Version Control using Git

Lee De Zhang
dezhanglee@comp.nus.edu.sg

School of Computing,
National University of Singapore

May 15, 2018

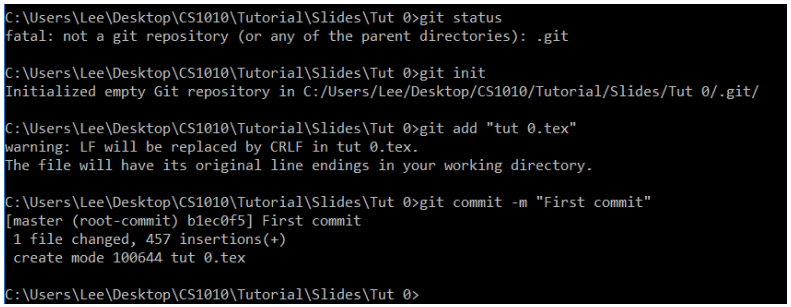
- Make sure you have Git and SourceTree (or any other Version Control GUI app) installed and properly set up on your PC. Verify by
  - Launching SourceTree without any error messages
  - Open the command terminal of your PC, type 'git' (without the ') and press enter. If git is successfully installed, then you should see something non trivial.
- Ensure that you have set up a Github account.

First, Managing Expectations

- We won't go into much detail, only the basics.
  - (Fairly reasonable) Assumption - You have never heard of Git before.
- If you are already know how to use Git and use it regularly, you probably won't see anything new here.

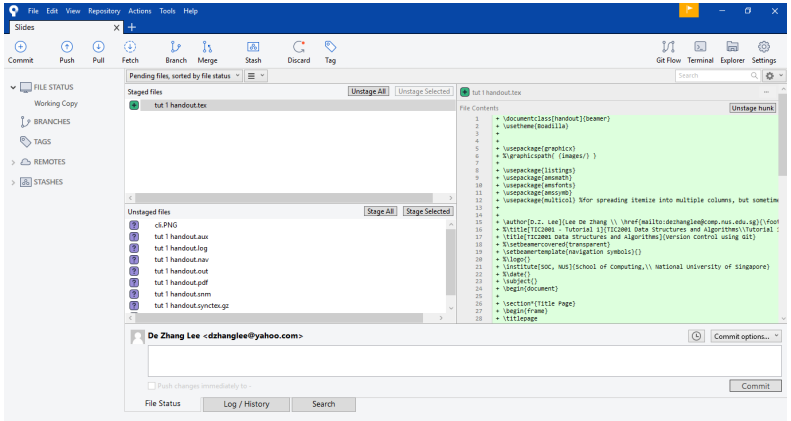
SourceTree

- We will be using a Graphical User Interface (GUI) tool for Git, instead of a Command Line Interface (CLI) approach
  - GUI - Like any most applications you use where you interact using both keyboard and mouse, eg Microsoft Word, Google Chrome, etc.
  - CLI - Text only applications - eg bash (Unix/Sunfire), cmd (Windows), terminal (Mac), etc.
  - Rationale - GUI looks less scary compared to CLI?



```
C:\Users\Lee\Desktop\CS1010\Tutorial\Slides\Tut 0>git status
fatal: not a git repository (or any of the parent directories): .git

C:\Users\Lee\Desktop\CS1010\Tutorial\Slides\Tut 0>git init
Initialized empty Git repository in C:/Users/Lee/Desktop/CS1010/Tutorial/Slides/Tut 0/.git/

C:\Users\Lee\Desktop\CS1010\Tutorial\Slides\Tut 0>git add "tut 0.tex"
warning: LF will be replaced by CRLF in tut 0.tex.
The file will have its original line endings in your working directory.

C:\Users\Lee\Desktop\CS1010\Tutorial\Slides\Tut 0>git commit -m "First commit"
[master (root-commit) b1ec0f5] First commit
 1 file changed, 457 insertions(+)
 create mode 100644 tut 0.tex

C:\Users\Lee\Desktop\CS1010\Tutorial\Slides\Tut 0>
```

(a) CLI



(b) GUI

- Without loss of generality, we will use SourceTree as our GUI tool.
  - Other apps should have the the same functionality/use the same terminology.

- SourceTree only available for Windows and Mac.
- Other alternatives available for Linux based system.
  - If you are *really really* using Linux, then you are better than most of us.
  - Should know how to find and install such alternatives on your own, or use CLI straight! =P

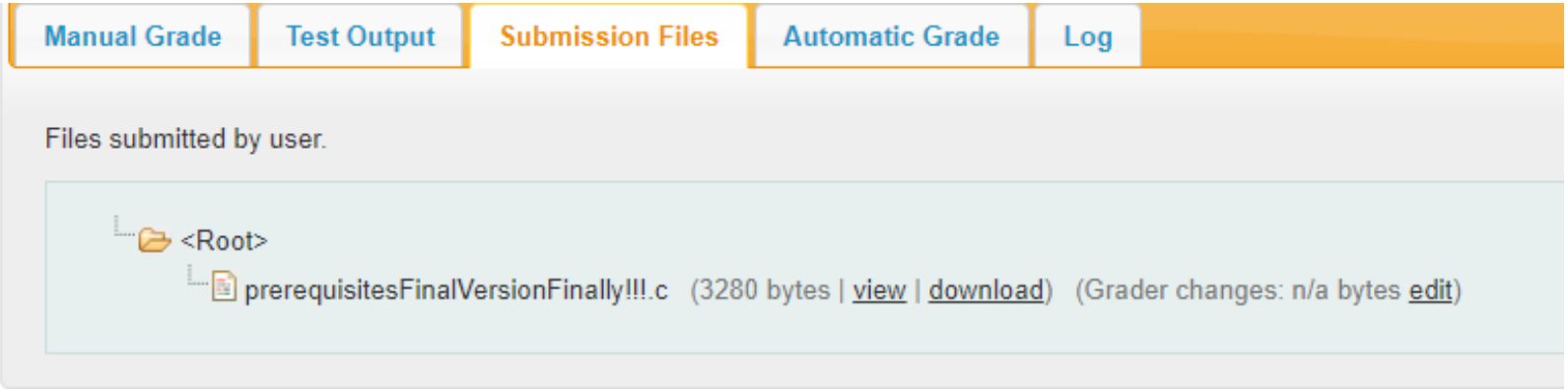
Version Control

Definition (*Version Control*)

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- Analogy 1: Saving a Game
  - Save a game at certain checkpoints, eg before fighting with a boss or trying something new.
  - If you mess up, you can go back to the checkpoint where you last saved.

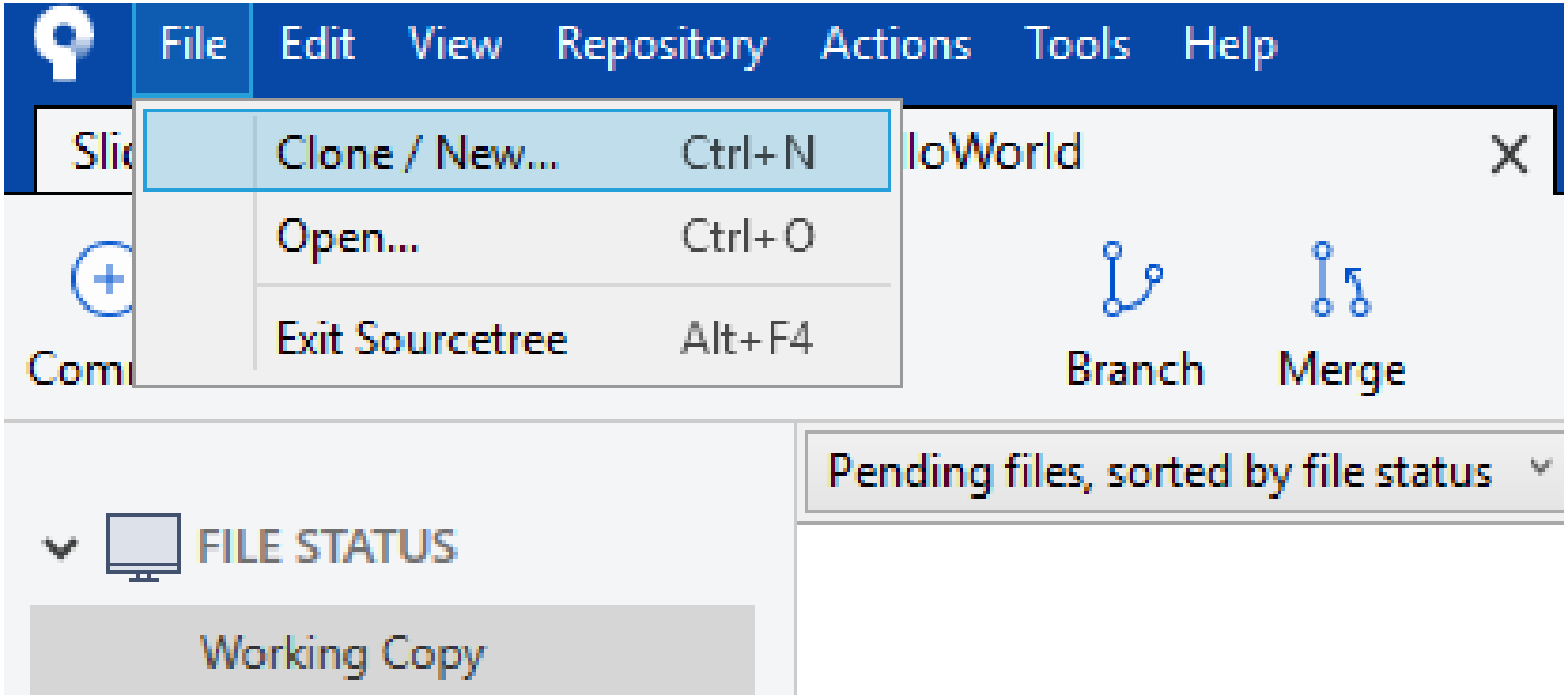
- Analogy 2: Assignment Writing
  - When you write a paper (or a programming assignment), you will save your work periodically
    - If something goes wrong (eg you accidentally deleted something), can go back to previous state
  - Problem: Only can capture one state at a time.
  - Solution: Save the same file under different names.
    - Can we do better?

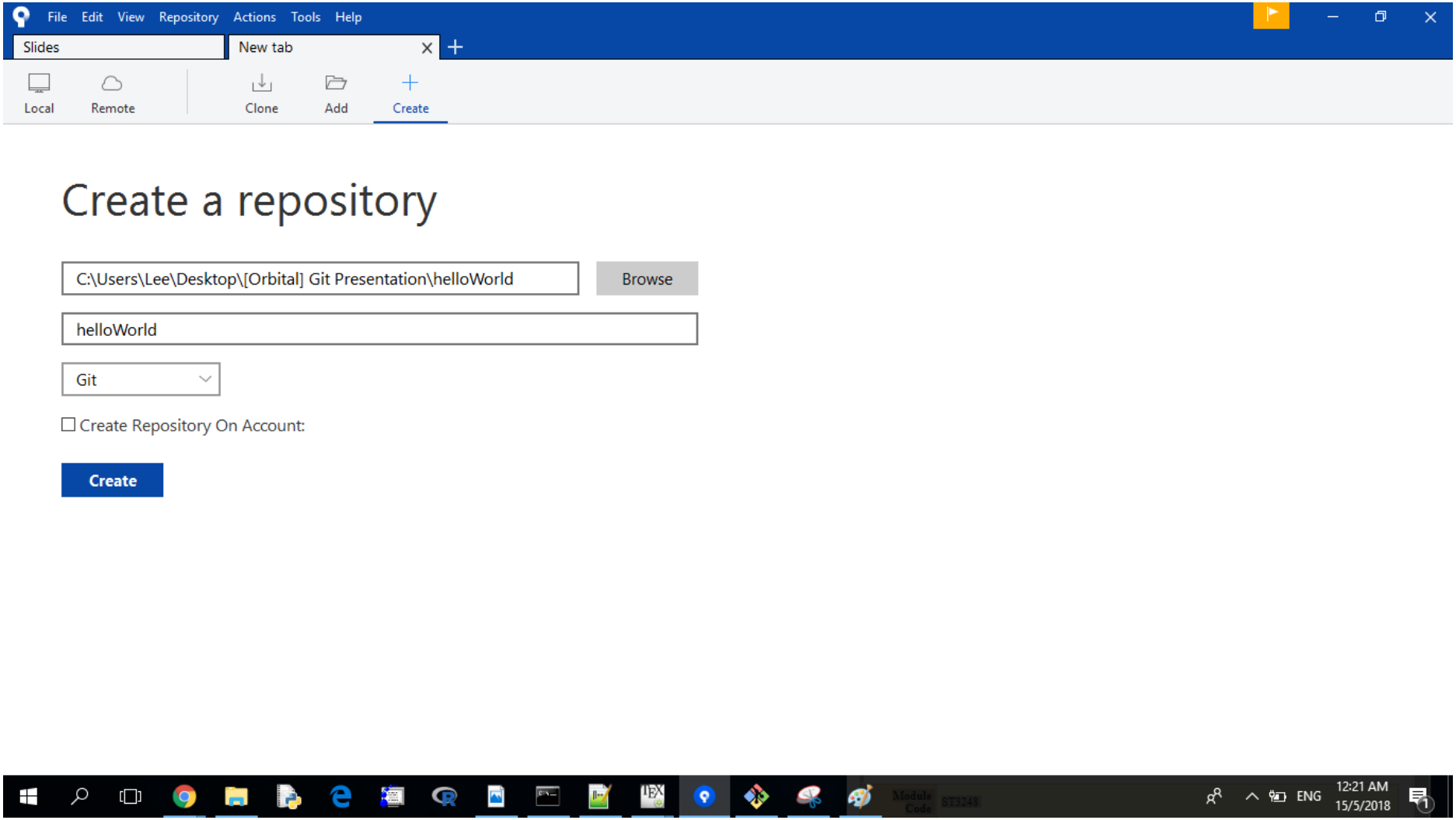


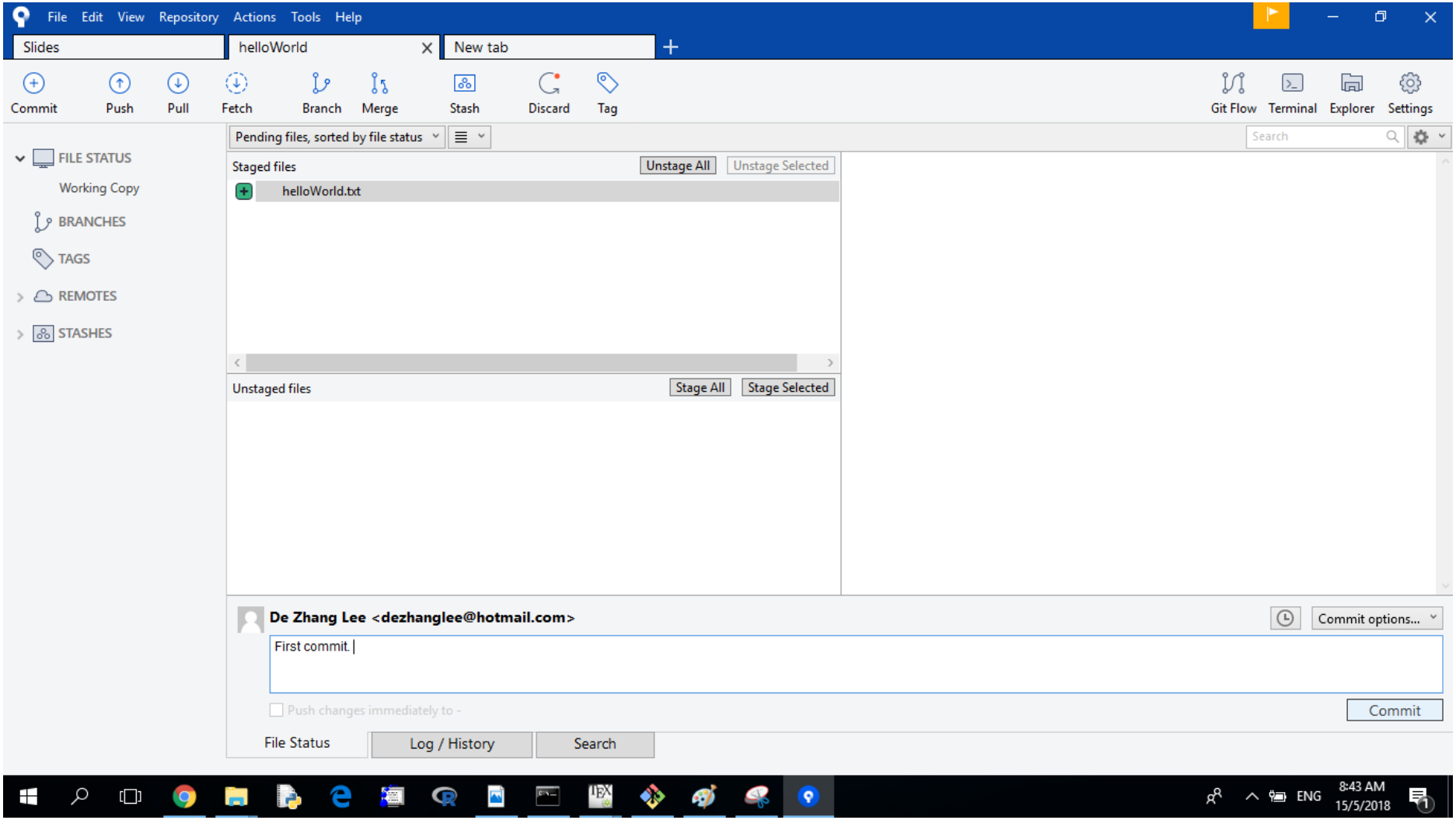
- Version control helps us to keep track of our past work, in a more systematic manner.
- Git is a popular version control utility.
  - Other version control utilities exist too. Eg Mercurial.
- SourceTree uses Git to carry out version control.

Adding Files to Git

- Let's start with a simple text file, 'helloWorld.txt' in an **empty folder**.
- We will now use SourceTree to track any changes made to 'helloWorld.txt'.
- On SourceTree, refer to next few slides for some screenshots.
  - File → Clone/New
  - Select 'Create'
  - Click 'browse', and select the folder you want to track.
  - Fill in the name of your repository (the folder you wish to track) in the next field.
  - For now, we leave the 'Create Repository on Account' option unchecked.
  - Click 'create'
  - You will be presented with a new screen, with 'helloWorld.txt' under unstaged files.
  - Click 'Stage All'
  - Enter a commit message (eg First Commit), and click 'commit'


File
Edit
View
Repository
Actions
Tools
Help
Clone / New... Ctrl+N
Open... Ctrl+O
Exit Sourcetree Alt+F4
Branch
Merge
Pending files, sorted by file status
FILE STATUS
Working Copy


File Edit View Repository Actions Tools Help
Slides
New tab
Local
Remote
Clone
Add
Create
Create a repository
C:\Users\Lee\Desktop\[Orbital] Git Presentation\helloWorld
Browse
helloWorld
Git
Create Repository On Account:
Create
ENG
12:21 AM
15/5/2018


File Edit View Repository Actions Tools Help
Slides
helloWorld
New tab
Commit
Push
Pull
Fetch
Branch
Merge
Stash
Discard
Tag
Git Flow
Terminal
Explorer
Settings
FILE STATUS
Working Copy
BRANCHES
TAGS
REMOTES
STASHES
Pending files, sorted by file status
Search
Staged files
Unstage All
Unstage Selected
helloWorld.txt
Unstaged files
Stage All
Stage Selected
De Zhang Lee <dezhanglee@hotmail.com>
Commit options...
First commit.
Push changes immediately to -
Commit
File Status
Log / History
Search
ENG
8:43 AM
15/5/2018

What Have We Just Done?

- First, we created a file, helloWorld.txt (can either be empty or not) inside a folder (let's call this folder *hw*).
- By doing the operations in the previous two slides, we created a Git repository using the folder *hw*.
  - In other words, git is actively tracking files inside this folder for changes.
  - We can specify files which are not to be tracked. More about this later.
- By staging the file 'helloWorld.txt', we are telling git that we intend to include this file in our version control.
- By committing, we have created a snapshot of our current (staged) files at this stage.
  - They can be retrieved at a later stage (even after 10,000 commits!)
  - No need to make several other files like helloWorld, helloWorld1, helloWorldFinal, etc.
  - More about committing later.

Equivalent CLI Commands - For Own Reading

In the order they should be called.

- cd dir
  - Replace dir with the path leading to the folder you wish to track
  - Example. cd `C:\Users\[Orbital]GitPresentation\helloWorld`
- git status
  - Make sure that there's no git repo already initialized here
- git init
  - Initialize the git repository here
- git add *
  - Add files into your git repository.
  - * is the wildcard operator, which selects all the files in the folder to be tracked
  - If only wish to add specific files, specify the filename.
    - Example. git add helloWorld.txt

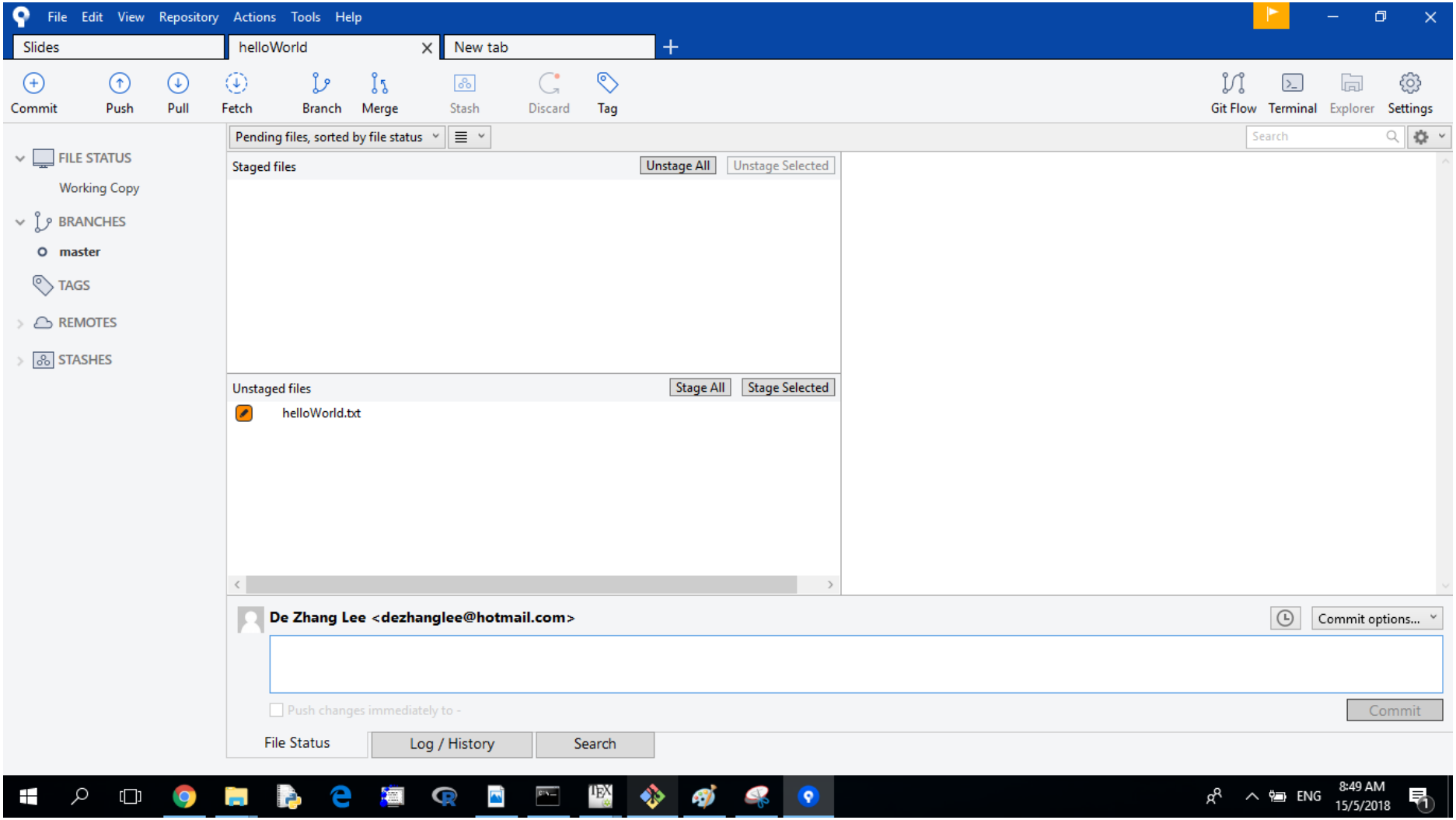
Committing (Saving) Changes

- Recall that in our earlier step, we have set up a git repository.
  - In other words, git is actively tracking our files for changes.
- Let's try changing our file, 'helloWorld.txt', and see what happens.

```
Lee@LAPTOP-CVDMNJN4 MINGW64 ~/Desktop/[Orbital] Git Presentation
$ cd helloWorld/

Lee@LAPTOP-CVDMNJN4 MINGW64 ~/Desktop/[Orbital] Git Presentation/helloWorld
$ ls
helloWorld.txt

Lee@LAPTOP-CVDMNJN4 MINGW64 ~/Desktop/[Orbital] Git Presentation/helloWorld
$ vim helloWorld.txt

Lee@LAPTOP-CVDMNJN4 MINGW64 ~/Desktop/[Orbital] Git Presentation/helloWorld
$ cat helloWorld.txt
Hello World!  :)
```

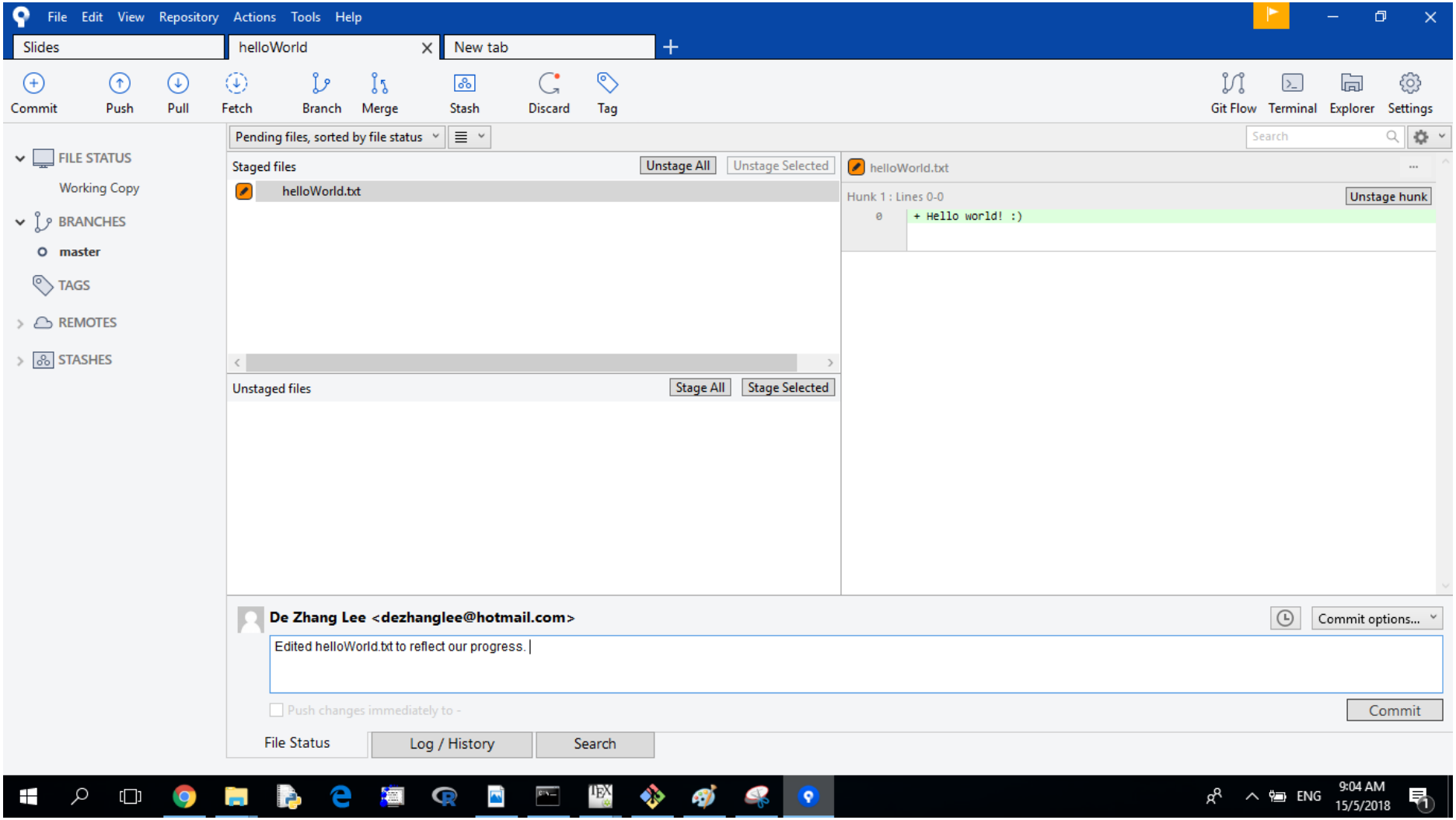
Editing the contents of 'helloWorld.txt'

SourceTree, After Editing The File



‘helloWorld.txt‘ is now ‘unstaged‘ again

- Recall that git is actively tracking the files in our repo.
- Therefore, after modifying its contents, git realizes that ‘helloWorld.txt‘ has been changed.
  - SourceTree reflects this by moving the file to the ‘unstaged‘ section
- To see the changes made, simply click the file and SourceTree will show you the difference between the current version, and the version in the last commit.
- Again, we can save a snapshot of our current state by using the commit option. Similar procedure as before.
  - Stage the files you wish to keep track off in the current commit.
  - Enter a *meaningful* commit message
    - Meaningful messages can save you when you yourself can't understand your own code, can trace and see what you have done up to this point.
    - You *may* hate yourself if you put commit messages like ‘ibisdbfsd‘, and are forced to revisit your code some time later.
  - Click ‘commit‘


File Edit View Repository Actions Tools Help
Slides
helloWorld
New tab
Commit
Push
Pull
Fetch
Branch
Merge
Stash
Discard
Tag
Git Flow
Terminal
Explorer
Settings
FILE STATUS
Working Copy
BRANCHES
master
TAGS
REMOTES
STASHES
Pending files, sorted by file status
Search
Staged files
Unstage All
Unstage Selected
helloWorld.txt
Unstaged files
Stage All
Stage Selected
helloWorld.txt
Hunk 1 : Lines 0-0
Unstage hunk
+ Hello world! :)
De Zhang Lee <dezhanglee@hotmail.com>
Commit options...
Edited helloWorld.txt to reflect our progress.
Push changes immediately to -
Commit
File Status
Log / History
Search
ENG
9:04 AM
15/5/2018

Equivalent CLI Commands - For Own Reading

In the order they are to be called.

- git status
  - See which files (if any) are changed. Also to ensure that there's a git repo here.
- git add *
  - Similar to before, wildcard operator * adds all the files.
  - If necessary, replace * with the individual filenames
- git commit -m "<Commit message>"
  - Commits the files specified in git add
  - Replace <Commit message> with your commit message

Putting Our Work Online

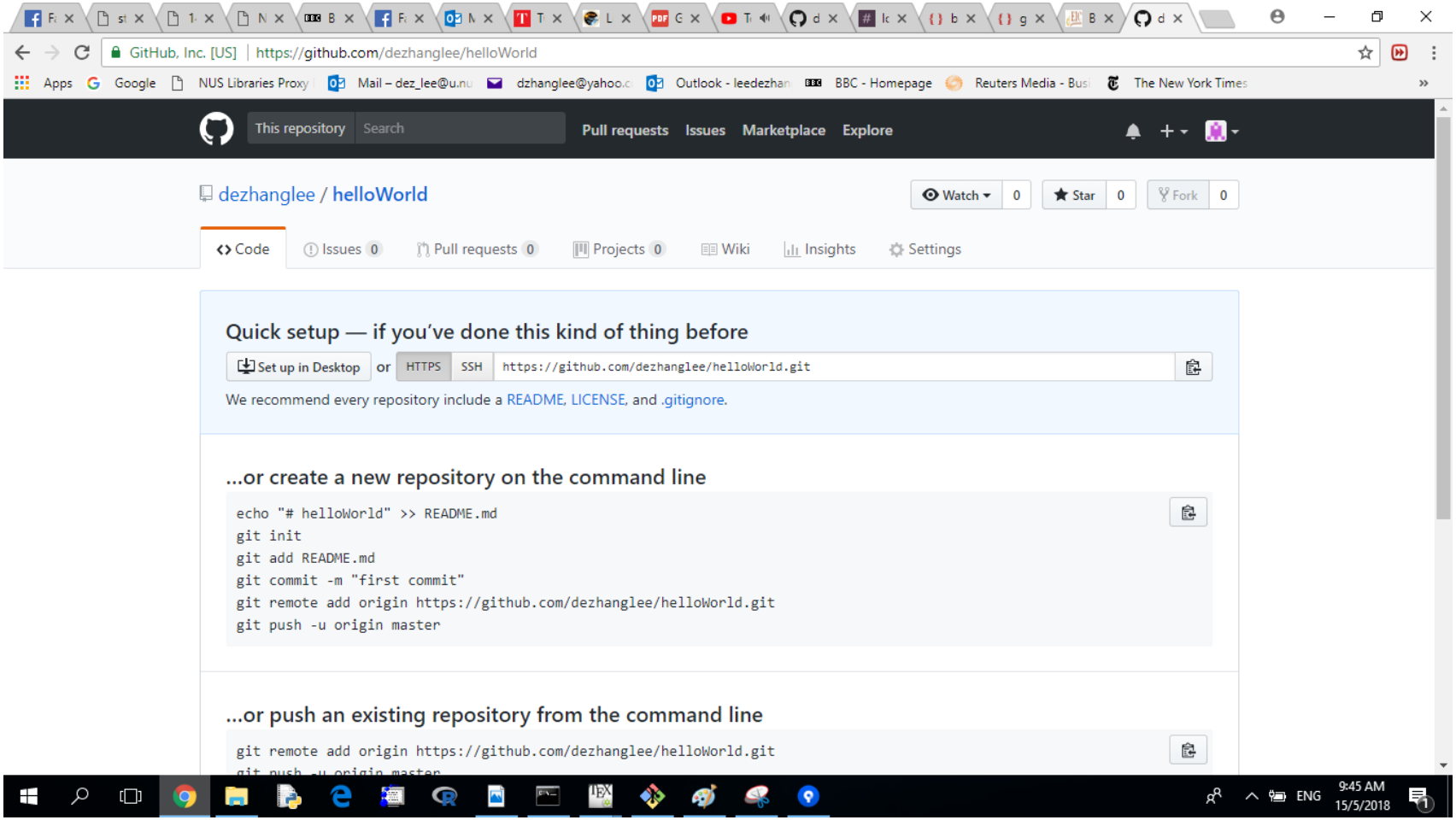
- Besides saving our commits locally, we can store them online.
- Some such online services include
  - Github
  - Bitbucket
  - Cloud Source (google)
- Without loss of generality, we shall use Github.
- First, ensure that you already have set up a Github account.

Creating a Repository on Github

- Click the '+' icon on the top right corner of your screen
- Click 'New Repository', and follow the on screen instructions.
  - For now, uncheck 'Initialize Repository with a README'
  - Set your repository as public. Private repos come with a paid package
    - There's a student package which allows you to create free private repositories on Github. If you are interested, google yourself.

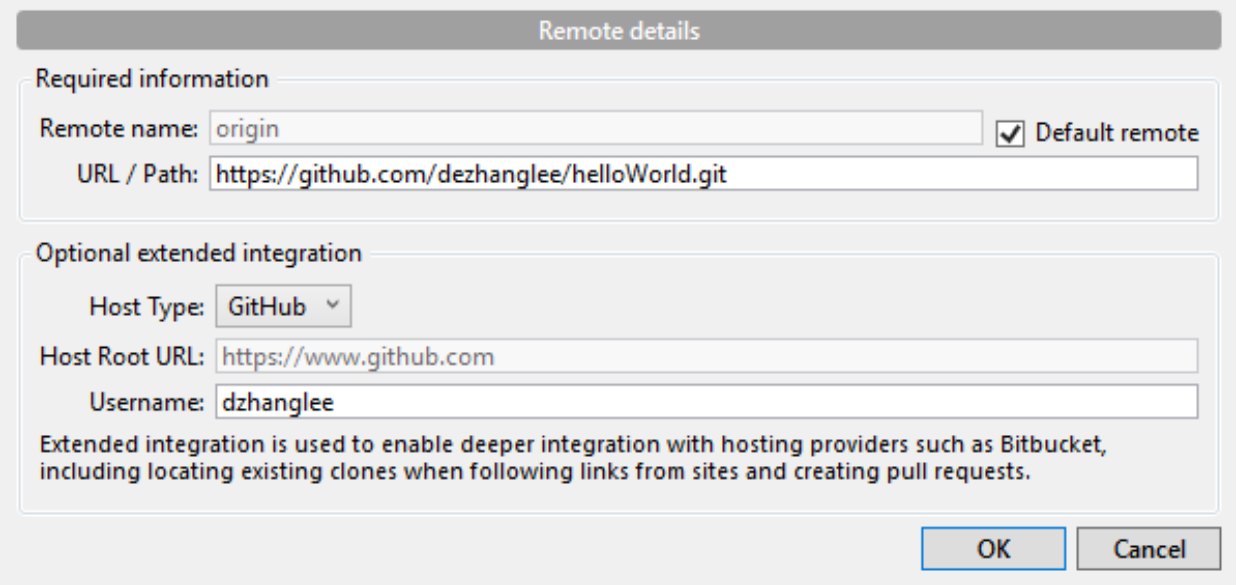
Publishing Your Local Repository on Github

- After creating your new repo on Github, you should be presented with the following screen

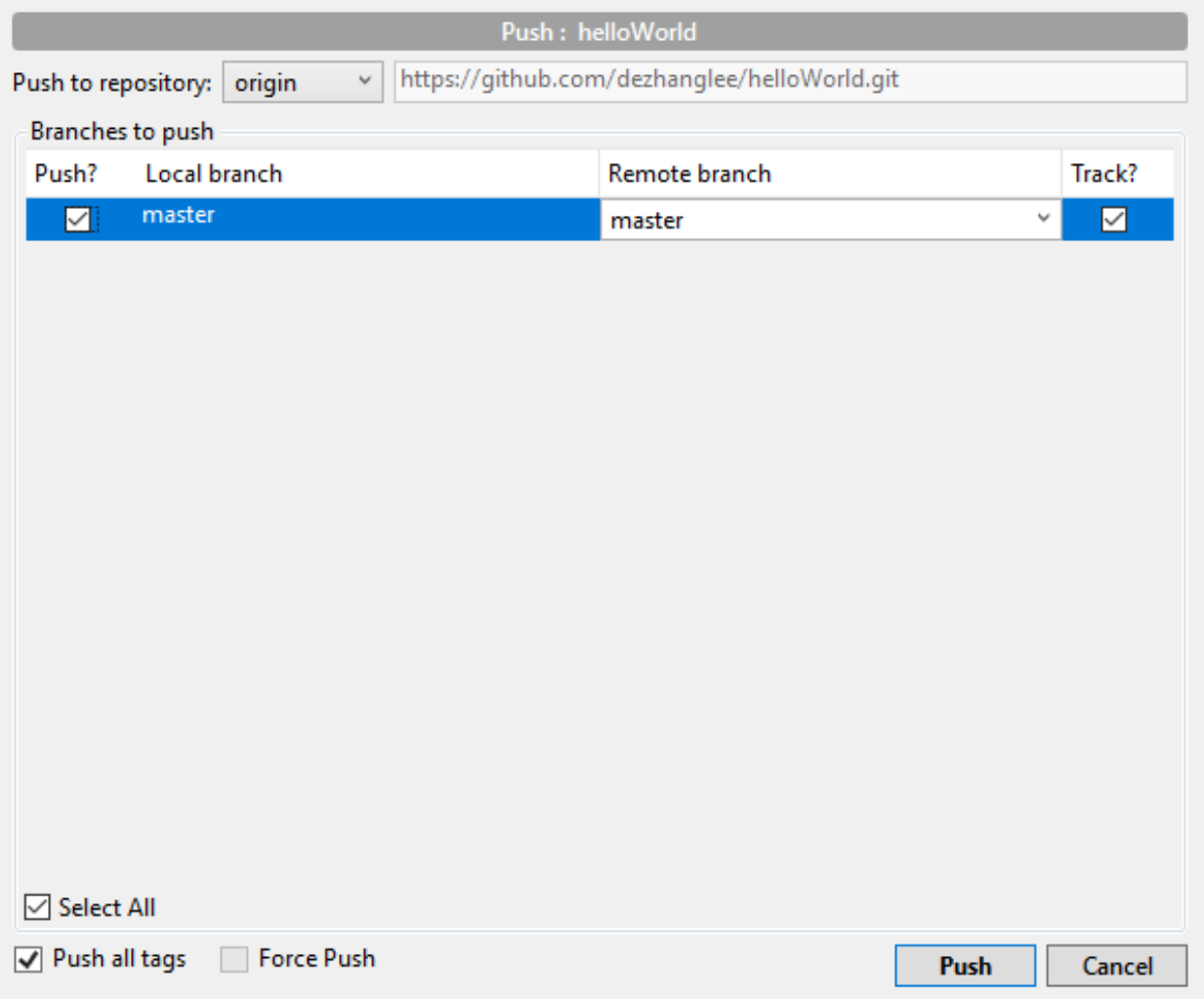


Pushing Local Files to Github

1. Copy the link in the section ‘Quick setup - if you’ve done this kind of thing before‘
2. On SourceTree
   - Select Repository → Repository Settings
   - Under ‘Remotes‘, click ‘Add‘
   - Paste the link copied from 1 to ‘URL/Path‘
   - Check ‘Default Remote‘
   - If the fields under ‘Optional Extended Integration‘ aren’t filled, please fill up accordingly.



- On SourceTree, click ‘push‘
  - ‘push‘ is a fancy name for upload
- Check the ’Push?’ checkbox for the first row
- Click ‘push‘
- If necessary, login using your Github ID and Password.



- On your Github repo, you should be able to see the repository you committed earlier.
  - You can view your past commits, and the associated files by clicking the following.

Collaboration using GitHub

- Since our repo is now online, we can collaborate with anyone easily.
- You will be working in pairs. Therefore, this is important.

Forking

- Basically, this means creating a copy of someone else's repo on Github to your own Github profile.
- Since this copy now belongs to you, you may now edit this and subsequently, propose changes to the original owner.
- For example,
  - Suppose John has an interesting program on Github, and you feel that you can add something.
  - But, since the repo belongs to John, you can't edit it directly.
  - However, you can make a copy of that repo by forking it to your own Github account.
  - Then, you can work on your own copy, commit the necessary changes
  - Afterwards, you may notify John of your proposed changes by submitting a pull request
    - More on pull requests later.
  - John can then either approve the changes and merge them into his repo, or reject them.

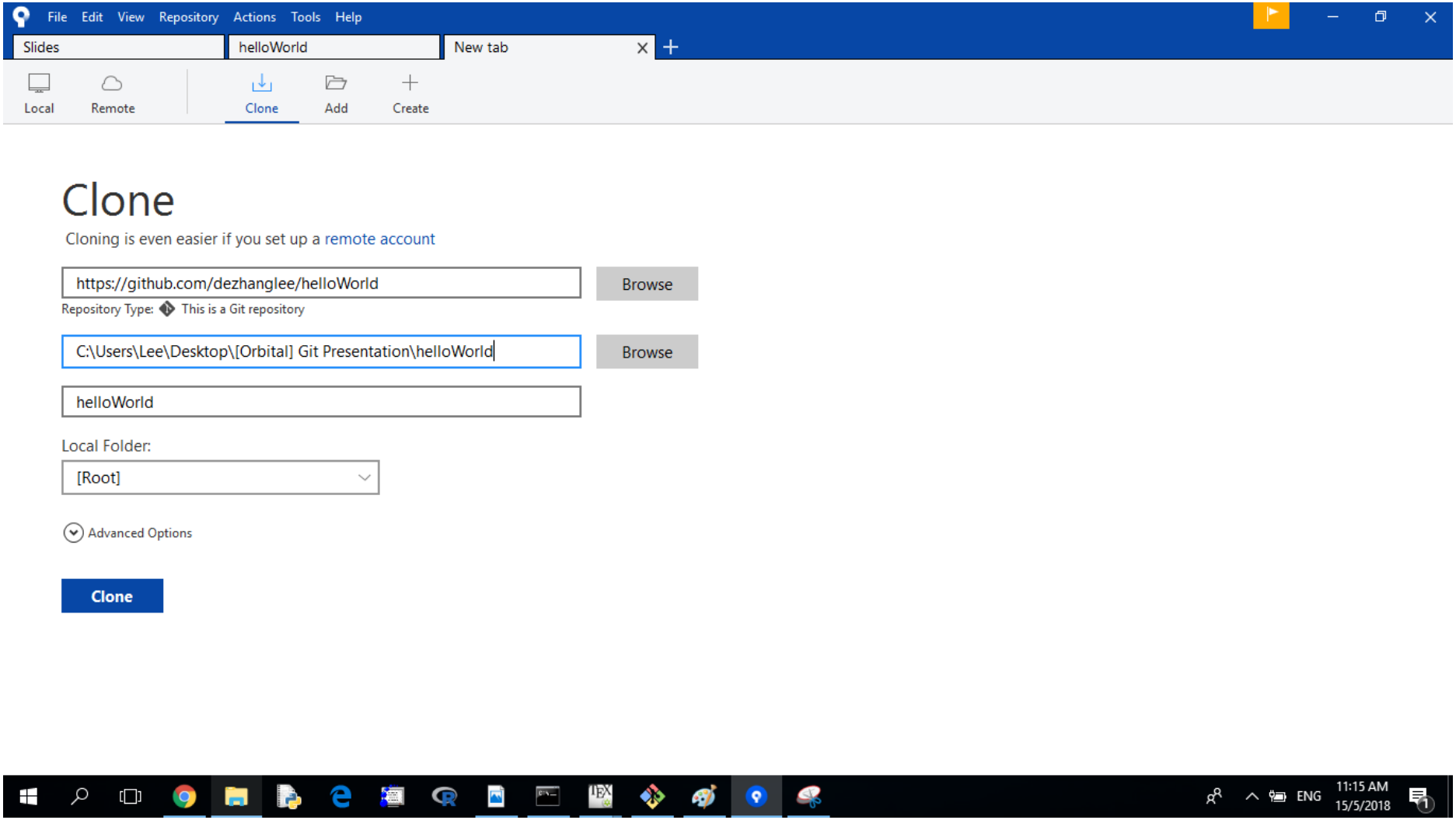
Forking on Github

- Go to any repo that doesn't belong to you on Github
- On the top right corner, click the 'fork' button.
- A copy of this repo should be on your Github.

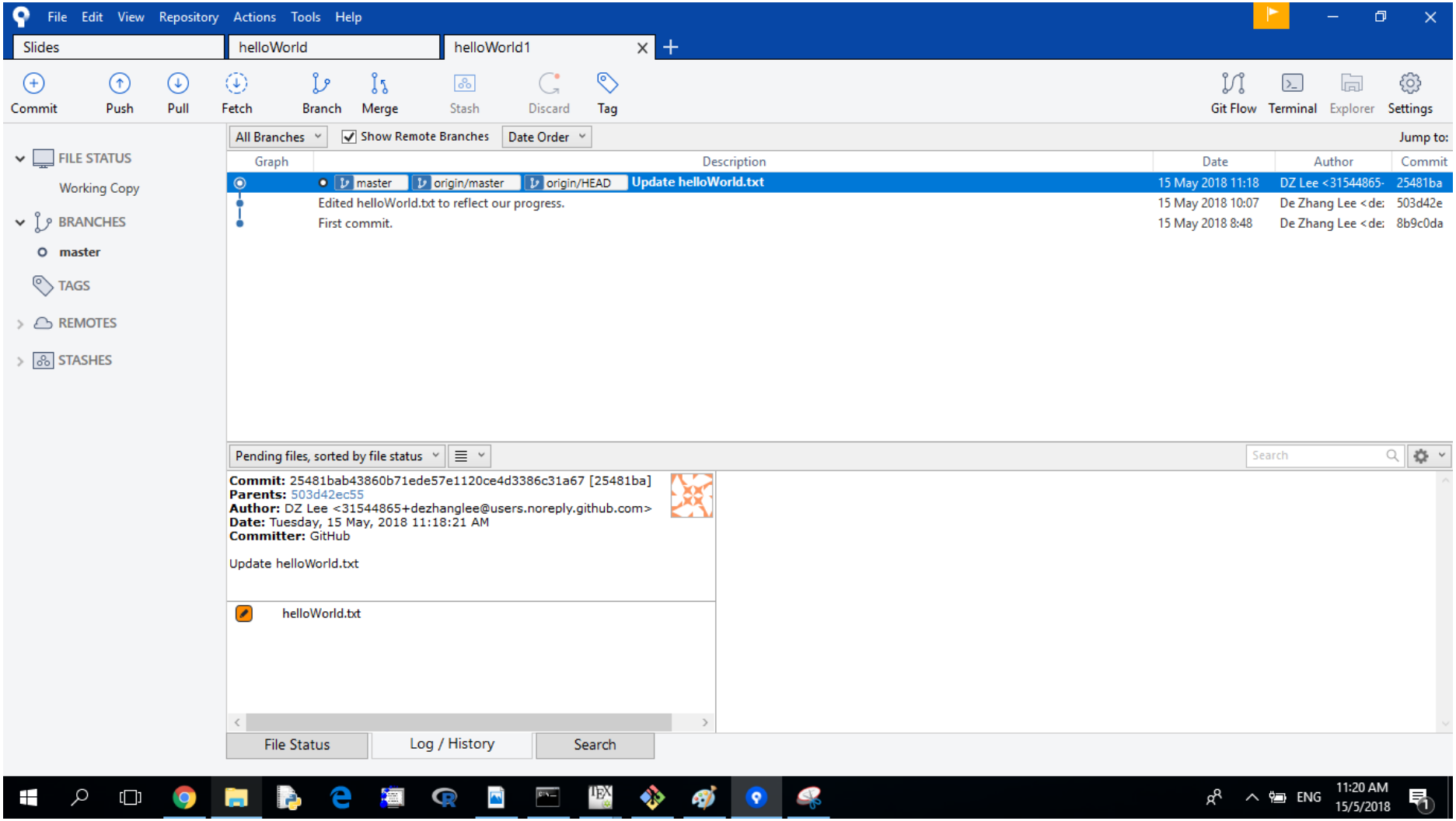
Cloning The Forked Copy To Your Computer

- Now that we have a forked copy of the repo you wish to modify, we should download it so that we can edit it on our computers
- This can be done through the clone function of Github.

- For this exercise, you may fork the following repo `https://github.com/dezhanglee/helloWorld`
- In the forked repo
  - Click the green ‘Clone or download‘ button.
  - Copy the link.
  - Alternatively, you may download a zipped copy. But for our case, we aim to complete this using SourceTree.
- On SourceTree
  - Click File → Clone/New
  - Paste the copied URL into the ‘Source Path/URL‘ text box.
  - Specify the folder you wish to download the repo to.
  - Click ‘clone‘


File Edit View Repository Actions Tools Help
Slides
helloWorld
New tab
Local
Remote
Clone
Add
Create
Clone
Cloning is even easier if you set up a remote account
https://github.com/dezhanglee/helloWorld
Browse
Repository Type: This is a Git repository
C:\Users\Lee\Desktop\[Orbital] Git Presentation\helloWorld
Browse
helloWorld
Local Folder:
[Root]
Advanced Options
Clone
ENG
11:15 AM
15/5/2018

Successfully Cloned the Repo



Proposing Changes to the Original Author

- Now that you have cloned the repo to your local PC, we may try to modify/add/delete the files.
- After making the necessary changes, we may suggest that the author incorporate our changes into the original repository.
- This can be done by making a Pull Request (PR) against the original repository
- Possible analogy
  - Suppose you wrote a paper, and published it online.
  - Your peers/colleagues may view the paper, and see if anything needs to be changed.
  - If they intend to propose changes, they may first download the paper, make the necessary changes and resubmit it back to the author
  - The author may then decide if he/she intends to keep or discard the changes.

Making a PR

- First, make the necessary modifications.
  - In our case, just add/delete some text to ‘helloWorld.txt‘.

```
C:\Users\Lee\Desktop\[Orbital] Git Presentation\helloWorld1>vim helloWorld.txt

C:\Users\Lee\Desktop\[Orbital] Git Presentation\helloWorld1>cat helloWorld.txt
Hello world! :)

The quick brown fox jumped over the lazy dog.

C:\Users\Lee\Desktop\[Orbital] Git Presentation\helloWorld1>
```

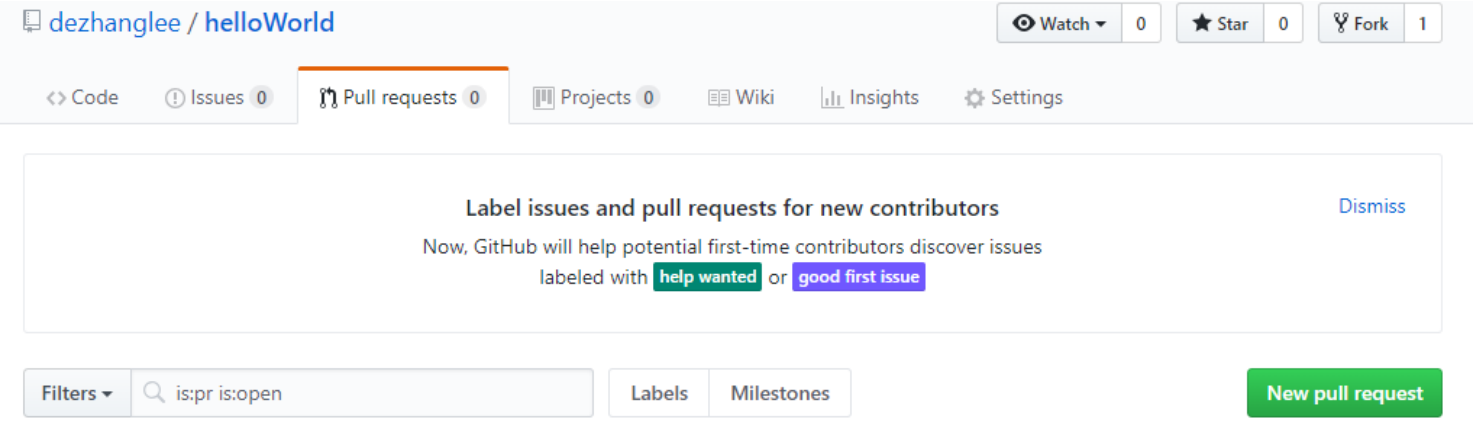
Edited ‘helloWorld.txt‘ with the above contents.

In Your Own Repository

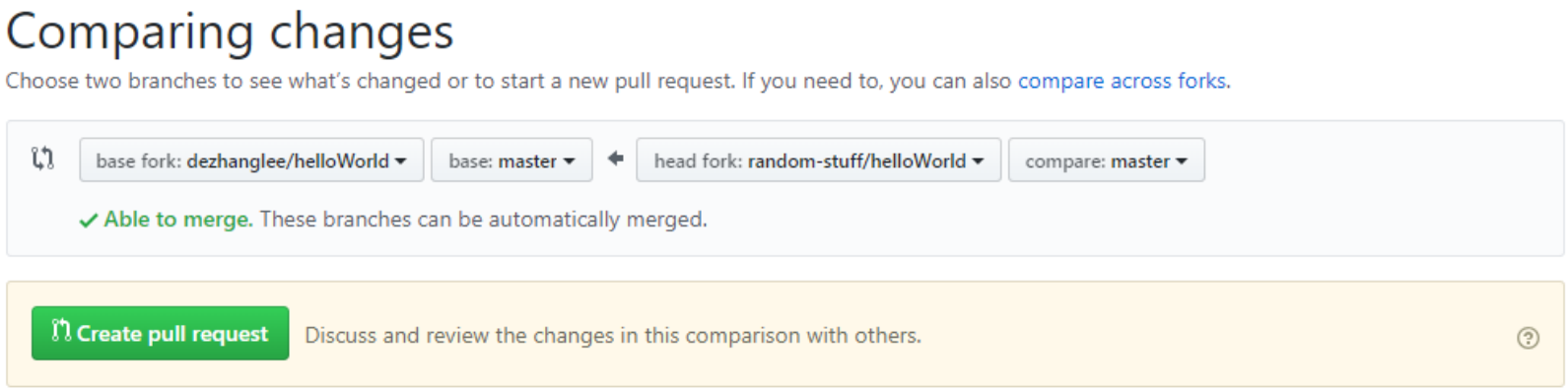
- Stage, commit and push the file(s) to your online repo.

In The Repository You Want To Propose Changes To

- Navigate to the repository you wish to propose changes (make a PR).
- Click on ‘Pull Requests‘, then the green ‘New Pull Request‘ button



- Click ‘Compare Across Forks‘ (Since we are using two different forks)
- Under base fork, select the repo you wish to propose changes to.
- Under head fork, select the repo you last modified.
- In our case
  - We wish to propose changes to *dezhanglee/helloWorld*, hence that’s our base fork.
  - These changes are from *random – stuff/helloWorld*, hence that’s our head fork.
- Click ’Create Pull Request’ to submit the PR for the author’s review. Include an appropriate description.



Merging a PR

- Now, we take on the role of the owner of the repo, who has received a PR.
- We first review the changes proposed by the PR, and decide accordingly.
  - This is why a good PR message (and good coding style) is important, especially if the PR is long and the codebase is complex
    - A real life codebase would be at least $\sim$ 10,000 lines of code. Windows 10 is around 50,000,000
- We may view all the 'open' (unresolved) PRs under the Pull Request tab.

The PR Page With Our PR From Before

GitHub, Inc. [US] | https://github.com/dezhanglee/helloWorld/pulls

This repository Search Pull requests Issues Marketplace Explore

dezhanglee / helloWorld

Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 1 Projects 0 Wiki Insights Settings

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

Dismiss

Filters is:pr is:open Labels Milestones New pull request

1 Open 0 Closed Author Labels Projects Milestones Reviews Assignee Sort

some changes to helloWorld.txt

#1 opened 8 minutes ago by dezhanglee

ProTip! Click a checkbox on the left to edit multiple issues at once.

© 2018 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub API Training Shop Blog About

Click On The PR You Wish To Review

GitHub, Inc. [US] | https://github.com/dezhanglee/helloWorld/pull/1

Code | Issues 0 | Pull requests 1 | Projects 0 | Wiki | Insights | Settings

some changes to helloWorld.txt #1

Edit

Open dezhanglee wants to merge 2 commits into `dezhanglee:master` from `random-stuff:master`

Conversation 0 | Commits 2 | Checks 0 | Files changed 3 | +5 −0

dezhanglee commented 10 minutes ago — Owner

Should fix bug #123

dezhanglee and others added some commits 42 minutes ago

- Update helloWorld.txt — Verified — 25481ba
- Added some text — 69f5f74

Add more commits by pushing to the **master** branch on **random-stuff/helloWorld**.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

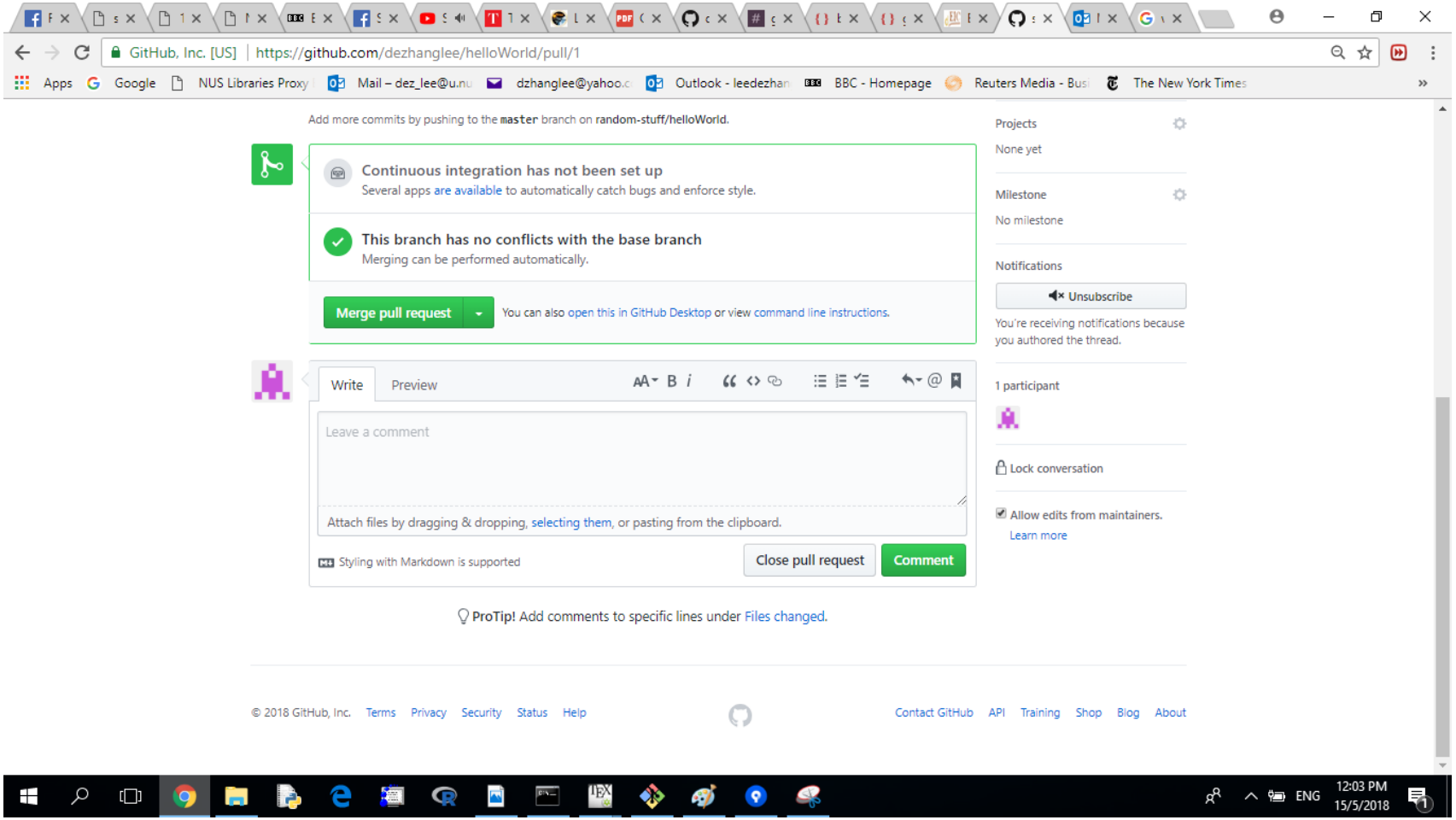
This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request — You can also open this in GitHub Desktop or view command line instructions.

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Notifications: Unsubscribe
You're receiving notifications because you authored the thread.

ENG 12:00 PM 15/5/2018

- Click the ‘Files Changed‘ tab to see the changes made.
- If you are satisfied with the changes and wish to merge it into your codebase, then click the green ’Merge Pull Request’ button, make the necessary commits.
- Else, you may close the pull request (and optionally, add a comment)



Merge Conflicts

- Note that sometimes, merge conflicts may occur.
- This means that
  - Between the time you forked the repo, and submitted the PR, the part where you edited has already been edited by someone else
  - Hence, there's a conflict between the edited codes.
  - Since we are now currently dealing with teams of 2, this shouldn't be a problem as you both can coordinate among yourselves to resolve this.

Putting Everything Together

- Let's try to build a simple Calendar app using Python, and keep track of our activity using Git.
- Very simple functionality
  - When app is launched, prompt for a username and password.
  - All usernames and passwords are stored in plaintext, in a .csv file (don't do this in your project!)
  - If the user inputs a valid username and password, then we output that user's activity.
- We shall initialize this project from scratch, and explore using git ignore.

Concluding Remarks

- Covered the basics of Version Control Using git.
- Did not cover many other useful features.
  - Branching - Read up on this
- Ask your questions on the slack channel, *lo – gitbasics* / email if its personal