

Using Git to Manage CogX KTH Code

Andrzej Pronobis

September 4, 2009

1 Benefits of Using Git

- *Branches and tags are managed by Git.* Merging can be performed easily without the need to manually track the origins of the code and worry about the same code being merged multiple times.
- *Distributed repositories.* The work can be performed locally and off-line. Branching/merging/committing does not require connection to any central server.
- *Cheap (topic) branches.* Creating local branches e.g. for testing/implementing a feature, maintenance etc. is quick and easy. A workflow based on topic branches allows to create a higher quality code and maintain stability of the master branch.
- *Commits are uniquely identified by an SHA hash code.* The SHA code is calculated based on the contents of the commit and therefore is the same even for unrelated repositories.
- *Non-linear history.* Git tracks the history of all branches and merges.
- *Performance.* Git is fast.

2 Documentation

- Git Community Book
<http://book.git-scm.com/>
- Git FAQ
<http://git.or.cz/gitwiki/GitFaq>
- Git for the lazy
http://www.spheredev.org/wiki/Git_for_the_lazy
- Git workflow for agile teams
<http://reinh.com/blog/2009/03/02/a-git-workflow-for-agile-teams.html>
- Git for computer scientists
<http://eagain.net/articles/git-for-computer-scientists/>

3 Installing Git

3.1 Ubuntu Linux

```
apt-get install git-core git-gui gitk
```

3.2 Windows

Full windows installer can be found at

<http://code.google.com/p/msysgit/downloads/list>

4 Setting Up the Local Repository

- Get access to the remote repository. Generate your public SSH key:

```
ssh-keygen -t rsa
```

and send the file `~/.ssh/id_rsa.pub` to the admin of the repository.

- Clone the remote repository into a local directory:

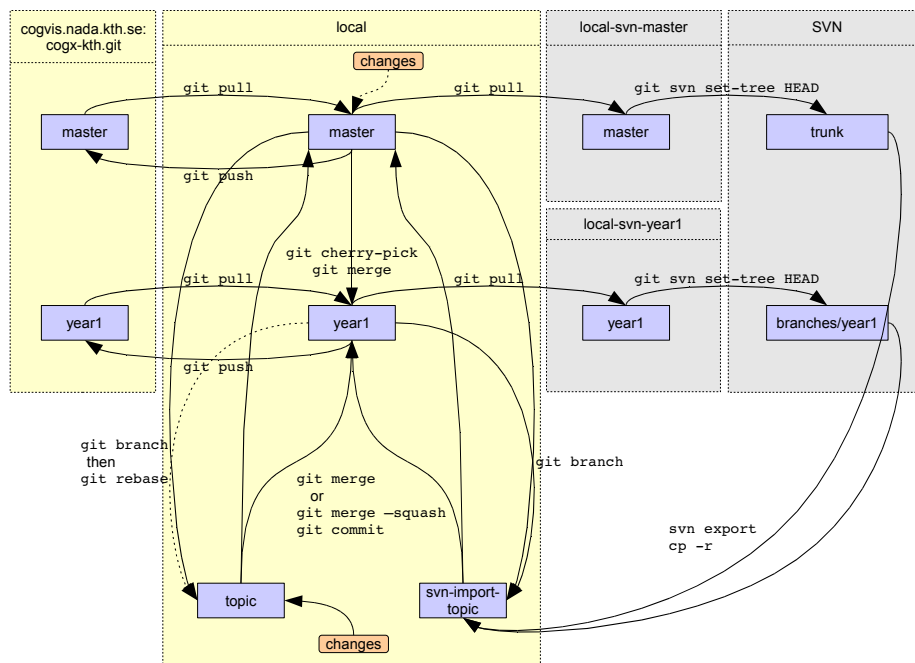
```
git clone git@cogvis.nada.kth.se:cogx-kth.git
```

- Create local branches tracking the remote branches e.g. `year1`:

```
git branch --track year1 origin/year1
```

5 Workflow for the CogX-KTH Code

5.1 Overview



5.2 Feature Development

- Checkout the master branch:

```
git checkout master
```

- Pull updates from the central server and merge them to master:

```
git pull
```

- Create and checkout a topic branch for the new feature:

```
git checkout -b topic
```

- Do some work in the topic branch:

```
echo "contents" > file
git add file
git commit -m "First_commit"
echo "contents2" > file2
git add file2
git commit -m "Second_commit"
```

- Merge with the master branch:

```
git checkout master
git merge --squash topic
git commit
```

- Push the changes upstream:

```
git push
```

- Do more work in the topic branch:

```
git checkout topic
echo "new_contents" > file
git add file
git commit -m "Third_commit"
```

- Pull some new updates from the central server and merge them to master:

```
git checkout master
git pull
```

- Rebasing the topic branch against the new changes:

```
git checkout topic
git rebase master
```

- Merge the changes with the master branch:

```
git checkout master
git merge --squash topic
git commit
```

- Push the changes upstream:

```
git push
```

5.3 Pushing Changes to SVN

Missing

5.4 Pulling Changes from SVN

Setup a joint GIT/SVN local repository:

- setup another local repository for instance by making a copy of your current git repository: `cp -r my_git_repo svn-git_repo`
- enter the directory with the repository: `cd svn-git_repo`
- checkout branch master: `git checkout master`
- remove all files apart from the `.git` directory
- run `svn co https://codex.cs.bham.ac.uk/svn/nah/cogx/code/kth/trunk`
- move all the files from trunk to your git working copy: `mv trunk/* .`

Now you have a joint git/svn working copy. To pull new changes from svn and merge them to git, do as follows:

- if you messed up sth with the files, you can do `git reset --hard`
- now, pull the most recent version of the master branch: `git pull`
- update the changes from the SVN: `svn up`
- commit the changes to git: `git add/git commit`
- push the changes to the shared repository: `git push`

6 FAQ

1. How to list branches and check which branch is checked out?

```
git branch
```

2. How to list all branches including remote branches?

```
git branch -a
```

3. How to view history of a branch or all branches?

```
gitk
gitk --all
```

4. How to throw away uncommitted change in a single file?

```
git checkout -- file
```