# Exercise: Source Control Systems

This document defines the exercise from the ["QA Fundamentals" Course @ Software University](https://softuni.bg/trainings/1166/qa-fundamentals-july-2015).

# Introduction

Version Control is an approach evolved to software systems, where a certain document is versioned, thus people have track for each change on the document. In the modern project management, version control systems are used in order people in the team to make changes in the project files (most probably the files contain code, but it could be in any other textual form) simultaneously and merge changes, taking a look in the history of the project changes and in some cases – reverting back to a previous change.

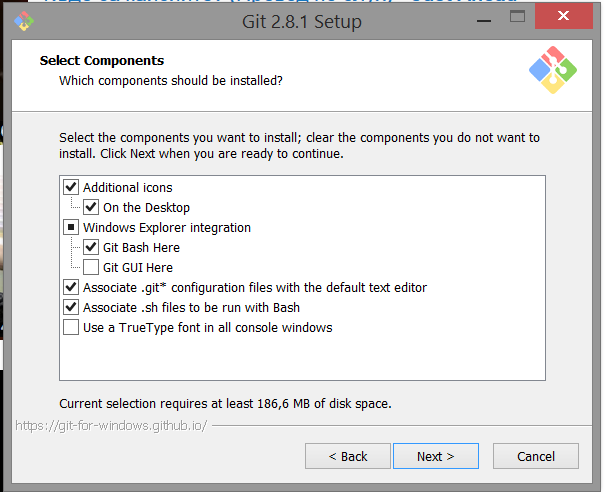
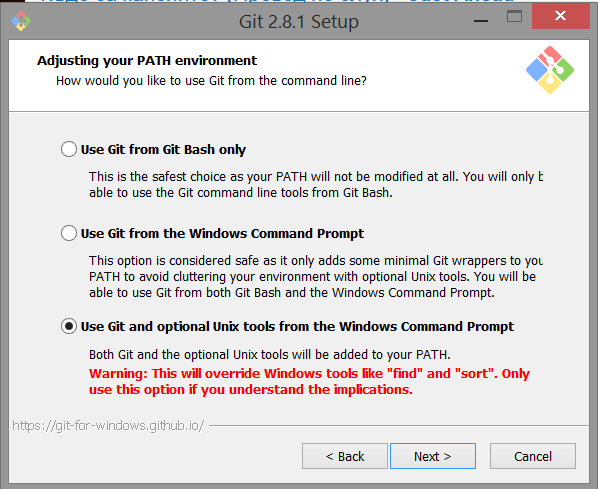
More information can be found on <https://en.wikipedia.org/wiki/Version_control>

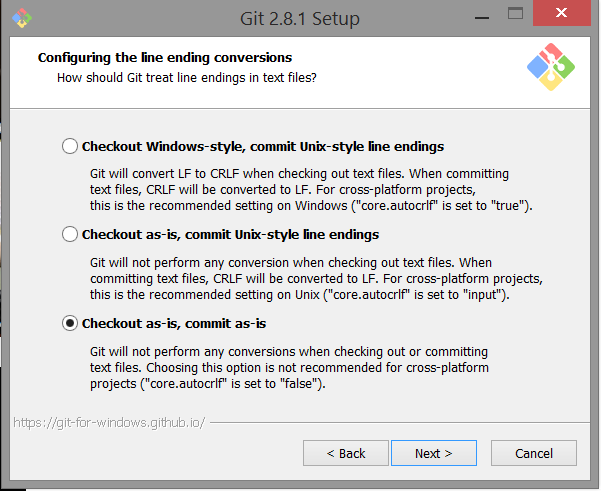
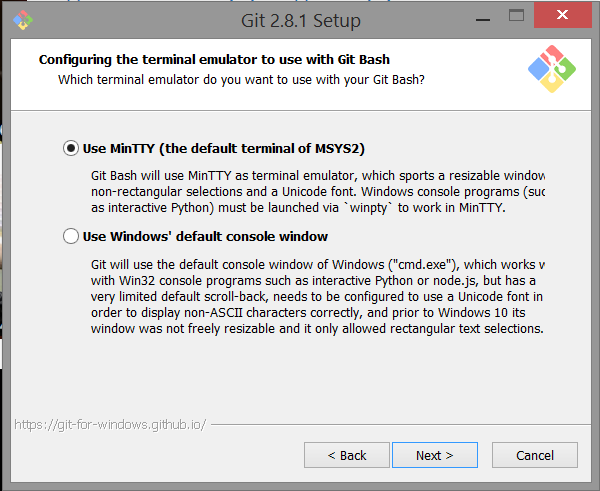
## Meet Git

Git is one of the most popular version control systems used nowadays. Beside the version control idea, git comes with its own tooling, standard library and a philosophy. For the sake of our exercise we will meet only few of its standard library

### Install Git

Git software is multiplatform. The installation guide will cover Windows installation. Go to [**https://git-scm.com/downloads**](https://git-scm.com/downloads) and download the distribution for your own OS.

** **

** **

## Meet GitHub

Version Control often requires a **working copy** (where the person make changes) and a **repository** (where the changes are stored, and other people can see them). GitHub is a platform which provides a toolset for version controlling one of which is a repository. You may open your own repository by joining GitHub

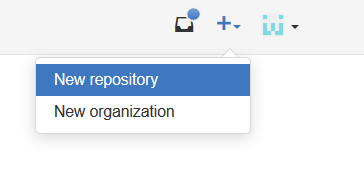
### Create a GitHub account

Go to [**https://github.com/join?source=header-home**](https://github.com/join?source=header-home) and enter your credentials for new account. You will be sent an email for verification. Verify your email too

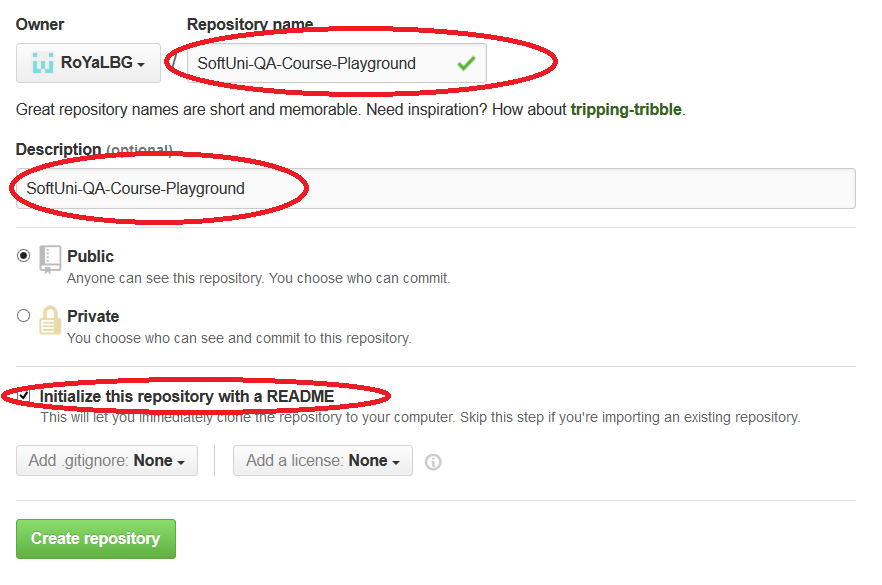
### Create a Repository

After you have successfully logged into your github account follow the steps below

1. Click on the “+” on the upper right corner and click “New repository”



1. Choose appropriate repository name e.g. “SoftUni-QA-Course-Playground” and a description. Initialize the repository with a README.



1. Click “Create repository”

## Learn Git Basics

Git is often required to be used directly on its standard library by using shell commands e.g. using the console client. This is due the full control than one have in the version control procedure. There are several graphical user interface clients, but the magic under them is unpredictable and they are often not chosen from the management in the standard toolset amongst the team (live with it, most probably your company will tell you what tools to and not to use ☹ )

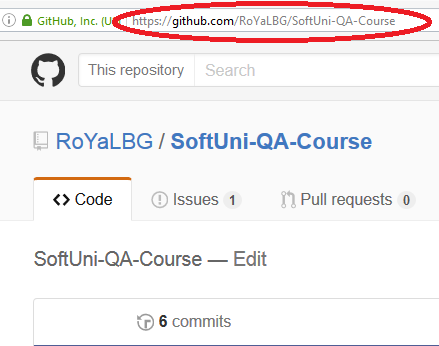
Basic commands will be covered below, but you can find most of them in <https://git-scm.com/docs>

### Clone a Repository

“Clone” is basically the process of getting a remote repository (let’s say GitHub) on your local machine and later sync changes.

1. Get your repository URL

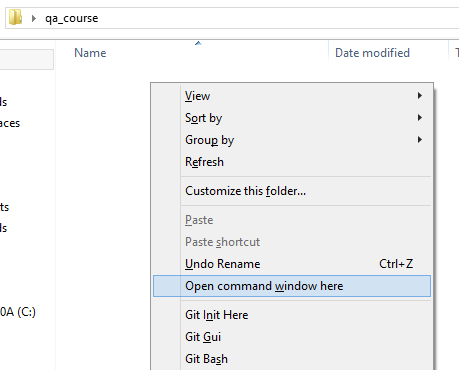
Navigate to your newly created repository in GitHub and copy the URL from the address bar.



1. Open command prompt in a directory where you would like to create the clone

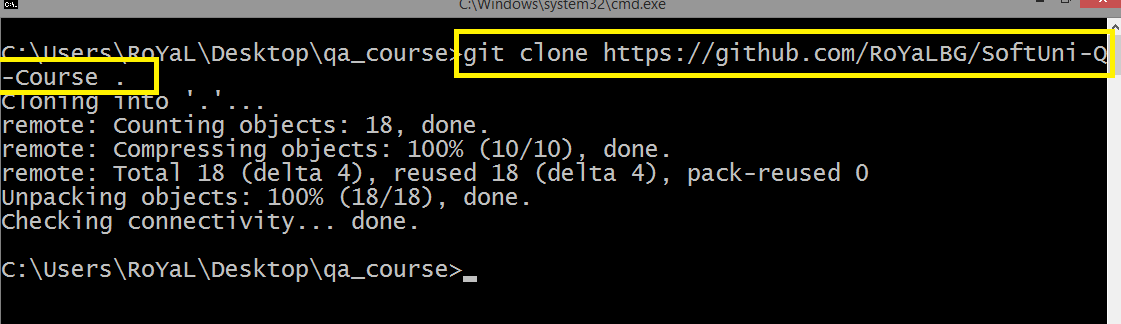
Let’s say in Desktop/qa\_course

Hold shift and click the right mouse button in the free white area. Then click “Open command window here”



1. Type “git clone {repository\_url} .” where {repository\_url} is the URL you have copied from step 1.

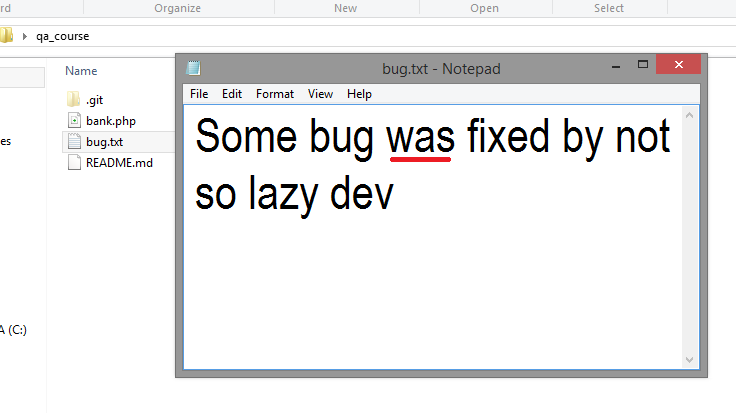
The dot “.” at the end means “clone it here, not in another folder”



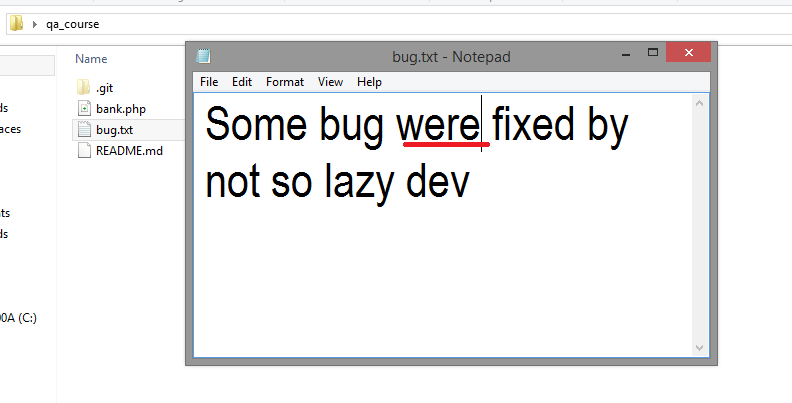
### Make a Local Change

The scenario we are trying to fake is when a new person comes to the team and have to work on a project that the rest of the team were developing until now. One needs to clone the repository and make the changes needed to finish their task. Let’s imagine we need to change one text to another text. So open a random file from your newly cloned repo in your PC and make a change.

\*\* Before



\*\* After

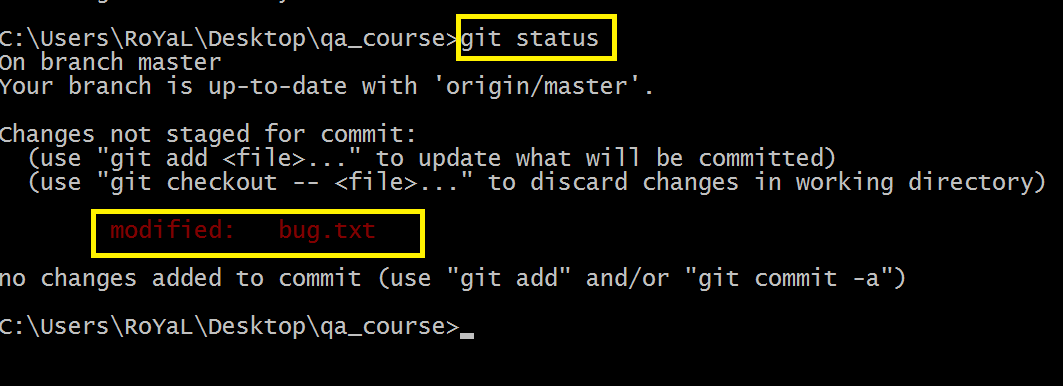


Save the file

### Check the Working Copy status

Often people need to check whether they have changed something or not, and what do they need to show the others as a change.

In the command prompt window from the clone step write “git status”

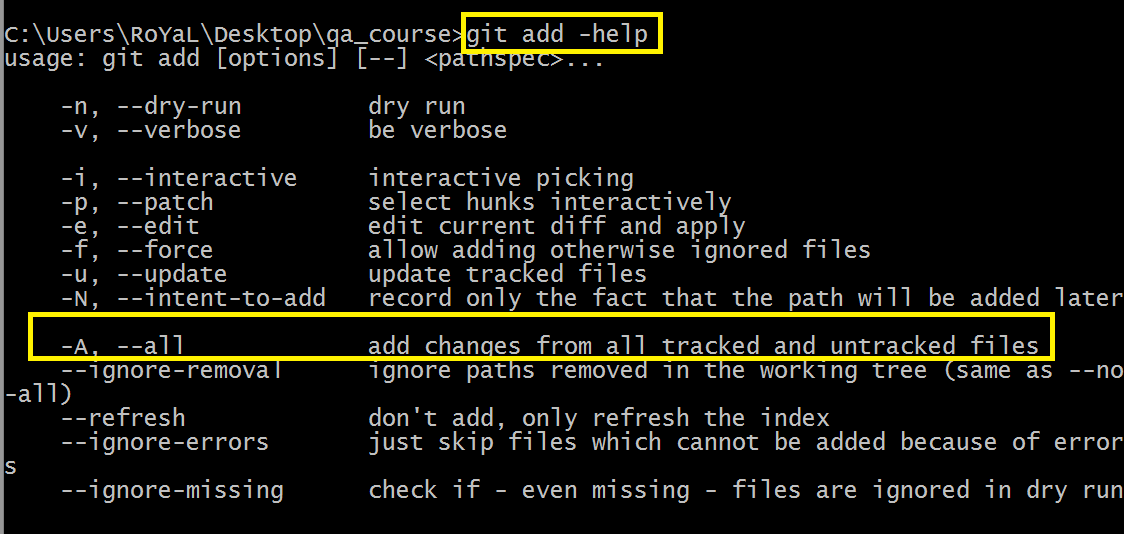


As it can be seen, Git tells us that we have modified “bug.txt”. All modified changes are only changes in the working copy. In order to be sent as a “commit” (commit is a transaction where at once a collection of changes is done) they need to be “staged”. Staging changes is done by “git add” command.

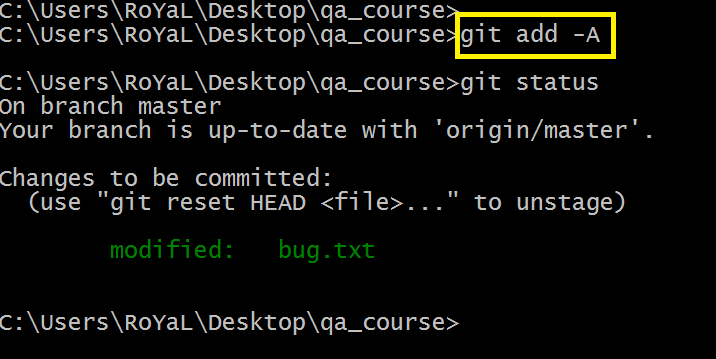
### Stage Changes

Git Add command accepts as parameters file names, wildcards, or flags. For instance if we have changed 5 files, but want in the next transaction to commit only bug.txt we can “**git add bug.txt**” only. If we want to stage all files with .txt extension we can “**git add \*.txt**”. If we want to stage the whole directory we can “**git add .**”. If we want to stage everything (which is the same as staging the root directory) we can send the flag **“-A**”.

If you run “**git add –help**” you will see the possible flags



Let’s just add everything

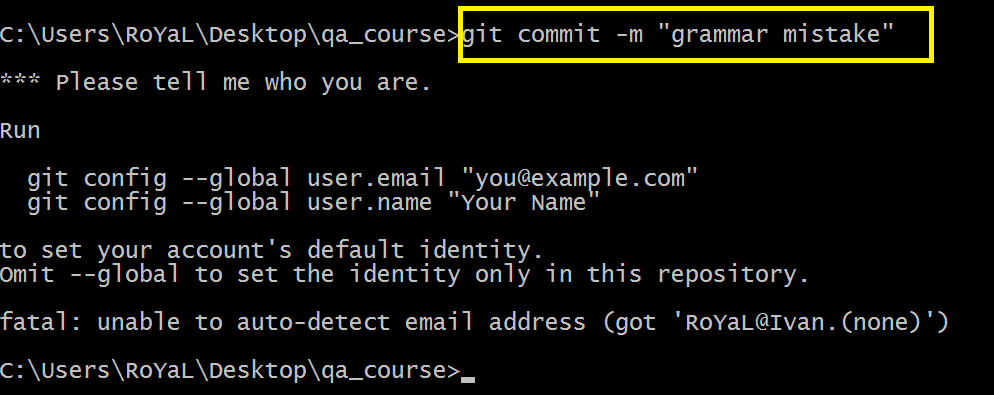


After running “**git status**” then, we can see in “green” the changes that will be processed in the transaction.

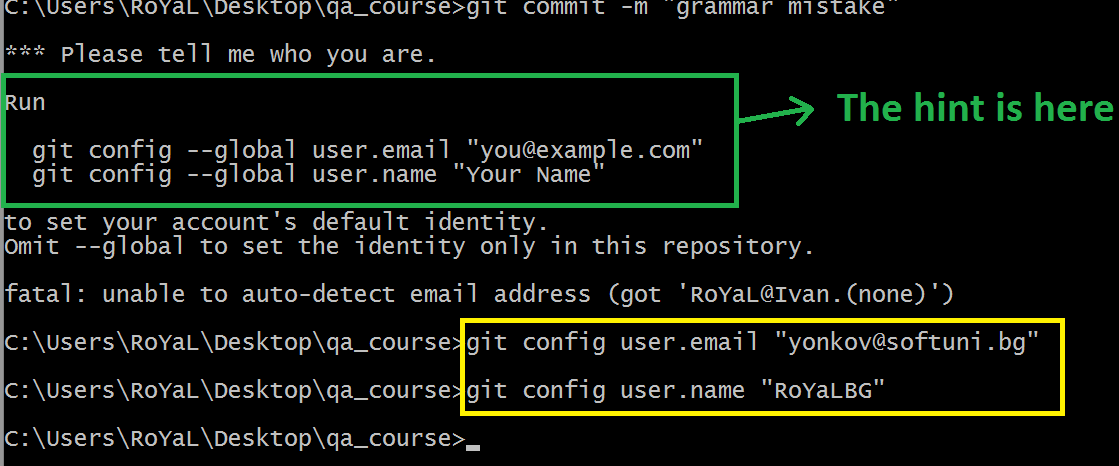
### Commit Changes

After we have successfully staged some changes it’s time to commit them. In other words to process them to our local repository. A local repository is used to version control things locally (it’s a repository, but hosted on your machine) and in some magic time to give access to someone else to clone the repository directly from your computer. We will not cover this scenario, but still the local repository exists in Git.

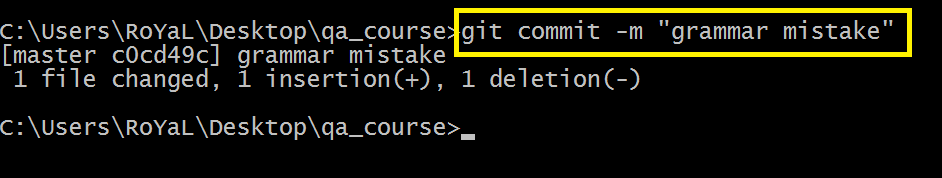
Committing changes is done via “**git commit**”. The command also accepts parameters and flags. In this particular scenario we will use the “**-m**” parameters which denotes a message. Message is used to inform others why you have done this change



If you run Git for first time, you need to tell your client how to identify you to the remote machines. You need to specify your GitHub email address and your GitHub username. There is an optional “**—global**” flag which will be used on the other repositories too (you will not be required to do this on the other repositories).



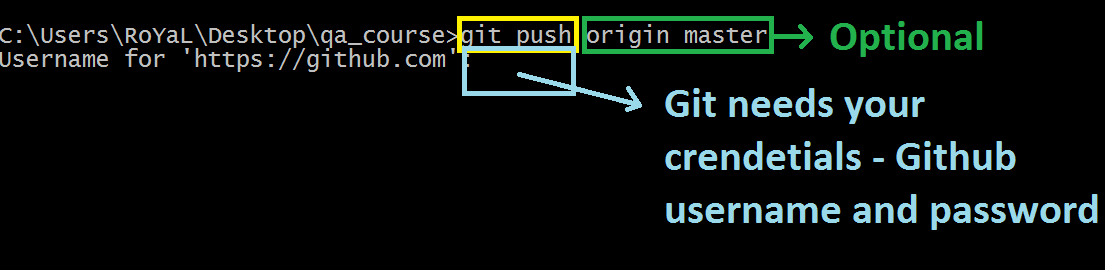
Let’s commit again

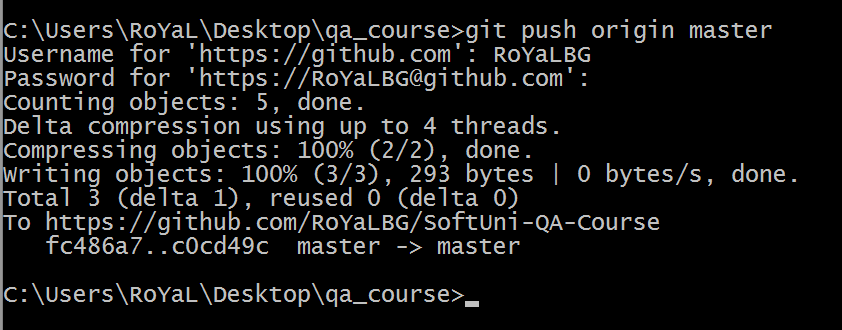


Now the changes are processed to your local repository. If by some rare case you give permissions to another computer to clone your local repository, the one will see these changes. However, they are still not on the Repository which the team uses – the one in github.

In order to process all of your commits **from your local repository to your synchronized repository (GitHub)** you need to “**push**” the commits

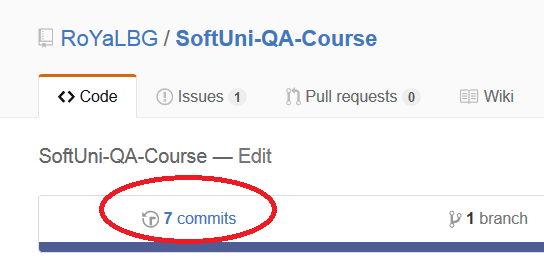
Depends on your version of git client you need to specify branch or not. It’s either only “**git push**” or “**git push origin master**”



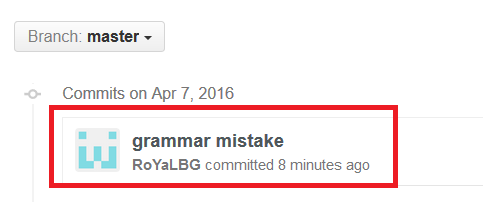


Changes are now processed to the GitHub repository.

### Check GitHub



We can see the commits done so far



And then see the changes in this commit

