Sage, Git, & Trac

Quickstart

Configuration

You only need to do this once:

```
git config --global user.name "Your Name"
git config --global user.email you@yourdomain.example.com
```

This data ends up in commits, so do it now before you forget!

Get the Sage Source Code

```
git clone https://github.com/sagemath/sage.git
```

Branch Often

A new branch is like an independent copy of the source code. Always switch to a new branch *before* editing anything:

```
git checkout develop                switch to the starting point
git branch new_branch_name                  create new branch
git checkout new_branch_name             switch to new branch
```

Without an argument, the list of branches is displayed:

```
git branch
  master
* new_branch_name                 * marks the current branch
```

When you are finished, delete unused branches:

```
git branch -d branch_to_delete
```

Where Am I?

Each change recorded by git is called a "commit". Examine history:

```
git show                          show the most recent commit
git log                         list in reverse chronological order
```

What Did I Do?

This is probably the most important command. Example output:

```
git status
  On branch new_branch_name              = current branch name
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:   modified_file.py      = file you just edited

Untracked files:
  (use "git add <file>..." to include in what will be
   committed)

        new_file.py                       = file you just added

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

Prepare to Commit

When you are finished, tell git which changes you want to commit:

```
git add filename                                add particular file
git add .                                 add all modified & new
```

The status command then lists the staged changes:

```
git status
On branch new_branch_name
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   modified_file.txt
        new file:   new_file.txt
```

Commit

The commit command permanently records the staged changes. The new commit becomes the new branch head:

```
git commit                        opens editor for commit message
git commit -m "My Commit Message"
```

Commits cannot be changed, but they can be discarded and re-done with the `--amend` switch. *Never* amend commits that you have already shared with somebody.

Summary

workspace is the file system: files that you can edit

```
git add filename                                copy file to staging
git reset HEAD filename                      copy staged file back
```

staging is a special area inside the git repository

```
git commit                                  commit all staged files
```

commits are the permanently recorded history

```
git checkout -- filename          copy file from repo to workspace
```

Merging

A commit with more than one parent is a merge commit:

```
git merge other_branch          incorporate other branch/commit
```

If there is no conflict this automatically creates a new merge commit. Otherwise, the conflicting regions are marked like this:

```
Here are lines that are either unchanged from the common
ancestor, or cleanly resolved because only one side changed.
<<<<<<< yours:source_file.py
Conflict resolution is hard;
let's go shopping.
=======
Git makes conflict resolution easy.
>>>>>>> theirs:source_file.py
And here is another line that is cleanly resolved or unmodified.
```

Edit as needed; To finish, run one of:

```
git commit                  commit your merge conflict resolution
git merge --abort                        discard merge attempt
```

Branch Heads

A git branch is just a pointer to a commit. This commit is called the branch `HEAD`. You can point it elsewhere with (`--hard`) or without (`--soft`, less common) resetting the actual files. That is, the following discards content of the current branch and makes it indistinguishable from a new branch that started at `new_head_commit`:

```
git reset --hard new_head_commit
```

There are various ways to specify a commit to reset to:

```
3472a854df051b57d1cb7e4934913f17f1fef820      40-digit SHA1
3472a85                  the first few digits of the SHA1
branch_name          the name of another branch pointing to it
6.2.beta6    a tag in the Sage git repo; Every version is tagged
origin/develop        the develop branch in the remote origin
HEAD~                              first parent of the current head
HEAD~2       first parent of the first parent of the current head
HEAD^2                            second parent of the current head
FETCH_HEAD   commit downloaded with the git fetch command
```

Trac and the Sage Git Repo

At `http://git.sagemath.org` you can browse our own git repository. On trac tickets, you can click on the links under **Branch:**

Git Trac Subcommand

We have added a `git trac` command to interact with our git and trac server. You can download and temporarily enable it via

```
git clone git@github.com:sagemath/git-trac-command.git
source git-trac-command/enable.sh
```

See the developer guide for how to install it on your system.

Configure Git Trac

To make changes to trac you need to have an account:

```
git trac config --user USER --pass PASS
```

Furthermore, our git repository uses your SSH keys for authentication. Log in on `https://trac.sagemath.org` and go to Preferences → SSH keys.

Downloading / Creating a Branch

```
git trac checkout ticket_number      branch for existing ticket
git trac create "Ticket Title"                create new ticket
```

This will get the branch from trac, or create a new one if there is none yet attached to the ticket.

Pull Changes from Trac

```
git trac pull optional_ticket_number
```

The trac ticket number will be guessed from a number embedded in the current branch name, or if there is a branch of the same name on a ticket already.

Push your Changes to Trac

```
git trac push optional_ticket_number
```

Getting Help

```
git help command              show help for (optional) command
git trac create -h                          help for subcommand
```

Sage developer guide: `https://doc.sagemath.org/html/en/developer/`