



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 vs. Trac

Main Page

bug tracking
revision history
wiki

cenpa/rat - GitHub

GitHub, Inc. [US] <https://github.com/cenpa/rat>

github SOCIAL CODING

cenpa / rat

Source Commits Network Pull Requests (0) Fork Queue Issues (7) Wiki (1) Graphs Branch: master

Switch Branches (1) Switch Tags (0) Branch List Search source code...

RAT is an Analysis Tool, SNO+ Edition
<http://snoplus.phy.queensu.ca>

SSH HTTP `git@github.com:cenpa/rat.git` Read+Write access

Added capability of partially filling AV with two different materials to...

name	age	message	history
doc/	2 days ago	Added capability of partially filling AV with two ... [kmarshall]	
rat/	2 days ago	Added capability of partially filling AV with two ... [kmarshall]	
tools/	June 28, 2011	Modification to the DBDcode to exclude fits that f... [ericvj]	

We couldn't find a README for this repository, we strongly recommend adding one. For more details on what formats we support, visit [github/markup](#)

github SOCIAL CODING

Terms of Service Privacy Security
© 2011 GitHub Inc. All rights reserved.

SNO+ trac&svn Integrated SCM & Project Management

logged in as mastbaum Logout Preferences Help/Guide About Trac

Wiki Timeline Roadmap Browse Source View Tickets New Ticket Search Doxygen Admin

Wiki: WikiStart

RAT is an Analysis Tool (RAT) - SNO+ Edition

RAT is a Monte Carlo and analysis tool developed by the [Braidwood collaboration](#) for modeling the behavior of a liquid scintillator detector surrounded by PMTs. Its goal is to be applicable to wide range of PMT and scintillator-based experiments.

News

- April 26, 2011 - As of [r512](#), RAT requires GEANT4 9.4 and is compatible with ROOT 5.28.
- March 18, 2010 - Trac site is alive!

Code

You can get the code for RAT using [Subversion](#), a revision control tool (like CVS) included in most modern Linux distributions. Anonymous checkout is available using the command

```
svn co https://www.snolab.ca/snoplus/svn/sasquatch/dev
```

When you first do this, you may be warned that the SSL certificate is not recognized, and asked what you want to do. Accept the SSL certificate permanently (option "p"), and you won't be bothered again. If asked for a password, hit enter, then specify the user "snoplus" and the standard collaboration password or your own if you have one.

For installation instructions, visit the [SASQUATCH Companion](#)

If you'll be doing code development, see also the [Subversion Tips](#).

Documentation

Documentation is available at the [SASQUATCH Companion website](#). It is included with the source distribution: see [sasquatch/dev/doc](#).

Quick Links

- [Simulation Tasks?](#)

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. [WikiFormatting](#) will give you a detailed description of available Wiki formatting commands.

Trac Documentation

- [TracGuide](#) -- Built-in Documentation
- [The Trac project](#) -- Trac Open Source Project
- [Trac FAQ](#) -- Frequently Asked Questions
- [TracSupport](#) -- Trac Support

For a complete list of local wiki pages, see [TitleIndex](#).

Edit this page Attach file Rename page Delete this version Delete page

Download in other formats:

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.

If you find a bug:

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

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

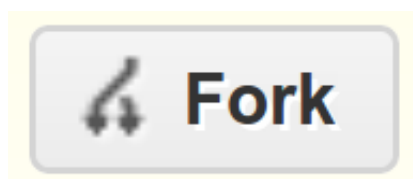
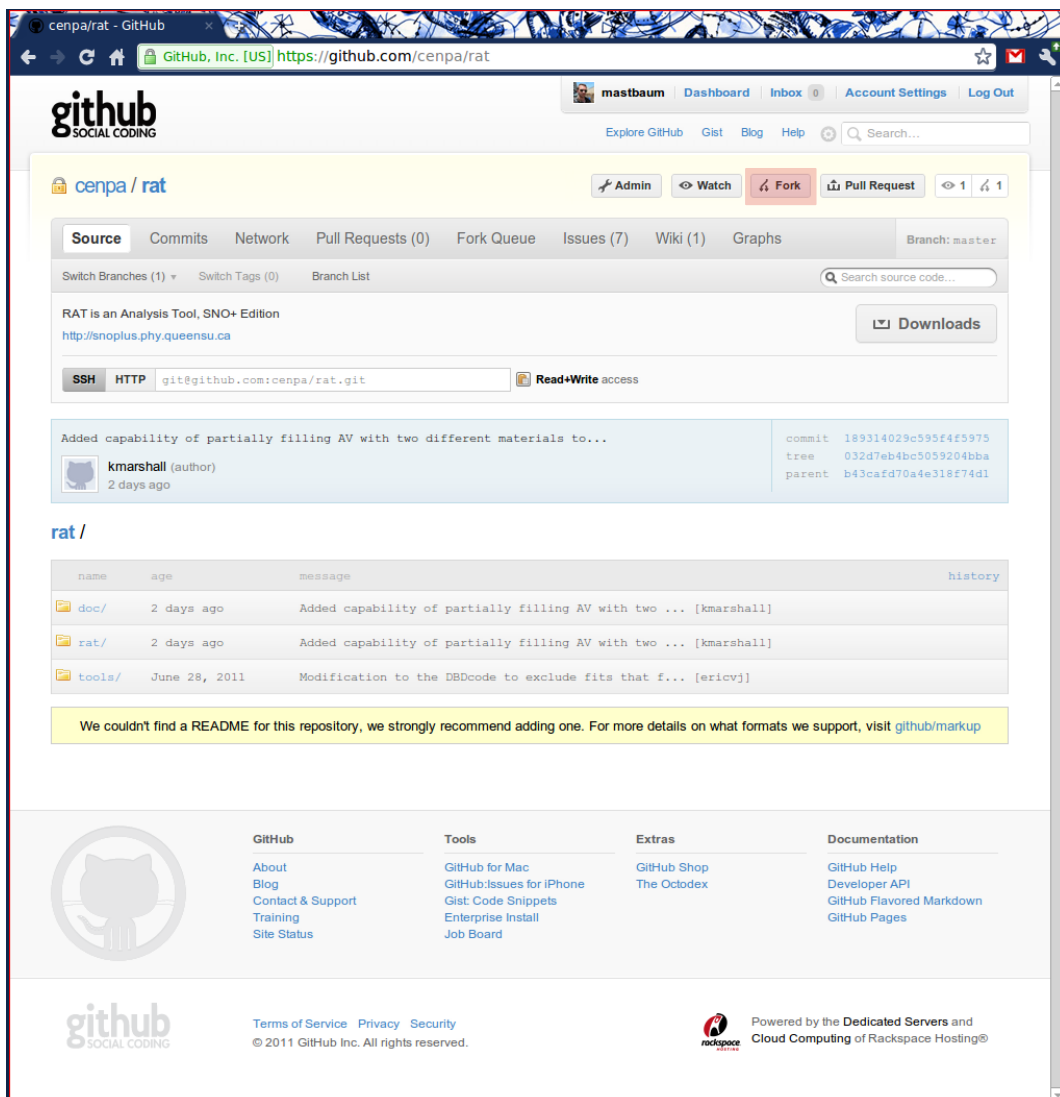
Once you've ensured that your report/request isn't already there, click "New Issue" to add it to the list.

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....

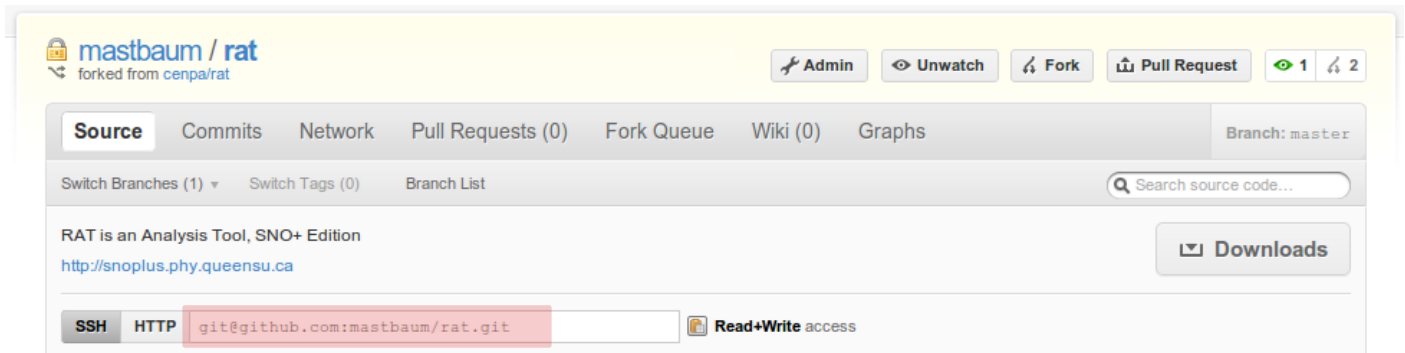
1. Fork the main RAT repository



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:mastbaum/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
```

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")
```

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:mastbaum/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)

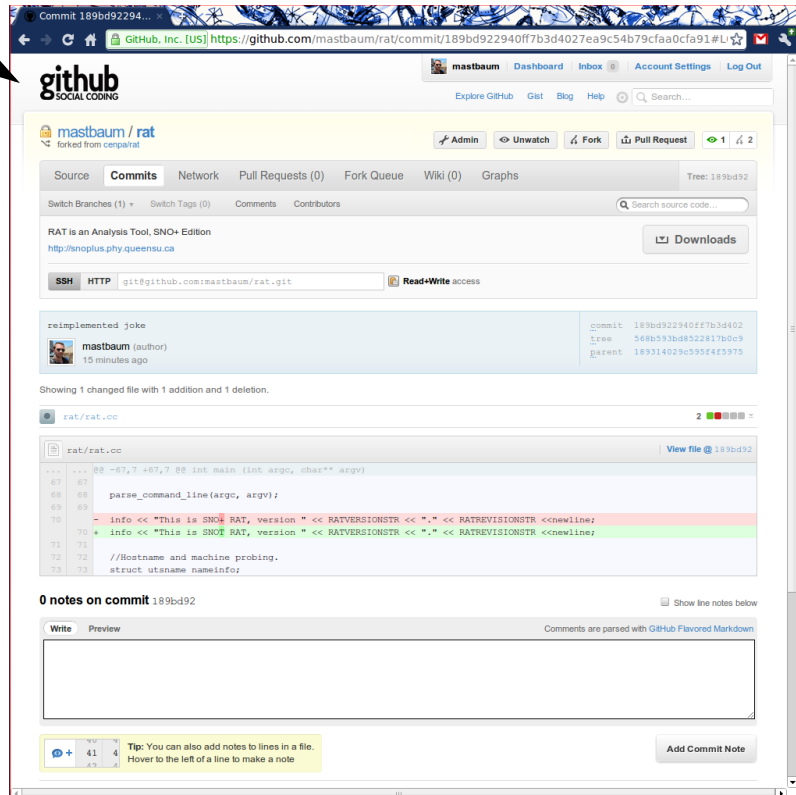
github Workflow

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

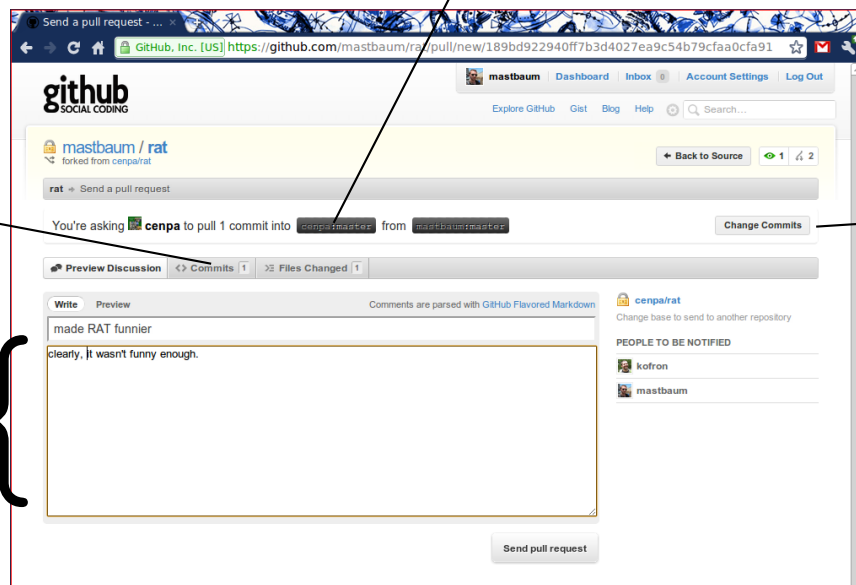
 **Pull Request**

Send to another RAT repository to share patches with collaborators

Preview included changes

Choose which commits are included

Explain why changes were made

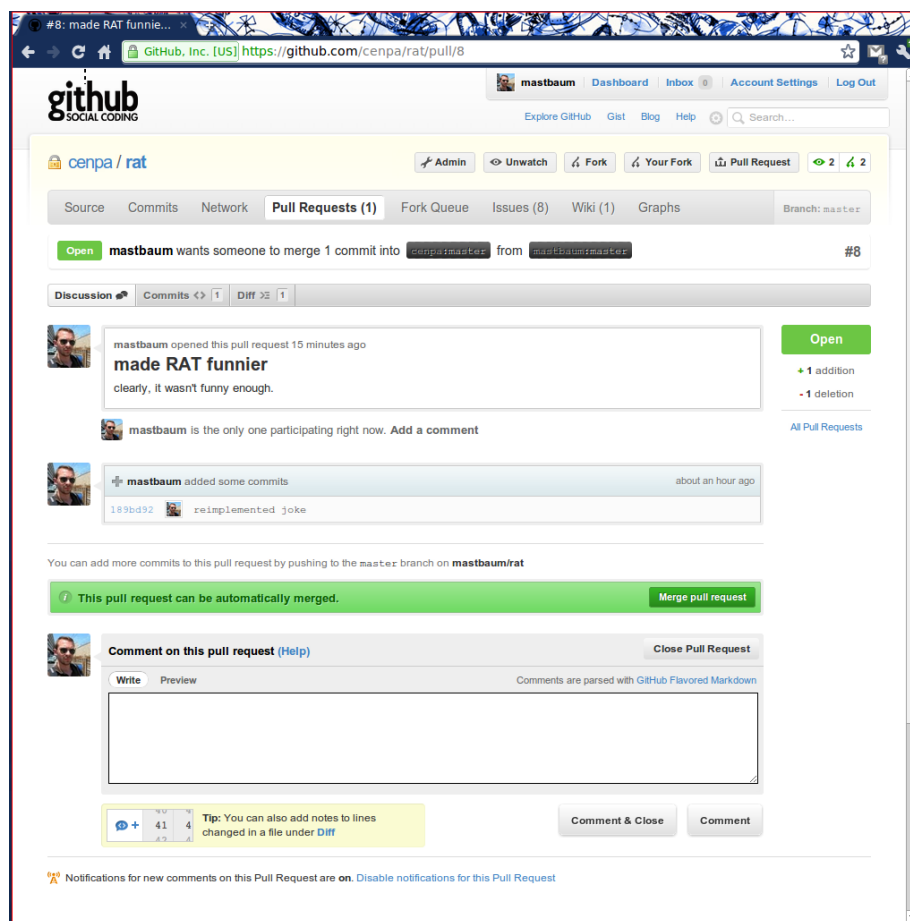


Send pull request

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.

github

Workflow

? Synchronizing your fork with the main • RAT repository

What to do:

```
1 $ git remote add upstream git://github.com/cenpa/rat.git
2 $ git fetch upstream
3 $ git checkout master
4 $ git merge upstream master
```

What it does:

- 1 Add main RAT as another “remote” repository called “upstream”
- 2 Download differences between your repository and the main one
- 3 Check out changes from local repository to current working copy
- 4 Merge in the changes you just checked out. You may need to resolve conflicts at this point if your changes are incompatible with recent changes to the main repository

As always, this is applied to the local repository.
Changes will appear on github when you “git push” them.

? 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/>

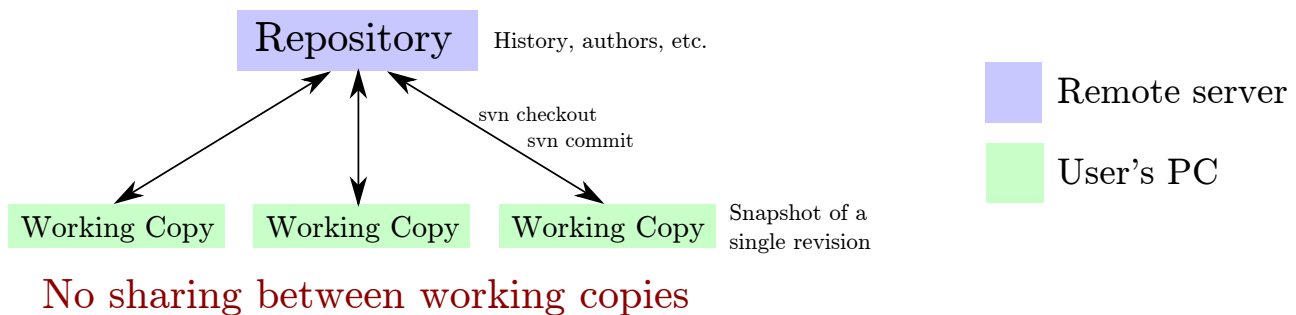
git(hub) vs. SVN

Repository Structure

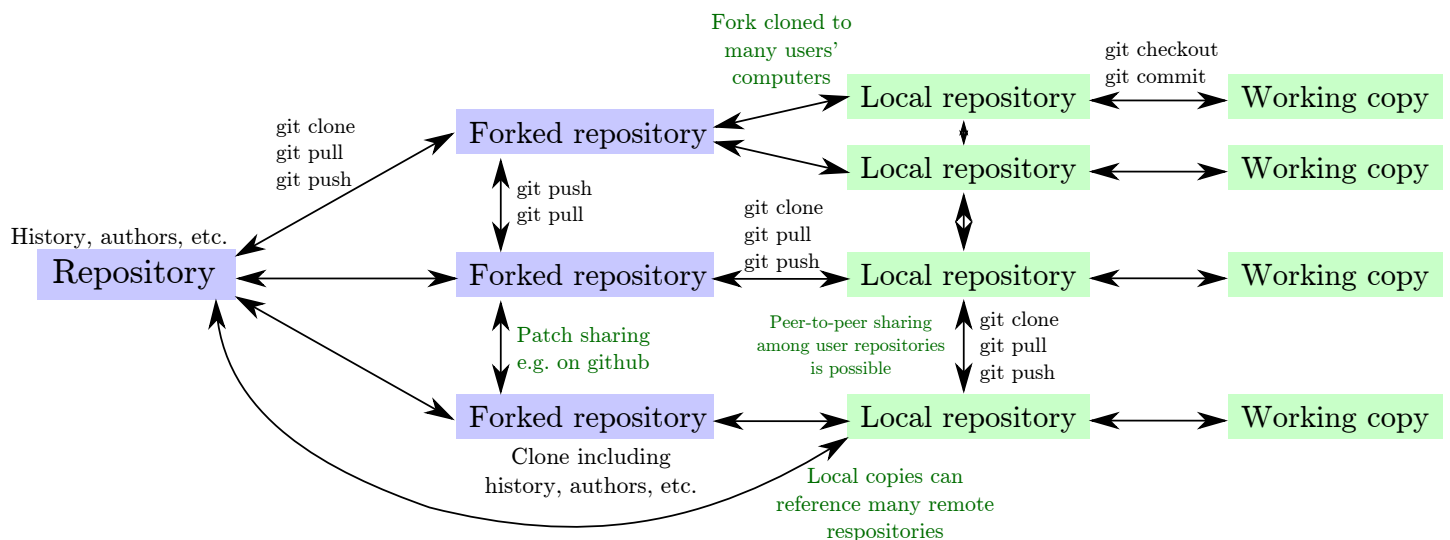
No version control:

```
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:



git 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.