Understanding Git management

We have already created our local repository **gittest** (should be with the same name as the name of your repository on GitHub) with the following commands:

```
$ mkdir gittest
$ cd gittest
```

Next we put our local folder **gittest** under Git management by typing:

```
$ git init
Initialized empty Git repository in e:/gittest/.git/
```

The above command needs to be executed only *once*.

The **git init** command created an empty Git repository or reinitialized an existing one. It created a repository in the current directory i.e. **`gittest`**

Next, the following command created an empty **`README.md`** file at the root of our project:

```
$ touch README.md
```

The above **touch** command created the file **`README.md`,** if the file did not already exist. If a file already existed, the accessed / modification time is updated for that file.

Staging

Note: In Git, you "stage" things before you commit them. You do this with the **git add** command (for example: **git add README.md**). This adds specific content to the 'stage'.

To *stage all changes* and new files, type:

```
$ git add .
```

The **git add .** command will take the working directory and all sub-directories and every single file, i.e. it adds all content to the 'stage' (this snapshot is now stored in a temporary staging area which Git calls the "index").

If you make any changes to a file after staging (but before committing), you'll need to **`git add`** the file again.

You can see what has and has not been staged and/or committed by typing:

```
$ git status
```

git reset HEAD

You can remove a file from staging with **`git reset HEAD <filename>`**.
HEAD is the snapshot of your last commit.

Let us say that I make some changes to the README.md file and committed it:

```
$ git add README
```

I now realize that I wanted to add something more to the README.md file.

The **git status** command reminds you:

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.md
#
```

Right below the "Changes to be committed" text, it says use git reset HEAD <file>... to unstage. So, let's use that advice to unstage the README.md file:

```
$ git reset HEAD README.md
Unstaged changes after reset:
```

The README.md file is unstaged.

The **git status** command shows:

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

Commit

When you get your changes just the way you want them added to the current revision, then you need to commit that revision to your local repository (this permanently stores the contents of the index in the repository).

We already have something staged and ready and it is time to *commit* it. We do it as follows:

```
git commit -m "brief description of commit"
```

Whenever you make a significant change while working on a project, get into the habit of committing it, even if it is only a line or two. It is quick and easy, and you will thank yourself later. Replace '**brief description of commit'** with a comment like **'First commit'**

```
$ git commit -m "First commit"
[master (root-commit)]: created 9311786: "First commit"
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

To check which files have been committed to the local repo, type:

```
$ git ls-files
README.md
```

Viewing the Commit History

After you have created several commits you'll probably want to look back to see what has happened. The most basic and powerful tool to do this is the git log command.

```
$ git log --pretty=format:"%h - %an, %ar : %s"
22a7b30 - Satish Talim, 5 hours ago : my first

where:

%h    Abbreviated commit hash
%an   Author name
%ar   Author date, relative
%s    Subject
```

git revert

git revert is used to record some new commits to reverse the effect of some earlier commits (often only a faulty one).

Example

```
git revert HEAD~3
```

Revert the changes specified by the fourth last commit in HEAD and create a new commit with the reverted changes.

```
git revert -n master~5..master~2
```

Revert the changes done by commits from the fifth last commit in master (included) to the third last commit in master (included), but do not create any commit with the reverted changes. The revert only modifies the working tree and the index.

Push changes to your repository on GitHub

Refer to the following url for details of the **git remote** command:
http://www.kernel.org/pub/software/scm/git/docs/git-remote.html

Type in the following:

```
git remote add origin https://github.com/SMTalim/gittest.git
```

(The *above* command needs to be executed only once in case you are not going to change the remote repository. **Note**: Substitute **SMTalim** with your **GitHubName** and **gittest** with *your* repository name on GitHub).

You can also confirm that this command has successfully executed by typing:

```
$ git remote show origin
```

On the contrary, if you type:

```
$ git remote show -n origin
```

In this case the command queries the local cache.

Once your changes are committed to your local repository, you need to *push* them to the remote repository for others to get at. To do that, you need to execute `git push`, which will push all the changes from your local repository up to the remote repository.

Git push takes several arguments:
```
git push <repository> <branch>
```

In this case, we want to push changes back to the original repository, which is aliased as `origin`, to the *master* branch.

Now type:

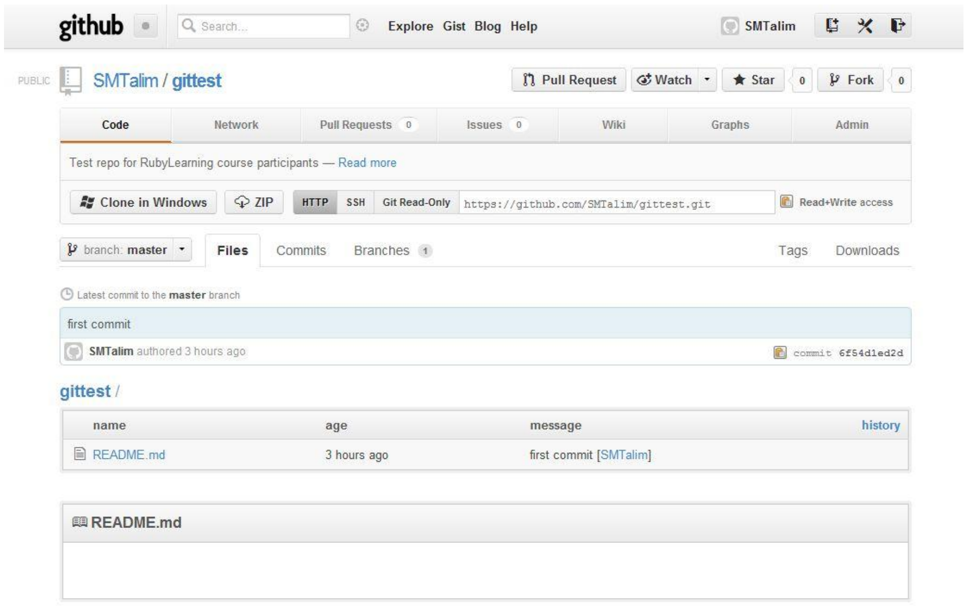
```
$ git push -u origin master
```

This command tells Git that you have a remote repository there and the **git push** command uploads your (committed) changes to Github.

Note: In case Git tells you that it failed to push some references, then refer to this blog post –

http://edspencer.net/2008/04/when-git-tells-you-it-failed-to-push.html

After pushing, here's how my repo looks on GitHub:



Test repo for RubyLearning course participants — Read more

https://github.com/SMTalim/gittest.git

Latest commit to the **master** branch

first commit

SMTalim authored 3 hours ago commit 6f54d1ed2d

gittest /

name	age	message
README.md	3 hours ago	first commit [SMTalim]

Note: Whenever you have committed changes that you wish to put into your Github repository in the future, just type:

```
$ git push origin master
```

Exercise 2

Modify the **README.md** with information about your repo and then push your changes to your GitHub repo. Post the URL (http://) of *your* GitHub repo.

Normal workflow

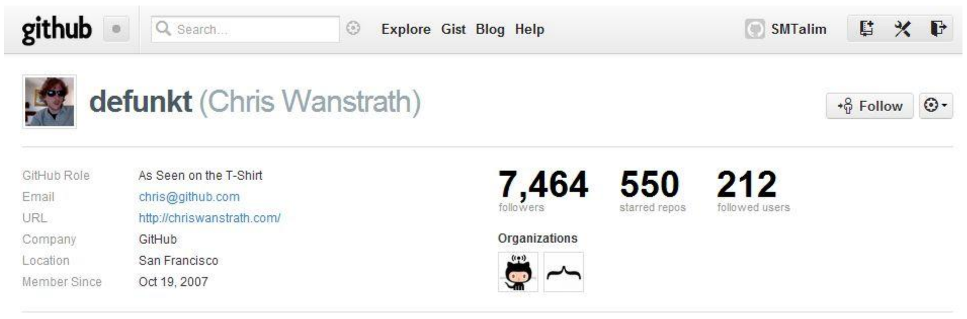
As we will soon see, once you have a Git repository and everything setup, the workflow is not going to be much different from working with any other source control system, the only real difference should be the staging process. The workflow will generally go something like this:

- synchronize with remote repository
- modify files locally
- see what you've changed locally
- stage the changes you want to commit, locally
- commit your staging area, locally
- rinse, repeat if necessary
- push to remote repository

Follow a Friend

One of the great features on GitHub is the ability to see what other people are working on and who they are connecting with. When you follow someone, you will get notifications on your dashboard about their GitHub activity.

Once you are on one of their pages, click the "**Follow**" button. You can follow me too!
https://github.com/SMTalim

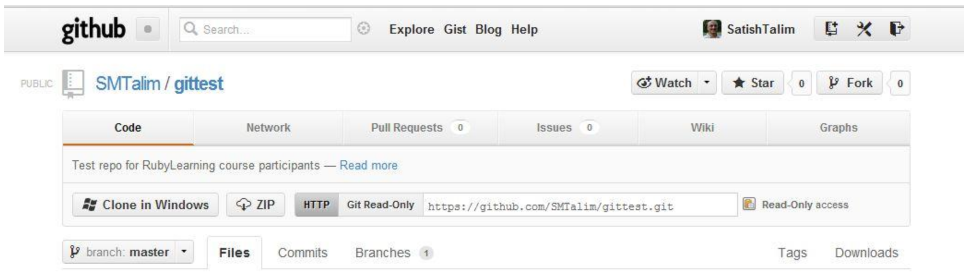


GitHub Role	As Seen on the T-Shirt
Email	chris@github.com
URL	http://chriswanstrath.com/
Company	GitHub
Location	San Francisco
Member Since	Oct 19, 2007

7,464 followers 550 starred repos 212 followed users

Watch projects

Go to my GitHub a/c and click on the **gittest** repo. Once you are on the project page, you will notice there is a "**Watch**" button at the top of the page. Click on it.



Test repo for RubyLearning course participants — Read more

https://github.com/SMTalim/gittest.git

Congratulations! You are now watching my **gittest** project. If I update it, you will see what happened in your dashboard.

Forking a repository

At some point you may find yourself wanting to contribute to someone else's project, or would like to use someone's project as the starting point for your own. This is known as "forking." Let us see an example of forking such a public repository. Ensure that you are logged into your GitHub account.

We shall fork RubyLearning's mentor Satoshi Asakawa's repository here –
https://github.com/ashbb/ruby_learning_participants

Go to the above url and click on the “Fork” button. You've successfully forked the repo, but so far it only exists on GitHub. To be able to work on the project, you will need to clone it to your local machine.

Also, whenever you fork a repository, you automatically watch that repository in your GitHub account. However, *fork does not automatically create a local folder*.

Now you need to *clone* the fork.

Make sure you use the “Your Clone URL” and **not** my Clone URL.

Remember you should be in your **gittest** folder.

Type:

```
$ git clone https://github.com/SMTalim/ruby_learning_participants.git
```

Remember that your clone url will be different.

Once the clone is complete, your repo will have a remote named “origin” that points to your fork on github. Don’t let the name confuse you, this **does not** point to the original repo you forked from.

Let’s confirm that, by typing:

```
$ cd ruby_learning_participants/
$ git remote show origin
```

To keep track of the original repo, you need to add another remote named **upstream**:

```
$ git remote add upstream
https://github.com/SMTalim/ruby_learning_participants.git

$ git fetch upstream
```

Pull in upstream changes

If the original repo you forked your project from gets updated, you can add those updates to your fork by running the following code:

```
$ git fetch upstream
$ git merge upstream/master
```

Pushing our changes

Now that we’ve got our fork, we need to make a few changes and commit them locally.

I added a comment to the file **README.txt**, namely:

```
Cheers from Satish Talim
```

In the **`ruby_learning_participants folder,`** I then typed:

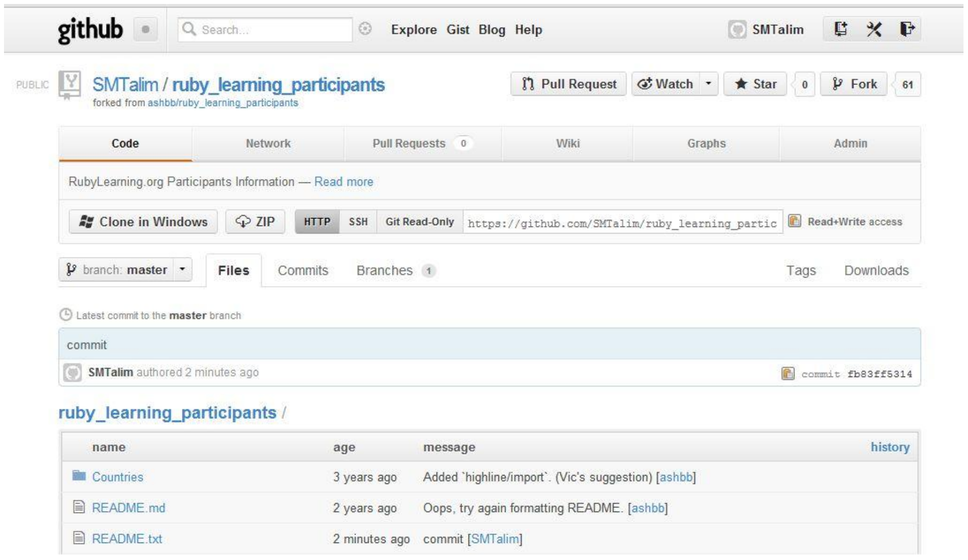
```
$ git add README.txt
$ git commit -m "added line in README.txt"
```

Once you've done this, it's time to push your updated branch.

```
$ git push origin master
```

Sending a pull request

After you've pushed your commit(s) you need to inform the project owner (ashbb) of the changes so they can pull them into their repo. From your project's page, click the "**Pull Request**" button (as shown in the next screenshot).



RubyLearning.org Participants Information — Read more

https://github.com/SMTalim/ruby_learning_partic

Latest commit to the **master** branch

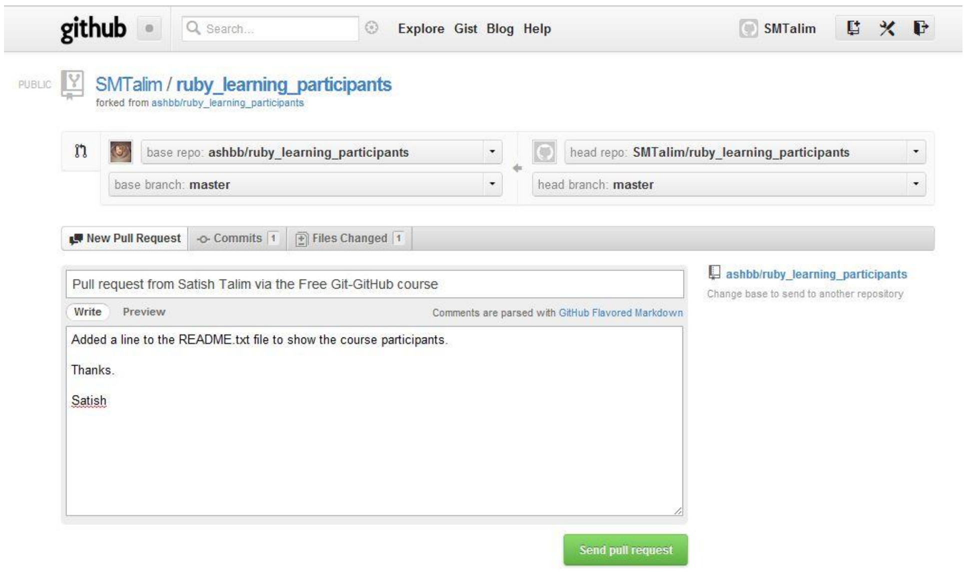
commit

SMTalim authored 2 minutes ago — commit fb83ff5314

ruby_learning_participants /

name	age	message
Countries	3 years ago	Added `highline/import`. (Vic's suggestion) [ashbb]
README.md	2 years ago	Oops, try again formatting README. [ashbb]
README.txt	2 minutes ago	commit [SMTalim]

Fill in a note and pick who to send the request to, as can be seen in the next screenshot -


github Search... Explore Gist Blog Help SMTalim

PUBLIC SMTalim / ruby_learning_participants
forked from ashbb/ruby_learning_participants

base repo: ashbb/ruby_learning_participants
base branch: master
head repo: SMTalim/ruby_learning_participants
head branch: master

New Pull Request | Commits 1 | Files Changed 1

Pull request from Satish Talim via the Free Git-GitHub course

Write Preview Comments are parsed with GitHub Flavored Markdown

Added a line to the README.txt file to show the course participants.

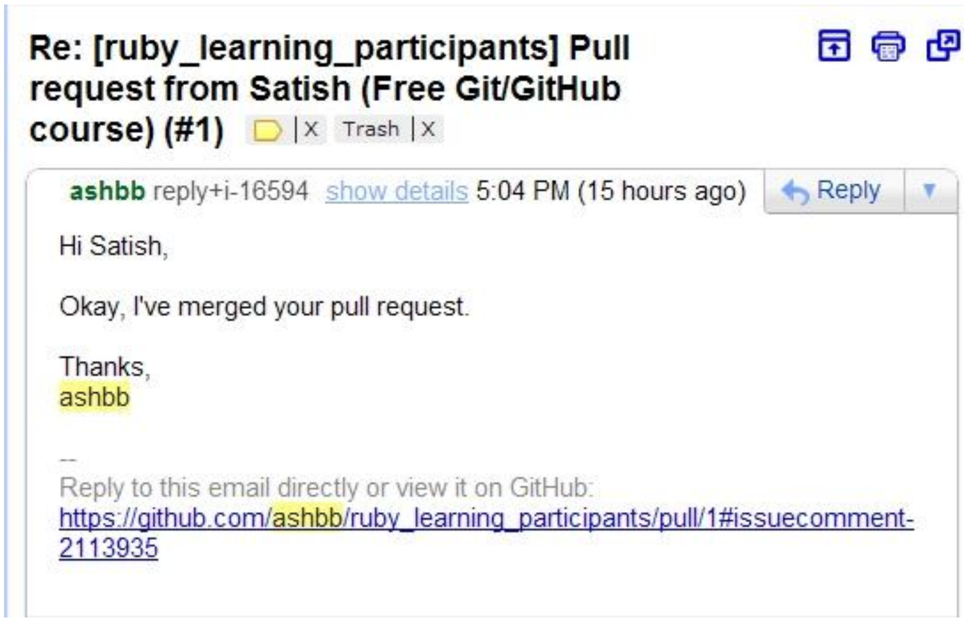
Thanks.

Satish

ashbb/ruby_learning_participants
Change base to send to another repository

Send pull request

Let us assume that ashbb has accepted my pull request. I will receive an email as shown:


Re: [ruby_learning_participants] Pull request from Satish (Free Git/GitHub course) (#1) | X Trash | X

ashbb reply+i-16594 show details 5:04 PM (15 hours ago) Reply

Hi Satish,

Okay, I've merged your pull request.

Thanks,
ashbb

--
Reply to this email directly or view it on GitHub:
https://github.com/ashbb/ruby_learning_participants/pull/1#issuecomment-2113935

Please note: Kindly, DO NOT SEND **ashbb** your pull request. You will get a chance to send a pull request in the exercise that follows in Day 3.