OBJECTIVES

- Introduction to GitHub
- Hands-on
  - Account Creation
  - Repository
  - Branching
  - Commit changes
  - Pull Requests
  - Collaboration
- UC local GitHub
- Helpful hints and Resources
WHAT IS GitHub?

- Free web based Public Repository for collaboration
- Web based Version Control System and Source Code Management based on ‘git’
- Mostly used for code in software development environments
- Can also be used for resource sharing and management in any team/environment

Popular collaboration platforms

- Microsoft SharePoint
- Confluence
- Asana
WHAT IS A VERSION CONTROL SYSTEM?

- Keeps records of changes
- Different users make changes to same shared documents at the same time
- Who made the changes?
- Revert changes to previous states
- Changes can be undone if needed

**GitHub** works best to track changes in text-based files, but can be used to version other types of files as well

- If you make a mistake, you can usually undo it. Even if you accidentally delete a branch, **GitHub** will let you undelete it
<table>
<thead>
<tr>
<th><strong>git</strong></th>
<th><strong>GitHub</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>is an open source distributed version control system</td>
<td>is a website that allows you to create online repositories or upload your <em>git</em> repositories online.</td>
</tr>
<tr>
<td>Repositories are created and can be shared with a local team who have various privileges to files (access control)</td>
<td>Gives a visual interface</td>
</tr>
<tr>
<td>Strictly command-line</td>
<td>Provides a backup for your local <em>git</em></td>
</tr>
<tr>
<td></td>
<td>You can share repos to outsiders and access other teams repos; basis for open source projects.</td>
</tr>
<tr>
<td></td>
<td><em>GitHub</em> version control is based on <em>git</em></td>
</tr>
<tr>
<td></td>
<td>Adds more functionalities; several collaboration features such as task management, bug tracking feature and wikis</td>
</tr>
<tr>
<td></td>
<td><em>GitHub</em> Desktop (Optional) – Can be locally installed on your computer to synchronize local code with github.com</td>
</tr>
</tbody>
</table>

*git* does not require the use of *GitHub* and vise versa; However it very common to use *GitHub* if you use *git*
Creating a GitHub Account

1. Choose the free account
2. You will receive a verification email at the address provided
3. Click the emailed link to complete the verification process
EXPLORING THE INTERFACE

▶ Your User Dashboard
1. The Login Landing Page
2. Overview of the exciting things happening on GitHub.
3. Create new projects
4. Customize the projects you are watching
5. Quick links to your own projects.

▶ Your User Profile
1. Top right corner
2. Contains your GitHub activity
3. Public (Open to employers)
4. Add some fun bio and awesome photo

▶ Your GitHub Repositories (‘Repo’)
1. Containers to holds everything related to a specific project.
2. Used to organize a single project
3. Can contain folders and files– anything needed for the project
4. A README file recommended
CREATE A REPOSITORY

1. Learn Git and GitHub without any code! Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.
2. Create a new repository. A repository contains all the files for your project, including the version history.
3. Initialize this repository with a README. This will be your initial commit.
WHAT IS A BRANCH

- The way to work on different versions of a repository at one time
- Default Branch – master
- Other branches are used to experiment and make edits before committing/merging to master
- A branch off the master is a copy or snapshot at that point in time
- Latter changes can be pulled to update the new branch while you were working on it
CREATE A NEW BRANCH

1. Add topics
2. Switch branches/tags
3. Create branch: readme-edits from 'master'
MAKE CHANGES, ADD COMMIT MESSAGES AND COMMIT CHANGES

- Changes to files can be made and saved by *commits*
- A *commit message* can be associated with each commit
- Commit messages describe or explain reason for the change
- Commit messages capture the history of changes for later reference by other contributors
OBJECTIVES

INTRODUCTION

GitHub BASICS

BRANCHING

PULL REQUEST

LOCAL GitHub @ UC

CONCLUSION

1. Introducing GitHub
2. Creating a new commit
3. Reviewing changes
4. Committing changes
5. Finalizing the commit
WHY DO A PULL REQUEST?

- Propose your changes
- Show differences (difs) of the content from both branches being compared
- Changes, additions, and subtractions are color coded in green and red
- Request peer reviews (feedback from specific team members)
- Have peers pull in your contribution and merge them into their own branches
CREATE A PULL REQUEST

1. Welcome to Pull Requests!
2. Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.
3. Comparing changes
4. Showing 2 changed files with 4 additions and 1 deletion.
5. Create pull request

Discuss and review the changes in a comparison with others.

Able to merge. These branches can be automatically merged.
CREATE A PULL REQUEST

1. Open a pull request
2. Write a description
3. Add and delete new file
4. Create pull request
5. Merge pull request
6. First Pull Request
7. Add, delete and new file
8. Able to merge
MERGE YOUR PULL REQUEST
Bring your changes together

1. 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.
   - Set up continuous integration to automatically test your code
     - Catch bugs, enforce style, and increase confidence in your code before you merge.
   - Explore GitHub Marketplace

2. Merge pull request #1 from CEASLIBRARY/readme-edits
   - First Pull Request
   - Confirm merge
   - Cancel

3. CEASLIBRARY self-assigned this 3 days ago
   - CEASLIBRARY merged commit 350884 into master 37 seconds ago
   - CEASLIBRARY merged commit 350884 into master 4 minutes ago
   - Pull request successfully merged and closed
     - You're all set—the readme-edits branch can be safely deleted.
   - Delete branch
   - Revert
   - CEASLIBRARY deleted the readme-edits branch 22 seconds ago
   - Restore branch
GitHub Basics

1. Go to Profile
2. Click “Add a bio”
3. Add an email address
LET'S COLLABORATE

1. Libraries Workshop
2. GitHub Basics
3. Branching
4. Pull Request
5. Local GitHub
6. Conclusion
LET’S COLLABORATE

- Commit changes
- Submit Pull Request: compare across forks
- Merge your changes after access is granted
ONLINE TUTORIALS & RESOURCES

► CEAS Library GitHub resources
  
  http://guides.libraries.uc.edu/GitHub

► Online LATEX links & tutorials
  
  ▶ UC main tutorial
  
  ▶ Github tutorials
  
  ▶ Youtube
Questions?
Thank you for attending the workshop!!