UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Version Control

An introductory tutorial

by:

Arné Schreuder & Hendri Potgieter

COS 301 - Software Engineering

February 15, 2013

1 Introduction

This tutorial covers the basics of Version Control with specific regards to Git, a tool that will be used extensively in both the mini-project as well as the main-project. This tutorial only roughly touches on the theoretical foundations that Version Control is built upon, however it is the user's responsibility to take the time and effort to read up more, get to understand and master the tool(s). It is widely accepted that Version Control forms part of the absolute necessary tools in any good Computer Scientist's tool kit and we will later discuss the reasoning behind this viewpoint.

1.1 What is Version Control?

Version control is essentially a control system used to keep track of modifications (additions, removal or changes) of documents or files in a project. Version control is also know as revision control or source control. VCS's (Version Control Systems) are known for being used for computer program source files, website files, documents and various other types of project related documents.

1.2 Why use Version Control?

Version control's usage can be motivated heavily by theoretical reasoning, but here we will only highlight the simple, yet most effective arguments. Have you ever lost a project the night before submitting? Version control helps to maintain a copy of the source code on some cloud location. It enables multiple users to simultaneously work together on the same/different versions of a project and without version control, keeping track of changes would be tedious and unnatural. It simply helps us to have control over the natural progression of our projects.

2 Git

2.1 What is Git?

Git is one of the most popular VCS's, especially to Linux users around the world. Git is simply one of the many Version Control Systems out there, but other popular tools include Mercurial and Subversion. Git is a distributed revision control/source management system and is built with an emphasis on speed.

2.2 The history of Git

Git was developed by Linus Torvalds (The creator of the Linux kernel) specifically for Linux kernel development. The developers of the Linux kernel first used BitKeeper, but later found that a more comprehensible tool was needed. The Linux kernel is available on Github, the world's largest online Git service. Feel free to go and follow Linus's tracking on the Linux kernel and take a look at the different branches, how modifications are pushed etc.

3 The basics of Git

3.1 Installing Git

Git is cross-platform compatible, but to install in a Linux environment such as Ubuntu run the following command:

```
$sudo apt-get install git-core
```

3.2 Typical Git cycle

```
$git pull user@server:~/Serverprojects/project
$cd project
... #do some work
$git add *
$git commit -m "Change log message here"
$git push origin master
```

3.3 Access to Git

Access to Git depends on how to server is set up. Here we will explain this by means of example. Github allows users to access repositories using HTML or SSL. SSL makes use of public and private keys that one needs to specify locally and then add on Github. Html uses normal username/password authentication.

3.4 Repositories

3.4.1 Server

```
$mkdir ~/Serverprojects/project
$cd project
$git init --bare
$git update-server-info          # If planning to serve through HTML
```

3.4.2 local

```
$cd ~/Localprojects/
$git clone user@server:~/Serverprojects/project
$cd project
... #development process continues
```

3.5 Adding/Removing files

When adding files, the local files have to be added first and committed before it can be pushed.

```
$git add main.cpp
$git commit –m "Added the old main file from backup"
$git push
```

When removing files, delete the local file first, then do update/commits on other files and then push.

```
$rm –rf /contentfolder
$git add –u
$git commit –m "Removed content folder"
```

3.6 Getting/pushing files

To get the latest files:

```
$git pull user@server:~/Serverprojects/project
```

To submit your branch as the latest version:

```
$git push origin master
```

3.7 Merging/Handling conflicts

This section of Version Control is very comprehensible and depends heavily on case to case scenarios. This section is thus left for self study.

3.8 Notes

Please note that the above commands are simply for demonstration purposes. It is up to you to adapt the commands for your own project, whether or not your work remotely or locally. Please do not accept the above commands as final and concrete, there are so many other options and alternatives and customization.

For more info type in command line:

```
$man git
```

4 Relevance to COS 301

4.1 Mini project Version Control protocol

The mini project protocol is as follows:

There will be 4 rounds in which groups will be assigned. On each round, the groups will renew and work on developing the next leg of the mini project. The groups for each round will be in the form of mixed groups, assigned by the COS 301 staff.

Therefore, the mini project protocol with regards to git is as follows:

There will be repositories for each group for each round. For example: COS301_GRPX_RNDY. Groups will use their repositories to "version control" their relevant documents each round and have both read and write access to these repositories. There will also be a master repository in which the COS 301 staff will push the finalized/model documents for each round. Finally the master repository will be used as the finalized project. Users will have read access to this repository throughout.

Note: Any changes to this arrangement will be via notification on the CS site under the COS 301 module.

4.2 Access details

Server details: **geth.cs.up.ac.za**
Login details: **Linux login details**

To get access to a repository, a user can simply make use of his/her Linux login details in combination with a Git clone and pull as such:

```
$git clone uXXXXXXXX@geth.cs.up.ac.za:/home/COS301/COS301_GRPX_RNDY
$git pull
```

5 Recommended reading material

[1] Pro Git book which can be found online at http://git-scm.com/book (ebook) and on github (https://github.com/progit/progit) - Please take the time to read through the book as Git is quite an extensive tool to use and definitely forms part of any good Computer Scientist's tool kit.

References

[1] Chacon, S 2009, *Pro Git (Expert's Voice in Software Development)*, e-book, accessed 11 Feb 2013, <http://git-scm.com/book>.