Git and GitHub for Beginners

- What is Git
- What Is GitHub
- Using GitHub
  - Forking an Existing Repository
  - Creating a New GitHub Repository
  - Creating a Local Copy of an Existing Repository
  - Keep Your Fork Synced
  - Comparing Repositories
    - List Current Status of Local Repository
    - Compare Differences before Commit
  - Saving Changes
  - Syncing to the Central Repository
  - Updating From Central Repository
- Summary of Basic Git Steps
- Where to Find More Help

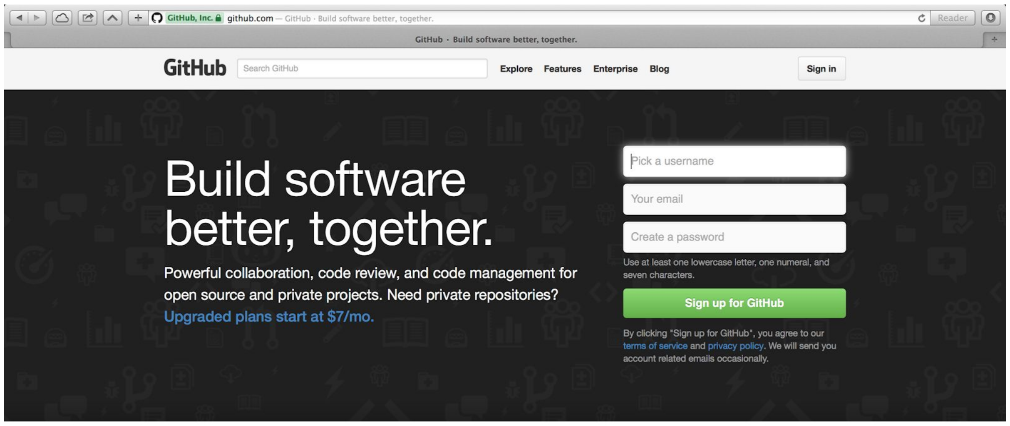
What is Git

Git is an open source version control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.
Every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.

What Is GitHub

GitHub (https://github.com/) is a Git repository web-based hosting service which offers all of the distributed revision control and source code management functionality of Git as well as adding personal features. Unlike Git, which is strictly a command-line tool, GitHub provides a web-based graphical interface and desktop as well as mobile integration.

- To set up your own, go to the GitHub web page and create a login -- "Sign up for GitHub"


GitHub · Build software better, together.

Build software better, together.

Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at $7/mo.

Use at least one lowercase letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.

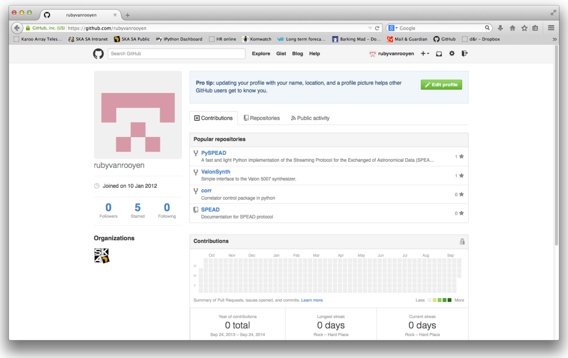
Using GitHub

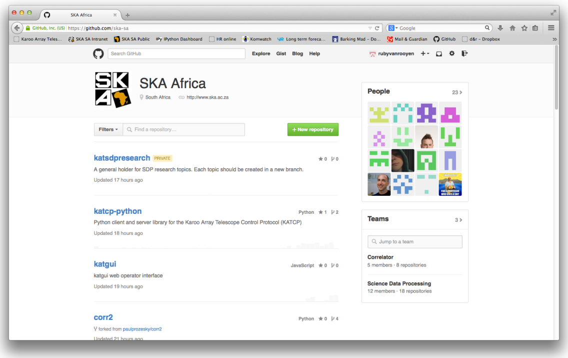
- To set up Git on your personal computer to access your GitHub account using a terminal, follow the instructions on the "Set Up Git" page for your system: https://help.github.com/articles/set-up-git
- Log into GitHub and go to your account page or to your organization account page.

The basic process we will follow

- Fork the repository, or create a new repository to get your own personal repository. This will become your *remote* repository.
- Clone your repository to your computer to actually do some work. This version on your local system is your *local* repository.
- Make your changes to the local repository
- Commit your changes
- Push your changes back to your remote fork on GitHub
- Submit a pull request from the forked repository back to the original repository

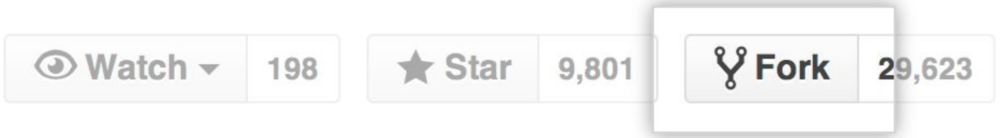
The pull request allows your changes to be pulled into the original repository.





Forking an Existing Repository

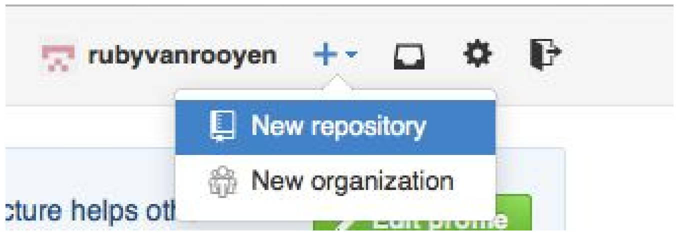
Fork is another way of saying copy. The term fork (in programming) derives from a Unix system call that created a copy of an existing process. **A fork is independent from the original repository.**
Simply navigate to the repository you want to create a private copy of and click on *Fork* in the top-right corner.



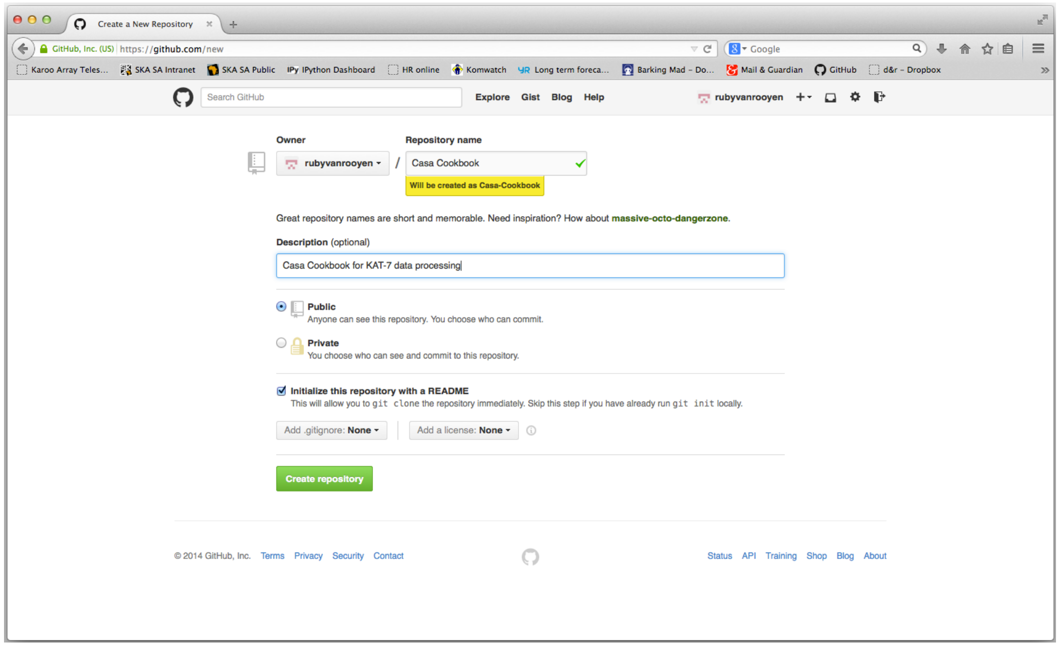
Forking is a *Github* thing, not a Git thing. When you fork a repo on Github, you're essentially making a copy of a repo at a particular point in time to your own account on Github.
Detailed instructions can be found on the GitHub Help pages:
https://help.github.com/articles/fork-a-repo/

Creating a New GitHub Repository

- Create a new repository by clicking on the "+" in the top menu bar, next to your user name.



- Use the web interface to create a new, empty repository.
  Detailed instructions can be found on the GitHub Help pages:
  https://help.github.com/articles/create-a-repo



Owner: rubyvanrooyen / Repository name: Casa Cookbook

Will be created as Casa-Cookbook

Great repository names are short and memorable. Need inspiration? How about **massive-octo-dangerzone**.

Description (optional): Casa Cookbook for KAT-7 data processing

● **Public** – Anyone can see this repository. You choose who can commit.

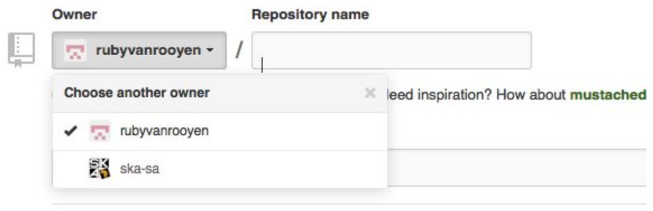
○ **Private** – You choose who can see and commit to this repository.

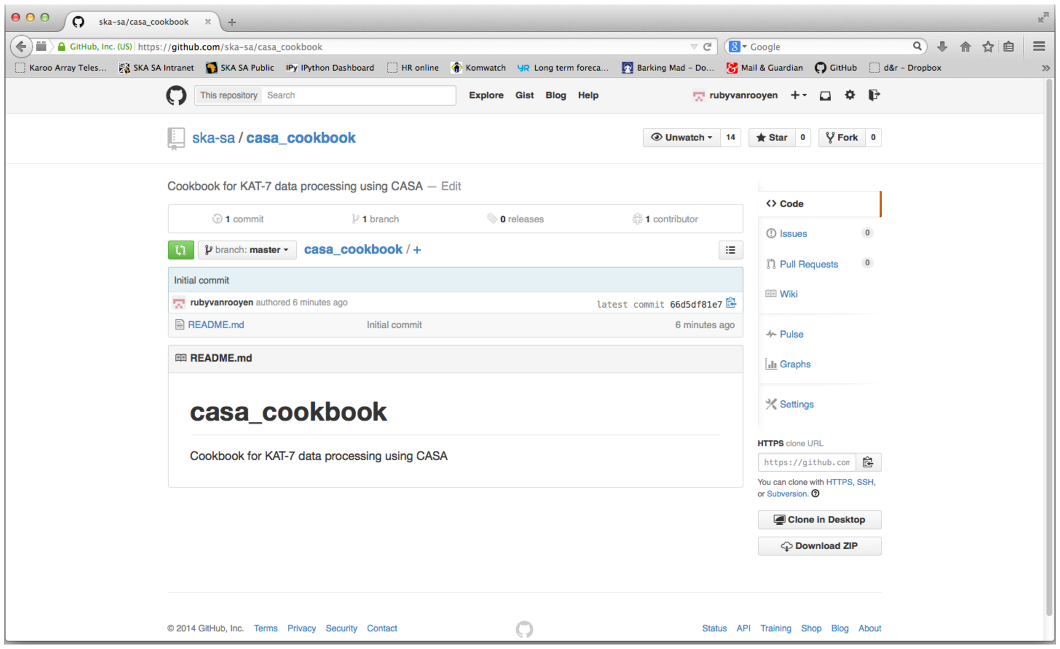
[✓] **Initialize this repository with a README** – This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None**

Create repository
© 2014 GitHub, Inc. Terms Privacy Security Contact
Status API Training Shop Blog About

- To add the repository to an organisation's repository, you can add a new repository from your account, select the organisation under "Owner" on the Create page.
  In this example the SARAO has a Git account named **`ska-sa`**





Cookbook for KAT-7 data processing using CASA — Edit

1 commit | 1 branch | 0 releases | 1 contributor

branch: master casa_cookbook / +

Initial commit

rubyvanrooyen authored 6 minutes ago — latest commit 66d5df81e7

README.md	Initial commit	6 minutes ago

README.md

casa_cookbook

Cookbook for KAT-7 data processing using CASA

You can clone with HTTPS, SSH, or Subversion.

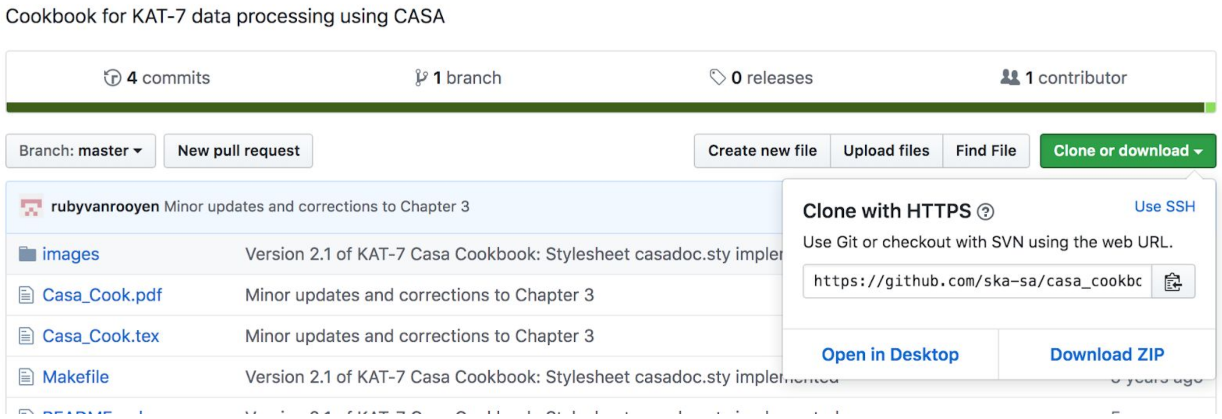
© 2014 GitHub, Inc. Terms Privacy Security Contact
Status API Training Shop Blog About

Creating a Local Copy of an Existing Repository

The **`git clone`** command copies an existing Git repository. This is sort of like **`svn checkout`**, except the "working copy" is a full-fledged Git repository—it has its own history, manages its own files, and is a completely isolated environment from the original repository.
Usage: **`git clone <repo url>`**

To clone a GitHub repository

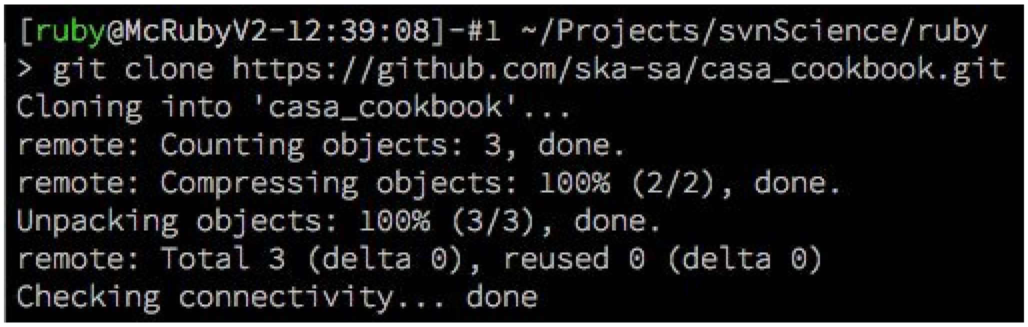
- Enter the repository you wish to add to and copy the "clone URL" available in the right bar menu using the "Copy to clipboard" icon .



rubyvanrooyen Minor updates and corrections to Chapter 3	
images	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty impler
Casa_Cook.pdf	Minor updates and corrections to Chapter 3
Casa_Cook.tex	Minor updates and corrections to Chapter 3
Makefile	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented

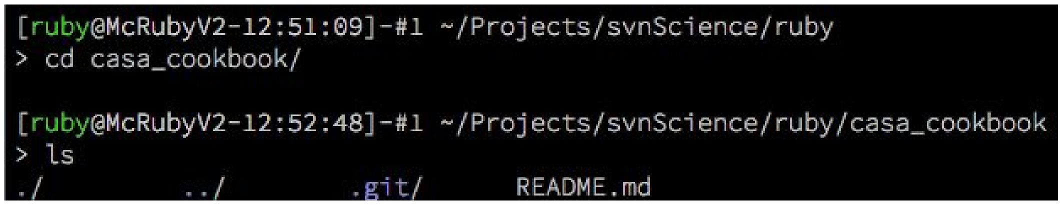
Click on the blue links (top right) to change the kind of access, Use SSH or Use HTTPS. Once you have selected the access type, click on the clipboard to copy the URL. The URL will then be available to "paste" on the command line.

- Navigate to where you want the repository to live and clone it onto your system



```
[ruby@McRubyV2-12:39:08]-#1 ~/Projects/svnScience/ruby
> git clone https://github.com/ska-sa/casa_cookbook.git
Cloning into 'casa_cookbook'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
Unpacking objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Checking connectivity... done
```

- The clone command will create a new repository (directory) with the name "casa_cookbook". Navigate into the directory and add your changes.

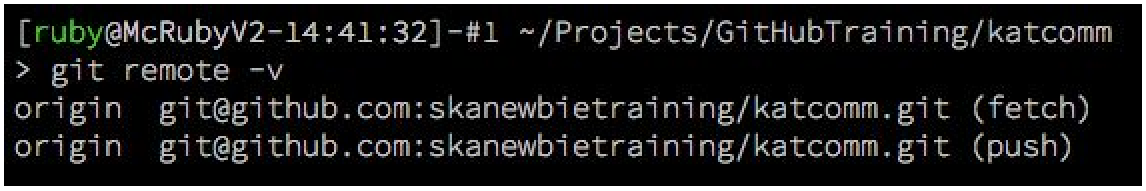


```
[ruby@McRubyV2-12:51:09]-#1 ~/Projects/svnScience/ruby
> cd casa_cookbook/

[ruby@McRubyV2-12:52:48]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> ls
./          ../          .git/       README.md
```

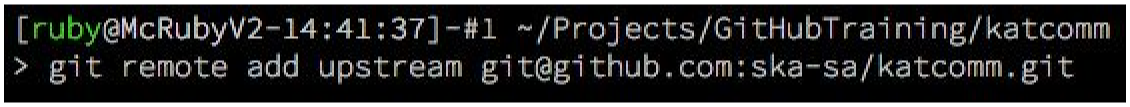
Keep Your Fork Synced

By default, when you clone a repository Git will save the parent repository under the name *origin*, so *origin* on your computer repository will be pointing to your GitHub fork.



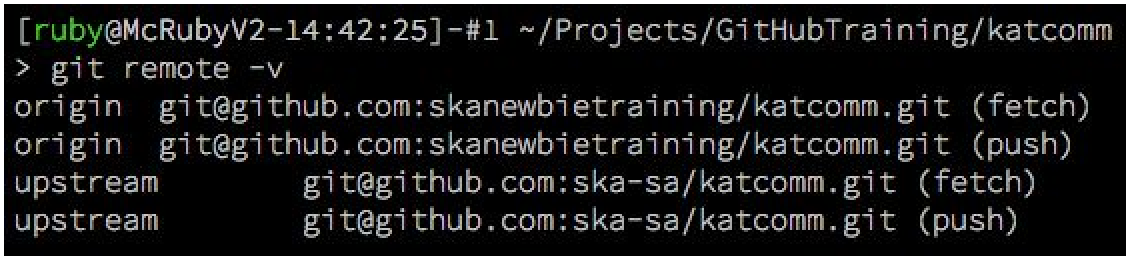
```
[ruby@McRubyV2-14:41:32]-#1 ~/Projects/GitHubTraining/katcomm
> git remote -v
origin  git@github.com:skanewbietraining/katcomm.git (fetch)
origin  git@github.com:skanewbietraining/katcomm.git (push)
```

When you fork a project, you get a copy of it at a specific moment in time. There aren't any built-in ways of automatically getting updates from the original repo after you forked it. You will have to fetch any updates through Git. You can configure Git to pull changes from the original, or *upstream*, repository into the local clone of your fork and to do that you'll need to add another remote that points back to the original repo. To add this, you'll need to be in your clone's directory, then use the `git remote` command to `add` the original repo as a remote.
Usage: `git remote add upstream <repo url>`



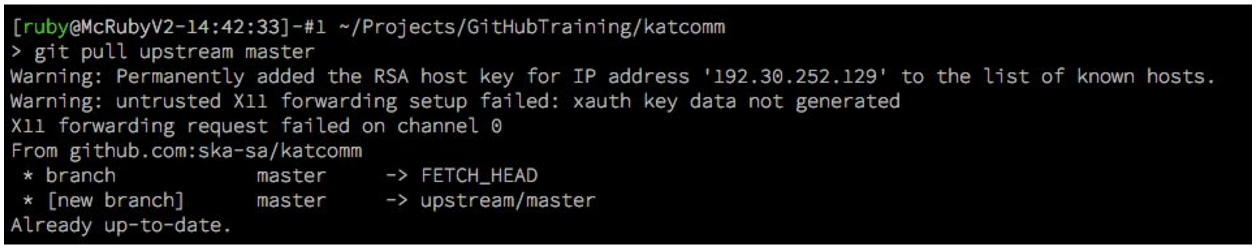
```
[ruby@McRubyV2-14:41:37]-#1 ~/Projects/GitHubTraining/katcomm
> git remote add upstream git@github.com:ska-sa/katcomm.git
```

This creates a remote named *upstream* to the remotes list, and points to the original repository. After adding this upstream remote, you should have *two* remotes. When you clone your fork, that creates the first remote, *origin*, that points to your fork on GitHub and that you can push back updates. The second one, *upstream*, points to the original repo you forked.



```
[ruby@McRubyV2-14:42:25]-#1 ~/Projects/GitHubTraining/katcomm
> git remote -v
origin  git@github.com:skanewbietraining/katcomm.git (fetch)
origin  git@github.com:skanewbietraining/katcomm.git (push)
upstream        git@github.com:ska-sa/katcomm.git (fetch)
upstream        git@github.com:ska-sa/katcomm.git (push)
```

Now you can get any updates from the original project, merge them back into your own repository, then push those back to your fork.
Usage: `git pull upstream master`



```
[ruby@McRubyV2-14:42:33]-#1 ~/Projects/GitHubTraining/katcomm
> git pull upstream master
Warning: Permanently added the RSA host key for IP address '192.30.252.129' to the list of known hosts.
Warning: untrusted X11 forwarding setup failed: xauth key data not generated
X11 forwarding request failed on channel 0
From github.com:ska-sa/katcomm
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master
Already up-to-date.
```

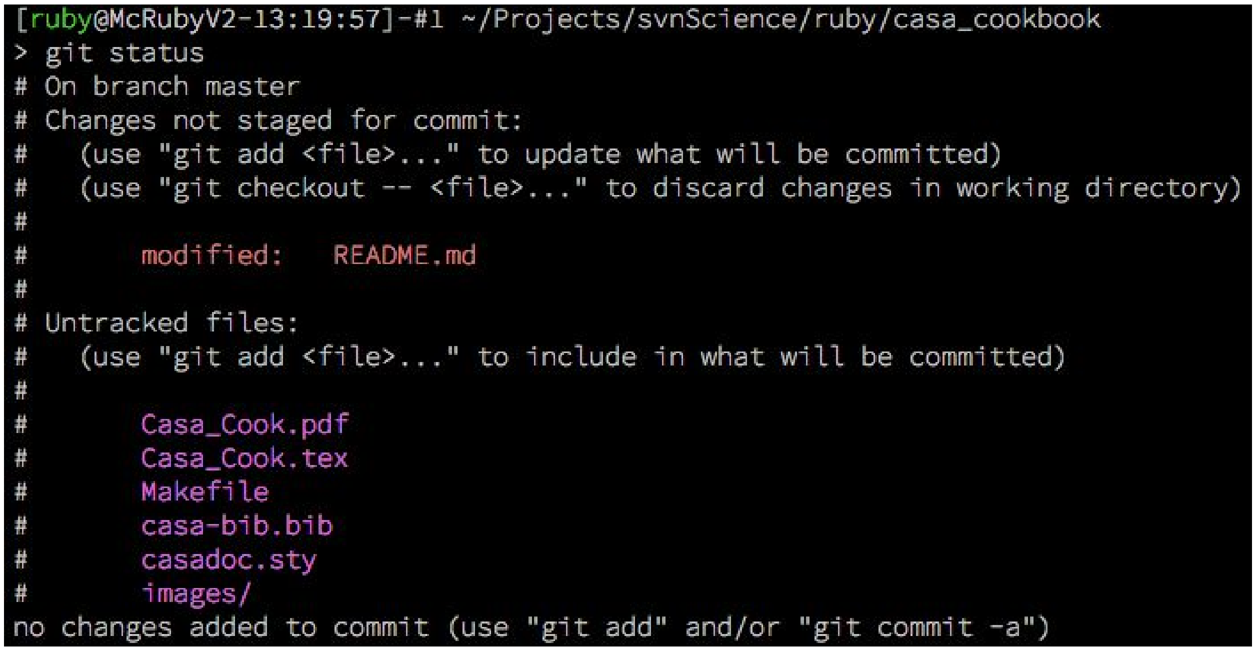
Now that you have your local master branch updated with changes from the original repo, you can start successfully add your own commits.

Comparing Repositories

List Current Status of Local Repository

The **`git status`** command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

Usage: **`git status`**

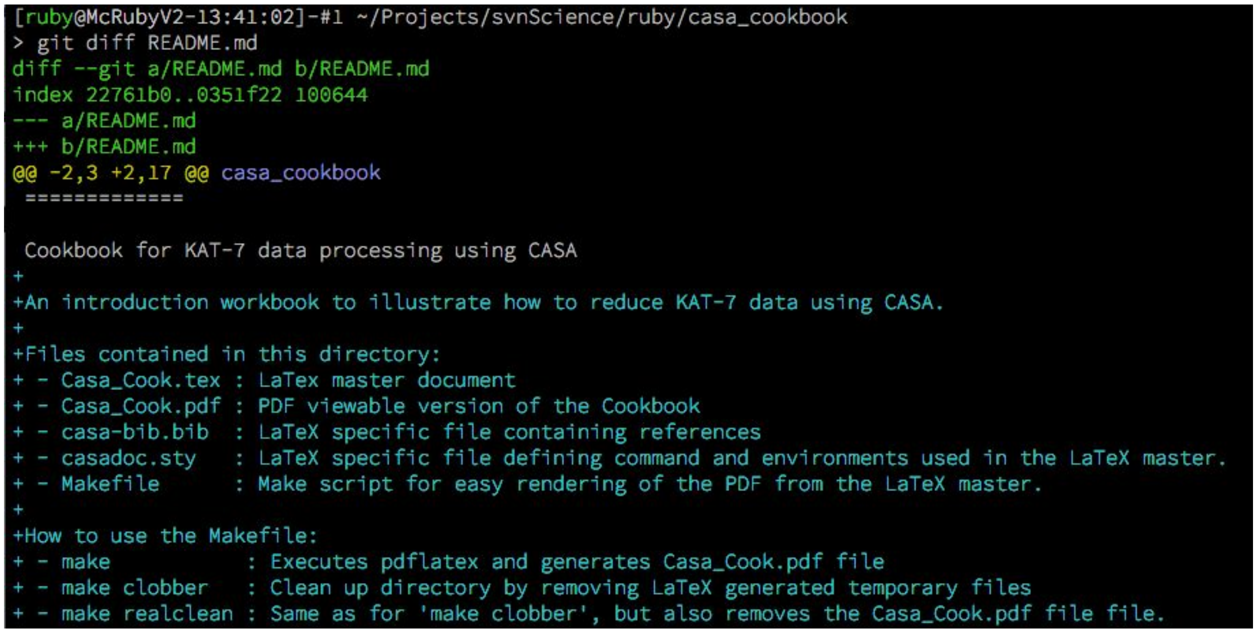


```
[ruby@McRubyV2-13:19:57]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README.md
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       Casa_Cook.pdf
#       Casa_Cook.tex
#       Makefile
#       casa-bib.bib
#       casadoc.sty
#       images/
no changes added to commit (use "git add" and/or "git commit -a")
```

Compare Differences before Commit

The **`git diff`** command shows you the difference between the version of a file in the working directory and most recent commit. This command will show conflicts which can be corrected before rebasing or merging.

Usage: **`git diff <filename>`**



```
[ruby@McRubyV2-13:41:02]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git diff README.md
diff --git a/README.md b/README.md
index 22761b0..0351f22 100644
--- a/README.md
+++ b/README.md
@@ -2,3 +2,17 @@ casa_cookbook
 =============

 Cookbook for KAT-7 data processing using CASA
+
+An introduction workbook to illustrate how to reduce KAT-7 data using CASA.
+
+Files contained in this directory:
+ - Casa_Cook.tex : LaTex master document
+ - Casa_Cook.pdf : PDF viewable version of the Cookbook
+ - casa-bib.bib  : LaTeX specific file containing references
+ - casadoc.sty   : LaTeX specific file defining command and environments used in the LaTeX master.
+ - Makefile      : Make script for easy rendering of the PDF from the LaTeX master.
+
+How to use the Makefile:
+ - make           : Executes pdflatex and generates Casa_Cook.pdf file
+ - make clobber   : Clean up directory by removing LaTeX generated temporary files
+ - make realclean : Same as for 'make clobber', but also removes the Casa_Cook.pdf file file.
```

Saving Changes

The `git add` and `git commit` commands compose the fundamental Git workflow. Developing a project revolves around three basic steps.

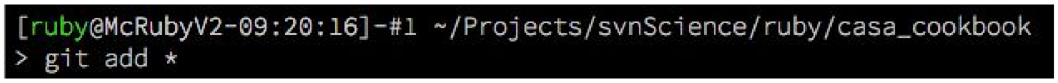
- First, you edit your files in the working directory.
- When you're ready to save the changes, you stage changes with `git add`.
- After that you commit it to the project history with `git commit`.

The staging area is one of Git's more unique features, if you're coming from an SVN background, it helps to think of it as a buffer between the working directory and the project history.

The `git add` command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit.
Usage: `git add <filename>`
or: `git add <directory>`

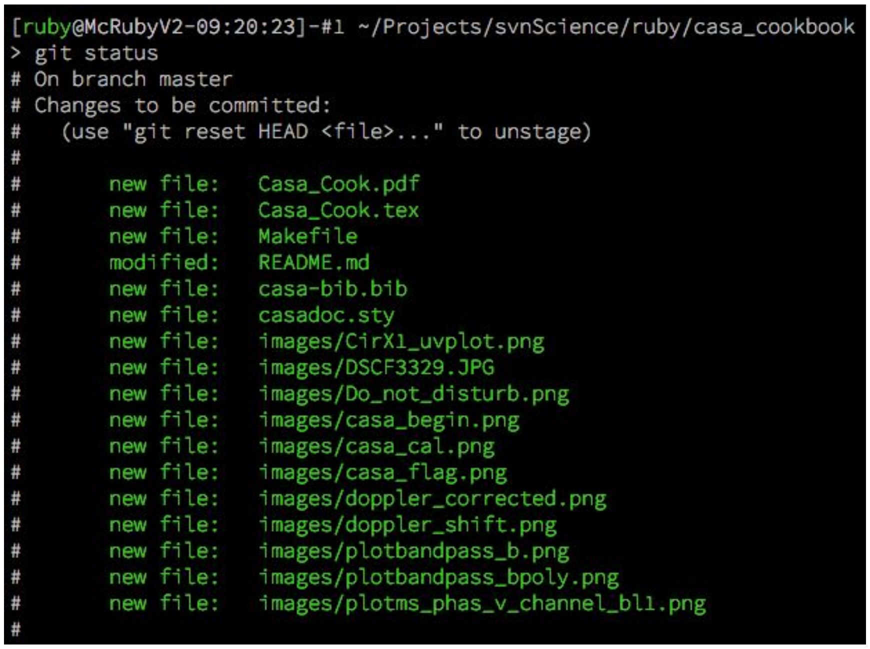
The `git add` command should not be confused with `svn add`, which adds file to the repository. `git add` doesn't really affect the repository in any significant way. This means that `git add` **needs to be called every time you alter a file**, whereas `svn add` only needs to be called once for each file.

- Adding new files/directories and modified files to the staging area.

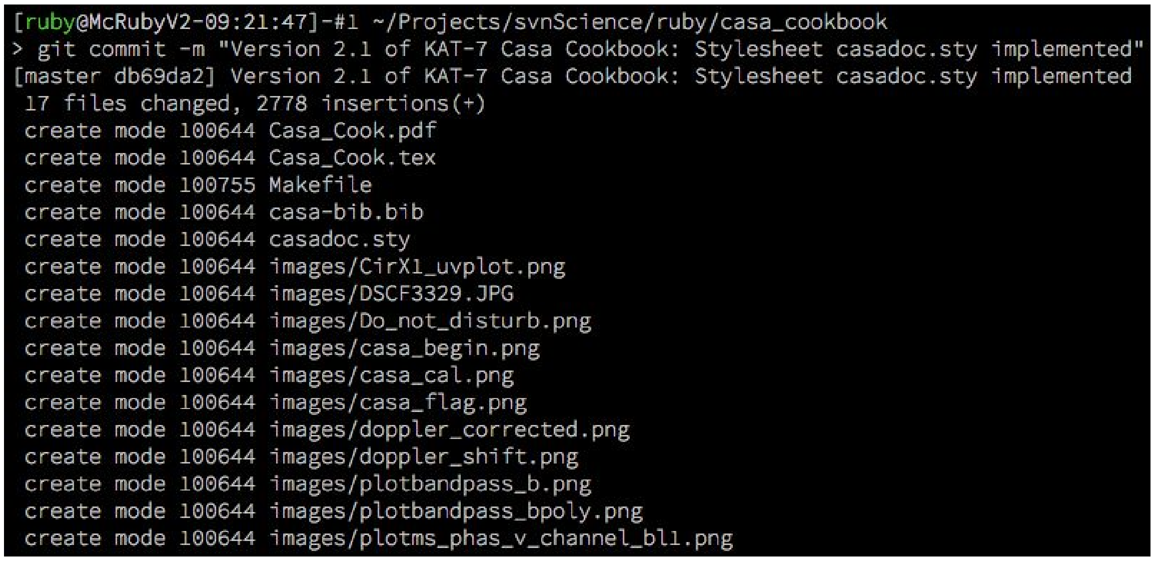


```
[ruby@McRubyV2-09:20:16]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git add *
```

- Verify state of working directory (this is the new snapshot to be committed).

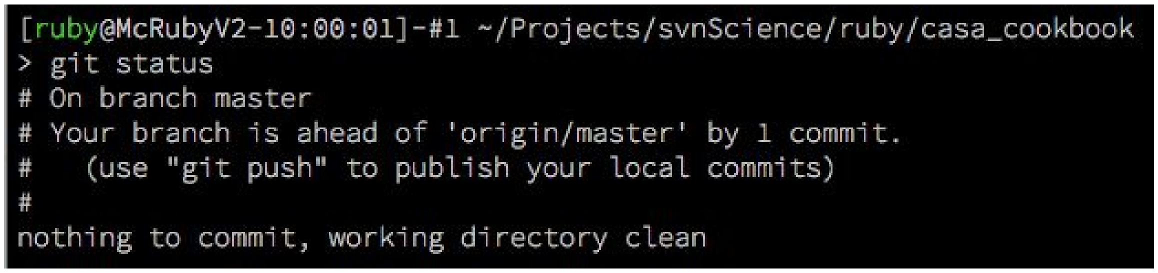

```
[ruby@McRubyV2-09:20:23]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   Casa_Cook.pdf
#       new file:   Casa_Cook.tex
#       new file:   Makefile
#       modified:   README.md
#       new file:   casa-bib.bib
#       new file:   casadoc.sty
#       new file:   images/CirX1_uvplot.png
#       new file:   images/DSCF3329.JPG
#       new file:   images/Do_not_disturb.png
#       new file:   images/casa_begin.png
#       new file:   images/casa_cal.png
#       new file:   images/casa_flag.png
#       new file:   images/doppler_corrected.png
#       new file:   images/doppler_shift.png
#       new file:   images/plotbandpass_b.png
#       new file:   images/plotbandpass_bpoly.png
#       new file:   images/plotms_phas_v_channel_bl1.png
#
```

The `git commit` command commits the staged snapshot to the project history. Changes are not actually recorded until you run `git commit`.
Usage: `git commit -m "Commit Message"`


```
[ruby@McRubyV2-09:21:47]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git commit -m "Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented"
[master db69da2] Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented
 17 files changed, 2778 insertions(+)
 create mode 100644 Casa_Cook.pdf
 create mode 100644 Casa_Cook.tex
 create mode 100755 Makefile
 create mode 100644 casa-bib.bib
 create mode 100644 casadoc.sty
 create mode 100644 images/CirX1_uvplot.png
 create mode 100644 images/DSCF3329.JPG
 create mode 100644 images/Do_not_disturb.png
 create mode 100644 images/casa_begin.png
 create mode 100644 images/casa_cal.png
 create mode 100644 images/casa_flag.png
 create mode 100644 images/doppler_corrected.png
 create mode 100644 images/doppler_shift.png
 create mode 100644 images/plotbandpass_b.png
 create mode 100644 images/plotbandpass_bpoly.png
 create mode 100644 images/plotms_phas_v_channel_bl1.png
```

While they share the same name, this command is nothing like `svn commit`. Snapshots are committed to the local repository, and this requires absolutely no interaction with other Git repositories.

Snapshots are always committed to the *local* repository. This is fundamentally different from SVN, wherein the working copy is committed to the central repository. In contrast, Git doesn’t force you to interact with the central repository until you're ready. Just as the staging area is a buffer between the working directory and the project history, each developer's local repository is a buffer between their contributions and the central repository.

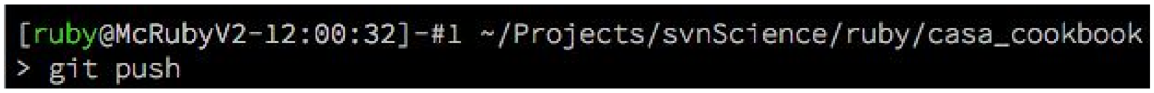


```
[ruby@McRubyV2-10:00:01]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
```

Syncing to the Central Repository

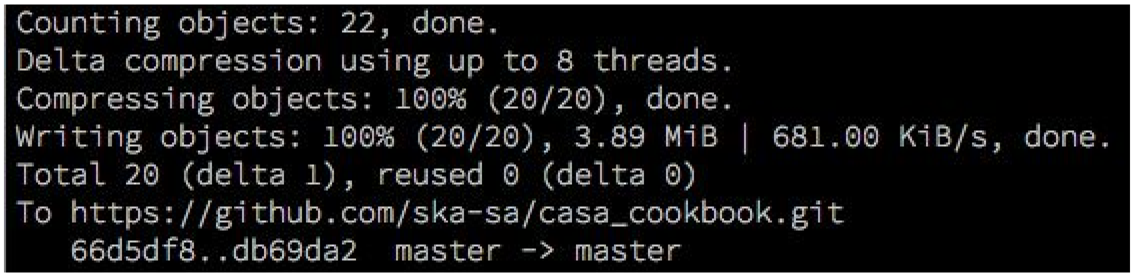
Git is designed to give each developer an entirely isolated development environment. This means that information is not automatically passed back and forth between repositories. Instead, developers need to manually `pull` upstream commits into their local repository or manually `push` their local commits back up to the central repository.

Pushing is how you transfer commits from your local repository to a remote repository.
Usage: `git push`
    Or: `git push <remote>`



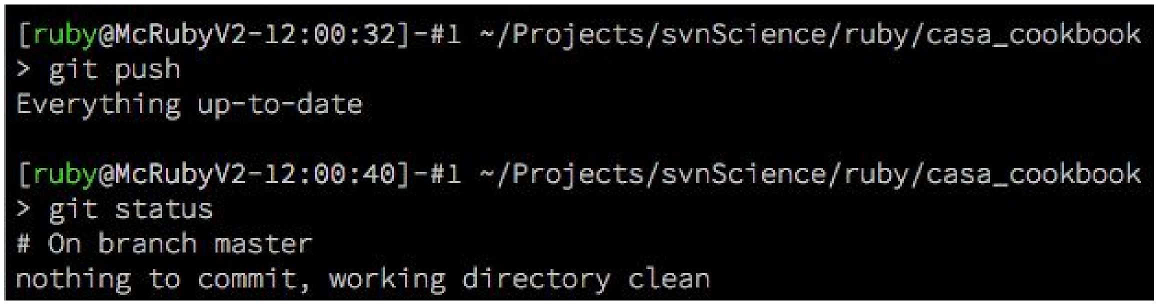
```
[ruby@McRubyV2-12:00:32]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git push
```

Which will show update to the central, master repository.



```
Counting objects: 22, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (20/20), done.
Writing objects: 100% (20/20), 3.89 MiB | 681.00 KiB/s, done.
Total 20 (delta 1), reused 0 (delta 0)
To https://github.com/ska-sa/casa_cookbook.git
   66d5df8..db69da2  master -> master
```

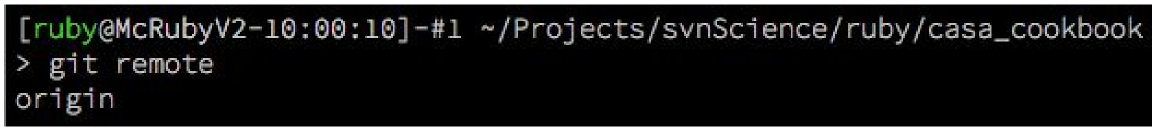
You can view the current status of your local repository to verify.



```
[ruby@McRubyV2-12:00:32]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git push
Everything up-to-date

[ruby@McRubyV2-12:00:40]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git status
# On branch master
nothing to commit, working directory clean
```

List the remote connections you have to other repositories.
Usage: `git remote`

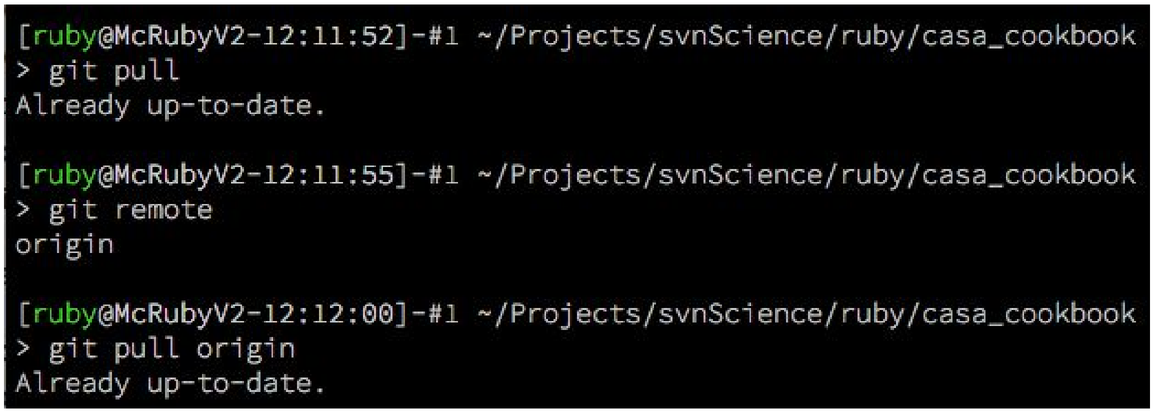


```
[ruby@McRubyV2-10:00:10]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git remote
origin
```

Updating From Central Repository

Merging upstream changes into your local repository by fetching the specified remote's copy of the current branch and immediately merge it into the local copy, using the `git pull` command.

You can think of `git pull` as Git's version of `svn update`. It's an easy way to synchronize your local repository with upstream changes.
Usage: `git pull`
    Or: `git pull <remote>`

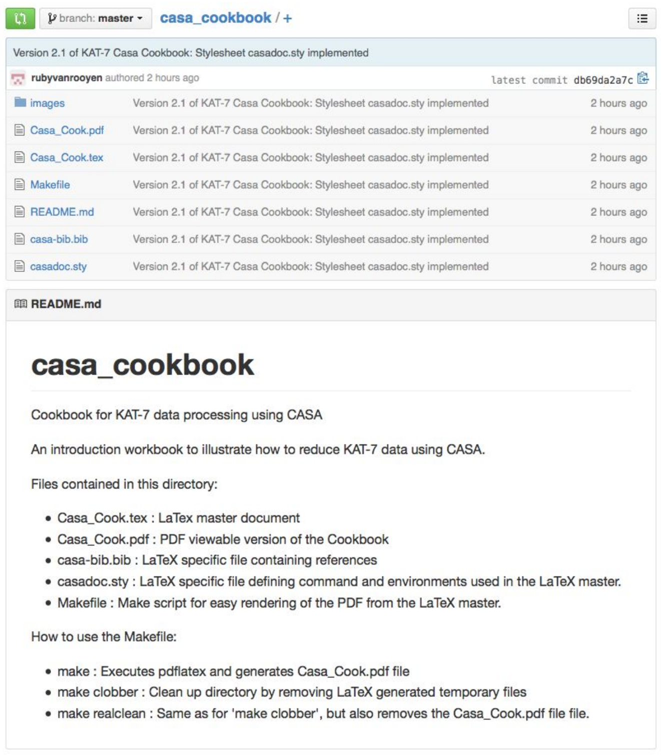


```
[ruby@McRubyV2-12:11:52]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git pull
Already up-to-date.

[ruby@McRubyV2-12:11:55]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git remote
origin

[ruby@McRubyV2-12:12:00]-#1 ~/Projects/svnScience/ruby/casa_cookbook
> git pull origin
Already up-to-date.
```

Remote repository on GitHub can be viewed and accessed anywhere via the web.


branch: master casa_cookbook / +

Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented

rubyvanrooyen authored 2 hours ago latest commit db69da2a7c

images	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
Casa_Cook.pdf	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
Casa_Cook.tex	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
Makefile	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
README.md	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
casa-bib.bib	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago
casadoc.sty	Version 2.1 of KAT-7 Casa Cookbook: Stylesheet casadoc.sty implemented	2 hours ago

README.md

casa_cookbook

Cookbook for KAT-7 data processing using CASA

An introduction workbook to illustrate how to reduce KAT-7 data using CASA.

Files contained in this directory:

- Casa_Cook.tex : LaTex master document
- Casa_Cook.pdf : PDF viewable version of the Cookbook
- casa-bib.bib : LaTeX specific file containing references
- casadoc.sty : LaTeX specific file defining command and environments used in the LaTeX master.
- Makefile : Make script for easy rendering of the PDF from the LaTeX master.

How to use the Makefile:

- make : Executes pdflatex and generates Casa_Cook.pdf file
- make clobber : Clean up directory by removing LaTeX generated temporary files
- make realclean : Same as for 'make clobber', but also removes the Casa_Cook.pdf file file.

Summary of Basic Git Steps

Git command	Description
`git clone </path/to/repository>` or `git clone <username@host:/path/to/repository>`	Create a working copy of a local or remote repository
`git pull`	Fetch and merge changes on the remote server to your working directory.
`git add <filename>` or `git add *`	Add one or more files to staging (also called index).
`git commit -m "Commit message"`	Commit changes to head, but not to the remote repository.
`git push origin master`	Send changes to the master branch of your remote repository.
`git status`	List the files you've changed and those you still need to add or commit

Where to Find More Help

- A very good and explanatory tutorial for all features Git can be found online:
  https://www.atlassian.com/git/tutorials/
- GitHub has its own help page which is very useful
  https://help.github.com/
- Basic Git Commands
  https://confluence.atlassian.com/display/STASH/Basic+Git+commands
- Git for SVN users
  http://www.git-tower.com/blog/git-for-subversion-users-cheat-sheet-detail/