GitHub Tutorial

Part 1 of 2

How to create a project

Our colleague has created a project that we later will download(clone) from Github and we will add new code and send it back to Github(commit & push) but that will be done in the second tutorial.

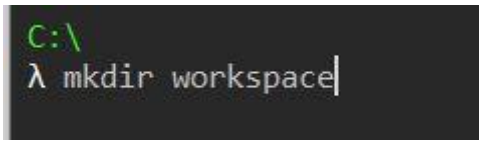
This on is to show how our colleague did to create a new project and add the first code. So below we will go thru that process.

First, we need to create a place on our computer where we want to store our project folder. (This is the same as you would go to file explorer and select create new folder).

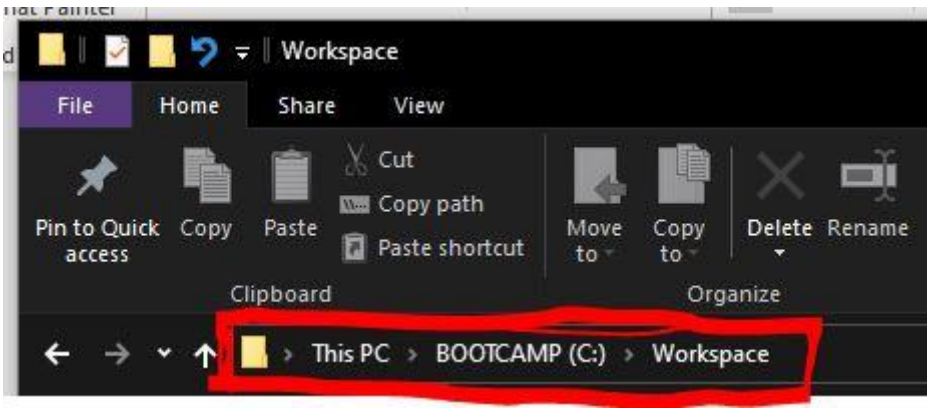
- Open your terminal (Git bash, iTerm, cmder or whatever you prefer to use)
- Go to root (as close to the hard drive we can put our directory)

- Use `cd ..` to step back down closer to root so you get here:

- Now we going to create a new folder called workspace, it is good to have a folder we always use for our projects, so we easy remember the path.



Now we have created a folder on our local computer where we can add projects:



- Now we need to step into our newly created folder (same as click to open folder in file explorer) We do this by using the command:
  cd workspace

Now we go to our IDE (intellij, Eclipse or whatever you use)

We create a new project, just create a simple java console application. Name it GitHubDemo.

And make sure to create it inside the folder we created above **c:\Workspace\GitHubDemo**

In the project we create a mainHello.java file that looks like this:

```java
public class mainHello {

    public static void main(String[] args) {

        PrintHelloWorld printer = new PrintHelloWorld();

         printer.print();


    }
}
```

And we create a file PrintHelloWorld.java

```java
public class PrintHelloWorld {
    public void print() {

        System.out.println("Hello World!!");
    }
}
```

And we are done here for now. Time to get this code up to GitHub.

Next step is to create the repository (repo)

We can check to see if there is any repo in this folder by using command: git status.

And to create the repo we use the command: git init

```
C:\Workspace\GitHubDemo
λ git status
fatal: not a git repository (or any of the parent directories): .git

C:\Workspace\GitHubDemo
λ git init
Initialized empty Git repository in C:/Workspace/GitHubDemo/.git/

C:\Workspace\GitHubDemo (master -> origin)
λ
```

As we can see we are now in the active branch master (this will be called main from now on, master was used before but is changing to main)

We can use the commando: git status to see all changes that has been made if we like.

And then we use: git add .

```
C:\Workspace\GitHubDemo (master -> origin)
λ git add .
```

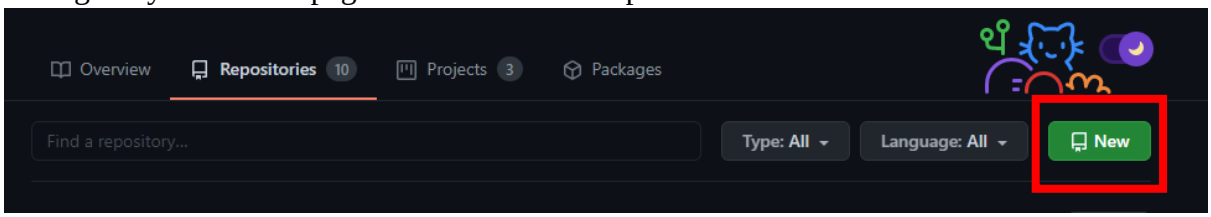
Then use: git status again and we will see the files added:

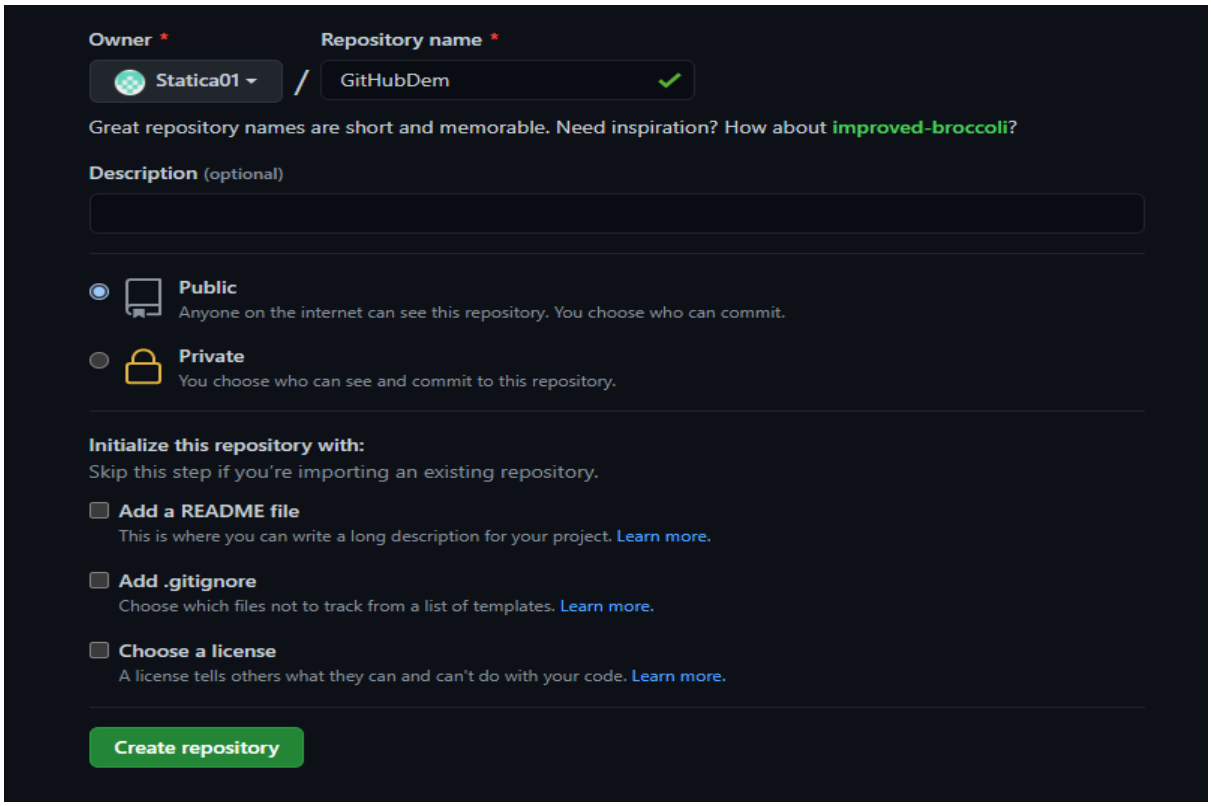
```
C:\Workspace\GitHubDemo (master -> origin)
λ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .idea/misc.xml
        new file:   .idea/modules.xml
        new file:   .idea/uiDesigner.xml
        new file:   .idea/vcs.xml
        new file:   .idea/workspace.xml
        new file:   GitHubDemo.iml
        new file:   src/PrintHelloWorld.java
        new file:   src/mainHello.java
```

We now need to create our commit. Since this is the first commit it is common to just name it “initial commit” or “first commit”

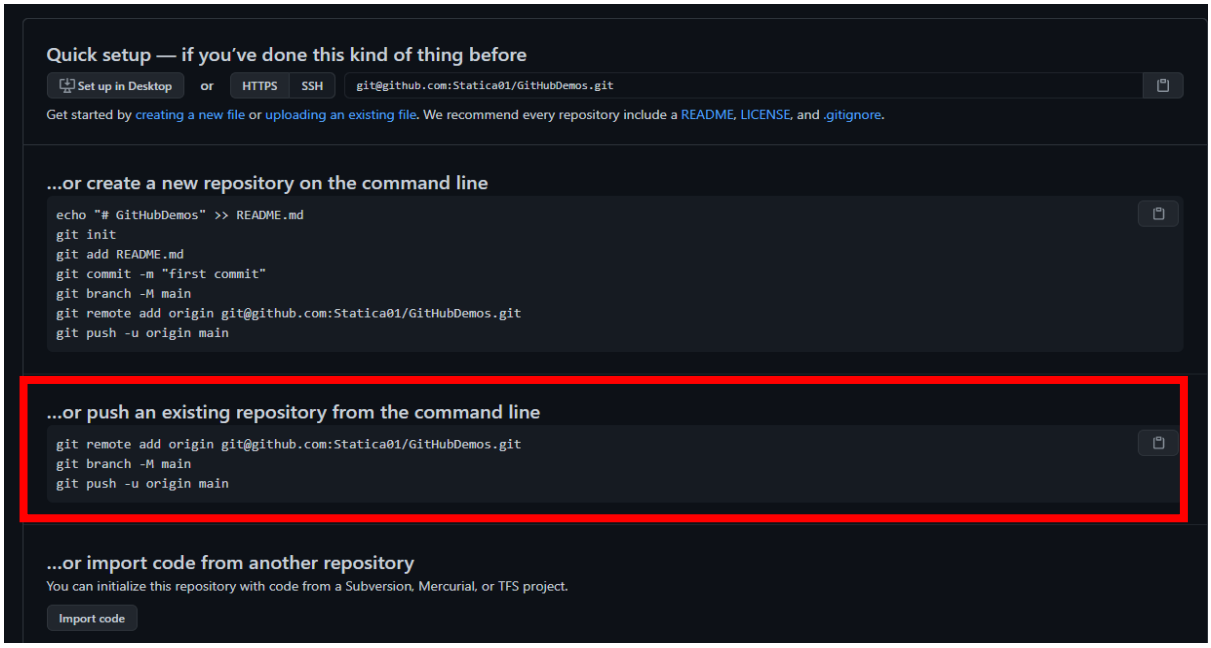
```
C:\Workspace\GitHubDemo (master -> origin)
λ git commit -m "first commit"
[master (root-commit) cf5ccfe] first commit
 8 files changed, 258 insertions(+)
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/uiDesigner.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .idea/workspace.xml
 create mode 100644 GitHubDemo.iml
 create mode 100644 src/PrintHelloWorld.java
 create mode 100644 src/mainHello.java
```

Now go to your Github page and create a new repo





Copy the first line you see in the … or push existing repository from the command line



Then go back to the terminal (command prompt) and paste all three lines, one by one. You need to paste first line then hit enter, then paste next -> hit enter. Not paste all lies at ones.

```
C:\Workspace\GitHubDemo (master -> origin)
λ git remote add origin git@github.com:Statica01/GitHubDemos.git

C:\Workspace\GitHubDemo (master -> origin)
λ git branch -M main

C:\Workspace\GitHubDemo (main -> origin)
λ git push -u origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 3.53 KiB | 602.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Statica01/GitHubDemos.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Now it is done, and we can go back to our github page and refresh, then we will see our project under the repository category.

GitHub Demo

Part 2 of 2

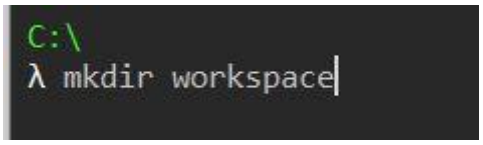
We will go thru the basics of git commandos to create a file path on our local computer. Next step is to clone a project from GitHub. And then add a file and commit this back to GitHub.

First, we need to create a place on our computer where we want to store our project folder. (This is the same as you would go to file explorer and select create new folder).

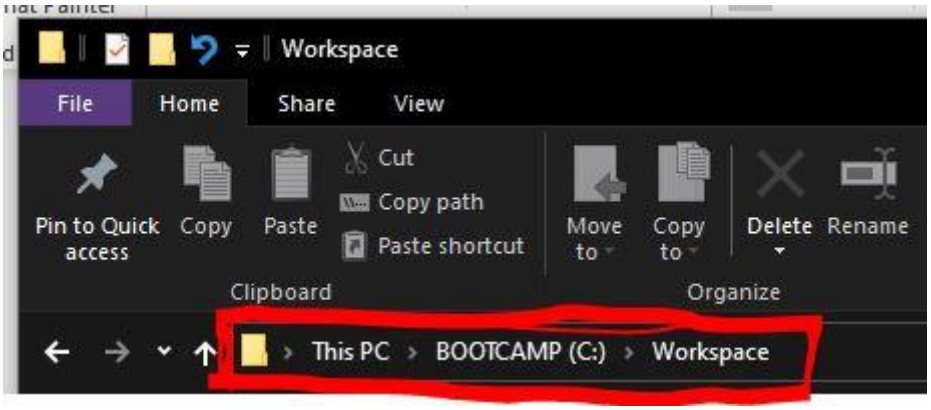
- Open your terminal (Git bash, iTerm, cmder or whatever you prefer to use)
- Go to root (as close to the hard drive we can put our directory)

- Use `cd ..` to step back down closer to root so you get here:

- Now we going to create a new folder called workspace, it is good to have a folder we always use for our projects, so we easy remember the path.



Now we have created a folder on our local computer where we can add projects:

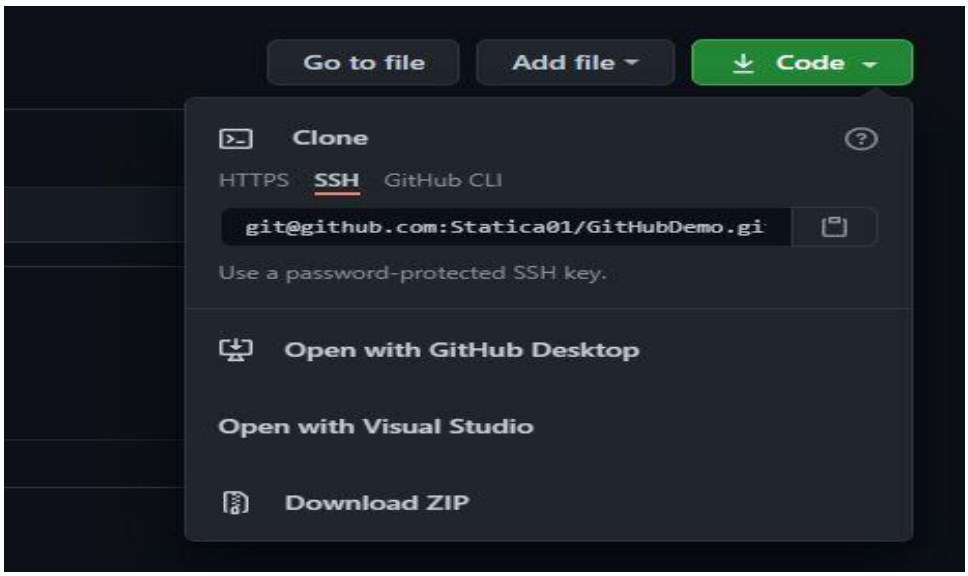


- Now we need to step into our newly created folder (same as click to open folder in file explorer) We do this by using the command:
  `cd workspace`

Now our colleague has created a GitHub repository for our project, and we need to go to GitHub and download it to our computer (we clone the project)

- Go to the gitHub page where you created the project in the tutorial above

  Here we click on the green Code button, make sure **SSH** is selected and click copy link:



- In our terminal we write this:

```
C:\Workspace
λ git clone git@github.com:Statica01/GitHubDemo.git
```

  This creates a folder with the name GitHubDemo in our workspace folder. And then it adds all files included in the project from GitHub. So, we have now copied the project and everything that have happened on GitHub down to our own computer.
  It should look like this:

```
C:\Workspace
λ git clone git@github.com:Statica01/GitHubDemo.git
Cloning into 'GitHubDemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

- Now we want to open and step into the folder we just downloaded (cloned)

```
C:\Workspace
λ cd githubdemo
```

When we do this, we will be standing in the main branch:

```
C:\Workspace\GitHubDemo (main -> origin)
λ
```

This branch is the projects main, master, head -branch. Which could be described as the branch where we only put code, files that are done and confirmed to work as we want to. We never work in this branch directly. See this branch should be seen as a "backup" branch where we always can go back and be sure that the code works and if we somehow lose our own code, we can clone this branch and get back to work.
So instead, we need to create our own branch:

- Since we haven´t created any branch for this project we will do that right away and with this commando below we create the branch and step into it to make it our active branch:

```
C:\Workspace\GitHubDemo (main -> origin)
λ git checkout -b ourBranch
Switched to a new branch 'ourBranch'

C:\Workspace\GitHubDemo (ourBranch -> origin)
λ
```

As we can see the directory is now changed to ourBranch -> origin

- If we already have a branch, we just use the commando:

Git **branch** ourBranch (without the -b)
Git **checkout** ourBranch (without the -b)

- If we are unsure if we created a branch earlier or not, we could check this by using the commando:

Git branch

this will give us a list of all branches that are connected to our project.

Now we are done and can start working on our project.
We can see that our colleague has already create two .java files. One is main.java:

```java
package java;

public class Main {

    public static void main(String[] args) {
        PrintHelloWorld printer = new PrintHelloWorld();

        printer.print();
    }
}
```

And the other is PrintHelloworld.java:

```java
package java;

public class PrintHelloWorld {
    public void print() {

        System.out.println("Hello World!!");
    }

}
```

Our work task is to add another method that print HelloWorld many times instead of just one.

- So, we go to the PrintHelloWorld.java file and we add this code:

```java
public void printManyTimes() {
    String printMany = "Hello Woooorld!";

    for (int i = 0; i < 5; i++) {
        System.out.println(printMany);
    }
}
```

- And after we created our method, we need to call it from our main class to get it to run:

```java
public class Main {

    public static void main(String[] args) {
        PrintHelloWorld printHelloWorld = new PrintHelloWorld();

        printHelloWorld.print();
        printHelloWorld.printManyTimes();
    }
}
```

- We made our changes, and we want to get this code up to GitHub.

```
C:\Workspace\GitHubDemo (ourBranch -> origin)
λ git add .

C:\Workspace\GitHubDemo (ourBranch -> origin)
λ
```

- Now our changes are registered and ready to be committed to Git. We do this by adding the command below, remember to give the commit a good descriptive name that tells what we been doing with the files. Also, a reminder to commit often and don´t let too many files be committed at the same time. This for two reasons: One, it is easier to see the changes and put good descriptions. Two, in case of merge conflicts it is easier to solve when not have to go thru tons of code at ones.

```
C:\Workspace\GitHubDemo (ourBranch -> origin)
λ git commit -m "added PrintManyTimes"
[ourBranch d0d3a30] added PrintManyTimes
 3 files changed, 6 insertions(+)
 create mode 100644 .idea/vcs.xml
 create mode 100644 out/production/GitHubDemo/java/Main.class
 create mode 100644 out/production/GitHubDemo/java/PrintHelloWorld.class

C:\Workspace\GitHubDemo (ourBranch -> origin)
```

- To check all commits that have been made we use git log:

```
C:\Workspace\GitHubDemo (ourBranch -> origin)
λ git log
commit d0d3a30a981c7918a16a6c06e0e7dfe23c83232c (HEAD -> ourBranch)
Author:
Date:   Mon Mar 8 14:26:24 2021 +0100

    added PrintManyTimes
```

- We also need to push this up to git. Right now, it is kind of only stored in a outgoing line with a title, waiting for our next order of what to do. We do that by using git push:
  If we haven´t created a remote branch (the branch that is connected to Git, not our local branch that only is used for our changes on our own computer. These branches have the same name so can be confusing in the beginning but it is just like working offline and online) we might get this message:

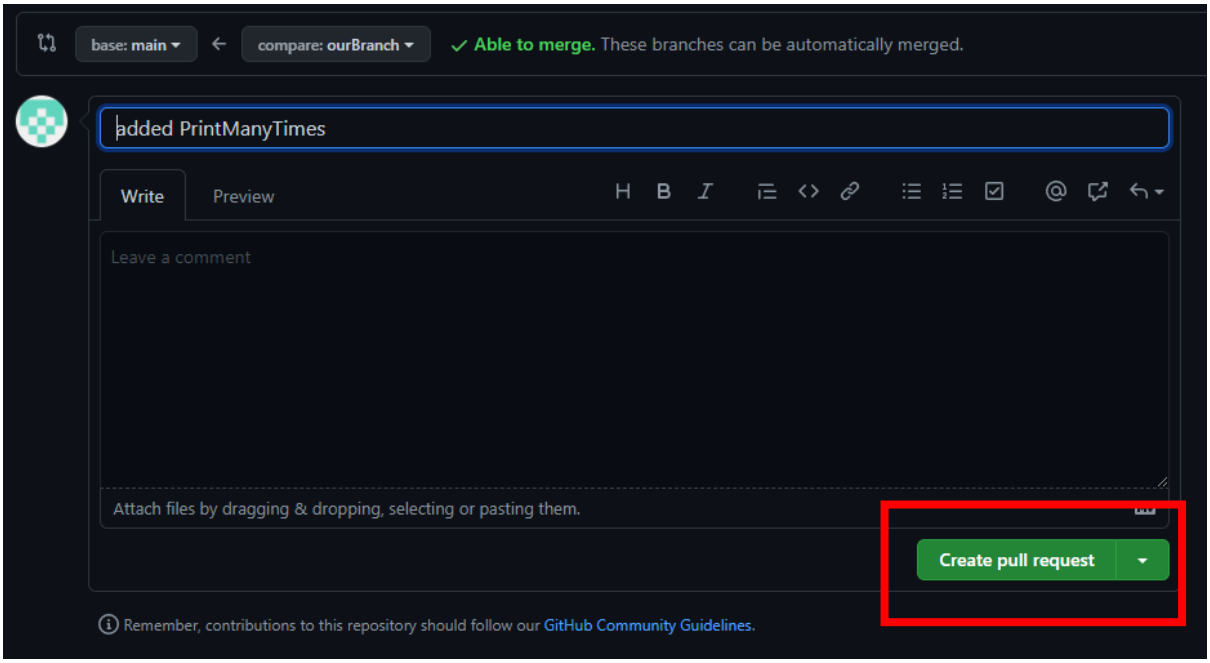
```
C:\Workspace\GitHubDemo (ourBranch -> origin)
λ git push
fatal: The current branch ourBranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin ourBranch
```

- All we have to do if this comes up is to add:

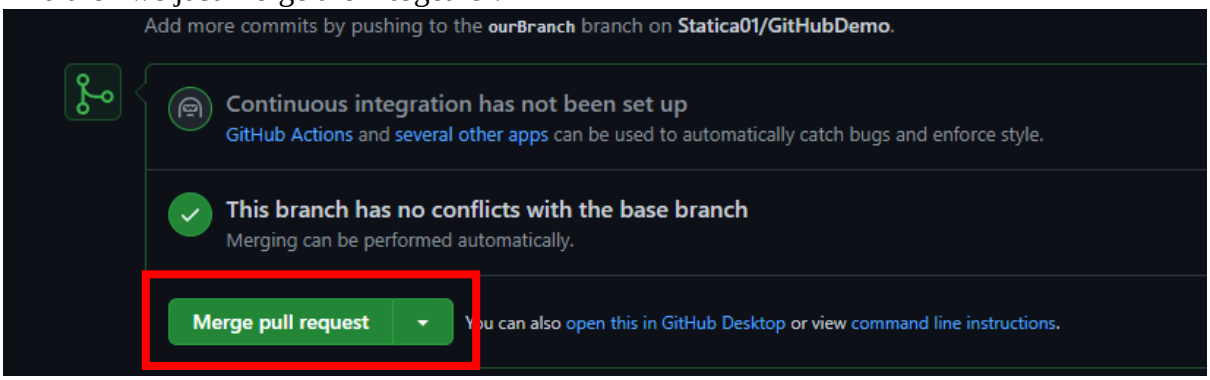
```
λ git push --set-upstream origin ourBranch
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 1.55 KiB | 318.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'ourBranch' on GitHub by visiting:
remote:      https://github.com/Statica01/GitHubDemo/pull/new/ourBranch
remote:
To github.com:Statica01/GitHubDemo.git
 * [new branch]      ourBranch -> ourBranch
Branch 'ourBranch' set up to track remote branch 'ourBranch' from 'origin'.
```

  Now we go to our GitHub page and refresh the window. We will see that it is updated:

- Click the Compare & Pull Request button.
  Now we want to merge this to our main branch. We do that by clicking the button in the right down corner:



- And then we just merge them together:



And we are done. Now our colleague can get access to our changes and work continue.