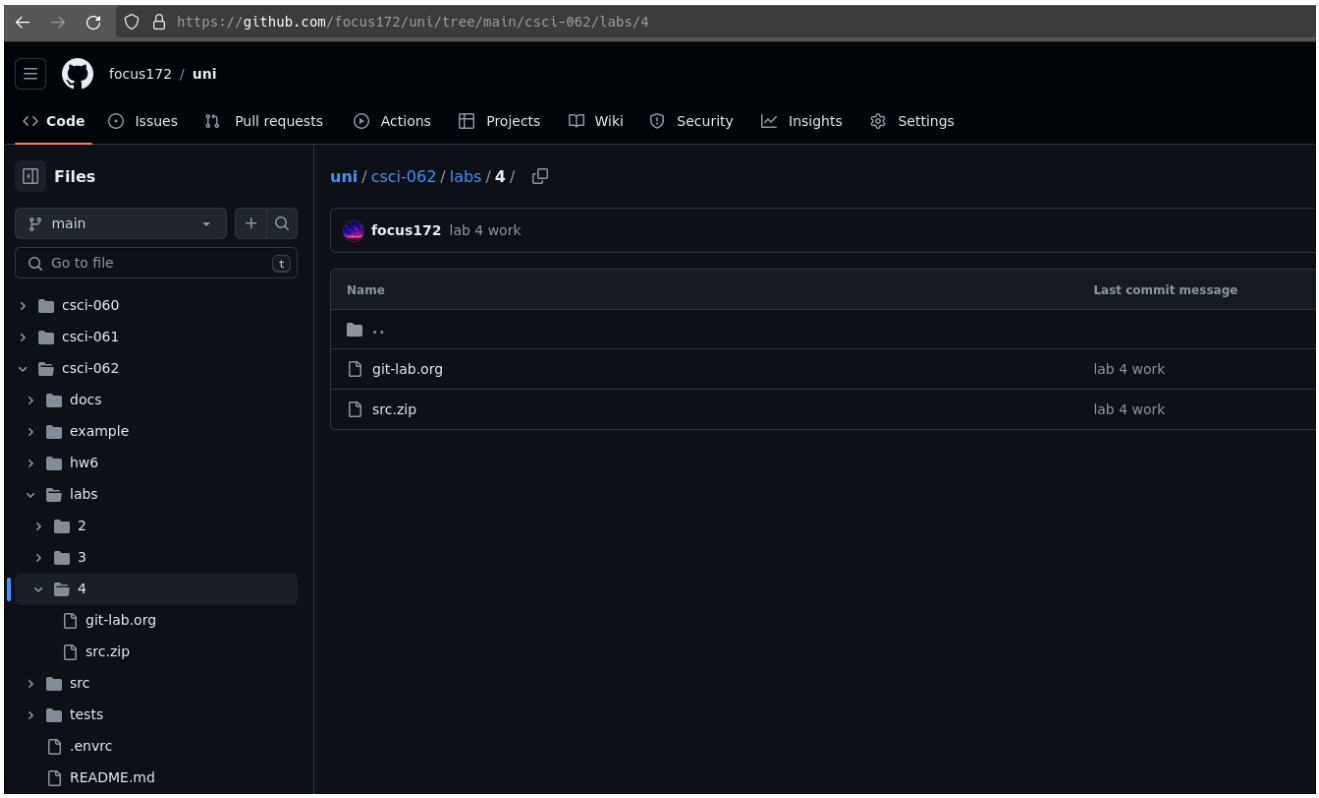
8.2. Pushing Code

Replace 'REMOTE-NAME' with the link to the repository created.

```bash
git remote add origin REMOTE-NAME
git push -u origin main
```

8.3. This Code

This most up to date version of this document is hosted online here.



Author: Evan Stokdyk

Created: 2024-05-30 Thu 15:05