

LAB 1

INTRO TO GIT

LAB DESCRIPTION:

THIS LAB WILL HELP YOU SETUP YOUR ACCOUNT, CREATE YOUR FIRST GIT REPOSITORY, AND THEN WALK YOU THROUGH LEARNING A FEW COMMANDS. THROUGHOUT THE TERM YOU WILL MAKE CHANGES AND LOG THEM USING SOURCE CONTROL TO SHOW YOUR PROGRESS AND IDENTIFY WHICH VERSION SHOULD BE GRADED. BECAUSE EACH LAB WILL ALTER YOUR PROJECT - WITHOUT SOURCE CONTROL TOOLS IT WOULD BE DIFFICULT TO GRADE. IT IS IMPORTANT THAT YOU PUSH OFTEN TO BUILD A PROPER LOG OF YOUR WORK. AFTER YOUR ACCOUNT IS CREATED START PUTTING YOUR STORYBOARD TOGETHER. THE STORYBOARD DESCRIBES YOUR AMBITIONS AND GOALS FOR THE TERM PROJECT. EACH WEEK YOU USE THE LAB AS A GUIDE TO ENHANCING YOUR PROJECT (WHAT TO ADD NEXT). THE STORYBOARD WILL BE YOUR VISUAL GUIDE TO HOW THESE ITEM WILL BE INTEGRATED.

LAB BREAKDOWN:

APPENDIX A: INTRO TO GIT
APPENDIX B: STORYBOARD
APPENDIX C: RESOURCES

LAB GOALS:



LEARN GIT CLONE, COMMIT, PUSH, AND OTHER USEFUL COMMANDS.



LEARN WHAT A STORYBOARD IS, AND MAKE ONE.



LEARN ABOUT GIT ReadMes



LEARN SKILLS TO COMMUNICATE YOUR PROJECT IDEAS.

Lab 1: Appendix A - Intro to Git

1. Create GitHub account
 - a.) Go to GitHub.com
 - b.) Enter username, email address and password
 - c.) Click create account
 - d.) Click chosen under 'free'
 - e.) Finish sign-up
 - f.) Verify email address in new tab
2. Create repository
 - a.) Click 'New repository' button



- b.) Give repository a name. If you can't think of one use the inspired repo name suggestion. If you don't like the inspired repo name suggestion, refresh the page to generate a new inspired repo name suggestion.

Create a new repository

A repository contains all the files for your project, including the revision history.

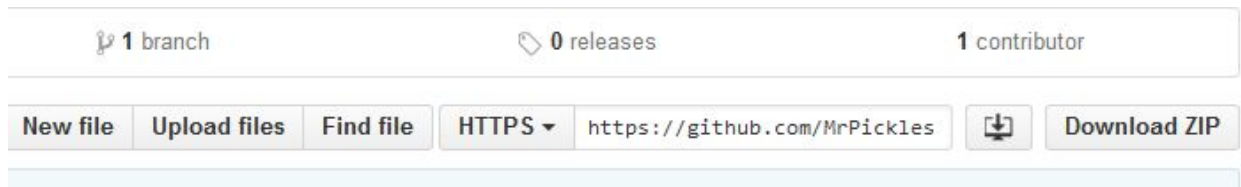
Owner	Repository name
 MrPickles212 ▾	/ <input type="text"/>

Great repository names are short and memorable. Need inspiration? How about **jubilant-octo-disco**.

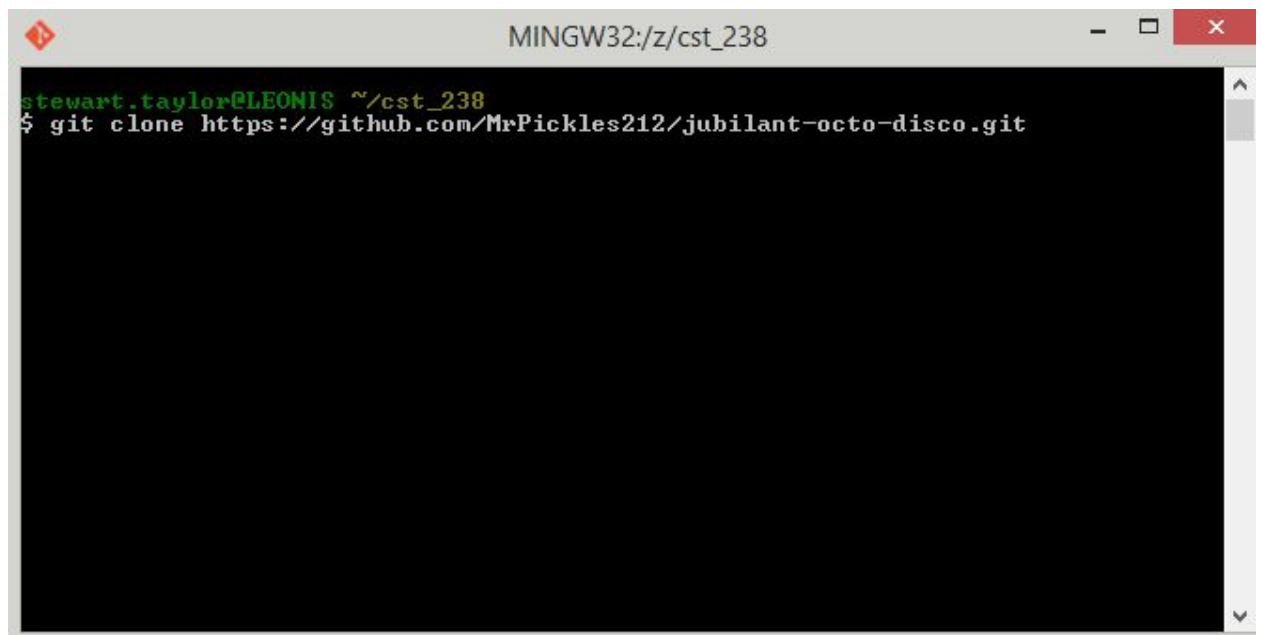
- c.) Give the repo a description.
 - d.) If you signed the waiver, make it public.
 - e.) Click initialize repo with README.
 - f.) Add .gitignore file by clicking dropdown menu. Select 'Qt' or whatever platform you are working with (e.g. 'C#' for WPF, 'Unity' for Unity).
 - g.) Add an open source license, if you wish.
 - h.) Click create repository.

3.) Set up your local working directory

- a.) Press windows key on keyboard
- b.) Type 'Git Bash' and press enter when Git Bash app pops up (you may have to wait a few seconds)
- c.) Enter 'pwd' and make sure the output is '\z' to make sure your present working directory is your Z drive.
- d.) Enter 'mkdir cst_238' then press enter. This creates a directory for us to work in.
- e.) Enter 'cd cst_238' then press enter. This changes our present working directory to the new folder we just created.
- f.) Go back to github and copy and paste the url to your newly created repository. You can do this by clicking inside the text box next to the HTTPS dropdown list, press Ctrl + A, then Ctrl + C. It should be <https://github.com/YourUsernameHere/YourRepoNameHere.git>



- g.) Enter 'git clone ' WITHOUT pressing enter.
- h.) Right click upper-left hand corner of git command prompt, scroll to edit, then click Paste. Press Enter.

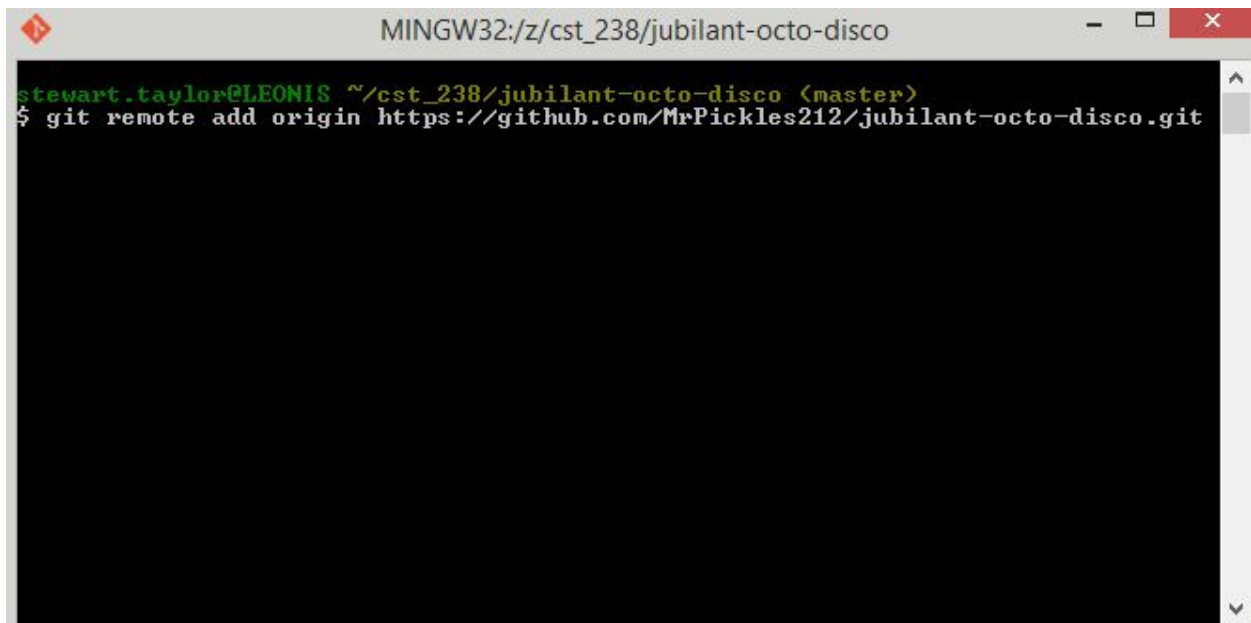


- i.) Do an 'ls' to ensure repo cloned properly.

- j.) Type 'cd ', press tab, then enter. Note: that's 'cd' then a space after it. If you don't add the space, tab is going to think you are looking for everything 'cd' related, instead of autofilling the directory you want to change to.

4.) Configure your repo

- a.) Type 'git remote add origin ', then type the link of your git repo (i.e. we already did this in a previous step.



```
MINGW32:/z/cst_238/jubilant-octo-disco
stewart.taylor@LEONIS ~/cst_238/jubilant-octo-disco <master>
$ git remote add origin https://github.com/MrPickles212/jubilant-octo-disco.git
```

- b.) Hah! Tricked you. You got an error. Why? Since you cloned the repo, git added the remote local for you. If you did not clone, however (i.e. let's say you did a git init), then you would have to use this command to link your working directory to your remote directory.
- c.) Do a git remote -v to confirm that your remote branch (origin) is indeed your current branch.
- d.) Type 'touch hello.c', press enter.
- e.) Type 'git status -s'
- f.) Type 'git add hello.c'
- g.) echo "Zdrastvootye" > hello.c, press enter.
- h.) Type 'git status -s' again. Notice the file has an 'M' to the left of the file name. That means that you just modified the file.
- i.) Type 'git stage hello.c'. This basically says to get the changes I made to the file ready to be committed (saved). Notice how the label on the filename turns from red (means not ready for commit) to green (ready for commit).
- j.) Type 'git config --global user.name "YourNameHere" '

k.) Type 'git config --global user.email "yourname@oit.edu" '

l.) Type 'git commit -m "Insert any random message here that describes the commit."', press enter.

m.) Type 'git config credential.helper store'. This is so you will never be prompted for your username or password after you enter them after this step.

n.) Type 'git push origin master', press enter.

o.) Enter your username when prompted.

p.) Enter your password when prompted.

When finished...begin your storyboard - described in **Appendix B - The Storyboard**

Lab 1: Appendix B - Storyboard Specifications

1:) Alter the readme to contain the following items at minimum (Not necessarily in this order):

- **Project Logo** - Draw a picture, use photoshop or screenshot an image from the internet of what you want your project to look like.
- **Description** - Tell us about the project in descriptive and detailed words.
- **Technologies** - What language / framework are you using? (E.g. Qt)
- **Screenshots/Mockup** - More are better than few
 - These can be screenshots of other apps that show what you are trying to explain.
 - Use descriptive about how images relate to your GUI concept.
 - Application flow - what screen do you see first? Where to go from there?
- **Installation Instructions** (How to install your application)?
- **Contribution guidelines** – Tell me how I can help out including wanted features and code standards. Do you use hungarian notation or anything out of the ordinary?
- **Bugs and TODO List** - What are you working on and what are you trying to fix?
- **Contributor list** – List the humans behind the project.
- **Credits, Inspiration, Alternatives** – Tell us why you picked this project.
- **License** - likely MIT but your choice.

2:) Make sure to have at least placeholders for items you don't know. Your storyboard is a live document that you will continue to modify through the term. Most items should have a fair amount of content to receive a full grade. Creativity is a big plus!

3:) Push the changes (often) to **git** to show the work history on your readme. Then type '**git tag lab_1_grade**' and push again with '**git push --tags**'.

Grading Breakdown

This readme would get an 'A':

<https://github.com/chessgames/play-zone>

This readme would get an 'B':

<https://github.com/StewartTaylor/Bishop-s-Fianchetto>

This readme would get an 'C':

<https://github.com/Tpimp/BluetoothTestHC-05>

Anything worse than this gets a 'D'...

Lab 1: Appendix C - Resources

GIT Links for examples of other well done ReadMe's:

<https://github.com/karan/joe>

<https://github.com/poma/HotsStats>

<https://github.com/rstacruz/hicat>

<https://github.com/b4b4r07/dotfiles>

Links for good storyboard examples:

<http://digitalstorytelling.coe.uh.edu/page.cfm?id=23&cid=23&sublinkid=37>

<http://screencrush.com/movie-storyboards/>

http://lfs.org.uk/documents/storyboard_1.pdf

<http://www.videomaker.com/article/15361-storyboard-examples>