**Software Development and Testing IT7320**

**ACTIVITY LAB 1 – GIT AND GITHUB**

Team Members: Submitted to:

Anjali Khandelwal Chalinor Baliuag

Kwino Pineda

Ralph Saplan

Thanh Huynh

Due Date: Friday 14 August 2015

Contents

[1. Understanding Version Control (Anjali) 2](#_Toc427250387)

[a. Launch Terminal of Git Shell 2](#_Toc427250388)

[b. Check Git installation with Git version: 3](#_Toc427250389)

[c. Launch GitHub for desktop: 3](#_Toc427250390)

[2. Surveying the GitHub Platform (Kwinno) 4](#_Toc427250391)

[a. Difference between the Git DVCS and GitHub 4](#_Toc427250392)

[b. Access the common project components of daily GitHub interactions 4](#_Toc427250393)

[c. Utilize project management components (Issues, Milestones, Collaborators and Teams) 5](#_Toc427250394)

[d. Recognize best document types for version control (code, CVS/TSV, small binaries) 7](#_Toc427250395)

[3. Creating a Hosted Repository (Thanh) 8](#_Toc427250396)

[a. Visit GitHub.com 8](#_Toc427250397)

[b. Create a repository and ways to interaction 8](#_Toc427250398)

[4. Touring Content Versioning on GitHub (Anjali) 12](#_Toc427250399)

[a. Browse to one of your repositories: 12](#_Toc427250400)

[b. Create file(s) and commit the new content : 13](#_Toc427250401)

[c. Editing files in your repository 13](#_Toc427250402)

[d. Remove a file and commit the removal 15](#_Toc427250403)

[e. Change a filename and commit the path change 15](#_Toc427250404)

[f. Examine the Commits page of change history: 17](#_Toc427250405)

[4. Acquiring Repositories Locally (Ralph) 18](#_Toc427250406)

# Understanding Version Control (Anjali)

## Launch Terminal of Git Shell

**Version control system (VCS)** is a software that helps software developers to work together and maintain a complete history of their work without overwriting each other’s changes.

**Git** is a distributed version control system. DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the sever goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository. Git does not rely on the central server and that is why we can perform many operations when we are offline. We can commit changes, create branches, view logs, and perform other operations when we are offline. We require network connection only to publish our changes and take the latest changes.

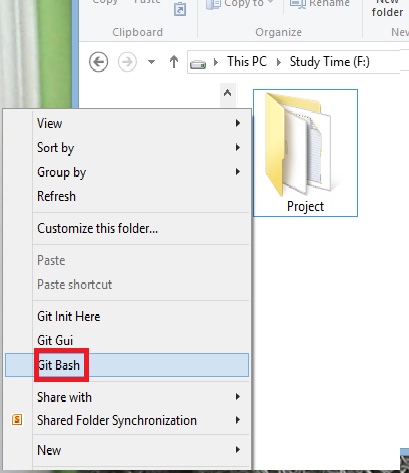
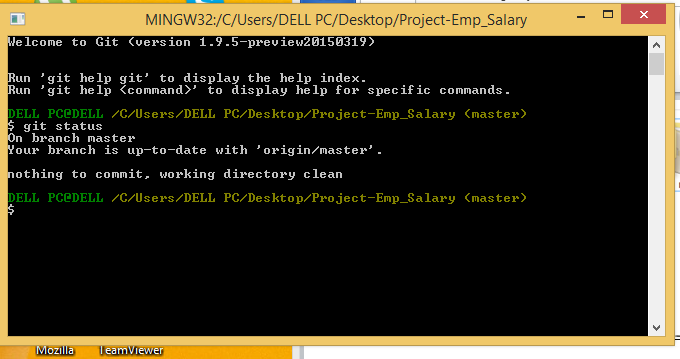
We can install Git from www.Git-scm .com. It helps developer to work on project on their local computer and they can copy there whole local repo on the server using Github to collaborate with other team members.

Here are all screenshots of installation of Git on local computer.

* Open [www.Git-scm.com](http://www.git-scm.com) to get the download link.
* Download Git by clicking on download link. Latest version 2.5.0.



* Follow the instructions and complete the setup of Git. Here are some screenshots of installation of Git.
* After completing the Git installation we can open a folder in Git bash and after initialize it as a working directory we can start working with Git.

****

## Check Git installation with Git version:

To check Git version type command “Git –version” in Git bash

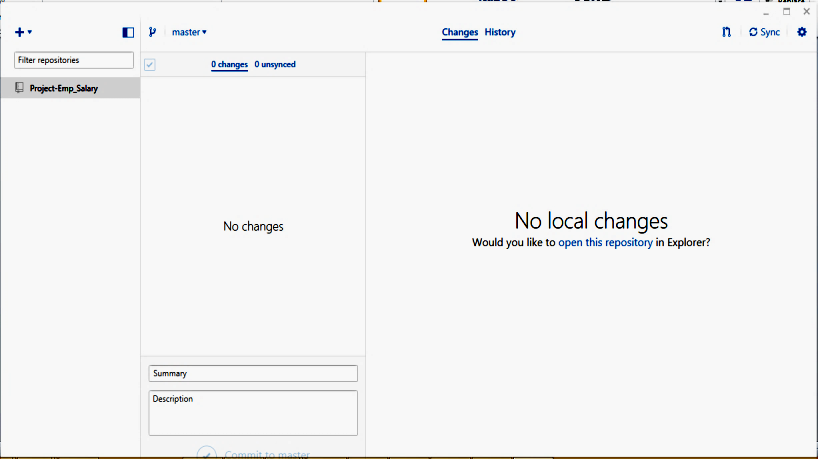
As command line is responding to Git commands, it shows Git has been installed on computer successfully and it is in working state

## Launch GitHub for desktop:

GitHub desktop is an application which can be installed on a local computer to manage remote repositories. Moreover we can perform Git commands like push, commit directly by clicking on it from the application rather than typing them in command line environment. In order to setup Github desktop application we need Git hub account.

Sign up on GitHub website [www.Github.com](http://www.github.com) to create GitHub account.

1. After that we are on the new webpage we need to sign up by type in the details like username, email address, password .
2. After finishing sign up we can sign in any time when we need to access GitHub account by click on sign in and type in username and password on website [www.Github.com](http://www.github.com)
3. To launch a GitHub Desktop download it from [https://windows.Github.com/](https://windows.github.com/) on local computer.



# Surveying the GitHub Platform (Kwinno)

## Difference between the Git DVCS and GitHub

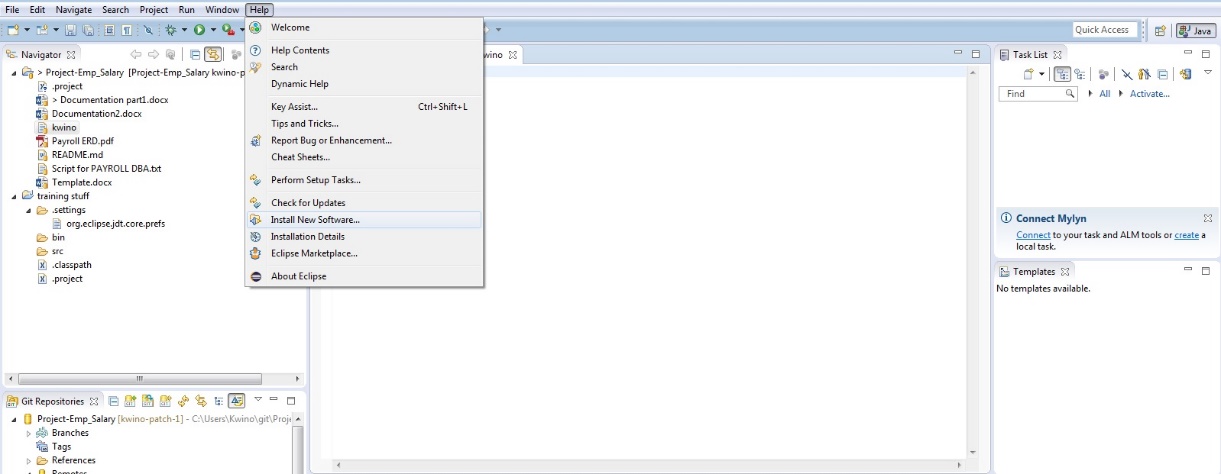
Git DVCS (Distributed Revision Control System) uses GitBash, a **command-line tool** to connect to your work locally and its working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server. (Wikipedia, 2015)[[1]](#footnote-1).

While on the other hand Github, is a web based Git repository hosting system which is free and open source used commonly with team collaborating with a project. It is a **web based graphical interface system** and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. (Wikipedia, 2015)[[2]](#footnote-2). Github also provides private account which needed to pay only few people used it. (Wikipedia, 2015). Github use can be used remotely using Gitbash or other application (Netbeans, Eclipse and etc.)

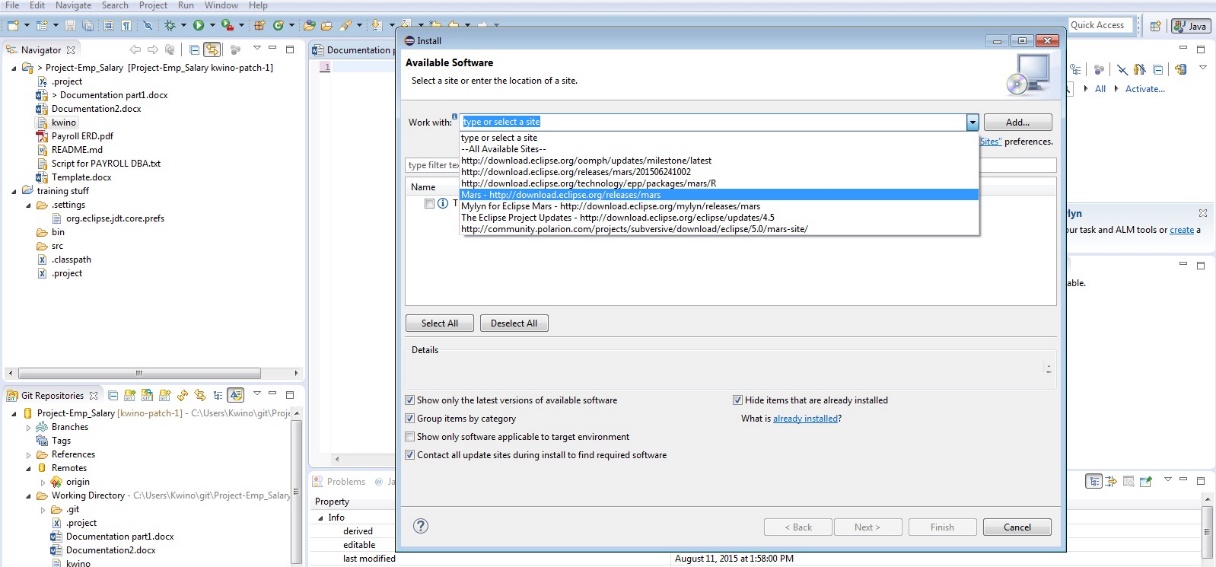
## Access the common project components of daily GitHub interactions

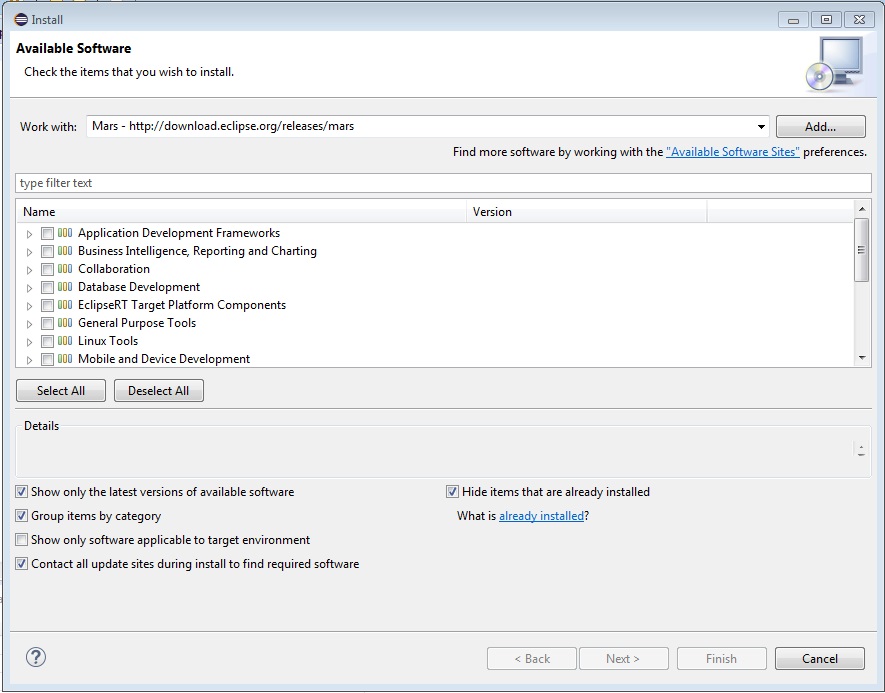
There are many ways to connect a project to Github, one of the way is using Eclipse. First install the Eclipse Mars Java IDE then after install, open eclipse and

* Go to Help menu.
* Click Install New Software.



* Choose <http://download.eclipse.org/releases/mars/> to download Github plugin
* Click Collaboration and Install all the Github plugins under Collaboration.

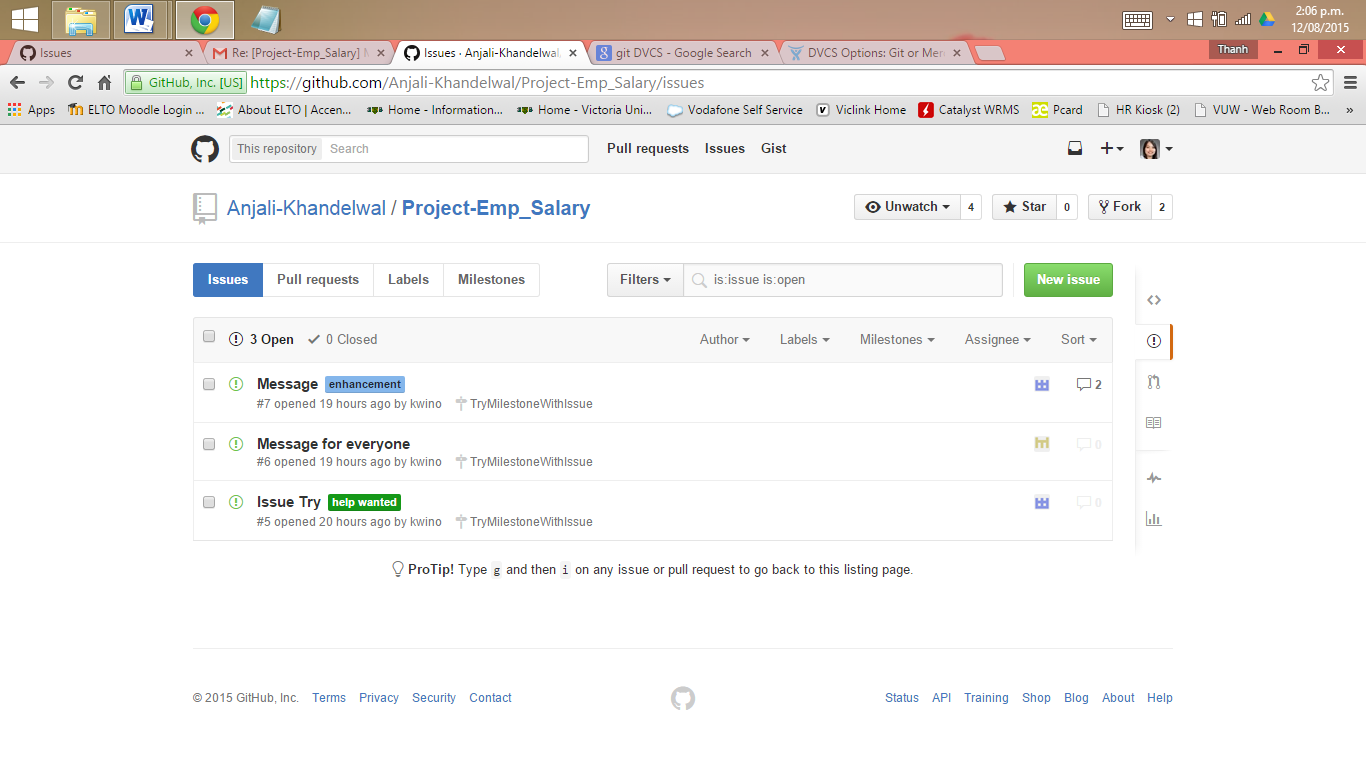




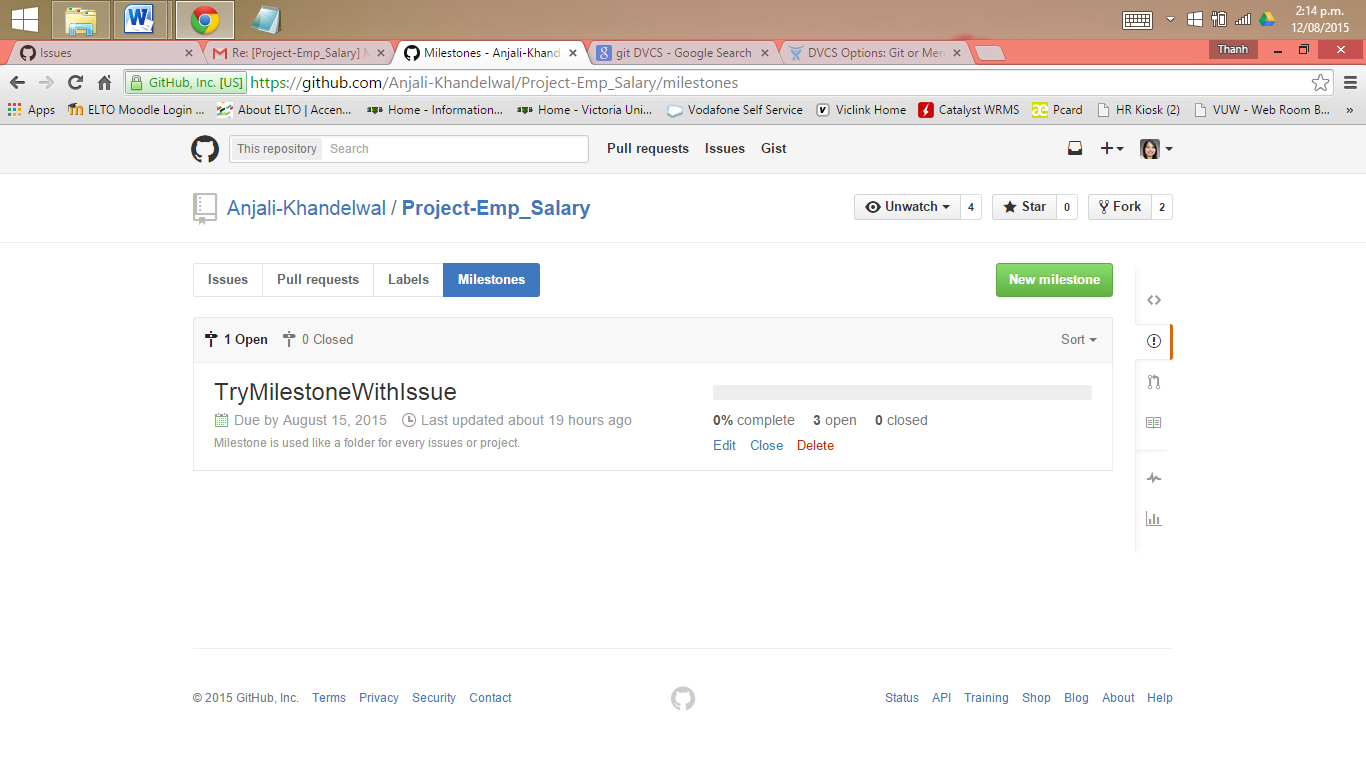
* Then Go To **File Menu** select import and copy the clone in Github paste it in the URI text box. Next is select the repository you want to fetch from Github to your local computer. And select as **Import as general project**.

## Utilize project management components (Issues, Milestones, Collaborators and Teams)

**Issues** is a feature of Github as an integrated bug and enhancement tracker where all the things such as request, suggestion and etc. are being post as message and discuss with the group. In **issues** you can assign to a specific member and put a **labels** and add color if it is a duplicate, bug, enhancement and others or make your own label to recognize what is the message all about.



You can create a **Milestones** for your issues and set the due date of your project to monitor. Creating milestone for every project is a great help all issue are being sorted or group by according to what project you are working on and all open, close of issue and completeness of the project were being recorded to display. Members of a team with issues regarding the project will be posted



In Github, a repository owned by a user account has 2 kinds of permission level (Github, n.d.). One is **repository owner** and **collaborators**. **Repository owner** has:

* all the permission
* full control of the repository
* owner can add collaborators
* change the visibility of the repository(from public to private or from private to public), and delete repository

There is only one owner of a repository owned by a user account.

**Collaborators** are users who are assigned to work on a project its either same or different project. A **collaborator** has the permission to:

* Push to (write), pull from (read), and fork (copy) the repository
* Apply labels and milestones
* Open, close, re-open, and assign issues
* Edit and delete comments on commits, pull requests, and issues
* Merge and close pull requests
* Send pull requests from forks of the repository
* Create and edit Wikis
* Create and edit Releases
* Remove themselves as collaborators on the repository (Github Help, 2015)[[3]](#footnote-3)

Under **collaborators** is a feature called **team** wherein Github are included and group by to work on a same project. The administrator of the team can assigned team members what privileged he/she can use it’s either pull only, push and pull, pull and administrative. The administrator of the project can limit the privilege and see the information of the members without their permission.

## Recognize best document types for version control (code, CVS/TSV, small binaries)

**CVS (Concurrent Version System)**, uses a client–server architecture: a server stores the current version(s) of a project and its history, and clients connect to the server in order to "check out" a complete copy of the project, work on this copy and then later "check in" their changes (Wikipedia, 2015)[[4]](#footnote-4). CVS can be use locally to your computer and it’s free to use by everyone such as Git, Mercurial and SVN.

While Code applications for version control are the IDE applications such as **Eclipse** and **NetBeans** where **Git plugins** are needed to download and install. Doing this there’s no need to use Git to add, commit, push, pull and etc. to remotely access your Github repository. With the help of IDE applications project files can import from Github to Eclipse or NetBeans. You just need to:

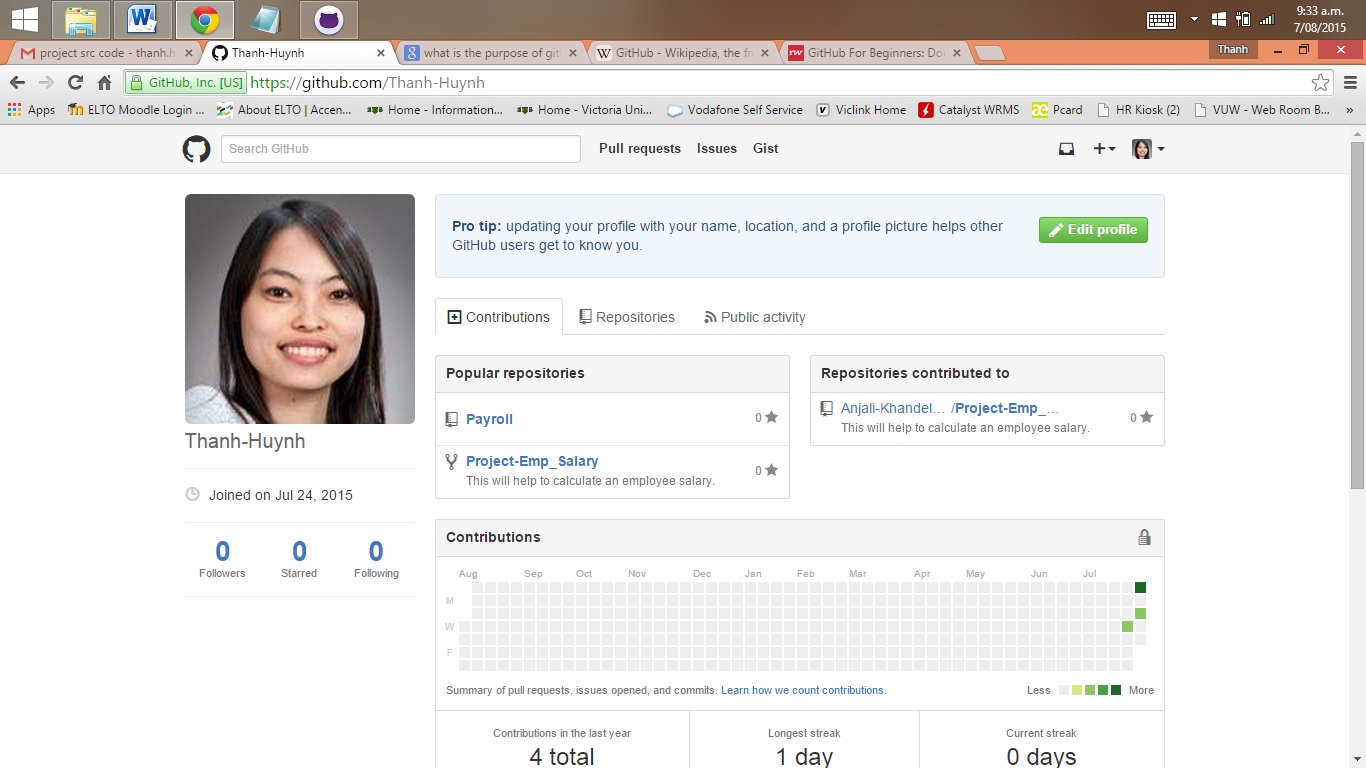
* connect it by creating an account to Github
* cloning the repository of your project from Github to Eclipse
* import project files from the repository
* commit all the changes from the imported project files to remotely control, change, monitor, and store your project online using Github.

By using IDE applications coders or programmers can use Github easily and more convenient. All the things that are in Git will be available and much better with the help of code applications.

# Creating a Hosted Repository (Thanh)

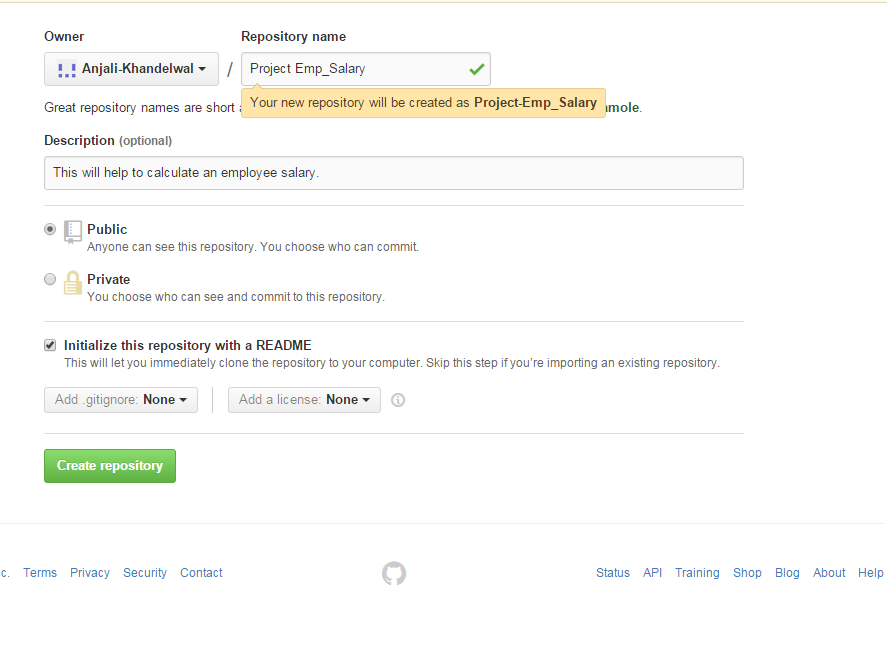
## Visit GitHub.com

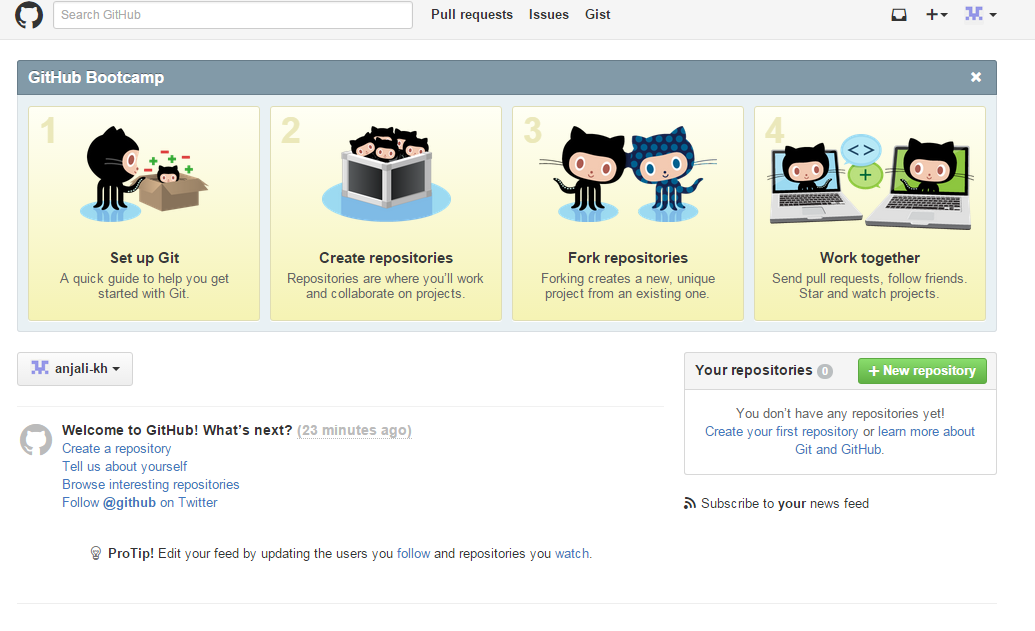




* + Go to the website GitHub.com website
  + Sign up an free account with username (Name to be recognized by team member), email address and password
  + The reason for having a GitHub account:
    - This is free for an unlimited quantity of public repositories.
    - This is a place where we build a profile, upload a project to share connect with the team members by “following” their accounts
    - GitHub can store any type of files like programs, code, text documents and so on in the project folders.
    - It is a public place where everyone can access and see your work. However GitHub does make sure the ownership is well retained. It states in their Term of Service that *“We claim no intellectual property rights over the material you provide to the Service. Your profile and materials uploaded remain yours.”*

## Create a repository and ways to interaction

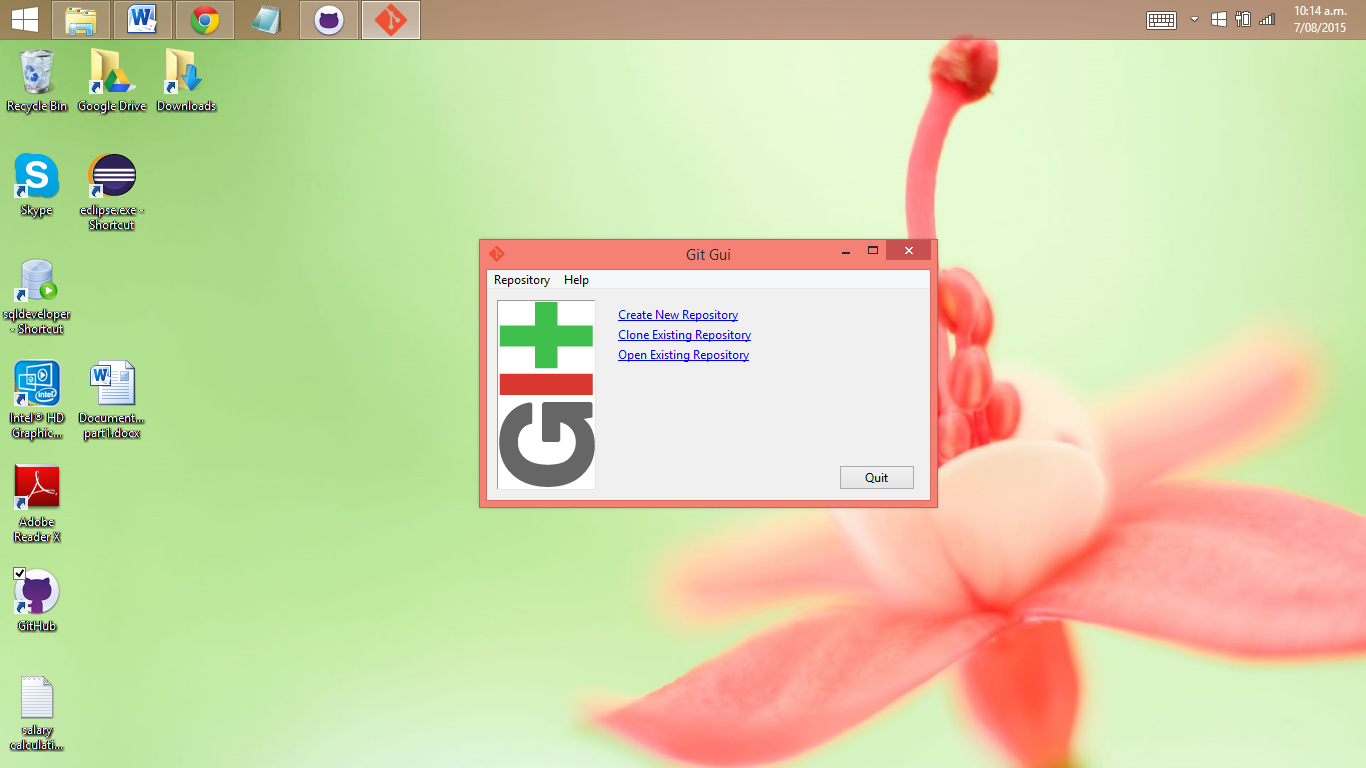
**Using GitHub Web user interface**



* + Create a short and meaningful name for your project repository. This is to help yourself and your team members to understand and remember. Our repository is **Project-Emp\_Salary**
  + Add a description of the project repository: **This will help to calculate an employee salary**
  + Choose public repository because it is free and can be accessed by everyone.
  + Select Initialize this repository with a README
  + Click Create repository. Now we are ready for the first commit

The above is one of the ways to create a repository using GitHub website user interface. There are other ways to create a repository such as:

**Using GitHub GUI**



**Using Git init at the command line**

There are two approaches:

1. Takes an existing project or directory and imports it into Git.

We will create a new Git repository skeleton \*.Git by go to the project’s directory and type”$ Git init”.

At this point, nothing in your project is tracked yet. To begin tracking those files and do an initial commit by

$ Git add \*.c

$ Git add README

$ Git commit -m 'initial project version'

At this point, you have a Git repository with tracked files and an initial commit.

1. Clones an existing Git repository from another server.

If there is a project you’d like to contribute to — the command you need is “Git clone [url]”. This will help Git to copy of all data that the server has.

**Using Eclipse**

Download EGit at [http://www.eclipse.org/eGit/download/](http://www.eclipse.org/egit/download/)

An Eclipse plug-in to use the distributed version control system Git.

The toolbar entries allow you to add an existing local Git repository to the view, clone a Git repository and to create a new Git repository.

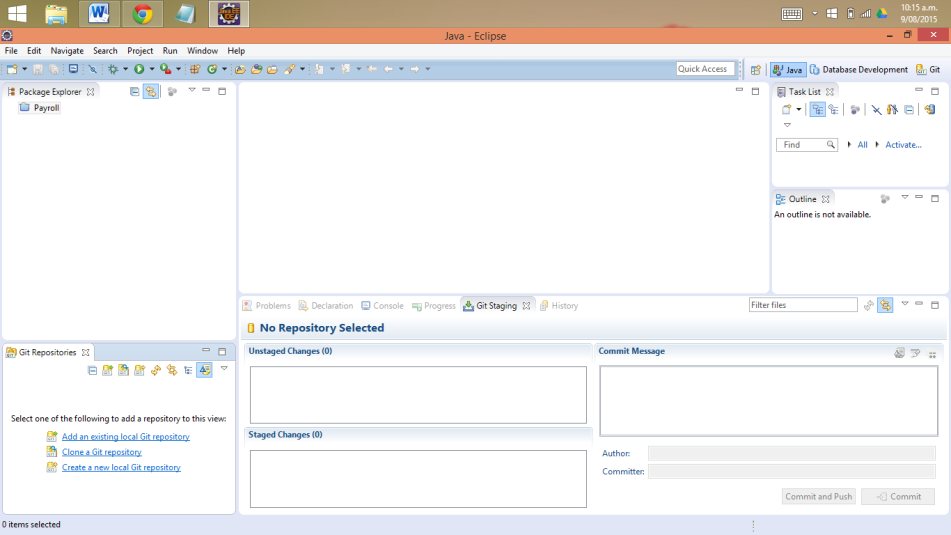
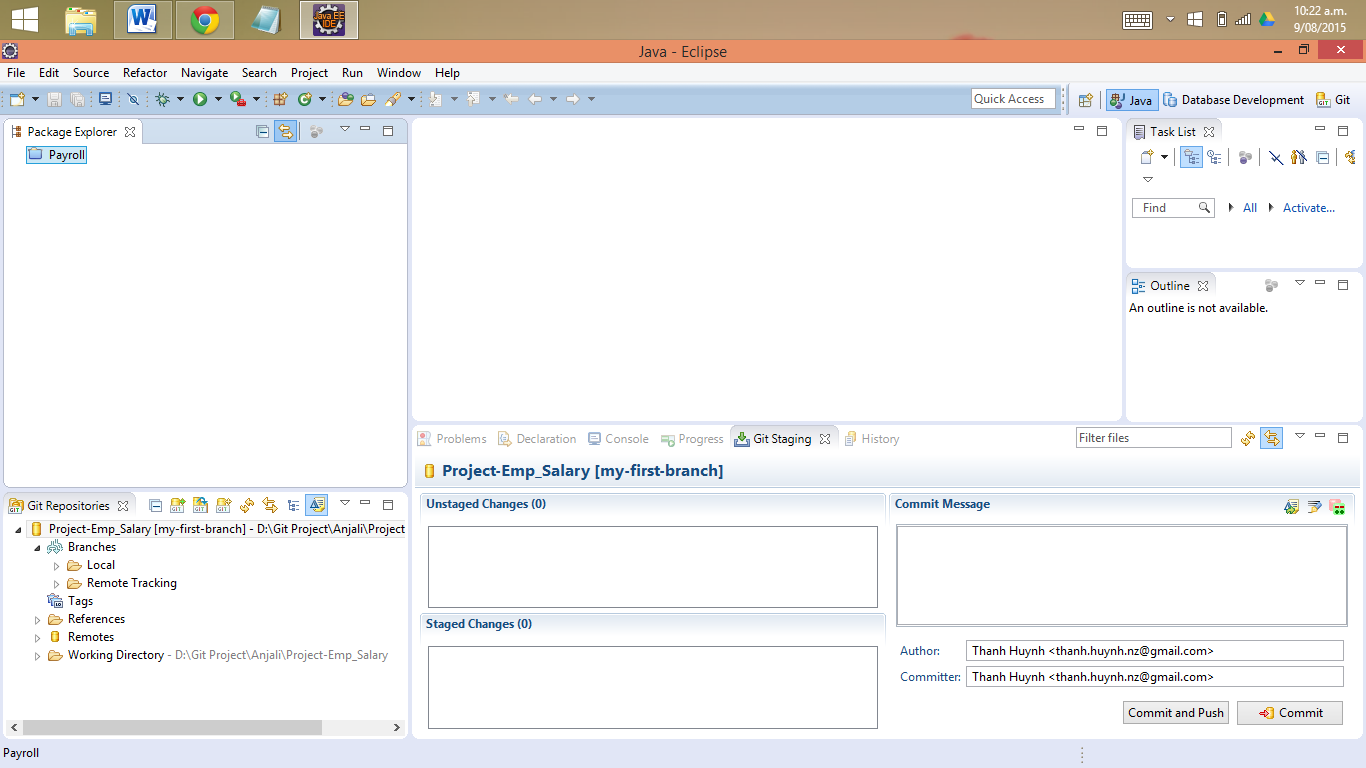
There is a recommendation to place your Git repositories outside the Eclipse workspace. This separates your Git repository from any additional meta-data which Eclipse might create.

Once you have added an existing local Git repository (Project-Emp\_Salary). It will show under the Git repositories view (as below). A right-click any element in the Git repositories view allows you to perform related Git operations. For example if you right-click on a branch you can “checkout” the branch or delete it.

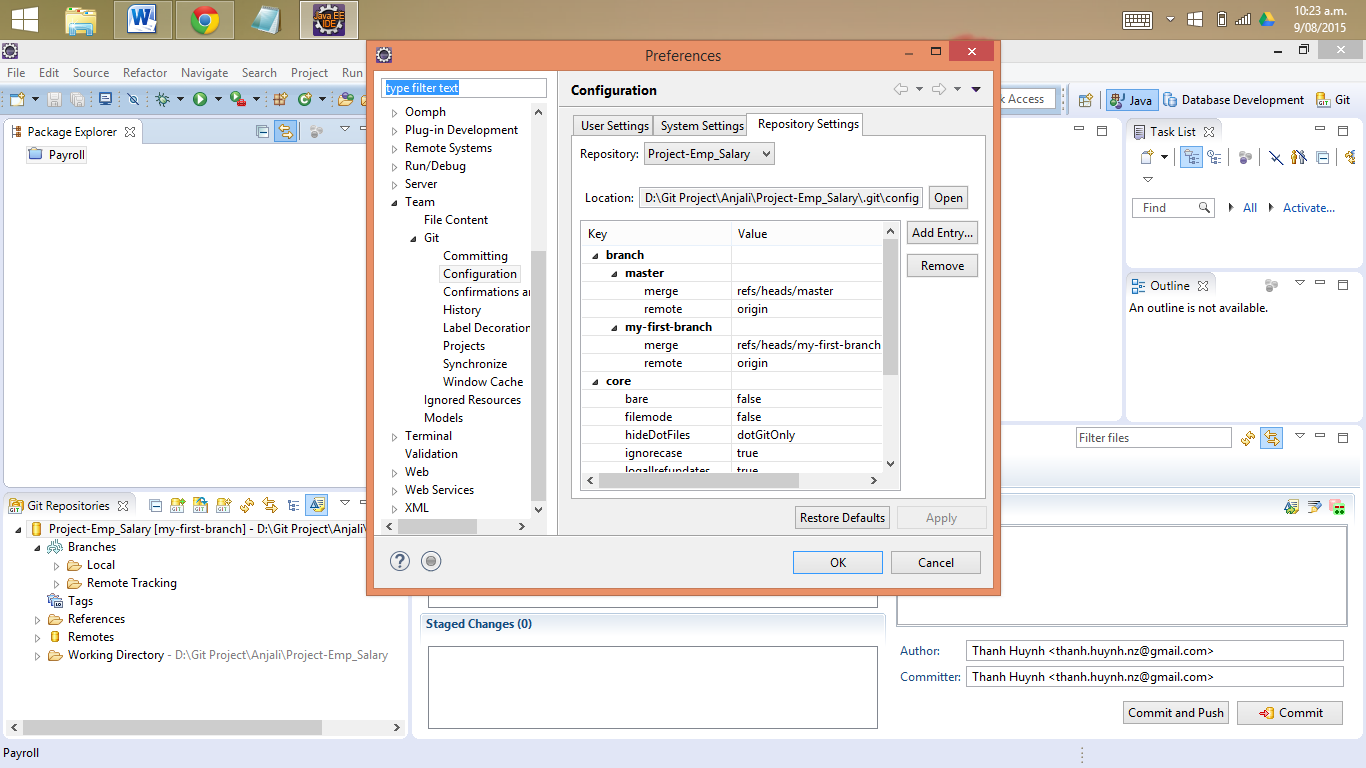
The Git Staging view is non-modal, you can switch between different repositories or even restart Eclipse without losing a commit message and it allows incremental staging for changes.

Open the Git Staging view via the Window → Show View → Other → Git → Git Staging menu.

Also you can choose any file which has changed and move them into the “Staged Changes” area. To commit the staged changes you write a comment message and select the “Commit”



The Git configuration settings can be adjusted via the Eclipse preference setting. Select Window → Preferences → Team → Git → Configuration to see the current configuration and to change it.

You can add entries to your Git configuration by pressing the Add Entries button on the Git Configuration preference page.

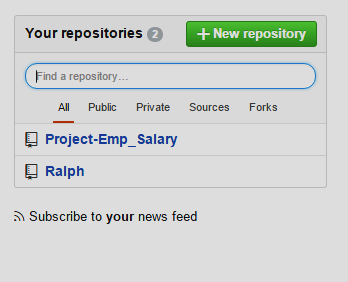
# Touring Content Versioning on GitHub (Anjali)

## Browse to one of your repositories:

We can create repository and browse that in many ways. We need to browse our repository on Github to see the changes done by other members, to push our changes from local Git repository. If we are working on a project in a team it is very important to be updated in order to collaborate properly and to make sure every member in team going in a right direction. Every member in team can get the whole copy of remote repository from GitHub in his local computer. The admin can make the remote repository public or private on Github according to the requirements. Public Repositories can be accessed by anyone while private repositories can’t be accessed without owner’s permission.

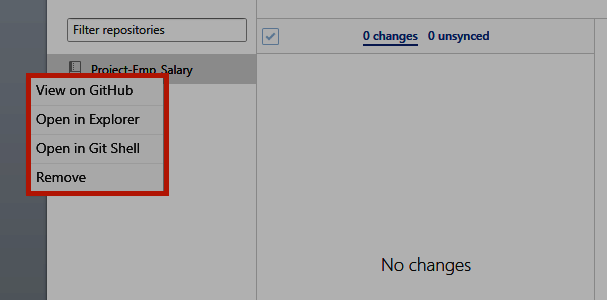
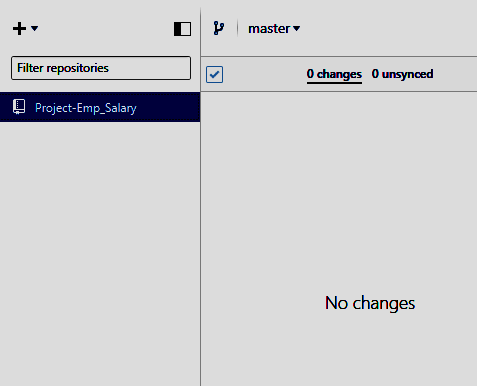
We can keep as many repositories as we need in GitHub Account and browsing to a particular one is quite simple task.

* **Browsing Repository from GitHub Account on web browser**



On GitHub account when we login we can select Repository by clicking on name or by typing the name in the blank space and we can filter them by clicking on public, private , sources etc.

* **Browsing Repository from GitHub Desktop App:**

****GitHub Desktop app provide multiple platforms to browse repository from one place by synchronized it to local Git, Git hub account. We can open repository in Git shell, file explorer, GitHub directly just by making right click on repo. There is no need to push or pull changes manually, using command prompt. This app makes easy to perform all Git commands and to browse repository, branches, history etc.

## Create file(s) and commit the new content :

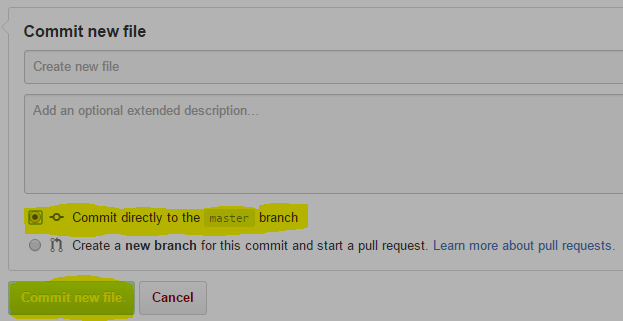
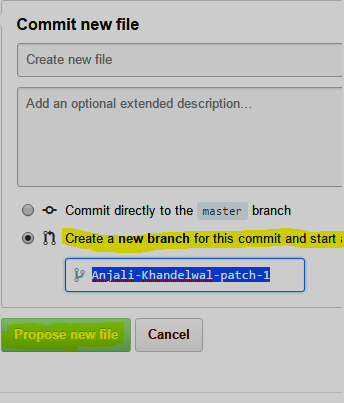
We can create files from GitHub web interface directly if we have access to the particular repo as a contributor or an owner.

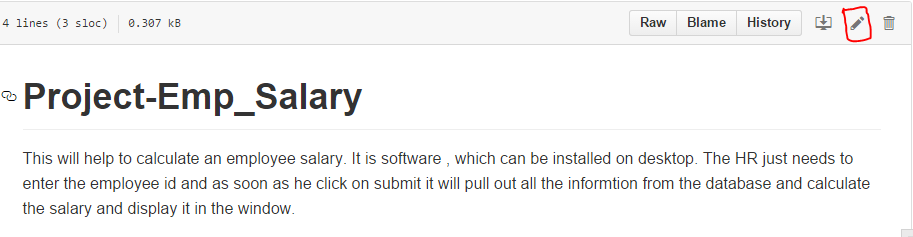


* On In your repository, browse to the folder where you want to create a file.
* Above the file list, click+.
* In the file name field, type the name and extension for the file. To create subdirectories, type the / directory separator.
* On the **Edit new file** tab, add content to the file.
* To review the new content, click **Preview**.
* At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file.
* Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is master, you should choose to create a new branch for your commit and then create a pull request.
* Click **Propose new file.**

## Editing files in your repository

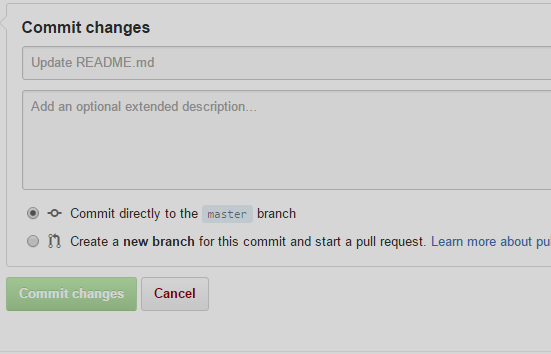
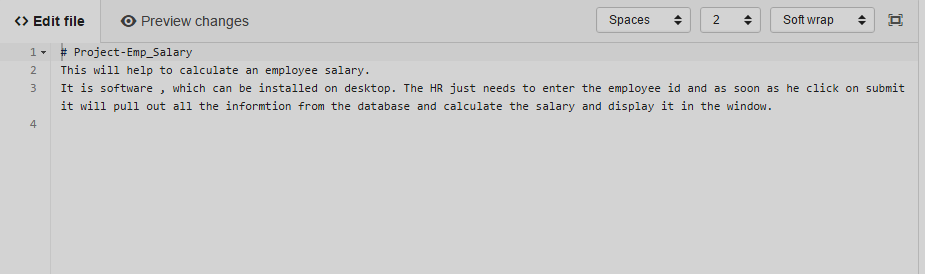
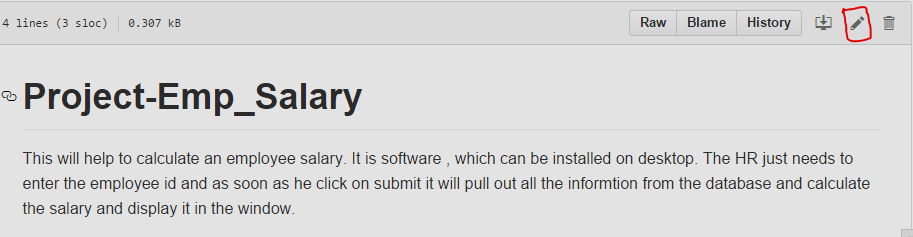
Edit an existing file's content and commit the change:

* You can edit files directly on GitHub in any of your repositories.
* In your repository, browse to the file you want to edit.



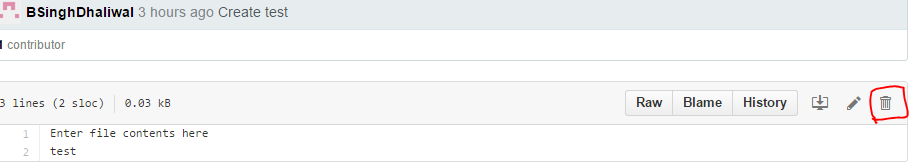
* In the upper right corner of the file view, click on edit icon to open the file editor.
* On the Edit file tab, make any changes you need to the file.
* Above the new content, click Preview changes

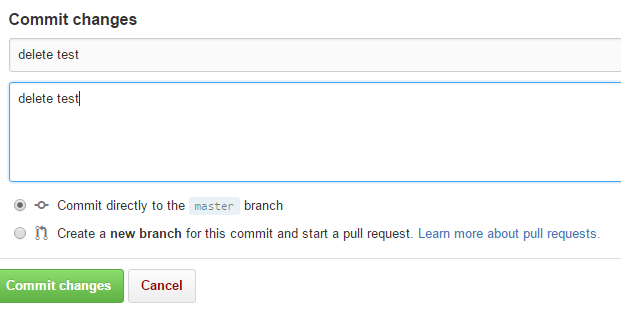
At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file.

Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is “master”, you should choose to create a new branch for your commit and then create a pull request.

## Remove a file and commit the removal

Browse the file you want to remove the click on removal icon on right side corner then type in the commit message in the bottom. Choose the default master branch or if you want to make another branch then pull request as well. And click on commit changes to complete the action.





## Change a filename and commit the path change

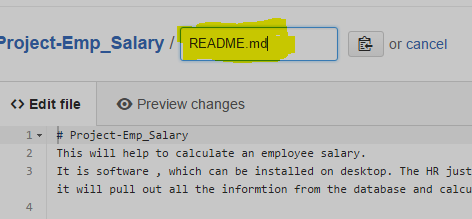
You can rename any file in your repositories directly in GitHub. Renaming a file also gives us the opportunity to move the file to a new location.

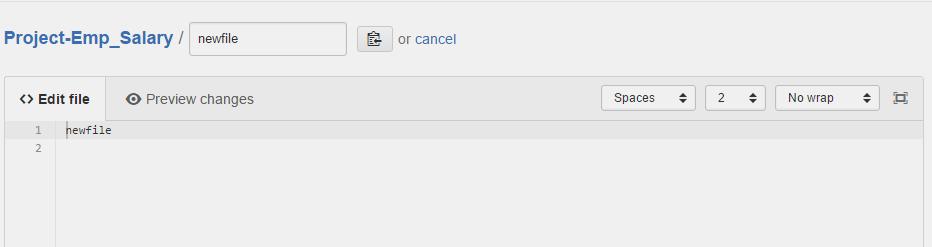
Browse the file which name we want to change. Click on edit icon and then in edit tab we can directly change the name of file. From Here e\we can also change the path of the file as well. The guidelines are listed below to change the path of any file.

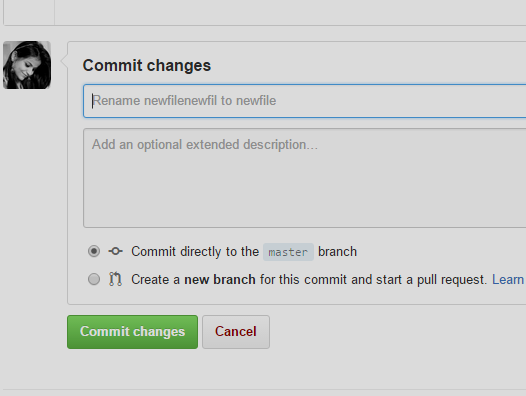
To move the file **into a subfolder**, type the name of the folder you want, followed by /. Your new folder name becomes a new item in the navigation breadcrumbs.

To move the file into a directory **above the file's current location**, place your cursor at the beginning of the filename field, then either type to jump up one full directory level, or type the Backspace key to edit the parent folder's name.

After changing the name at the bottom we need to type in a meaningful commit message.

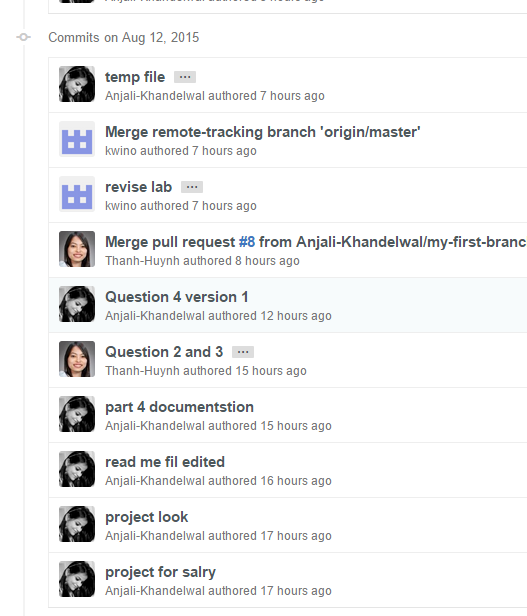






## Examine the Commits page of change history:

We can check the all history of commits and changes in files, folders and repositories with attached commit messages from the history tab in GitHub web interface.



# Acquiring Repositories Locally (Ralph)

1. Wikipedia (2015). Retrieved from https://en.wikipedia.org/wiki/Git\_(software) [↑](#footnote-ref-1)
2. Wikipedia (2015). Retrieved from https://en.wikipedia.org/wiki/GitHub. [↑](#footnote-ref-2)
3. Github Help, (2015). Retrieved from https://help.Github.com/articles/permission-levels-for-a-user-account-repository [↑](#footnote-ref-3)
4. Wikipedia (2015). Retrieved from <https://en.wikipedia.org/wiki/Concurrent_Versions_System> [↑](#footnote-ref-4)