

CMPE414 SOFTWARE ENGINEERING

Instr:Cain Kazımoglu

Subject: Word Counter Software by Using GitHub

PREPARED BY:

20121364 JAMAL

20111475 EMRE TURAL

20080548 ÖMER FARUK KENGER

CONTENT:

# 1-Why use Version Control System

# 2-What are Git and GitHub

# 3-Basic Git Commands

# 4-Using GitHub in a Project

# 1-Why use Version Control System

Version control is a system that records changes to a file or set of files over time so that you can recal specific versions later. A VCS allows you to revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduce an issue and when. Using VCS also means that if you screw things up or lose files, you can generally recover easly.

# 2-What are Git and GitHub

In computer programing, a distributed revision control (DRC) offers multiple developers to work on a given project without requiring them to share a common network. So, in simple words, Git is a distributed revision control.

Consider this example:your client asks you to remove a feature, let’s call it “Feature-X”. How will a developer remove the code of “Feature-X” from code base safely?

Using branches is the solution to the above mentioned common development problem. Although, Git wasn’t the first version control system (VCS) that introduce the concept of branching, it is the first that makes it user fiendly.

GitHub is a web-based Git repository hosting service, which offers all the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. It provide access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

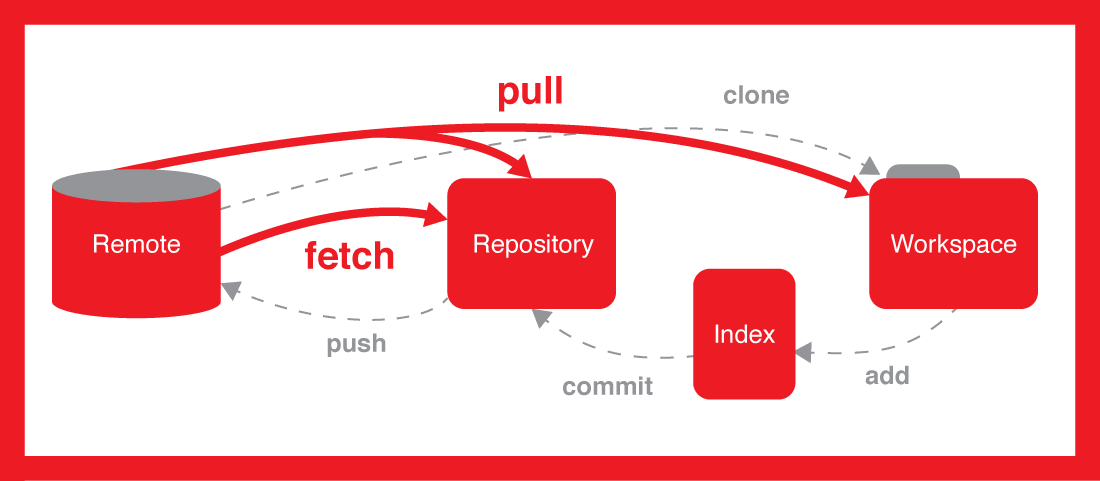


Figure1:Basic Git topology.

# 3-Basic Git Commands

For simplicty, the commands are divided into four section: initializing, staging, committing, and maintaining.

Specifically, i will be covering these commands: config, init, clone, add, status, commit, push, pull, log, and reset.

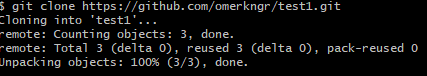
3.1- Initializing

Before you can start taking snapshots though, you have to establish a repository by either creating one from scratch or by copying an already established repository.

git init: initialize a new empty repository on your local machine.



git clone <url> clones or copies a git repository to your local machine. When you clone, you do not need to initialize a new repository because cloning copies all file, include the actual repository.



3.2- Staging

The goal of any VCS is to save periodic snapshots of your project. Once you have a snapshot saved, you can feel safe working on your project as you can always revert back to an earlier snapshot if you make a huge mistake.

git add <filename> : adds a new file to staging. This file is ready to be commited.

git add . : adds all files in the current directory and all sub directories to your local working directory.

git status : list all files ready to be committed, which have been added to staging.

3.3- Committing

After taking a snapshot, you want to move the snapshot from staging to your actual repository. You have the option of continuing to work locally or sharing those commits to aremote repository,perhaps on your web hosting platform or on GitHub.

git commit -am ‘<add note>’  :  commits new and updated files - moving them from the staging queue to your local repository. Make sure the note you add is relevant - that is, it summarizes the changes or updates you’ve made.

git push origin master gathers all the committed files from your local repository and uploads them to a remote repository. Keep in mind that not all files are included in the upload; only new files and files that contain changes. Put another way, this command syncs your local repository and external repository so they are *exactly* the same.

3.4- Maintaining

git log is used to view the history of your repository.

$ git log

commit 15f4b6c44b3c8924caabfac9e4be11946e72acfb

Author: Michael Herman <hermanmu@gmail.com>

Date: Thu Oct 10 22:56:30 2013

Update readme.md

commit 93127eed8fa0c3b4df7bbdabd7d6aefa312c31a3

Author: Michael Herman <hermanmu@gmail.com>

Date: Thu Oct 10 22:45:14 2013

initial commit

Here we can see the local commits, along with the commit number, author info, date, and the note from the commit - which is exactly why it’s good to use detailed messages with your commits.

# 4-Using GitHub in a Project

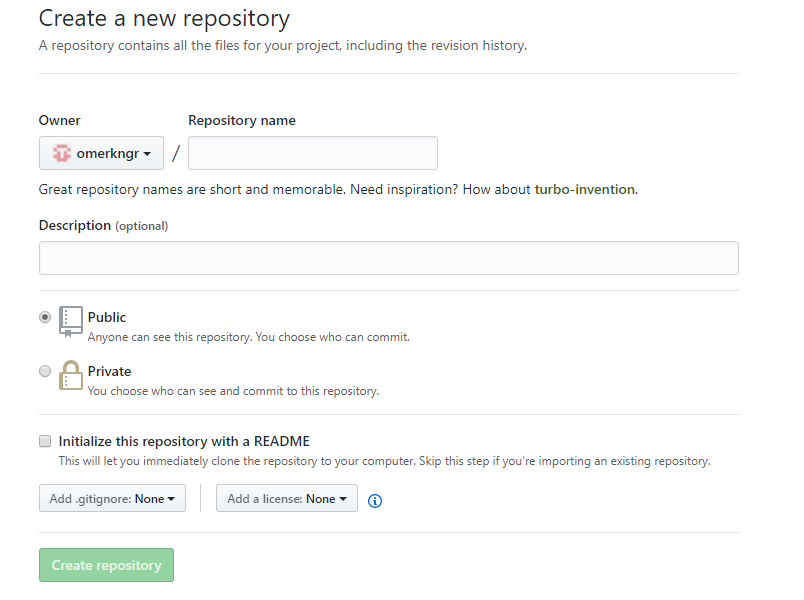
Using Github in a web browser.

* **Create a Repository**:

**1-** go to [**https://github.com**](https://github.com)

**2-**log in

**3-**click + to create new repository



**4-**Click initialize with README

**5-**Click Create Repository

* **Create a commit**

**1-**click on a file

**2-**edit in browser

**3-**Scroll down to “Commit changes”

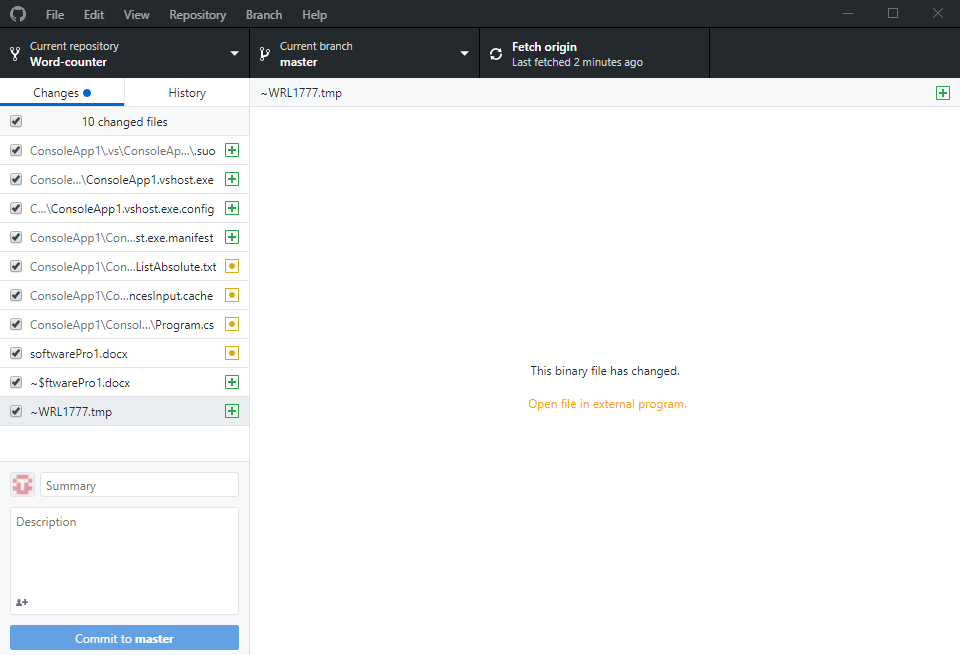
**4-**fill in the information describing the changes

**5-**Click “Commit changes”

Useing Github in a GUI application:

* **Create repository**

**1-** Click file , then chose “new repository” or “clone repository”



**2-** Choose your repository from the list

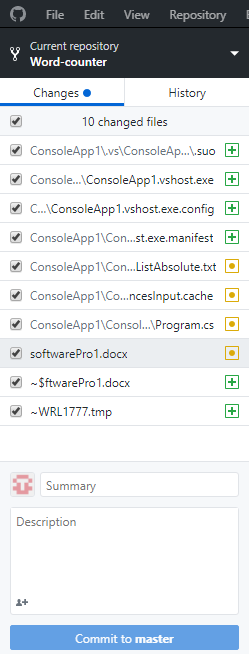
**3-** Choose directory to save in (leave default)

* **Create a commit:**

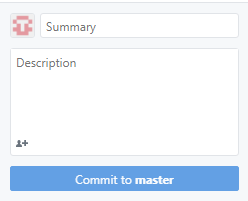
**1-** Open the folder

**3-** Change the a file

**4-** Check changes on Github app



**5-**Create a Commit message



**6-**Click "Commit to master" Sync:

**7-** Click Sync button (this pushes our local changes to the GitHub, plus pulls any updates from the web version)