

Introduction to GitHub

By Erik Barrow

Contents

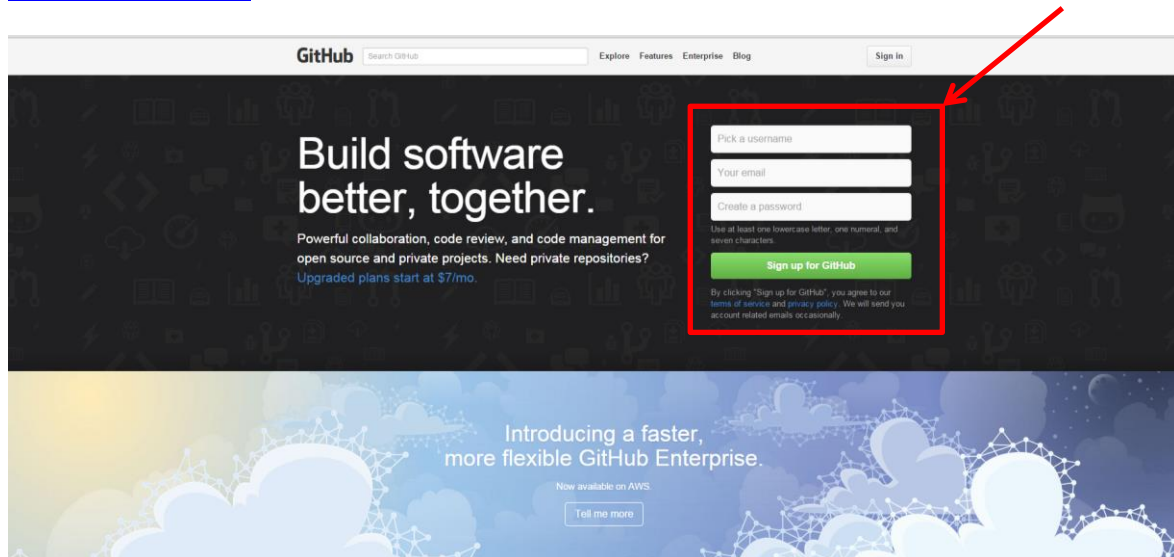
What Is GitHub?	2
Signing Up For GitHub.....	2
Getting Free Private Repository's For Your Group	3
Using GitHub	4
Creating Your Groups Repository	4
Adding Members To Your Private Repository.....	5
Adding A Readme.....	5
Using Issue Tracking.....	6
Creating An Issue	6
Viewing Issues	7
Closing Issues	7
Creating A Project Wiki	8
Using Git.....	10
Installing GIT On Your Computer	10
Using Git Clone.....	11
Using Git Push	12
Using Git PULL.....	14
Creating A Branch	14
Merging Branches	16
Reverting a change.....	18
Bibliography	19

What Is GitHub?

GitHub is an online repository tool that uses the GIT protocols. It allows you to create repository's for your projects and host them online, allowing teams to access the repository from any machine with internet access.

Signing Up For GitHub

If you haven't already, you should sign up for GitHub. You can do this by going to <https://github.com/>.



Fill in the sign in box on the home page, and then choose the free plan!

Welcome to GitHub

You've taken your first step into a larger world, @CovStudent.

✓ Completed
Set up a personal account

Step 2:
Choose your plan

Step 3:
Go to your dashboard

Choose your personal plan

Plan	Cost (view in GBP)	Private repos	
Large	\$50/month	50	<input type="button" value="Choose"/>
Medium	\$22/month	20	<input type="button" value="Choose"/>
Small	\$12/month	10	<input type="button" value="Choose"/>
Micro	\$7/month	5	<input type="button" value="Choose"/>
Free	\$0/month	0	<input type="button" value="Chosen"/>

Don't worry, you can cancel or upgrade at any time.

Each plan includes:

- Unlimited collaborators
- Unlimited public repositories
- ✓ Free setup
- ✓ SSL Protection
- ✓ Email support
- ✓ Wikis, Issues, Pages, & more

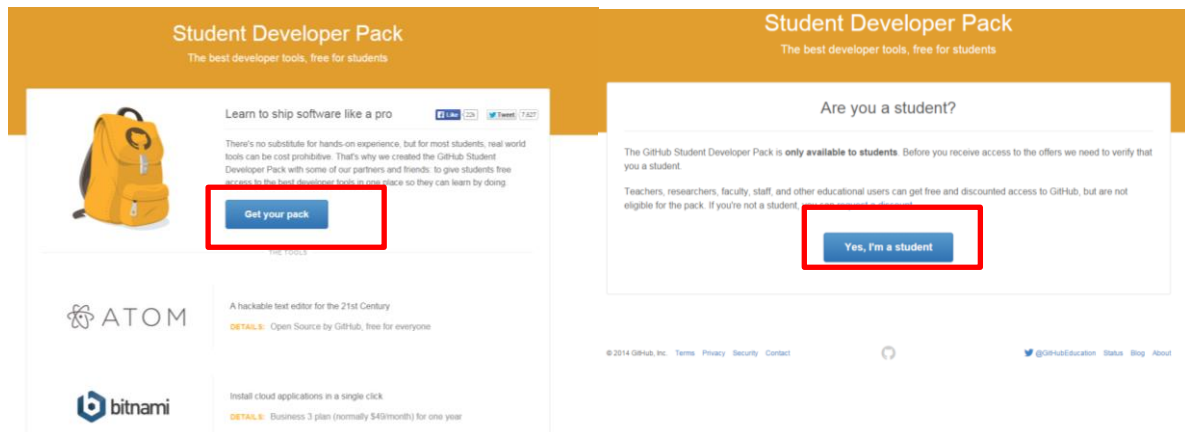
☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations.](#)

Congratulations, you now have your very own GitHub account!

Getting Free Private Repository's For Your Group

With GitHub education you can get a free micro account on GitHub. This will allow you to create private repositories. This means that only your team members are able to edit your project. There are also many other free programs that you can get access too, but we are mainly interested in GitHub.

Go to <https://education.github.com/pack> and click Get Your Pack, and then click Yes I am A Student



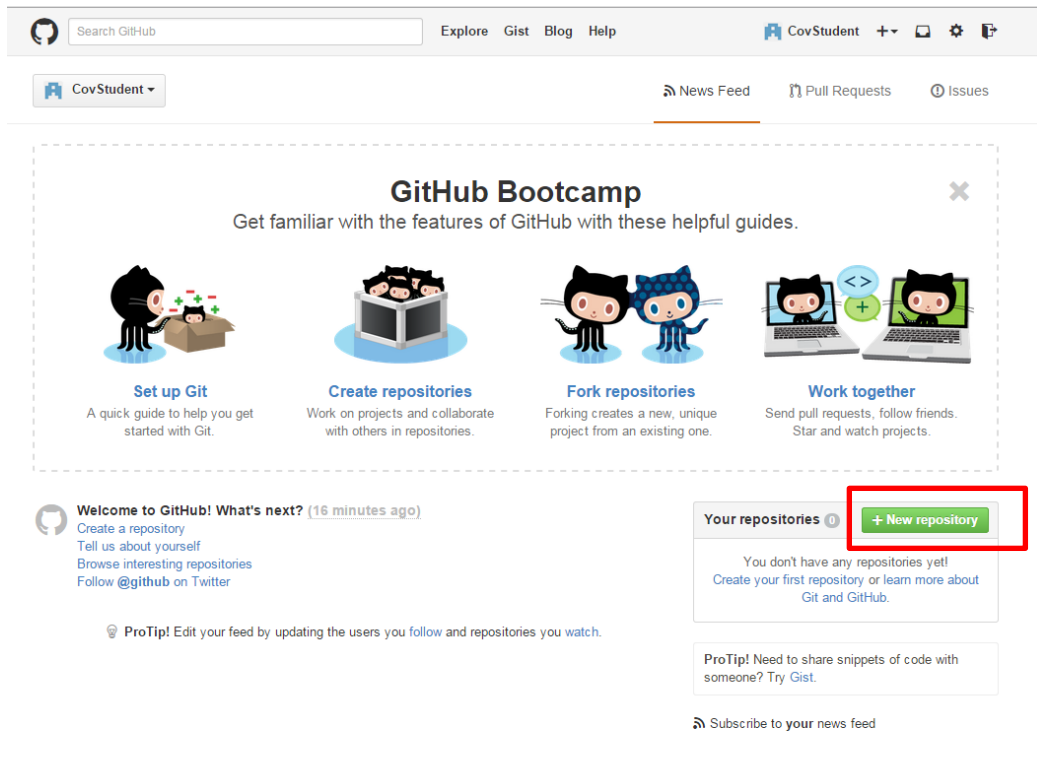
Then Fill in your details to confirm you are a student.

The image shows a screenshot of the GitHub Education 'Request a discount' form. The form is set against a teal background with the GitHub Education logo and navigation links at the top. The main heading is 'Request a discount' with the subtext 'Discounted and free plans are available for educational use'. The form is divided into two steps: 'Step 1: Tell us what you need' and 'Step 2: Tell us about you'. In Step 2, there are several input fields: 'Name' (with a placeholder 'CoStudent'), 'Verify academic status' (with a red error message: 'You don't have any verified email addresses. Please add and verify your school-issued email address – or a contact email if you don't have one – by clicking here. When done, refresh this page.'), 'School name', 'Graduation year' (a dropdown menu showing '2014'), and 'How do you plan to use GitHub?' (a text area).

Using GitHub

Creating Your Groups Repository

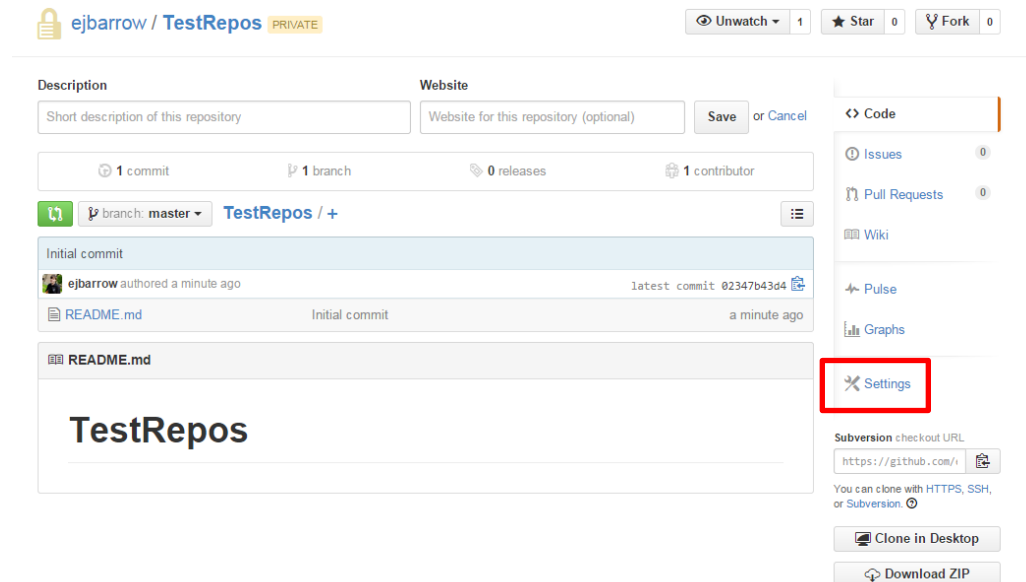
Now you have free private repositories one member of your group should create a private repository for you all to share. To start off with make sure you are logged into the homepage of GitHub. Select “+New Repository” to create your repository.



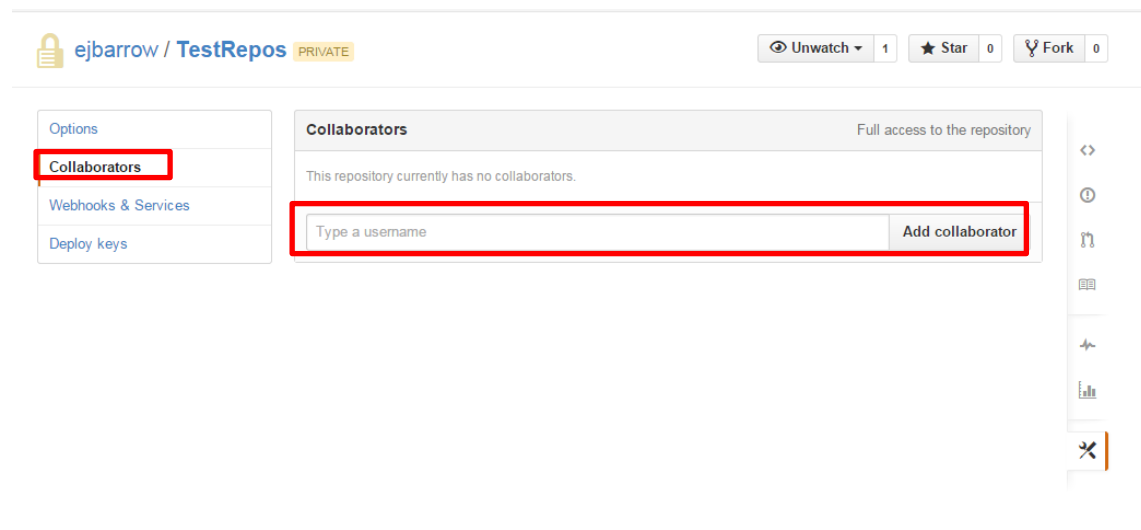
On the following page you need to choose the details of your repository. Give It a Name (Include your group Number at the beginning), choose **Private**, and select “**initialise this repository with a README**”. Note down the details on the page following this.

Adding Members To Your Private Repository

In your repository you will now want to add collaborators. To do this select settings, and then select collaborators.



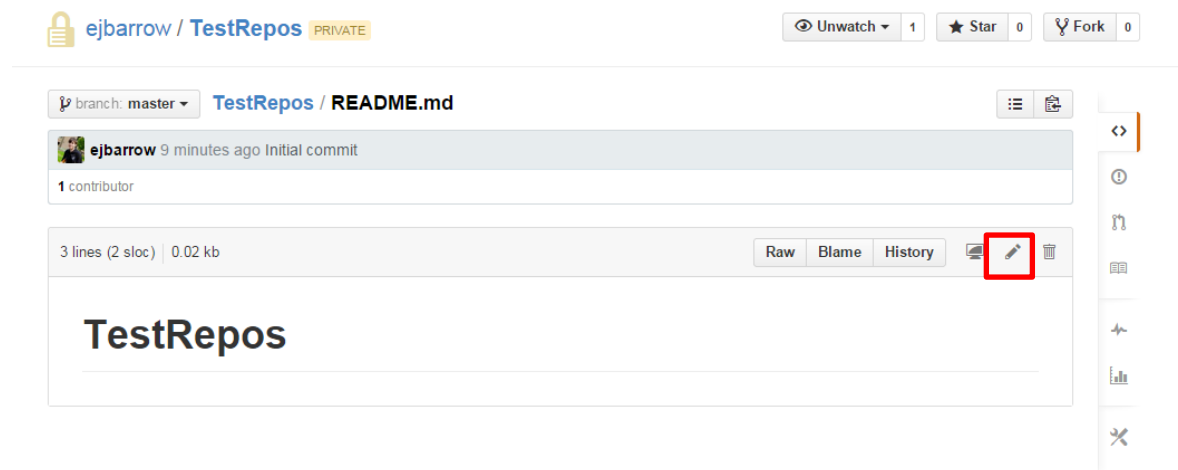
Add the username of your group members and click “Add collaborator” one by one for each team member.



Adding A Readme

You should keep a readme in your repository to explain the project, and to also explain anything a new developer would need to start working with your code. You can edit this on GitHub by clicking

on the initial README.md file that was created, and clicking on the little pencil to edit the file.



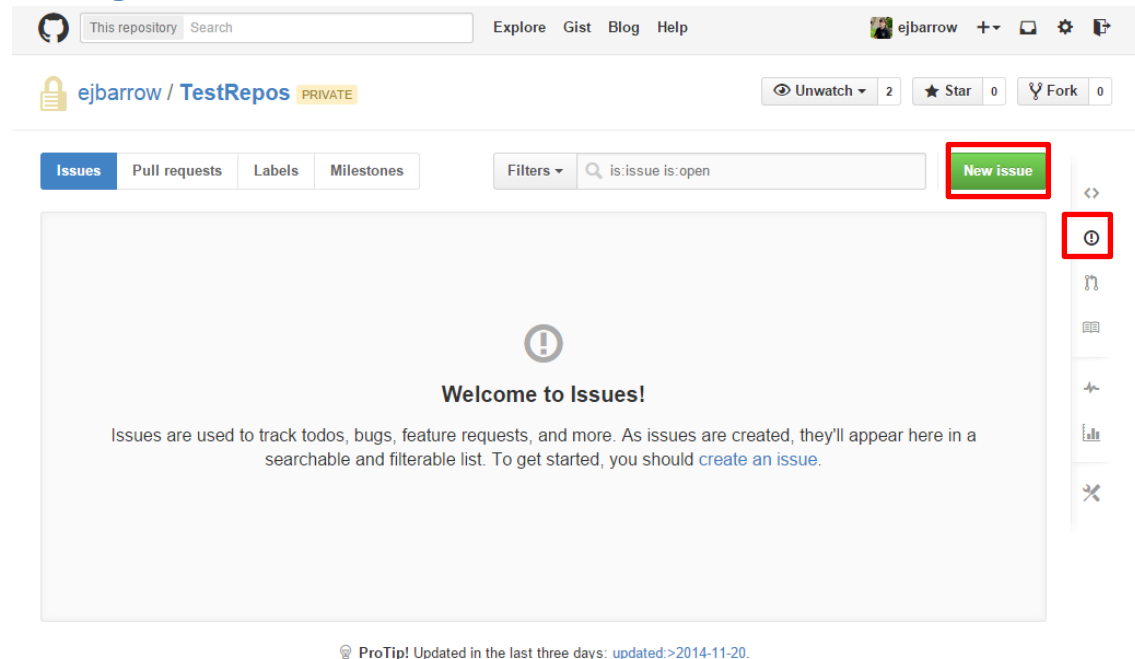
After editing your readme, don't forget to click the "commit changes" button.

Using Issue Tracking

Issue tracking allows you to do several things with your GitHub repository. You can use it to report bugs in the program, list enhancements that need working on, ask questions to your team members, and ask for help.

For your projects you may find it useful to use this to list all your requirements as enhancements to your project. GitHub allows you to assign a team member to the enhancement/bug so you know who is responsible for implementing or fixing that part of the project.

Creating An Issue



Click on the issues icon on the right hand side of the page. Then Click on "New Issue"

Issues Pull requests Labels Milestones



Title

No one is assigned

No milestone

Write Preview

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Submit new issue

Assign someone to this issue

Filter people

ejbarrow Erik Barrow

CovStudent

Markdown supported Edit in fullscreen

Labels

bug

duplicate

enhancement

help wanted

invalid

question

wontfix

Start by giving your issue a title. You can then write a full description in the comment area describing what the issue is related to. Click on a label on the right hand side to select a category for the “issue”. You can even assign a member of your team by clicking the gear icon. When finished click on “submit new issue”.

Viewing Issues

To view your issue click back on the exclamation icon on the right hand side of GitHub. You can open an issue in more detail by clicking on it. From here you can also add comments to the issue to show progress.

Closing Issues

Bug Test 1 #1

Open ejbarrow opened this issue a minute ago · 0 comments



ejbarrow commented a minute ago

testing bug feature

ejbarrow added the **bug** label a minute ago

CovStudent was assigned by ejbarrow a minute ago



Write Preview Markdown supported Edit in fullscreen

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close issue

Comment

Edit New issue

Labels

bug

Milestone

No milestone

Assignee

CovStudent

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

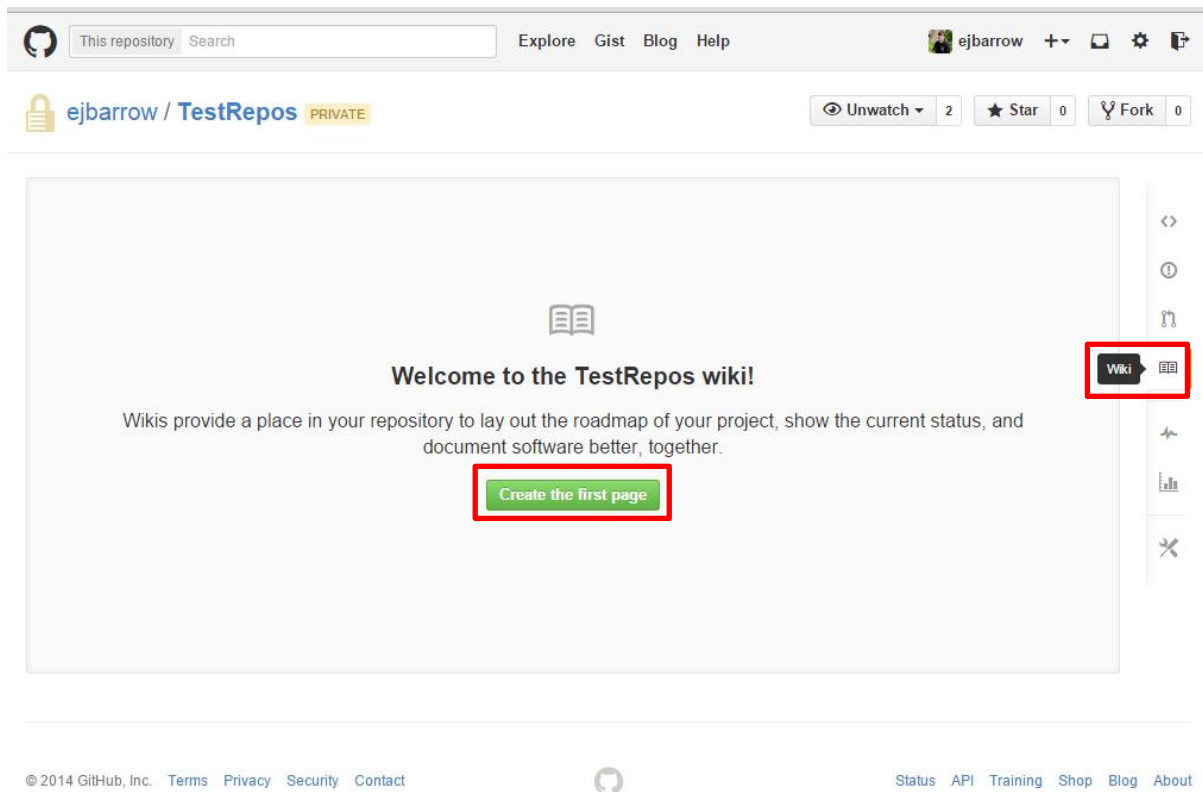
2 participants

Lock issue

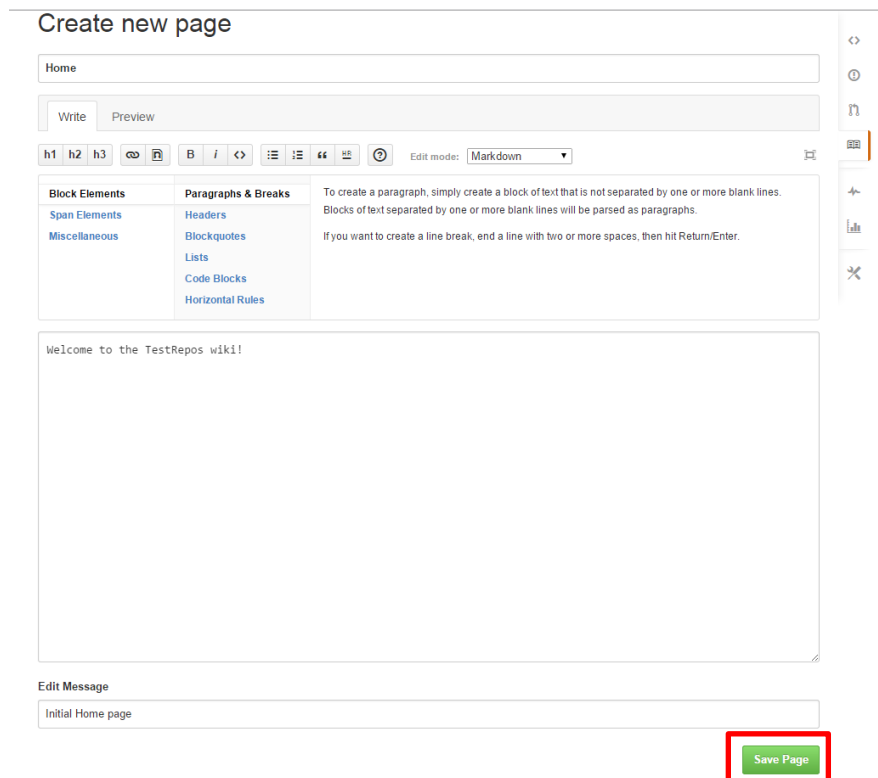
You can close an issue by opening it in the detailed view. Then click the close issue button.

Creating A Project Wiki

To start you wiki you should open your repository. On the right hand menu select the icon that looks like an open book. From here click “Create First Page”.

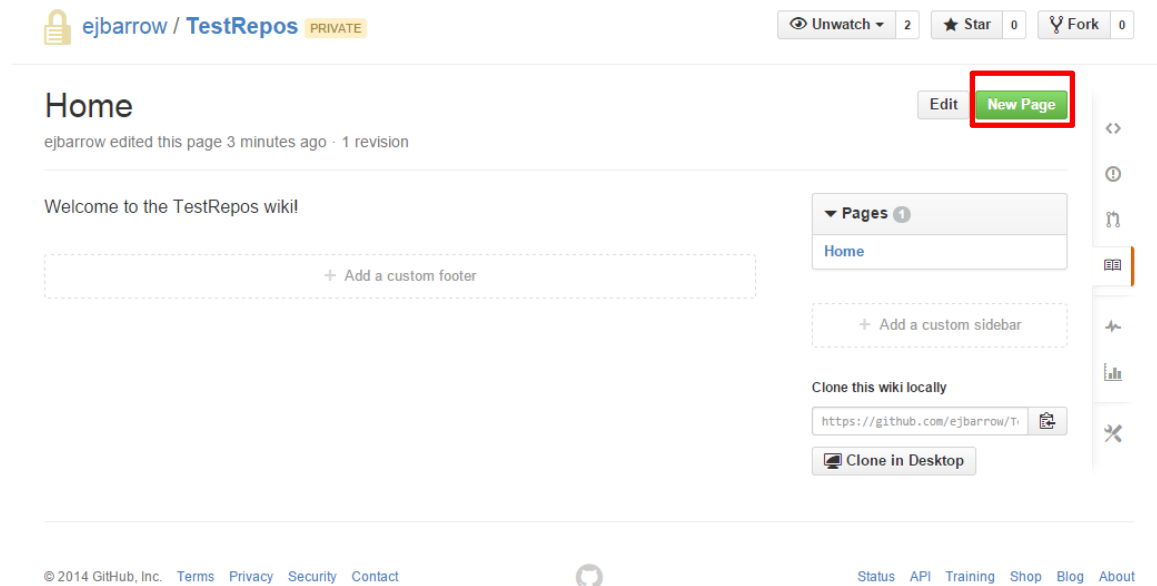


Create your first page with an introduction to your group’s project.



From here you can add more pages to your wiki about your project. You might like to add your groups other work here, such as the breakdown of your team's roles and responsibilities. It is also a good place to put copies of other planning items such as your pseudo code, storyboards, and data flow diagrams.

To add more pages just open your wiki and click "add new page". You can view these pages in the navigation on the right hand side.

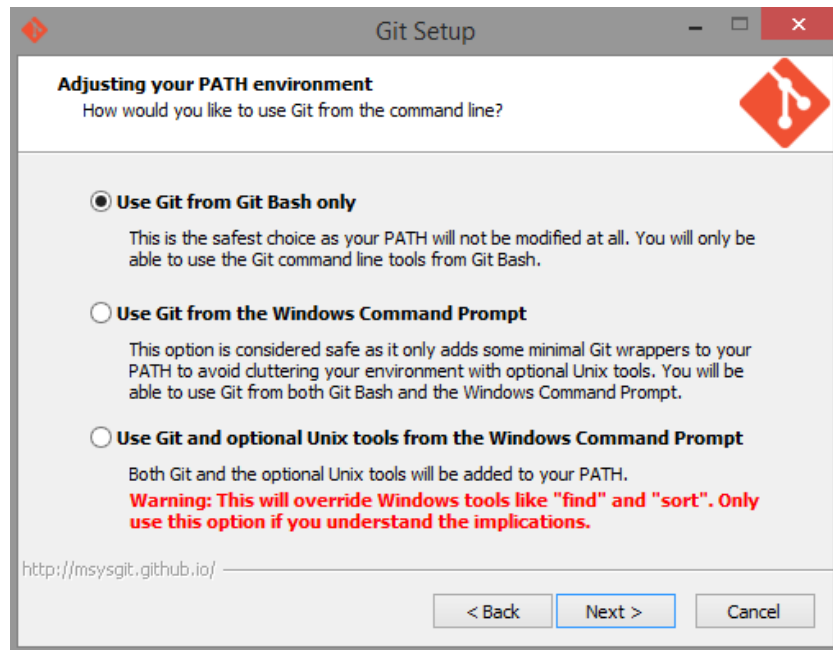


Using Git

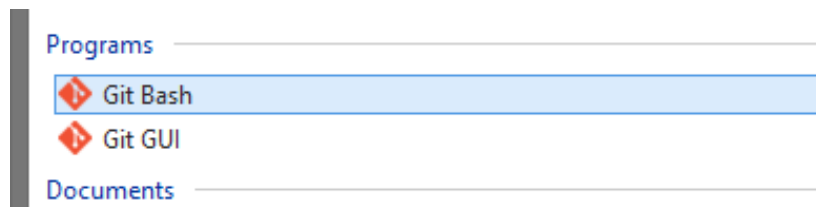
Git is a program that allows you to interact with your git repository's, and pull a copy of your groups work from the server. Once you have edited this work you can then push this back onto the server.

Installing GIT On Your Computer

To install GIT you first need to download the install file from <http://git-scm.com/downloads>. On running the installer make sure "use git from Git-Bash only" is selected.



Follow the rest of the installation using the default selections.



Open Git-Bash to continue with the tutorial.

Before we do anything else you need to configure git to your user. This allows commits that you make to be properly labelled with your details.

First type

```
git config --global user.name "Your Name Here"
```

Secondly

```
git config --global user.email "your_email@youremail.com"
```

```
$
Erik@FRODO ~
$ git config --global user.name "Erik"
Erik@FRODO ~
$ git config --global user.email "ab3065@coventry.ac.uk"
Erik@FRODO ~
$
```

Using Git Clone

You should start by creating and moving to a folder that you are going to use to sync your work with the GitHub server. Then initialize your local repository with “git init”

```
Erik@FRODO /d
$ mkdir myfirstrepos

Erik@FRODO /d
$ cd myfirstrepos

Erik@FRODO /d/myfirstrepos
$ git init
Initialized empty Git repository in d:/myfirstrepos/.git/

Erik@FRODO /d/myfirstrepos (master)
$
```

Then you will want to clone your online repository. The URL of your Repository is available on GitHub on the right hand side.

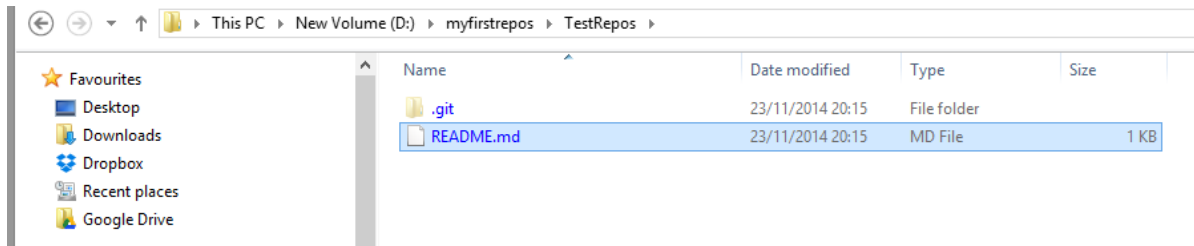
```
git clone YourReposAddressHere
```

You will be prompted for a username and password. This is the login that you used when you made your GitHub account.

```
Erik@FRODO /d/myfirstrepos (master)
$ git clone https://github.com/ejbarrow/TestRepos
Cloning into 'TestRepos'...
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.

Erik@FRODO /d/myfirstrepos (master)
$
```

You can now navigate to the folder you made. Your repository has been added as a folder. You should be able to see the Readme file that we edited earlier.



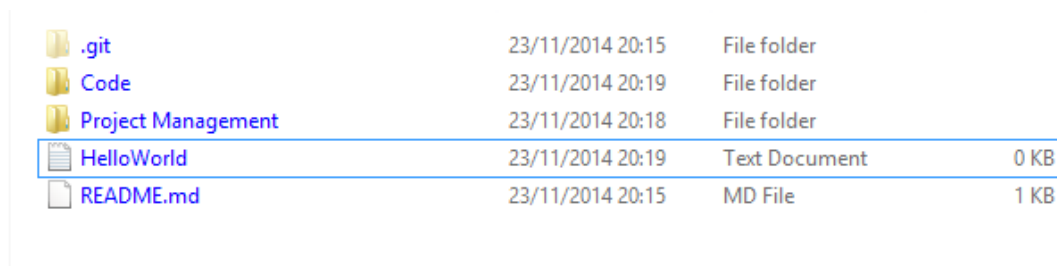
Using Git Push

Make sure you CD (change directory) to the new repository.

```
Erik@FR0DO /d/myfirstrepos (master)
$ cd testrepos

Erik@FR0DO /d/myfirstrepos/testrepos (master)
$
```

Add some files to your local copy of the repository or edit a current one. I added a HelloWorld.txt file and some folders to store some other files in.



To commit a file back to the server you need to add the file to the commit. You can do this like the following.

```
git add HelloWorld.txt
```

Or to quickly track all changed files you can use.

```
git add .
```

Use the following command to see the status of files to be committed.

```
git status
```

```
Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$ git add .

Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   HelloWorld.txt
```

You can now commit the changes to your local repository. Ensure you add a comment to the commit.

```
git commit -m 'description of commit'
```

```
Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$ git commit -m 'hello world added'
[master c050161] hello world added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 HelloWorld.txt

Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$
```

Finally now you can push your local commit to back to the server! You will be prompted for your login details again.

```
git push origin master
```

```
Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$ git push origin master
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ejbarrow/TestRepos
 02347b4..c050161  master -> master

Erik@FR0D0 /d/myfirstrepos/testrepos (master)
$
```

Using Git PULL

Once you have the initial local repository cloned you will only need to pull the changes. You can do this with the following command. You will be prompted for your GitHub user details.

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git pull
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
Already up-to-date.

Erik@FRODO /d/myfirstrepos/testrepos (master)
$
```

In my example I am already up to date. But as team members add and change files that will change. It is important to PULL an up to date copy before pushing your changes to the server, as you may find that you are prevented from pushing outdated files to the server.

Creating A Branch

Sometimes you may not want to edit your main branch of your repository. You might want to make a different version of your program, or test out the feasibility of a new feature without damaging the main build. A common use is to work on a program bug.

To create a branch you use

```
git branch nameofbranch
```

You can view branches with

```
git branch
```

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git branch bug/bug1

Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git branch
bug/bug1
* master

Erik@FRODO /d/myfirstrepos/testrepos (master)
$
```

To switch to a branch you use the checkout feature.







```
git checkout nameofbranch
```

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git checkout bug/bug1
Switched to branch 'bug/bug1'

Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$
```

After checking out a branch your local repository will change to represent the branch. For example the folder that you can see in windows is now that of the branch, not the master repository.

Let's add another file and commit it to the branch.

	.git	23/11/2014 20:41	File folder	
	Code	23/11/2014 20:19	File folder	
	Project Management	23/11/2014 20:18	File folder	
	bugfix	23/11/2014 20:43	Text Document	0 KB
	HelloWorld	23/11/2014 20:19	Text Document	0 KB
	README.md	23/11/2014 20:15	MD File	1 KB

```
Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$ git add bugfix.txt






Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$ git commit -m 'Bug fixed'
[bug/bug1 f549e0a] Bug fixed
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 bugfix.txt

Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$ git push origin bug/bug1
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 279 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To https://github.com/ejbarrow/TestRepos
 * [new branch]      bug/bug1 -> bug/bug1
```

Now let's switch back to the master branch.

```
Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Erik@FRODO /d/myfirstrepos/testrepos (master)
```


Name	Date modified	Type	Size
 .git	23/11/2014 20:45	File folder	
 Code	23/11/2014 20:19	File folder	
 Project Management	23/11/2014 20:18	File folder	
 HelloWorld	23/11/2014 20:19	Text Document	0 KB
 README.md	23/11/2014 20:15	MD File	1 KB

Notice how after the switch the file we added to the branch isn't in the folder where it was a minute ago. It will only show when we change to that branch.

To add it to the master branch now we have "fixed" the bug, we need to merge the branches.

Merging Branches

To start you need to check out the branch we are merging to. We want to merge into the master branch, so we check out that branch. Then we merge the branch we want to join into it, with the following command.

```
git merge branchname
```

```
Erik@FRODO /d/myfirstrepos/testrepos (bug/bug1)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git merge bug/bug1
Updating c050161..f549e0a
Fast-forward
 bugfix.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 bugfix.txt

Erik@FRODO /d/myfirstrepos/testrepos (master)
$
```

Let's push the branch back to the server.

```

Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git commit -m 'branches merged'
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

Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git push origin master
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/ejbarrow/TestRepos
c050161..f549e0a  master -> master

```

You may have issues merging branches like you do when you try to push a repository that is out of date. This may occur when a team member has edited the main branch and the changes conflict with the changes you are trying to push. Git will attempt at best to merge differences automatically.

Let's look at GitHub, you can see the files that we added have been pushed to the server.


ejbarrow / TestRepos PRIVATE
Unwatch 2
Star 0
Fork 0

Description
Website

Short description of this repository
Website for this repository (optional)
Save or Cancel

3 commits
2 branches
0 releases
1 contributor

branch: master TestRepos / +

Bug fixed
Cov Student authored 16 minutes ago
latest commit f549e0a83c

HelloWorld.txt	Bug fixed	16 minutes ago
README.md	Initial commit	3 days ago
bugfix.txt	Bug fixed	16 minutes ago

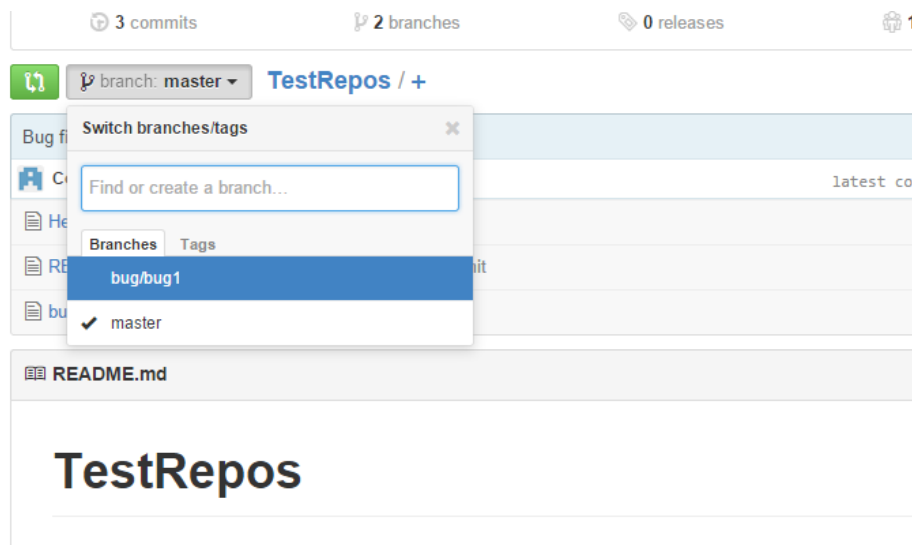
README.md

TestRepos

Code
Issues 1
Pull Requests 0
Wiki
Pulse
Graphs
Settings

Subversion checkout URL
https://github.com/
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

You can also see any branches on the branch drop down menu.



Reverting a change

Well, what happens if I make a mistake and I break something? Well git keeps a copy of all the changes you make. You can simply roll back your project to an earlier version. This is why naming your commits is useful; as it will help you to find the commit you want to roll back to.

`git log`

Git log will allow you to see the log of things that have been happening on the branch you currently have checked out. See why those comments were handy now...

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git log
commit f549e0a83c3c42ccc8a05a5b7b243c8945560843
Author: Erik <ab3065@coventry.ac.uk>
Date: Sun Nov 23 20:44:24 2014 +0000

    Bug fixed

commit c050161f33bf884093880ae54696d99e23877d7c
Author: Erik <ab3065@coventry.ac.uk>
Date: Sun Nov 23 20:28:03 2014 +0000

    hello world added

commit 02347b43d4609387c711dc446ca33d9b5beb6eb6
Author: Erik Barrow <ejbarrow@users.noreply.github.com>
Date: Thu Nov 20 12:12:07 2014 +0000

    Initial commit

Erik@FRODO /d/myfirstrepos/testrepos (master)
$
```

To revert back to a commit use the following command. NOTE: you will lose any changes made since that commit.

```
git reset --hard commitID
```

You can replace the “commitID” with the id of the commit you want to roll back to (that long string of jumbled letters and numbers)

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git reset --hard f549e0a83c3c42ccc8a05a5b7b243c8945560843
HEAD is now at f549e0a Bug fixed

Erik@FRODO /d/myfirstrepos/testrepos (master)
$
```

Don't forget to push the changes back to the remote server.

```
Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git commit -m 'reverted change'
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean

Erik@FRODO /d/myfirstrepos/testrepos (master)
$ git push origin master
Username for 'https://github.com': ejbarrow
Password for 'https://ejbarrow@github.com':
Everything up-to-date
```

Bibliography

galleryhip, n.d. *git-logo*. [Online]

Available at: <http://galleryhip.com/git-logo.html>

[Accessed 20 November 2014].

inc, G., n.d. *GitHub*. [Online]

Available at: <https://github.com>

Tyers, M., n.d. *305CDE*. s.l.:s.n.