Git Hands-On

Version: 0.2.1

Anders Nilsson
andersn@control.lth.se

November 2, 2015

1 Introduction

It is assumed that you have working installation of git on your computer. This hands-on tutorial will straight through use text mode git as run in a terminal window. Feel free to use a graphical client of your choice instead if that makes you feel better.

2 Setting up

Begin by opening a web browser and go to `http://gitlab.control.lth.se` and sign up for a new account, or authenticate using a google ID if you prefer that.

Authentication when accessing repositories on the gitlab server is handled using SSH keys, we should therefore generate a key pair (if you don't already have keys) and upload the public key to the gitlab server. First, check for possibly existing keys:

```
andersn@heisenberg:/local/home/andersn\$ ls ~/.ssh/
authorized\_keys  id\_dsa.keystore  id\_rsa      known\_hosts
id\_dsa           id\_dsa.pub       id\_rsa.pub
```

You most probably don't have all the files I have, the important ones to look for are the ones starting with id_dsa or id_rsa. If you have at least one of these pairs, just skip to the next step and upload the key to the server.

To generate a key pair just run the following:

```
andersn@heisenberg:/local/home/andersn\$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/local/home/andersn/.ssh/id_dsa): id_dsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_dsa.
Your public key has been saved in id_dsa.pub.
The key fingerprint is:
SHA256:/TKfNkR7SEED5SRe2dH6GdECF6J7sFBUar+i4t5cN0A andersn@heisenberg
The key's randomart image is:
+---[DSA 1024]----+
|          .==X+++.|
|          o Bo+o.o|
|         . E .. o.|
|          = =o .. |
|         S =ooo .o|
|            ++....|
|           =.=.   |
|      .o o *oo    |
|     oo.+  .o.    |
+----[SHA256]-----+
```

Important! When it asks for a passphrase, just press ¡enter¿.

Ok, now we have SSH keys so let's then upload the public key to the server. On the gitlab webpage in your browser, find 'Profile Settings' , 'SSH Keys', 'Add SSH Key'. Copy-Paste the contents of your public key id_dsa.pub and click to add the key.

Now you can set up the environment for the rest of the exercise. Start by telling git your name and email:

```
git config --global user.name "Anders Nilsson"
git config --global user.email "andersn@control.lth.se"
```

Then let's clone the playground repository:

```
git clone ssh://git@gitlab.control.lth.se:/FRT090-2015/playground.git
cd playground
```

3 edit - commit - push

Open the file *Readme* in a text editor and add some lines of text to the file. Then save.

```
gedit Readme
```

Use `git status` to check for modified files in the working copy.

```
andersn@stodola:/tmp/playground\$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

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:   Readme

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

Commit your changes to the local repository, with some useful commit comment.

```
andersn@stodola:/tmp/playground\$ git commit -am"Added a new line."
[master 50c3451] Added a new line.
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Run `git status` again to see what has happened.

```
andersn@stodola:/tmp/playground\$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working directory clean
```

This means that you have local commits which are not yet available on the git server. To publish your changes on the server, run `git push`.

```
andersn@stodola:/tmp/t/playground\$ git push
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100\% (2/2), done.
Writing objects: 100\% (3/3), 306 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To ssh://git@gitlab.control.lth.se:/FRT090-2015/playground.git
   8a8ad08..eb02c3c  master -> master
```

4 Handling Conflicts

If the previous push command went as shown, you happened to be the first in the group to perform a push. Instead you might see something similar to

```
andersn@fiol:../andersn/work/playground\$ git push
To git@gitlab.control.lth.se:frt090-2015/playground.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@gitlab.control.lth.se:frt090-2015/playground.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

I basically says that someone else performed a succesful *push* to the repository server sometime between your latest succesful *pull* operation and the failed *push*. The message also tells you what to do to solve the problem, perform a *git pull* and see what happens.

```
andersn@fiol:../andersn/work/playground\$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100\% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100\% (3/3), done.
From gitlab.control.lth.se:frt090-2015/playground
   eb02c3c..0bf182c  master     -> origin/master
Auto-merging Readme
CONFLICT (content): Merge conflict in Readme
Automatic merge failed; fix conflicts and then commit the result.
```

If git manages to merge the changes you will just get a message saying all went well. If the different versions can not be automatically merged together you will be notified about the conflict instead, as shown above. To resolve the conflict you have to open the file(s) where there is a conflict and decide yourself which version should be used. In this example the *Readme* file now looks like this.

```
A phony repo used for tutorial purposes.
<<<<<<< HEAD
A second line.
=======
Another second line change.
>>>>>>> 0bf182c3ae6abc94a75c0c4e27df85f6cf7d82e4
```

Here we see the two versions of the second line within conflict markers. Edit the file as you wish, then save, commit, and try to push again.

5 Continue

Keep on playing with the playground repository to get familiarized with git.

- Add new files to the repository.
- Try some other git commands; *diff*, *log*, *blame*
- Try to add and/or modify a *.gitignore* file. It is very useful for hiding away certain file names (or patterns) that you do not want to have within version control. For example .class files if you program in Java, or .o files from C/C++.
- Point a web browser to `http://gitlab.control.lth.se`, find the playground project and play around with the functionality in gitlab.