Technical University of Moldova

CIM Faculty

Report

Interactive Development Environment
Laboratory work #1

Done by:	student, gr. FAF-121
	Alexa Cristina
Verified by:	assistant lecturer
	MSc Truhin Alexandr

Chisinau 2014

Tasks and Points

Points	Task
0	**•Connect to a remote server via SSH**
0	**•Initialize a repository on server**
2	**•Create a file in repository folder, write in your name, save it and commit it**
1	Connect to server using public key (1 pt)
1	•Create 2 more branches with at least one unique committed file per branch (1 pt)
1	Set a branch to track a remote origin on which you are able to push (ex. github, bitbucket or a custom server) (1 pt)
1	•Reset a branch to previous commit, reset a branch to some specific commit (1 pt)
1	•Restore a reset branch back to its previous state (1 pt)
1	•GIT cherry-pick, rebase (1 pt)
1	•Create a VCS hook (1 pt)
1	•Make your CLI text editor to highlight code (1 pt)

1	•Create a VCS alias (1 pt)
1	•Master any CLI editor (ex. VIM). Learn 10 commands' sets (a/A/i/I/o/O is one set) to prove your mastery (1 pt)
2	•Create your own server (ex. virtual machine) (2 pt)
1	•Create a VCS merge conflict and solve it (1 pt)
15	Total

Laboratory Work #1

Command Line Interface; CLI Editors; Setting Server Environment; Version Control Systems

Prerequisites:

•IDEs: Command Line Interface (CLI), CLI Editors

•Languages: bash

•Frameworks:

•Technologies: Version Control Systems (VCS)

•Time: 3 hours

Objectives:

•Understanding and using CLI (basic level)

•Administration of a remote linux machine using SSH

•Ability to work remotely (remote code editing)

•Ability to work with VCS

Technical Prerequisites:

•Connection to a remote server via SSH (you can use a virtual machine as a remote server)

•VCS on remote server

•CLI text editor (vi, vim, emacs or nano) with necessary plugins (if necessary)

Mandatory Tasks:

•Connect to a remote server via SSH

```
$ ssh -T git@github.com
```

•Initialize a repository on server

```
$ git init
```

•Create a file in repository folder, write in your name, save it and commit it

```
vim file.txt
i
Alexa Cristina
Esc
:wq
git add file1.txt
git commit -m "file add"
```

Tasks With Points:

•Connect to server using public key (1 pt)

First, I had to create a pair of keys so I typed the following command: `$ ssh-keygen -t rsa -C "cristina.alexa92@gmail.com"`
After that I transfered the public key to the remote server that I created. I used the following command: `$ ssh-copy`

•Create 2 more branches with at least one unique committed file per branch (1 pt)

I've been new to this, but after reading about branches in the git guide, I've found out that this is not difficult.

```
$ git checkout -b branch
$ touch myfile2.txt
$ git add .
$ git commit -m "created file in first branch
$ git checkout -b branch2
$ touch myfile3.txt
$ git add myfile3.txt
$ git commit -m "commitment of myfile3.txt"
```

•Set a branch to track a remote origin on which you are able to push (ex. github, bitbucket or a custom server) (1 pt)

```
git remote add origin alexa@192.168.0.103:myfolder/.git
```

The result to see if it worked is this, I guess.

```
git push origin master
alexa@192.168.0.103's password:
Everything up-to-date
```

•Reset a branch to previous commit, reset a branch to some specific commit (1 pt)

It actually depends on what is actually meant as to reset: because there to ways as told reset but one of it gets to the needed commit and makes the changes from the last commits unstaged, if it is soft reset; or it deletes all changes from last commits and they can not be reverted. In this case I used command for --soft reset, which is by default, in order to make it possible to restore the reset branch back to its previous state. First, I typed the command `git log` to see the id of commit wanted to reset.
Then, I typed the command `git reset 01e22`

•Restore a reset branch back to its previous state (1 pt)

Here I needed `git reflog` command to see all the commits, their ID-s and the previous addresses of HEAD. To get to previous state, I used the command: `git reset HEAD@{1}`

•GIT cherry-pick, rebase (1 pt)

First, I was on master I typed:

```
git cherry-pick ..branch
```

I got two files created in master from branch1. Now, I made some changes in master and commited them. So, I am going to do rebase for branch2 from master. `git rebase master` What was obtained is the files from master that were commited later were now in branch1. I understood the difference between cherry-pick and rebase. Cherry-pick only copies commits

•**Create a VCS hook (1 pt)** It was a bit tricky but interesting to find out what a hook means and how it can be made. So, the hook files are usually in `.git/hooks/` directory. I created a post-commit file in the following way:

```
vim post-commit
i
#!/bin/sh
echo Congratulations, you made a commitment
ESC
:wq
```

After the that I made the file created executable with the command: `chmod +x post-commit`

Afterwards, I made a commitment, to see if it had worked, and yes it had worked. :) The tutorial used for this task is here.

•**Install a code-highlighter plugin in your CLI text editor (1 pt)**

I installed python.vim for hilighting code in vim. It seemed to be easy, just copied the downloaded file in the directory:`.vim/syntax/folder`

•**Create a VCS alias (1 pt)**

```
git config --global alias.cm commit
```

After writing this commmand I can use cm instead of commit.

•**Master any CLI editor (ex. VIM). Learn 10 commands to prove your mastery (1 pt)**

I decided to work with vim, as far as I was a bit familiar with it. So the commands that I am "mastering" are:

- Close/Write+Close File: `:q`/`:wq`
- Activate/Dezactivate the insert mode: `i`/`Esc`
- Copy: `y`
- Cut a word: `dw`
- Paste before/after cursor: `P`/`p`

- Select/Select Lines/Select Columns: v/V/Ctrl-V

- Undo: `u`
- Redo: `Ctrl-R`
- Delete: `d`
- Move Page Up/Page Down: `Ctrl-B`/`Ctrl-F`

- **Create your own server (ex. virtual machine) (2 pt)**

I have created an ubuntu server on virtual machine, Connected it via ssh. Almost all the tasks for this laboratory work were done on this specific server. I followed a tutorial for installing the server. Also, I consulted a bit the Ubuntu Server Installation Guide. It seemed to be simple. Do not know if it would be very effective for a real server but at least for a trial it worked. :)

- **Create a VCS merge conflict and solve it (1 pt)**

For this specific case, I edited a file in branch1 and branch2 in different ways. Then, tried to rebase branch1 to branch2, and obtained a conflict (content). So, the way to solve the conflict is to modify the files such that they complement each other and wrote the following: `git rebase branch2 --continue`

Conclusions:

The tasks in this Laboratory Work made me get familiar with CLI, VCS, server installation, server connection through secure shell and so on. The thing that I can say after doing the tasks is that there are a lot more interesting stuff to learn about this, but in the same time I found out a lot of new interesting things that I can use. Most of all, I was impressed by git, and what you can do with; like creating hooks and specific customization.