FIC (Nokia) lab activity 2

Nokia Connecting Robots using Cloud Solutions

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
| Author | Cristian Cosariu, Alexandru Iovanovici, Lucian Prodan |
| Owner | Nokia Timisoara, Politehnica University of Timisoara |
| Organization | Nokia Romania |
| Approver |  |
| Document ID |  |
| Document location | Public |
|  | |  |
|  | | |  |

Contents

[1 GIT 3](#_Toc463295832)

[1.1 About version control 3](#_Toc463295833)

[1.2 Git commands 3](#_Toc463295834)

[1.3 Workflow example 5](#_Toc463295835)

[2 My virtual machine (MobaXterm / VNC) 6](#_Toc463295836)

[2.1 Accessing the machine trough ssh 6](#_Toc463295837)

[3 Resources 7](#_Toc463295838)

# GIT

## About version control

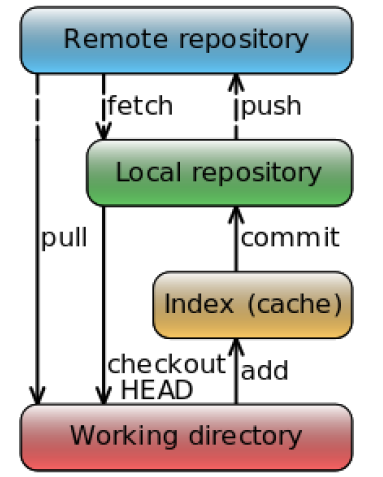
Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Local version control / centralized version control / distributed version control.

## Git commands

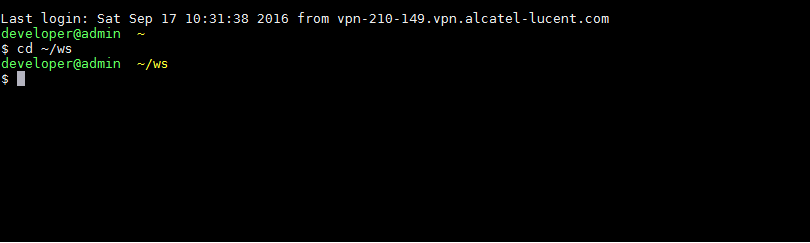
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git workflow:



1. Open a terminal
2. Go to the folder ws using the command line:

**cd ~/ws**

****

1. Inside the folder clone your project

**git clone [url]**

1. A folder with the repository has been created. Go inside the folder using the cd command:

**cd git\_demo\_project**

1. Change file content
2. In the project folder create a new file with your name and the extension .txt : [your\_name].txt. Add some text inside and save it.
3. In order to see the files that you have modified just execute the git status command:

**git status**

1. The next step is to add the modified files to the staging area:

**git add [your\_name].txt**

In case you want to add all files just use the “.” instead of the file name ( **git add .** )

1. Commit your changes into your repository:

**git commit –m “[Your name]: [a specific message for what has been changed]”**

Obs: git will notice if your project is not up to date

1. In order to avoid conflicts you need to pull changes before commiting your code (best practice). To get the latest changes use the command:

**git pull**

1. To send the changes into your repository, use the command:  
   **git push**
2. Now all the changes are in the repository and other developers from the project will be able to see them.

## Workflow example

1. When a new unit of work is started, either a JIRA issue is created or an existing issue is taken.
2. A new local branch is created from fresh master. Branch name starts with an issue number and contains a short summary of issue, dash-separated. For example, for issue IMCMD-628 the new branch is created as follows:

git checkout -b 0628-measurements-no-scale

git push --set-upstream origin 0628-measurements-no-scale

Having ID first and sound summary will help you and others later.

1. Developer works on his branch committing locally and occasionally pushing to central repo.

# ... fix some stuff

git add . && git commit -m "IMCMD-628 fixed some stuff"

# ... add some new stuff

git add . && git commit -m "IMCMD-628 added some new stuff"

# ... share the awesomeness with the world!

git push

1. Developer merges changes from master on regular basis in order to integrate with others' work and make final merge smaller and easier:

git checkout master

git pull

git checkout 0628-measurements-no-scale

git rebase master

We prefer rebasing because it puts developer's changes on top of the history (and doesn't create additional "merge commit"). Merge can be used instead of rebase. The difference is described [here](https://www.atlassian.com/git/tutorials/merging-vs-rebasing/).

**Note that rebasing rewrites the local history, so you should be aware of that**

1. After the work is done, developer rebases on master once again and pushes the result.
2. Right before opening the merge request, developer renames the branch on the server to start with mr-. This is done so that GitLab CI can pick up and build that branch (by default only master, master-\* and mr-\* branches are built by CI).

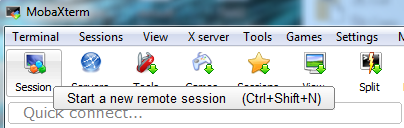
git push -u origin 0628-measurements-no-scale:mr-0628-measurements-no-scale

1. A new merge request is opened on GitLab
2. Any changes required by reviewers are pushed to the upstream task branch.
3. After MR is complete, changes are merged to master by one of the code guards.
4. The issue may now be resolved.

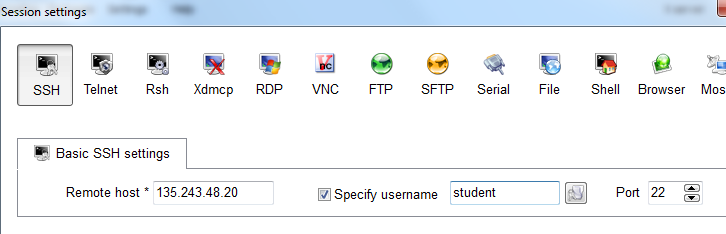
# My virtual machine (MobaXterm / VNC)

## Accessing the machine trough ssh

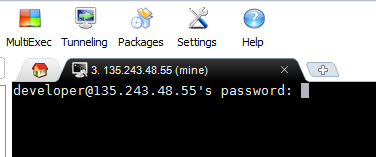
1. Open MobaXterm from the desktop
2. Open a new session



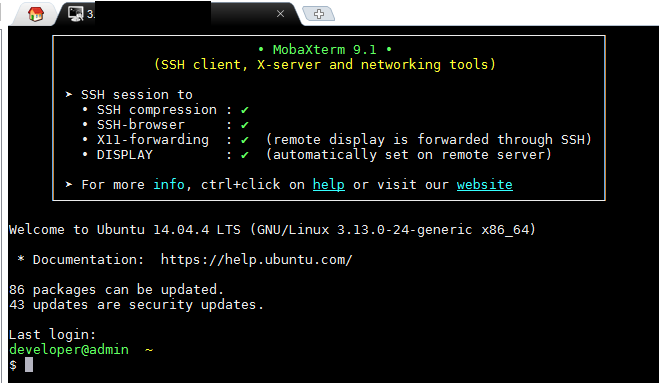
1. On the SSH tab add your virtual machine IP Address, click on specify username and add your username. The required information can be found on the paper that you received.



1. After you click ok a terminal will open and you will be asked for your password



1. You are now connected to your virtual machine



# Resources

<https://git-scm.com/>

<https://tortoisegit.org/>

<http://cebula.emea.nsn-net.net:8888/ftw15/wiki/wikis/git-workflow>

<http://learngitbranching.js.org/>

<https://www.codecademy.com/learn/learn-git>

<https://git-scm.com/downloads>

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing/the-golden-rule-of-rebasing>

<http://mobaxterm.mobatek.net/demo.html>

<http://mobaxterm.mobatek.net/download.html>