

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA

UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRAREA DE LABORATOR#1

Version Control Systems si modul de setare a unui server

Author:

Ana SUTREAC

lector asistent:

Irina COJANU

Laboratory Work 1

1 The purpose of the laboratory

To study bases and principles of creating and working with Git repository

2 Objective

Version Control Systems (git || bitbucket || mercurial || svn)

3 Tasks and points

Basic Level (mark 5 || 6):

- initializing a repository
- configure your VCS
- create branches(min 2 branch)
- minimum 1 commit per branch

Normal Level (mark 7 || 8):

- set a branch to track a remote origin
- reset a branch to previous commit
- save temporary changes without committing them
- use .gitignore file

Advanced Level (mark 9 || 10):

- merge 2 branches
- resolve conflicts in 2 branches
- to know git commands

4 Laboratory work description

Link to my repository- <https://github.com/anashutrac/MIDPS/tree/master/Lab>

Basic Level

- initializing a repository and configure VCS

We can get a git repository using 2 methods. The first method takes an existing project(document) and imports it to Git with command:

```
git init repository_name
```

but for this we need to configure our VCS with generating a SSH key to connect to your existing GitHub account without supplying username and password at each visit.

The second method is to create an account on GitHub which will contain your MIDPS repository. To clone an existing repository to local map we have to use it's SSH link. Before using git clone sshlink command we have to configure our VCS with commands:

```
git config --global user.name "YourName"
```

```
git config --global user.email "youremail@domain.com"
```

We can add new file with `git add` command and use `git status` command to determine which files are in which state.

- create new branches

The default branch in Git is master, `git init` command creates it by default. As we start making commits, master branch will point to the last commit we made. Every time we commit, it moves forward automatically. We create a new branch with command:

```
git branch new_branch
```

This creates a new pointer to the last commit. To switch to an existing branch we have to run command:

```
git checkout branch_name
```

For tracking our position Git use a special pointer named HEAD. When we switch branches, HEAD pointer moves to branch we switched.

If we want to see our history of commits and existing branches use:

```
git log --oneline --decorate
```

- commits

Commits are saved changes in our project. Each commit has an associated commit message, which is a description explaining why a particular change was made. We can committing our changes with:

```
git commit -m "Message" command or
```

```
git commit -a -m "Message" add and commit at the same time
```

```
git log - commit history
```

Normal Level

- track remote origin

Remote repositories are versions of local project that are hosted on the Internet or other computers. This option make easier to collaborate with others and share your data with pushing and pulling commands.

Remote branches are references (pointers) to the state of branches in your remote repositories. They're local branches that you can't move; they're moved automatically for you whenever you do any network communication.

Tracking branches are local branches, but they have direct connection to remote branches. To set a local branch to track a remote origin we have to use command:

```
git checkout --track origin/branch_name
```

- reset a branch to previous commit

To reset a branch to previous commit we use command :

```
git revert <SHA>
```

`git revert` will create a new commit that will contain everything from old commit.

`<SHA>` - is commit hash that is given when commit

- save temporary changes without committing

To clean our working copy we use stash feature that work as a clipboard: it take all changes in your working copy and saves them for you on a new clipboard. Later, at any time, you can restore

the changes from that clipboard in your working copy - and continue working where you left off.

git stash

git stash list

an overview of current stashes

git stash pop

will apply the newest stash and clear it from stash clipboard

- use .gitignore

If you create a file in your repository named .gitignore, Git uses it to determine which files and directories to ignore, before you make a commit.

A .gitignore file should be committed into your repository, in order to share the ignore rules with any other users that clone the repository.

Advanced Level

- merge two branches

Merge two branches means that Git take the independent lines of development created by git branch command and integrate them into a single branch;

git merge branch_name - merge the specified branch into the current branch.

- resolve conflicts in two branches

If the two branches you're trying to merge both changed the same part of the same file, Git won't be able to figure out which version to use. When such a situation occurs, it stops right before the merge commit so that you can resolve the conflicts manually.

The great part of Git's merging process is that it uses the familiar edit/stage/commit workflow to resolve merge conflicts. When you encounter a merge conflict, running the git status command shows you which files need to be resolved. Then, we can go in and fix up the merge to our liking.

5 Imagini

Clone an existing repository

```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git remote add origin https://github.com/anashutrac/MIDPS.git

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 225 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
Branch master set up to track remote branch master from origin.
To https://github.com/anashutrac/MIDPS.git
 * [new branch]      master -> master

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Add and committing my first file

```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.txt

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

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git add README.txt

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.txt

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git commit -m "Add README"
[master (root-commit) c536f65] Add README
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.txt
```

Add a folder for Laboratory 1

```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Lab#1/

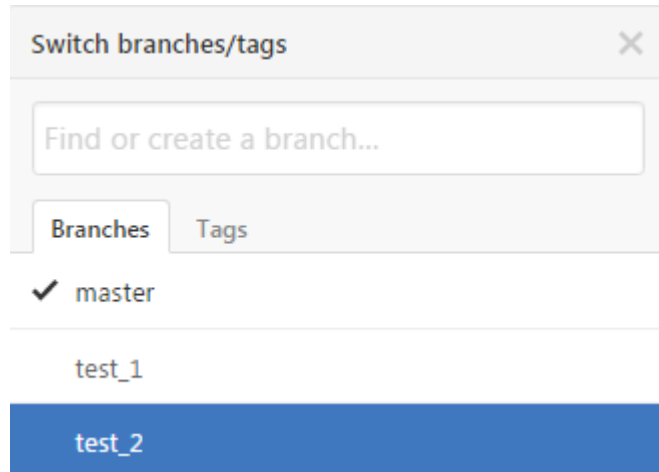
nothing added to commit but untracked files present (use "git add" to track)

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git add <folder>/# Lab#1
bash: folder: No such file or directory

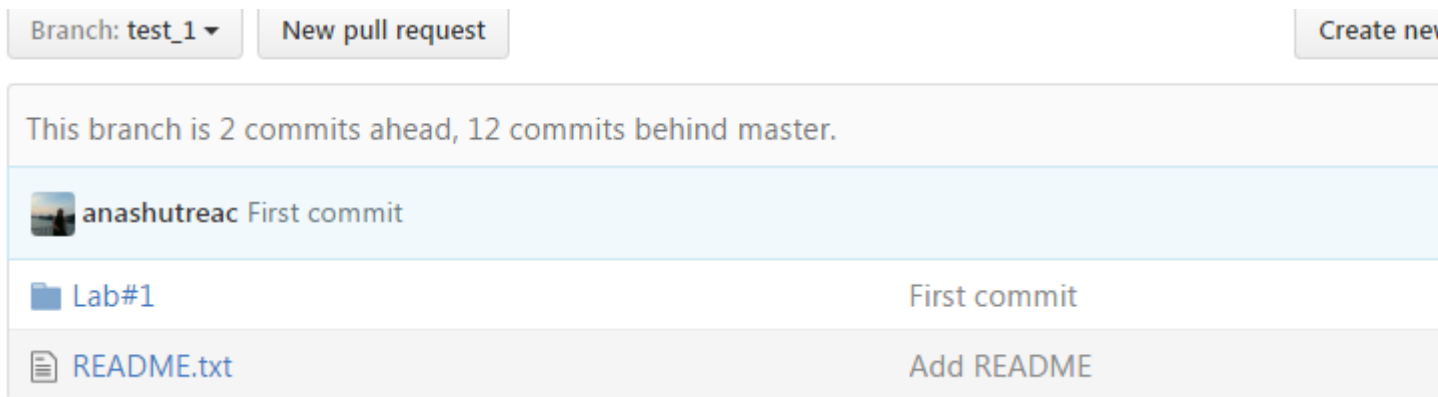
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git add Lab#1

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git commit -m "Added Lab 1 folder"
[master 889e4a1] Added Lab 1 folder
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Lab#1/README.txt
```

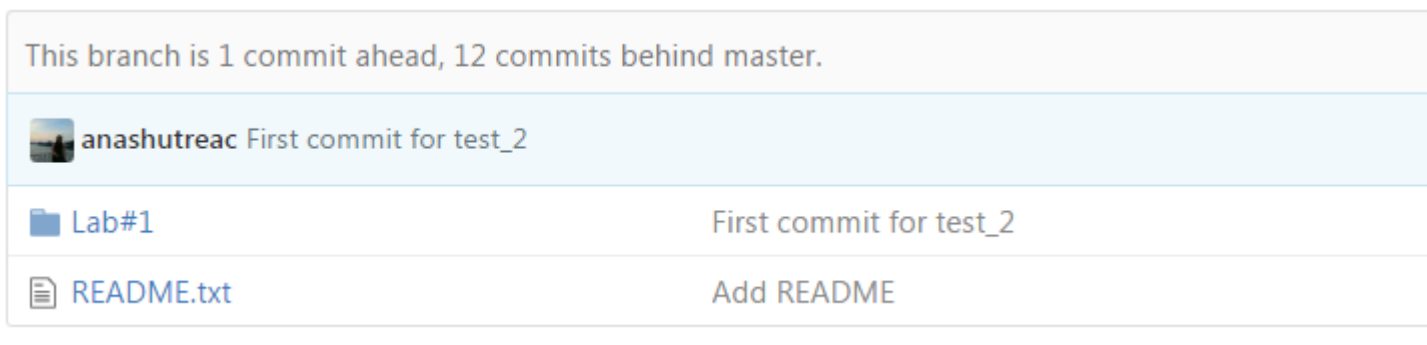
Two new branches



Commit on test1 branch



Commit on test2 branch



Set branch test3 to track a remote origin

```
Ana@Ana-PC MINGW64 ~/MIDPS/Lab#1 (master)
$ git checkout --track origin/test_3
Branch test_3 set up to track remote branch test_3 from origin.
Switched to a new branch 'test_3'
```

Save uncommitted changes

```
Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git status
On branch test_3
Your branch is ahead of 'origin/test_3' by 2 commits.
  (use "git push" to publish your local commits)
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Lab#1/test_3_1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Lab#1/test_3_2.txt

no changes added to commit (use "git add" and/or "git commit -a")

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git stash list

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git stash
Saved working directory and index state WIP on test_3: d237d98 new file in branch test_3
HEAD is now at d237d98 new file in branch test_3

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git stash list
stash@{0}: WIP on test_3: d237d98 new file in branch test_3

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git status
On branch test_3
Your branch is ahead of 'origin/test_3' by 2 commits.
  (use "git push" to publish your local commits)
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Lab#1/test_3_2.txt

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

Git stash pop command

```
Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git checkout master
A       Lab#1/test_3_2.txt
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git checkout test_3
A       Lab#1/test_3_2.txt
Your branch is ahead of 'origin/test_3' by 2 commits.
(use "git push" to publish your local commits)
Switched to branch 'test_3'

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git stash pop
On branch test_3
Your branch is ahead of 'origin/test_3' by 2 commits.
(use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Lab#1/test_3_2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Lab#1/test_3_1.txt

Dropped refs/stash@{0} (52cad884642bc824024c3e0a54d5f3623ce5cabb)










Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git commit -a -m "commit latest changes"
[test_3 8ae8cf6] commit latest changes
2 files changed, 7 insertions(+), 1 deletion(-)
create mode 100644 Lab#1/test_3_2.txt

Ana@Ana-PC MINGW64 ~/MIDPS (test_3)
$ git status
On branch test_3
Your branch is ahead of 'origin/test_3' by 3 commits.
(use "git push" to publish your local commits)
nothing to commit, working tree clean
```

Reset previous commit

```
Ana@Ana-PC MINGW64 ~/MIDPS/Lab#1 (test_3)
$ git reset --hard c536f65
HEAD is now at c536f65 Add README
```

Demonstrate that .gitignore file work well

 .git	2/17/2017 4:11 PM	File folder	
 Lab#1	2/17/2017 4:09 PM	File folder	
 Lab#2	2/17/2017 3:59 PM	File folder	
 Lab#3	2/17/2017 3:59 PM	File folder	
 Lab#4	2/17/2017 3:59 PM	File folder	
 Lab#5	2/17/2017 3:59 PM	File folder	
 README	2/17/2017 3:59 PM	Text Document	5 KB
 README	2/9/2017 10:03 PM	Text Document	0 KB
 Source	2/13/2017 3:24 PM	Object File	45 KB


```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

Merge two branches

```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git merge test_1 -m "to learn how"
Merge made by the 'recursive' strategy.
 Lab#1/test_1_1.txt | 7 ++++++
 1 file changed, 7 insertions(+)
 create mode 100644 Lab#1/test_1_1.txt
```

Resolve conflict between two branches

```
Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git merge test_2
Auto-merging Lab#1/README.txt
CONFLICT (content): Merge conflict in Lab#1/README.txt
Automatic merge failed; fix conflicts and then commit the result.

Ana@Ana-PC MINGW64 ~/MIDPS (master|MERGING)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
  new file:   Lab#1/test_2_1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   Lab#1/README.txt

Ana@Ana-PC MINGW64 ~/MIDPS (master|MERGING)
$ git commit -a -m "Resolved merge conflict"
[master 8fb81d9] Resolved merge conflict

Ana@Ana-PC MINGW64 ~/MIDPS (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

6 Conclusion

During the laboratory execution, I have studied Version Control System concepts, Git commands - most known VCS and GitHub - web based version control repository. Using this utilities for documents (in special source code) management I have noticed some advantages:

One of the biggest advantages of Git is branch making. When a developer want to change something in his/her project without changing the main product they create a new branch. This ensures that master branch will save code quality.

Documents in Git are better organized. You always can return to your earlier changes if you have commit number and messages related to commits help to identify the right one.

Moreover, if you are working on a branch and have to change to over branch but your work is not ready to commit, you can save the current status on that branch and return to it in the future.

Git allows to work in team for same project. Every one has their copy of remote repository and make local changes. This changed files can be pushed to remote repository and everyone participating in project can see changes, when and who made them.

7 Bibliography

1. <https://git-scm.com/book/en/v2>
2. <https://www.atlassian.com/git/tutorials>