TUTORIAL GIT AND GITHUB EXPRESS - 2018

Note on this one page express tutorial

It may take from days to years to learn git. According to an expert on git, he is still learning git even after more than 5 years of working on git. However, in this page, I'll try to make the process of learning as simple as possible: one chapter tutorial!

Though we can use GUI version, git is absolutely a command line utility. So, we'll be working mostly on command line mode. As we learn git, we'll realize the command line is really a first class citizen, and it's the core of a git.

After finishing this one page tutorial, we'll learn:

1. Git install
2. Creating a git repository
3. Adding a file to a git
4. Github - repository
5. Collaborative working via forking central repo
6. Github - clone
7. Branching
8. Pull request
9. Pulling from a central repo & merging

In this express tutorial, we'll work on Ubuntu 14.04 as a local repository and we'll use two separate Github accounts: one for a primary repo and the other one for contributor to the primary repo.

Git Install

We can install the Git command line tool using the command below:

$ sudo apt-get install git

$ git --version

git version 1.9.1

For more details on installation:

1. [GIT and GitHub - 1. Installation](http://www.bogotobogo.com/cplusplus/Git/Git_GitHub_Installation.php)
2. [GIT on Ubuntu and OS X - Focused on Branching](http://www.bogotobogo.com/cplusplus/Git/Git_Ubuntu.php)

Creating a git repository

$ git init project1

Initialized empty Git repository in /home/k/GitTraining/project1/.git/

Note that we do not have any server, and there is no background daemon. We just used local file system to create the project1 directory and the nested .git directory.

$ cd project1

$ ls

$ ls -al

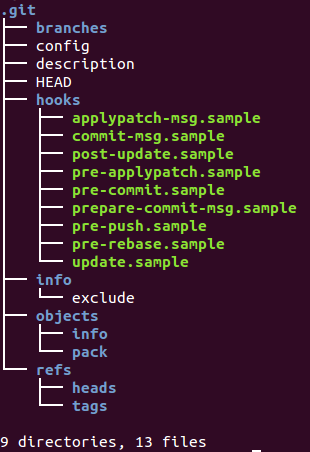
total 12

drwxrwxr-x 3 k k 4096 Jun 3 09:52 .

drwxrwxr-x 3 k k 4096 Jun 3 09:52 ..

drwxrwxr-x 7 k k 4096 Jun 3 09:52 .git

$ tree .git



Unlike other source control system such as CVS, there is only one **.git** folder at the top level. Only one **.git** per repository!

Also, note that we do not have any file in the repository yet:

$ git status

On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

Now we make our first ifle: **first.txt**.

Let's see how the git think of the file:

$ git status

On branch master

Initial commit

Untracked files:

(use "git add ..." to include in what will be committed)

first.txt

nothing added to commit but untracked files present (use "git add" to track)

Adding a file to a git

$ git add first.txt

The **git add** is merely telling the git our intention of adding for the next transaction. It's not adding the file to a repo yet. It just signals our participation. We do not have a permanent recode of the file yet.

We can see the changes to be committed using **git status**:

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached ..." to unstage)

new file: first.txt

Now we can commit to the **master**:

$ git commit -m "My first commit"

[master (root-commit) b025f57] My first commit

1 file changed, 50 insertions(+)

create mode 100644 first.txt

Here, the **b025f57** is a global unique identifier. The **644** indicates the user can read and write and others and group just can read the file.

Now we have permanent record of the file and we can see our current directory is clean:

$ git status

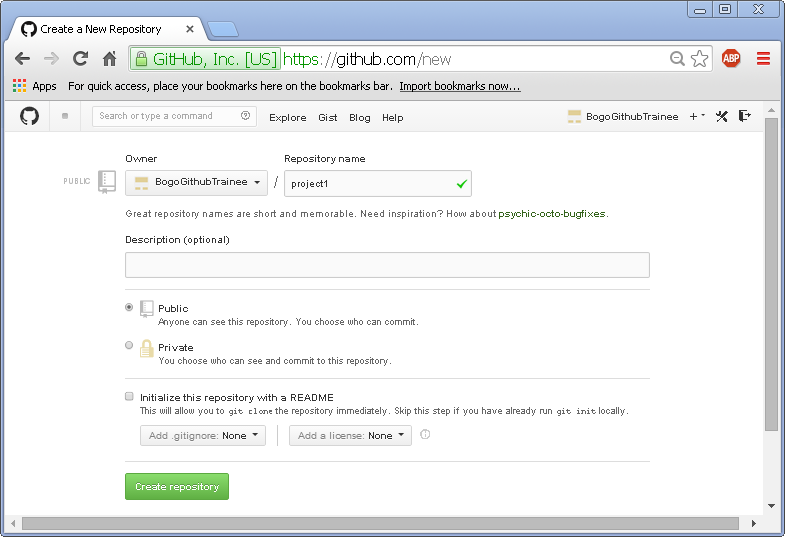
On branch master

nothing to commit, working directory clean

So far, we've been working on **local**. Now, we may want to use network (remote): **github**.

Github - repository

We created an empty public repository (**project1**) in GitHub under **BogoGithubTrainiee** account:



Github provides us some instructions, and the most import one is how to **push**: "Push an existing repository from the command line".

git remote add origin https://github.com/BogoGithubTrainee/project1.git

git push -u origin master

So, we copy & paste the two git commands into our local git command window:

$ git remote add origin https://github.com/BogoGithubTrainee/project1.git

$ git push -u origin master

Username for 'https://github.com': bogogithubtrainee

Password for 'https://bogogithubtrainee@github.com':

Counting objects: 3, done.

Delta compression using up to 2 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 899 bytes | 0 bytes/s, done.

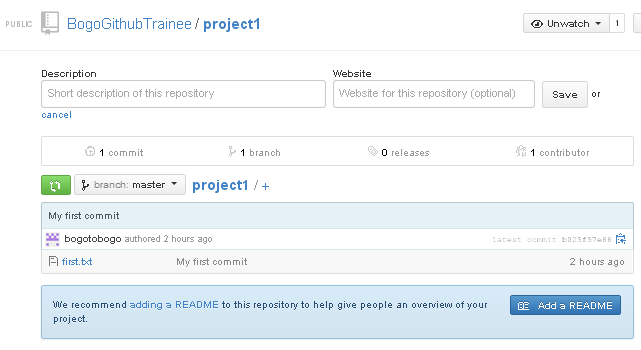
Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/BogoGithubTrainee/project1.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

We can see the **push** has an immediate effect on the Github as shown in the picture below:

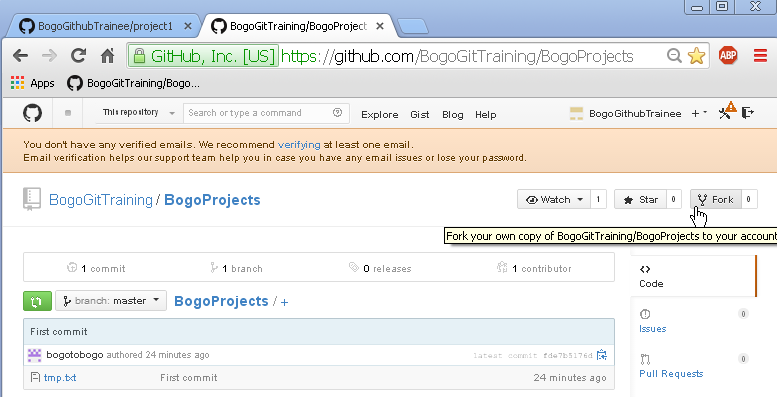


The **first.txt** has been pushed into the Github!

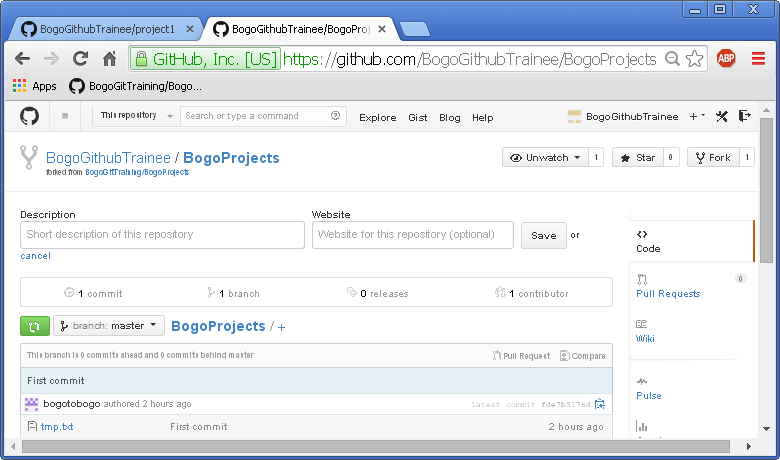
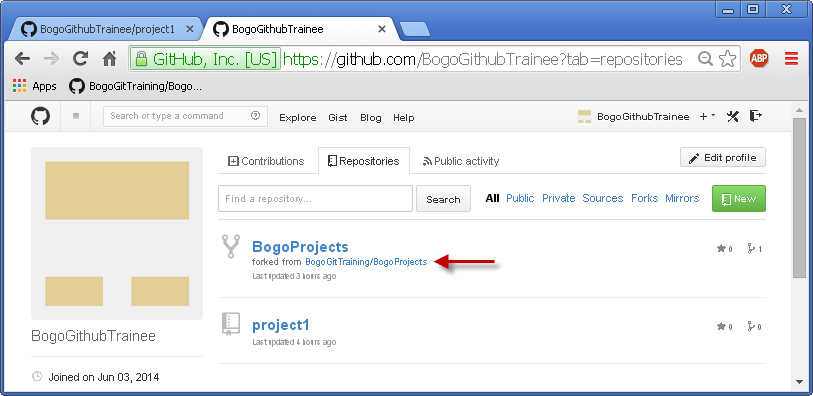
Github has records all the information at the time of the push: who did it, when, etc.

Collaborative working - fork

Most of the projects require collaborative working environment. We can do it via Github using **fork**.



We get the project from other account (**BogoGitTraining**), and the result is shown below:

Actually, now the **BogoGitHubTrainee** account has photocopy of the repository which owned by **BogoGitTraining**.

Github - clone

Now we want to transfer the copied project to a local repository:

$ cd ..

$ git clone https://github.com/BogoGithubTrainee/BogoProjects

Cloning into 'BogoProjects'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 3 (delta 0)

Unpacking objects: 100% (3/3), done.

Checking connectivity... done.

Then ask git for the status:

$ ls

BogoProjects project

$ cd BogoProjects

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean

The **git clone** brought all the information include all branches. All have been pulled to our local machine:

$ git branch -a

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/master

Branching

In this section, we want to do a **branch** operation for the cloned local master. First, we make a new branch:

$ git branch bogolocal

Then, we need to toggle to the new branch:

$ git checkout bogolocal

Switched to branch 'bogolocal'

Note that at this point there is no difference between the **master** and the new branch, **bogolocal**.

Then, we make a new file called "bogolocal\_new.txt".

$ git status

On branch bogolocal

Untracked files:

(use "git add ..." to include in what will be committed)

bogolocal\_new.txt

nothing added to commit but untracked files present (use "git add" to track)

$ git add bogolocal\_new.txt

$ git commit -m "new file to the new branch"

[bogolocal 49c85fd] new file to the new branch

1 file changed, 1 insertion(+)

create mode 100644 bogolocal\_new.txt

Note that we're working on local repository.

Pushing back to github

$ git push -u origin bogolocal

Username for 'https://github.com': BogoGitHubTrainee

Password for 'https://BogoGitHubTrainee@github.com':

Counting objects: 4, done.

Delta compression using up to 2 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 299 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/BogoGithubTrainee/BogoProjects

\* [new branch] bogolocal -> bogolocal

Branch bogolocal set up to track remote branch bogolocal from origin.

In the command the **-u** option stands for **upstream**. It would refer to the main repo that other people will be pulling from. The **-u** option automatically sets that upstream for us, linking our repo to a central one. That way, in the future, Git knows where we want to push to and where we want to pull from, so we can use git pull or git push without arguments.

One more thing while we are on local - file system is updated when we switching to another branch. In other words, when we switch to the **master** branch, the update file system does not have **bogolocal\_new.txt** file:

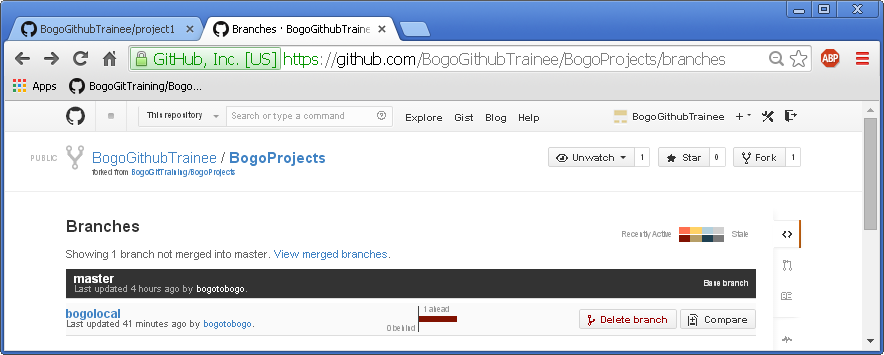
$ git checkout master

Switched to branch 'master'

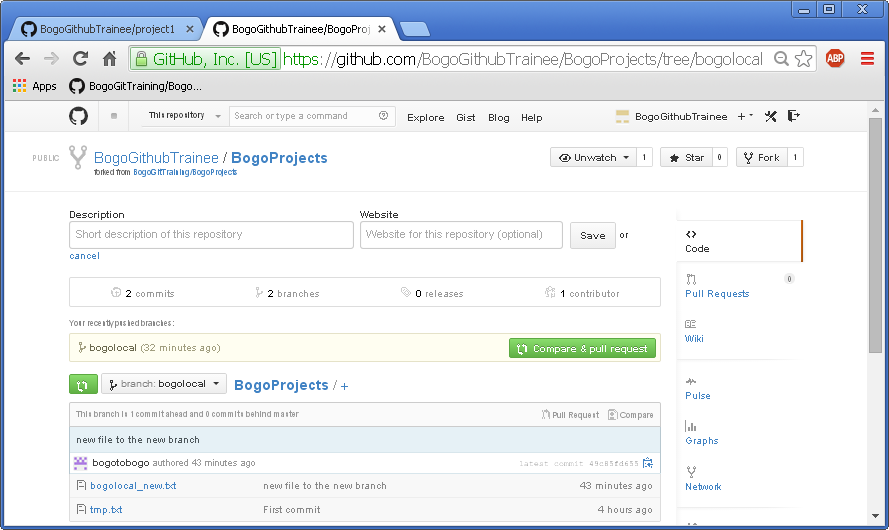
Your branch is up-to-date with 'origin/master'.

Back to github

When we refresh github under **BogoGithubTrainee**, we can see we have the pushed branch, **bogolocal** in the repository:

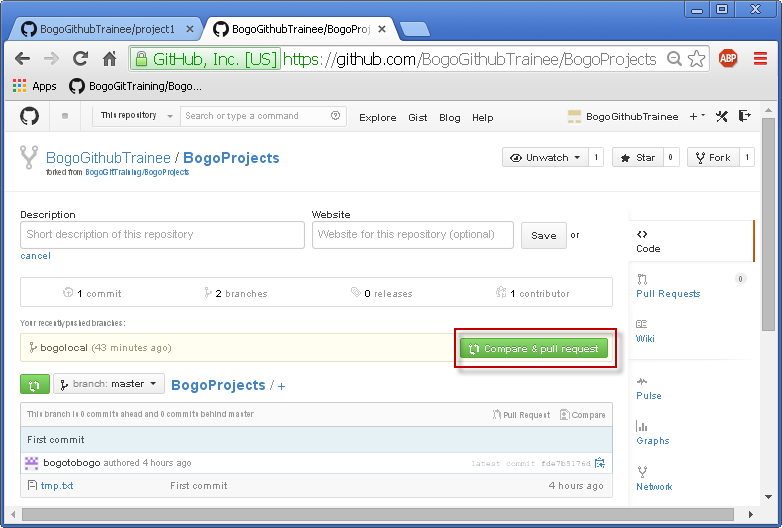


We can get more detail about the branch:



Pull request

Since we've updated the cloned repo, now it's a time to contribute our work to the central repo under different account owned by a primary maintainer of the project.

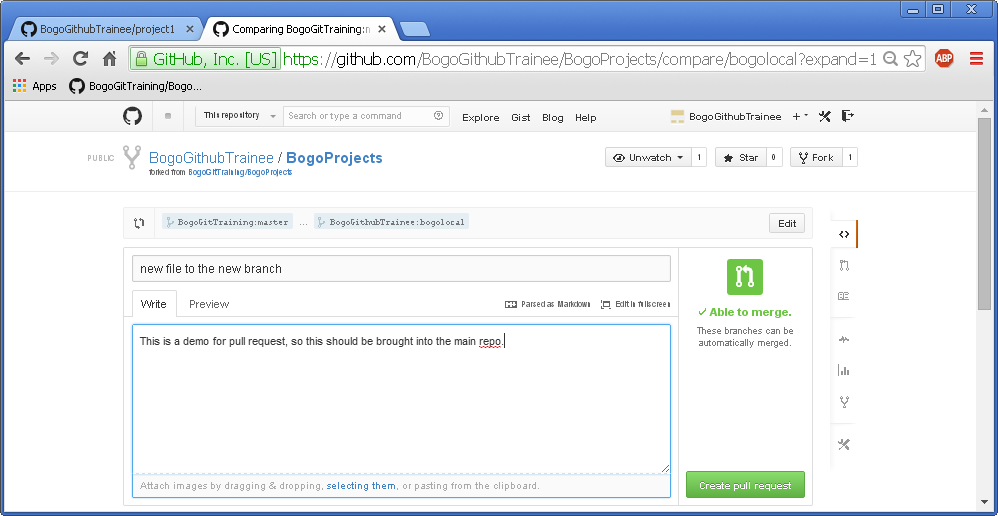


The **pull request** is a way of offering our changes to the central repo.

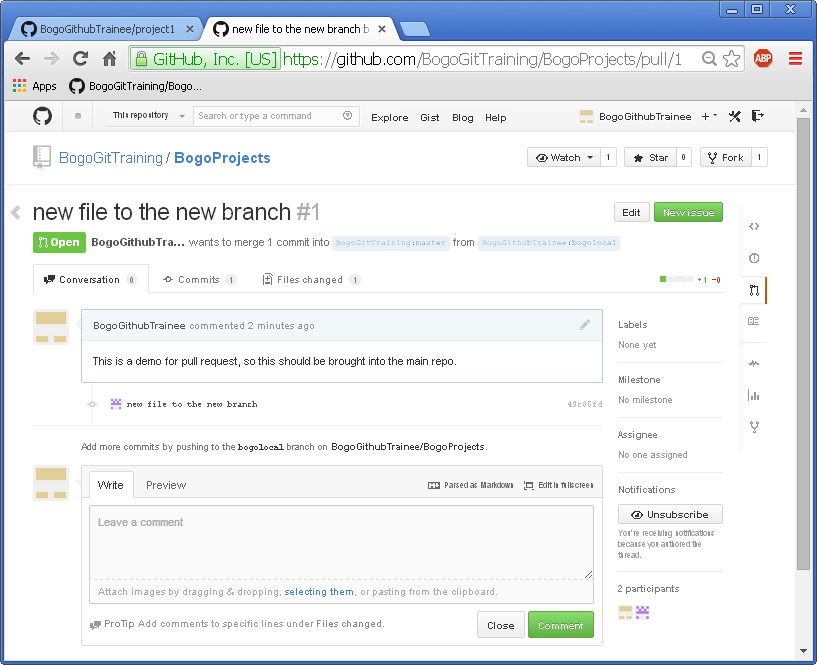
Why it is **pull** instead of **push**?

That's because the primary maintainer of the central is the one who decides whether bring our changes into the central repo.

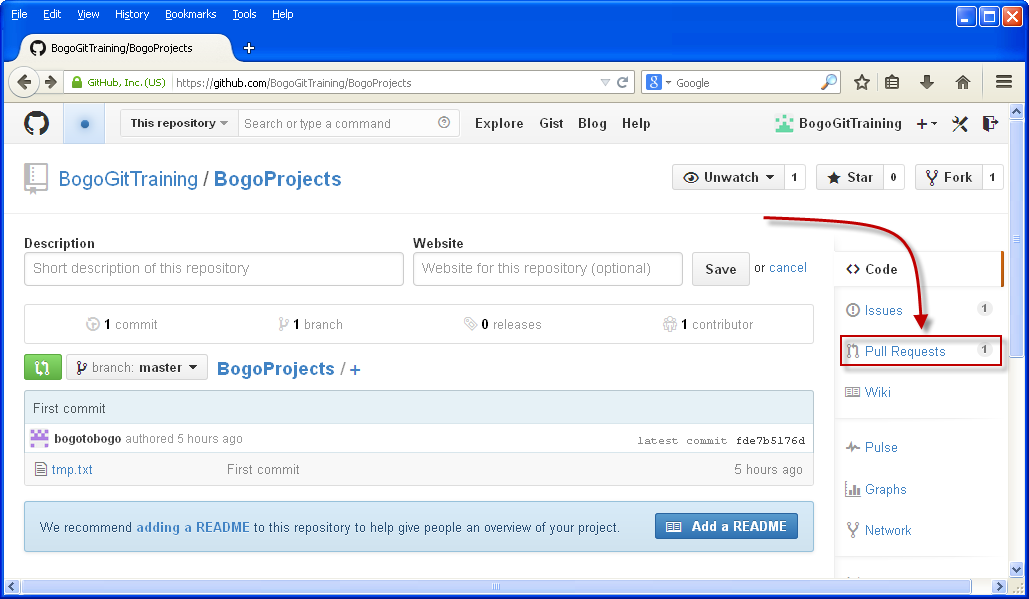
Note that we can describe why our changes should be brought into the central repo:



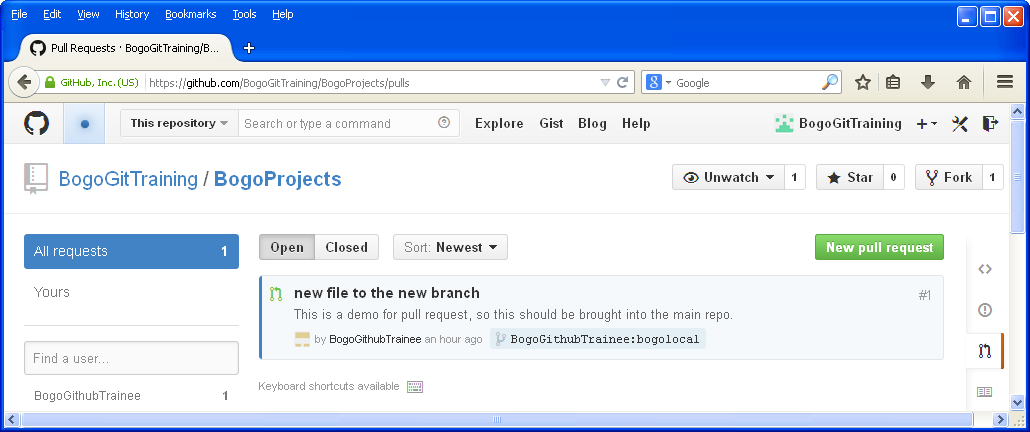
We can ask the primary maintainer to pull my change by clicking the **Create pull request** button.

  
  
Pulling from a central repo & merging

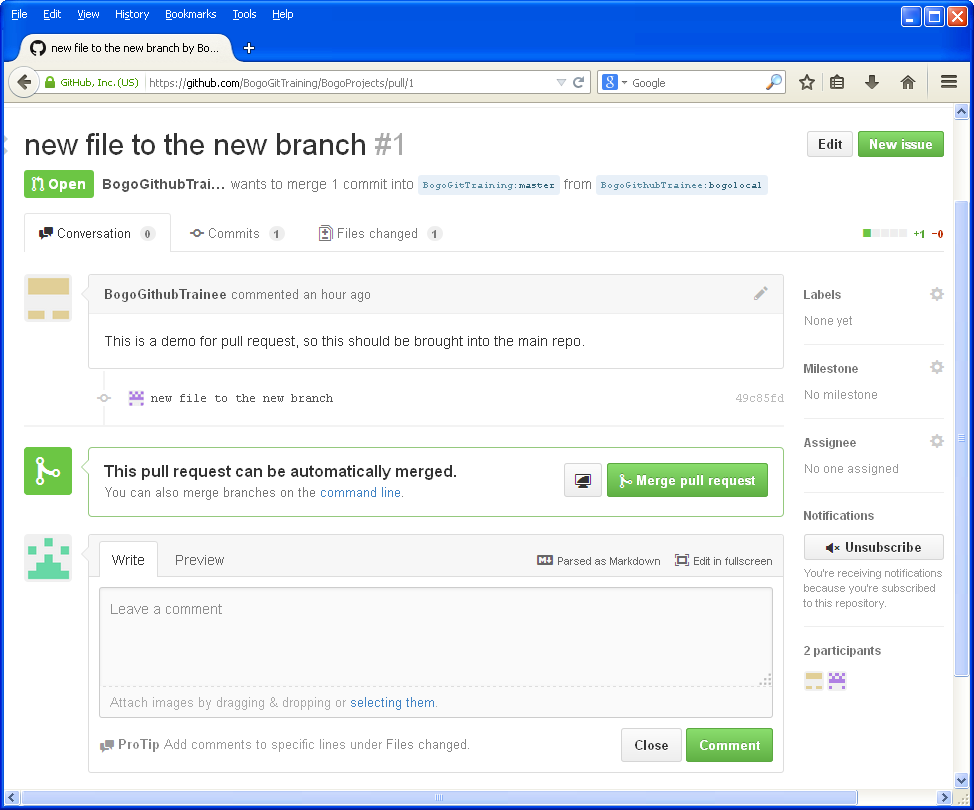
To pull the requested changes, we need to switch to another Github account (**BogoGitTraining**) that is maintaining the central repository for **BogoProjects**.



Now we can see the requests for pulling:



Click the "new file to the new branch" link, we get:



At this point, we can exchange comments with the contributor before **Merge pull request**.

