



github vs. Trac+SVN

For SNO+ RAT

A walkthrough of user and developer workflow
with git and github

Andy Mastbaum
mastbaum@hep.upenn.edu

git(hub) vs. SVN

Introduction

git is a distributed version control system. github is an online service that provides repository hosting and a suite of source code management tools for projects that use git.

No version control:

- ✗ Revision tracking: Arcane folder numbering schemes
- ✗ Code sharing: Email, scp, snail mail printout
- ✗ Collaborative development tools: Email/Gchat
- ✗ Bug tracking: Email, sticky notes
- ✗ Repository backup: None, probably

SVN version control:

- ✓- Revision tracking: Snapshots of code numbered sequentially
- ✗ Code sharing: Email, scp, ...
- ✗ Collaborative development tools: email/Gchat
- ✓ Bug tracking: Trac tickets
- ✗ Repository backup: Do it yourself

git version control with github:

- ✓ Revision tracking: changesets numbered, can be exchanged between repositories, picked and chosen individually
- ✓ Code sharing: git push/pull (pull requests)
- ✓ Collaborative development tools: github code commenting system
- ✓ Bug tracking: github Issues
- ✓ Repository backup: automatic with github

In SVN, there is one repository and users have “working copies” between which there is no way to share patches. With git, everything is a repository and code is easily shared between peers. Individual commits can be pushed and pulled from one repository to another.

github

Workflow

git Quick Reference

	SVN	git
Remote Repositories	<code>svn checkout [url]</code>	<code>git clone [url]</code>
	<code>svn update</code>	<code>git pull</code>
	<code>svn info</code> (remote repository location)	<code>git remote -v</code>
Committing	<code>svn checkout [url]</code>	<code>git clone [url]</code>
	<code>svn update</code>	<code>git pull</code>
	<code>svn diff (-r[rev] [path])</code>	<code>git diff ([rev] [path])</code>
	<code>svn status</code>	<code>git status</code>
	<code>svn revert [path]</code>	<code>git checkout [path]</code>
	<code>svn rm</code> <code>svn add</code> <code>svn mv</code>	<code>git rm</code> <code>git add</code> <code>git mv</code>
	<code>svn commit</code>	<code>git commit -a</code> <code>git push</code>
Browsing	<code>svn log</code>	<code>git log</code>
	<code>svn blame</code>	<code>git blame</code>
Branches	<code>svn copy [server]/trunk \ [server]/branches/branch</code>	<code>git branch [branch]</code>
	<code>svn switch \ [server]/branches/branch</code>	<code>git checkout [branch]</code>
	<code>svn merge -c [rev] [url]</code>	<code>git cherry-pick [rev]</code>
Tags	<code>svn copy [server]/trunk \ [server]/tags/name</code>	<code>git tag -a name</code>

For a more complete guide and explanations, see <http://git.or.cz/course/svn.html>
See also <http://help.github.com/git-cheat-sheets/>

github vs. Trac

Main Page

bug tracking
revision history
wiki

The screenshot shows the GitHub repository page for 'cenpa/rat'. The repository is owned by 'mastbaum'. The page displays the repository's source code, commits, and a list of files. The 'Source' tab is selected, showing the repository's structure. The 'commits' tab is also visible, showing a list of commits. The repository description is 'RAT is an Analysis Tool, SNO+ Edition'. The repository is hosted on GitHub, and the source code is available via SSH or HTTP. The repository is public, and the user 'kmarshall' is the author of the latest commit. The repository is a part of the 'cenpa' organization, and the repository is named 'rat'.

The screenshot shows the Trac project page for 'SNO+'. The project is managed by 'SNO+ trac&svn'. The page displays the project's source code, tickets, and a list of files. The 'Wiki' tab is selected, showing the project's wiki page. The wiki page is titled 'RAT is an Analysis Tool (RAT) - SNO+ Edition'. The page contains information about the project, including its purpose, news, code, documentation, and quick links. The project is managed by 'SNO+', and the source code is available via Subversion. The project is a part of the 'SNO+' organization, and the project is named 'RAT'.

github

Workflow

If you just want to use RAT
(not change it)...

0. Clone the RAT repository

```
$ git clone git@github.com:cenpa/rat.git
remote: Counting objects: 7363, done.
remote: Compressing objects: 100% (2742/2742 (delta 4790)
remote: Total 7563 (delta 4790), reused 7563 (delta 4790)
Receiving objects: 100% (7564/7563), 611.27 MiB | 1.75 MiB/s, done.
Resolving deltas: 100% (4790/4790), done.
$ cd rat
$ ls
doc rat tools
```

That's it! Build RAT with scons as always.

Tip: you are cloning the entire repository with history,
so the initial checkout will take longer than with SVN.

To update your copy of RAT,

```
$ cd rat
$ git pull
```

If you find a bug:

To submit a bug report or request a feature, go to

<https://github.com/cenpa/rat/issues>

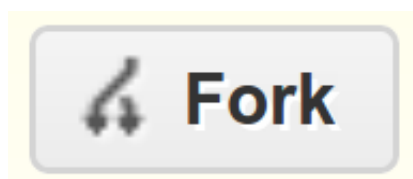
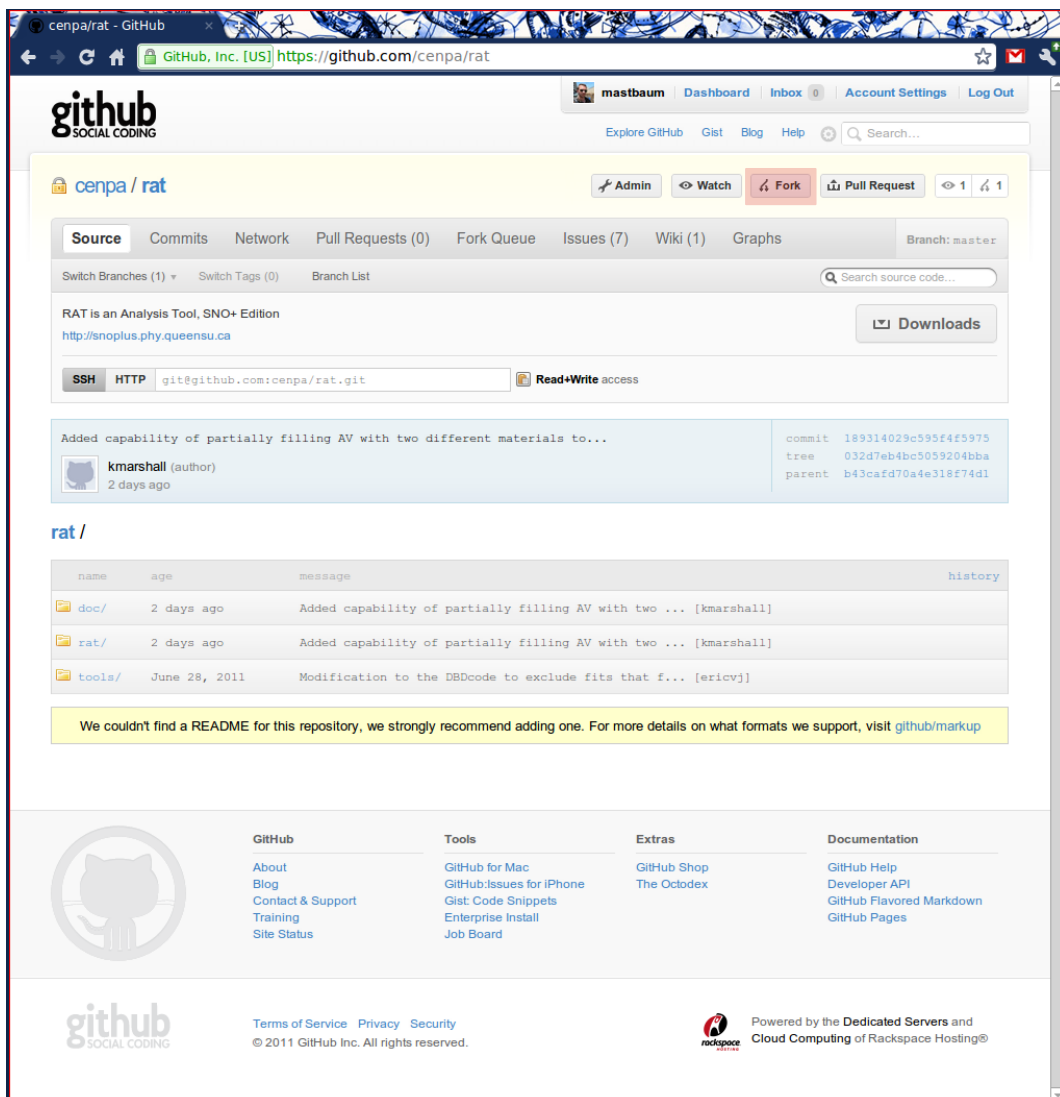
Once you've ensured that your report/request isn't already
there, click "New Issue" to add it to the list.

If your bug/feature is already reported, make a note of
your failure mode/use case in the comments!

github Workflow

If you are contributing code to RAT....

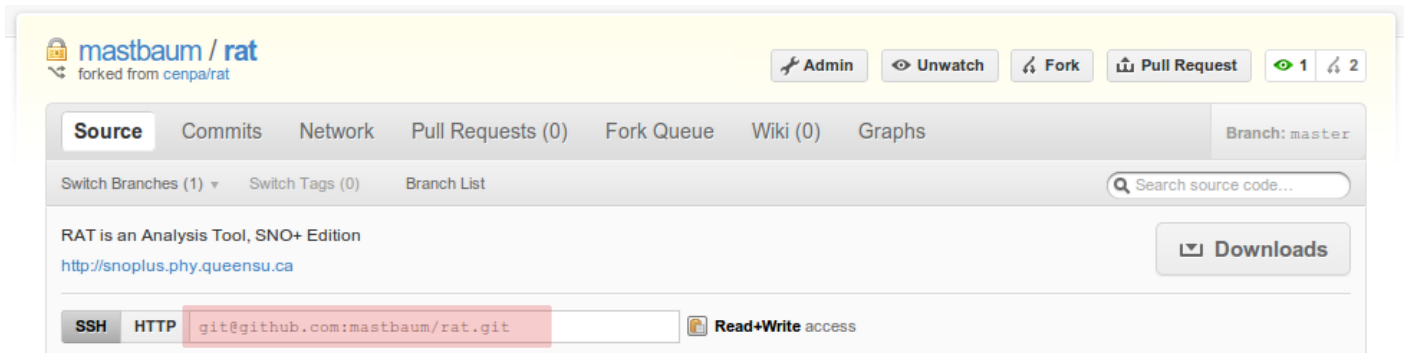
1. Fork the main RAT repository



For more information about forks, see <http://help.github.com/fork-a-repo/>

github Workflow

You now have your
own clone of RAT:



2. Clone your fork locally

```
$ git clone git@github.com:[YOUR USERNAME HERE]/rat.git
remote: Counting objects: 7363, done.
remote: Compressing objects: 100% (2742/2742 (delta 4790)
remote: Total 7563 (delta 4790), reused 7563 (delta 4790)
Receiving objects: 100% (7564/7563), 611.27 MiB | 1.75 MiB/s, done.
Resolving deltas: 100% (4790/4790), done.
$ cd rat
$ ls
doc rat tools
```

Tip: you are cloning the entire repository
with history, so the initial checkout will
take longer than with SVN.

github

Workflow

3. Make changes to the code

```
HACK HACK HACK
```

See a summary of your changes:

```
$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in the working directory)
#
#       modified:   rat/rat.cc
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Look at the diff:

```
$ git diff
diff --git a/rat/rat.cc b/rat/rat.cc
index fd4bf25..7025af9 100644
--- a/rat/rat.cc
+++ b/rat/rat.cc
@@ -67,7 +67,7 @@ int main (int argc, char** argv)

    parse_command_line(argc, argv);

-   info << "This is SNO+ RAT, version " << RATVERSIONSTR << "." << RATREVISIONSTR << newline;
+   info << "This is SNOT RAT, version " << RATVERSIONSTR << "." << RATREVISIONSTR << newline;

    //Hostname and machine probing.
    struct utsname nameinfo;
```

Tip: git is far more powerful than SVN. For more information on the many git features and subcommands, see <http://help.github.com/git-cheat-sheets/>

github

Workflow

3. Commit your changes to the local repository

```
$ git commit -a -m "reimplemented joke"
[master 189bd92] reimplemented joke
1 files changed, 1 insertions(+), 1 deletions(-)
```

- ! At this point, your change is only committed to the local repository on your computer (unlike SVN, where there was no local repository).

Tip: you can commit lots of changes locally (say you're on a plane), and they will be queued and applied individually when you synchronize later

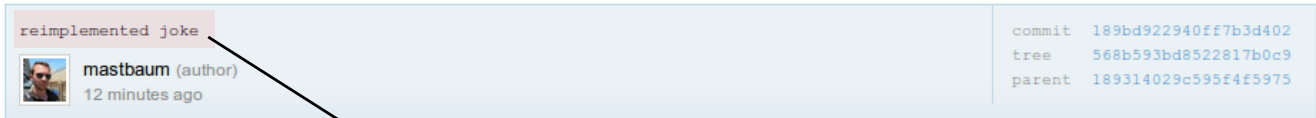
4. Synchronize with the remote repository

```
$ git push
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 390 bytes, done.
Total 4 (delta 2), reused 0 (delta 0)
To git@github.com:[your username]/rat.git
1893140..189bd92 master -> master
```

This pushes all changes committed to the local repository on to your repository on github (your fork of RAT)

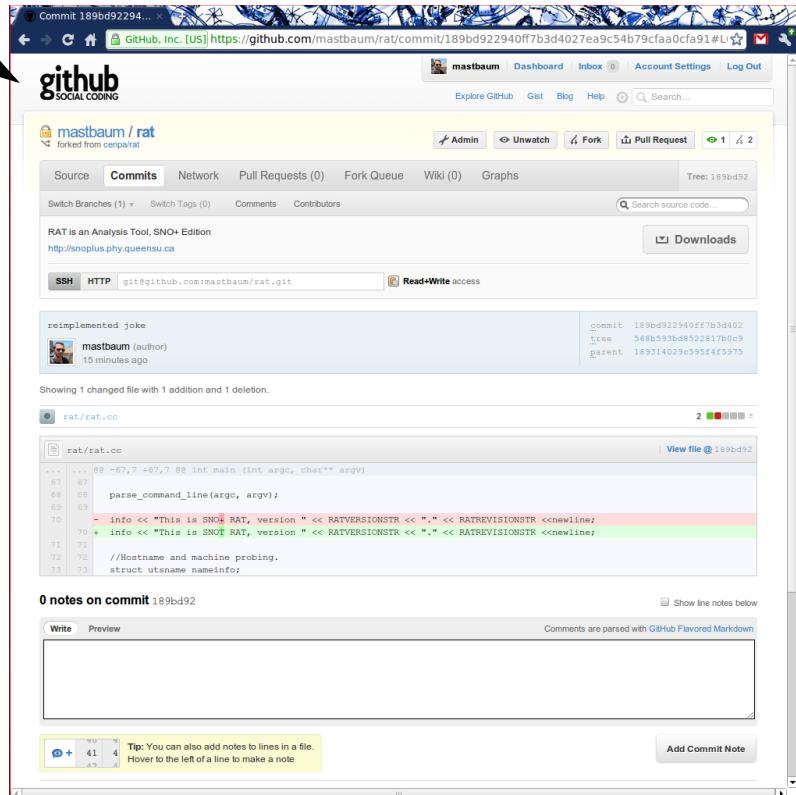
github Workflow

The changes, now on github:



Click for changeset
details

Make comments
on the commit, or
individual lines



Remember that your fork is a perfectly legitimate RAT repository. This means that:

1. Others can fork your fork for collaborative development
2. You can pull changes in from other repositories (share patches)
3. You can submit a “pull request” to have your changes merged into another repository
3. You can create your own issue tickets, milestones, etc.

github Workflow

5. Submit a pull request to have your changes merged into the main RAT

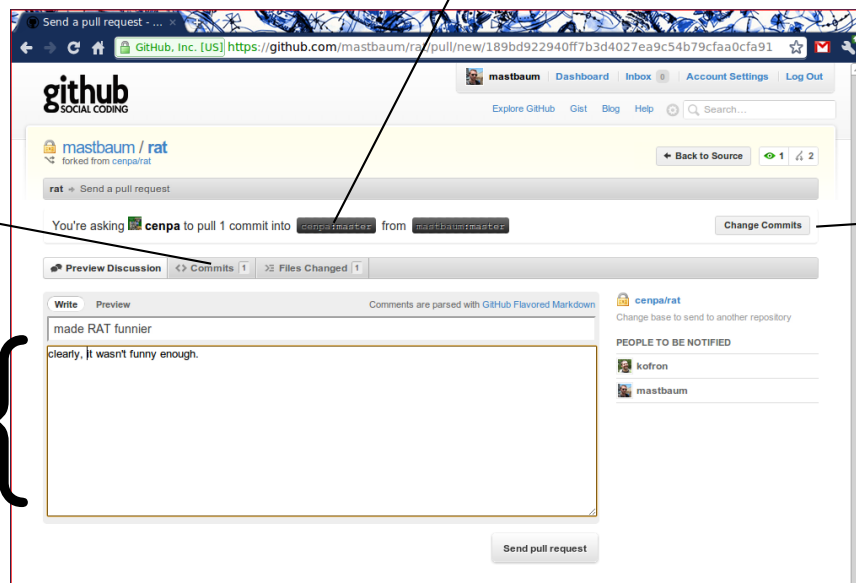
 **Pull Request**

Send to another RAT repository to share patches with collaborators

Preview included changes

Choose which commits are included

Explain why changes were made

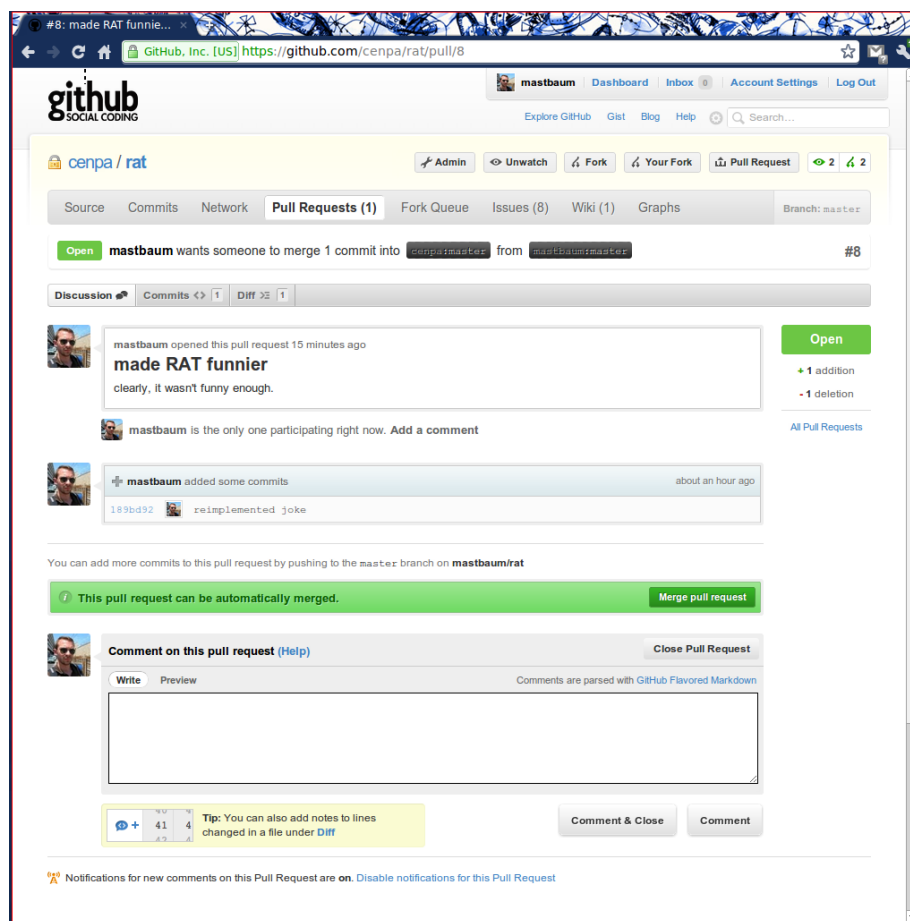


Send pull request

See <http://help.github.com/send-pull-requests/> for more information on pull requests

github Workflow

The CIC will review changes and merge code into the main RAT repository



If a pull request is rejected, the reviewer will provide comments on the reasons and make suggestions for improvement.

Commits and comments can then be added, creating a dialogue between the developer and reviewer.

github

Workflow

? Synchronizing your fork with the main • RAT repository

Note: If you are not using a fork, or just want to synchronize your local repository with your fork, just use “git pull.”

Situation: While you are developing a feature on your fork, code that you want has been committed to the main RAT repository (or someone else’s fork). You wish to bring your repository up to date with those changes.

- 1 Browse to your fork on github (<http://github.com/username/rat>)
- 2 Click on “Fork Queue” to see all commits that are in other forks but not yours.
- 3 Select the commits you would like to pull into your fork. To bring up to date with the main RAT repo, just “Select All” the commits under cenpa/rat.
- 4 Under the “Actions” drop-down, choose “Apply”

To see these changes in your local copy, you will need to pull them from github:

```
$ cd rat  
$ git pull
```

? Additional resources

github Introduction:

<http://help.github.com/>

git Cheat Sheets:

<http://help.github.com/git-cheat-sheets/>

Complete git command reference:

<http://gitref.org/>

git/SVN Command Reference

<http://git.or.cz/course/svn.html>

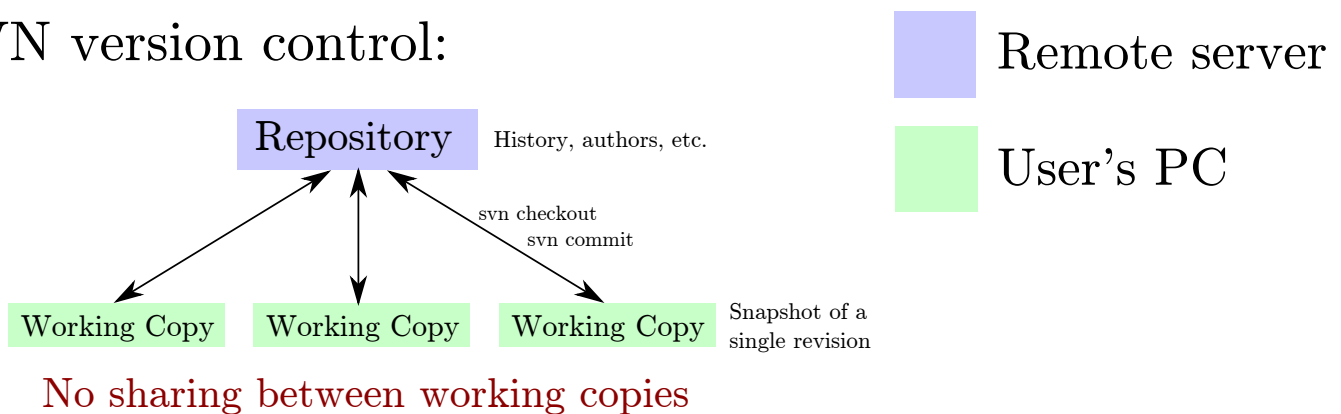
git(hub) vs. SVN

Repository Structure

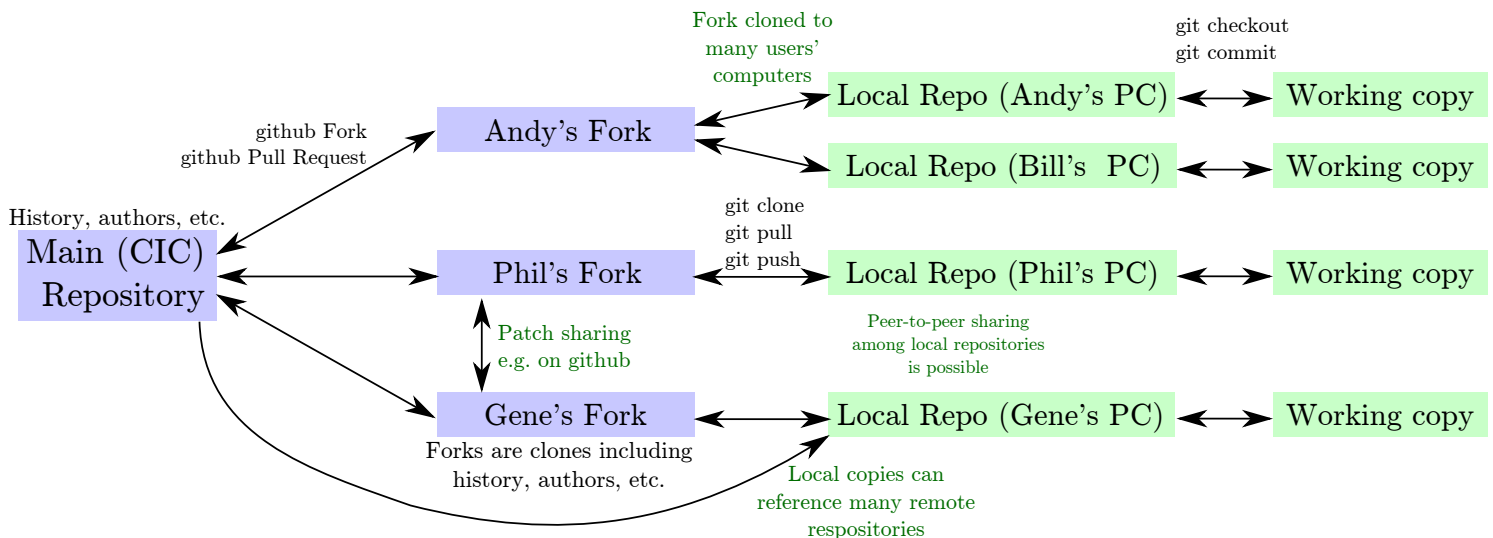
No version control:

```
me@mycomputer ~$ ls
mycode      mycode_2      mycode_2.1
myrealcode  mycode_2a     myproject4
myrealcode2 therealcode    a
temp        mycode_temp   mycode_test
mycode080511 mycode_r423   mycode_real
to_do_list1 todo          notes
notes_old   todo_list1_old_a
```

SVN version control:



github version control (example):



Everything is a repository, allowing patch-sharing and source code management impossible in SVN. git and github's tools make the actual workflow very simple, as shown in this walkthrough.