Based on 100 points total, roughly 1.3 per check box unless noted as a %. Extra credit evaluated separately.

Lab 1 - Version Control Software.

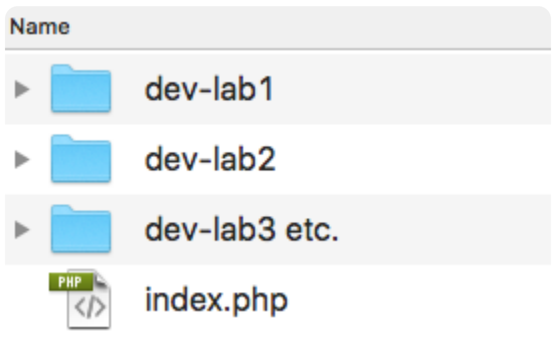
The purpose of this exercise is to:

> Set up a template using GIT and GitHub Orgainziation for the class. If that is not working you cannot receive a grade higher than 50% even if you have completed more than half of the check boxes.

As a reminder all labs are due by noon. Many classes have them due by midnight but I have extended all the due dates to noon the next day to allow you a little more time.

Version Control

- ✓ Before starting this lab be sure to have git installed on your computer.
- ✓ Be sure to have accepted the invitation from GitHub so you can create your repo in the organization.
- ✓ On your computer set up this folder structure (under your cs148 folder) only create dev-lab1:



cs 148 folder structure.

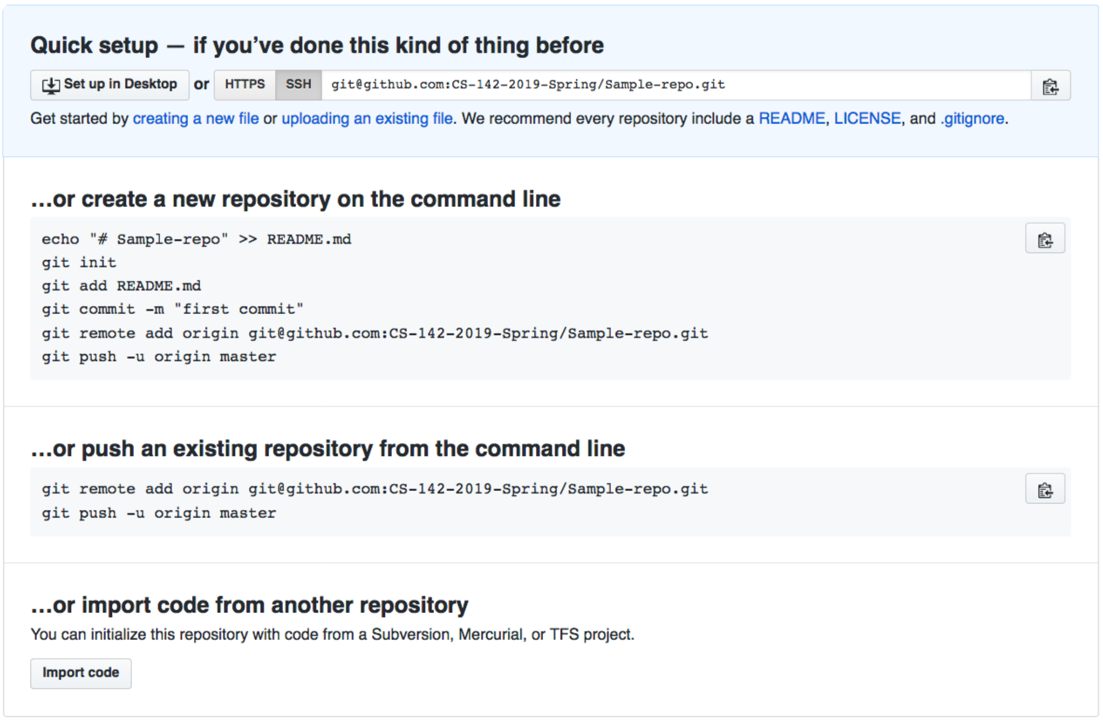
- ✓ Create a sitemap for the class naming it index.php, it belongs in your cs 148 root folder and is NOT part of your git project. It is just there. Here is the structure I want you to follow: sample . This is a three column format I want you to follow. View the source code, you can copy the structure and css.

Setting up a project using version control

- ✓ Create a plain text file named README.md and on the first line just put your First and Last Name.
- ✓ Create a new private GitHub project in the class Repo named Lab-1-FirstName-LastName. For

example mine would be Lab-1-Robert-Erickson.

- [x] Verify that you have used the case and hypens in your repo name as detailed above.
- [x] Keep the content of the first GitHub page (has all the git commands) in a text file named git-Commands.txt as you will need the git remote add origin command (note it is SSH).



Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:CS-142-2019-Spring/Sample-repo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# Sample-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:CS-142-2019-Spring/Sample-repo.git
git push -u origin master
```

...or push an existing repository from the command line

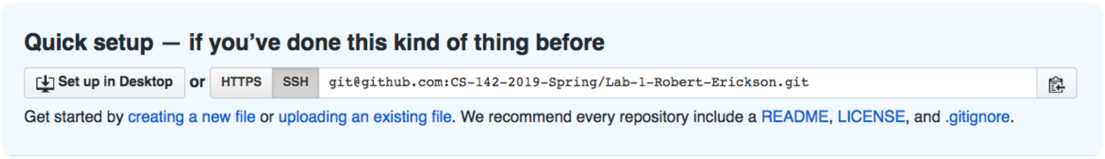
```
git remote add origin git@github.com:CS-142-2019-Spring/Sample-repo.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

git commands to save



Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:CS-142-2019-Spring/Lab-1-Robert-Erickson.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

SSH vs HTTPS - use ssh

- [ ] Add a link to this git repo under your lab one supporting documents on your sitemap.
- [x] Set up your dev-lab1 folder to be a git repo (git init) on your computer.

```
git init
```

- [x] Now connect your local computer to your git repo:

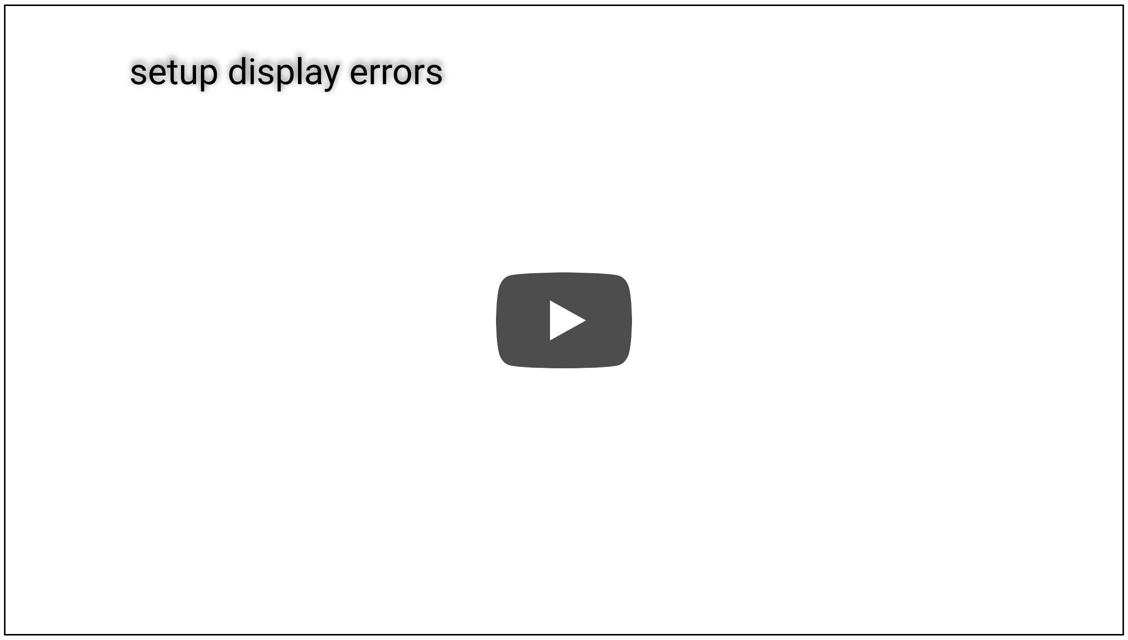
```
git remote add origin [using the repo name as shown in above figure]
```

- [x] Configure your git setup with your username and email that matches the email address you used for Github. For example:

```
git config --global user.name "Robert Michael Erickson"

git config --global user.email "robert.erickson@uvm.edu"
```

- ✓ Create a file named: .user.ini



Setting up a .user.ini file to display errors

- ✓ .user.ini has one line in it:

```
display_errors = 1;
```

- ☐ Create a file named .gitignore

Creating a .gitignore file

☐ .gitignore should contain this:

```
.htaccess
    .user.ini
ADMIN
```

☐ SFTP all the files to your dev folder on the server.

☐ Check to make sure your lab is working (dev).

☐ Set up the ssh key on your computer (this step you only need to do once on your computer, the other steps you do for every repo/project)

Generate the SSH key (only the first four steps)

Copy the key to GitHub (note pbcopy may or may not work but you can look at the key with this unix command: `cat ~/.ssh/id_rsa.pub`
highlight and copy from ssh to the end of your email address, MAC Command c to copy, PC generally highlighting it copies it. Notice the email address must be the same as you configured git to use and your GitHub account)

☐ When your dev folder on the server is working, add this code to your version on your computer.

```
git add -A
```

☐ Make a commit that is telling git this version is good:

```
git commit -m "message about what you did"
```

NOTE: If you forget the -m with a message you will go into the vim editor. Press these keys in this order to quit and write (save and exit)

```
ESC
 :
qw
```

☐ Send your version to your cloud repo (ie github)

```
git push origin master
```

Set up the live site.

☐ Using Terminal or git bash shell, connect to the server and get into your class folder (note below is minus lower case L):

```
ssh w3.uvm.edu -l yourusername [hit enter, type your password]

    cd www-root/cs148/ [hit enter]
```

☐ Make the live folder:

```
    mkdir live-lab1 [hit enter]
```

☐ Go into the live folder:

```
    cd live-lab1 [hit enter]
```

Set up the ssh key on the server.

Same as for your computer only the commands are on the server now.

☐ Generate a new key on the server.

☐ Copy the key to GitHub naming this key with a different name. You will now have two keys in GitHub.

☐ Check your status, if it is not a repo you will need to initialize it. Of course at this moment it should NOT be a repo as we have not initialized it yet:

```
git status

    git init
```

- [ ] Connect your live folder with the GitHub repo. This is the same as connecting your local machine to the repo using the git add remote command you have saved in gitcommands.txt
- [ ] The last step is pull your files down from the repo to your live site.

```
    git pull origin master
```

- [ ] Check the live site, it should work the same as the dev site except errors and warning do not show because you do not have the .user.ini file. We added that to the .gitignore file as we don't want errors to show on the live site if they slip by us.

Html Templates

Use version control by making a commit after you create each file. You do not have to push each time if you dont want to.

- [ ] Create a file named top.php, you should have the same content as I do: top.php
- [ ] Make a commit for top.php
- [ ] Create a file named header.php, you should have similar content as I do: header.php
- [ ] Make a commit for header.php
- [ ] Create a file named nav.php, you should have similar content as I do: nav.php NOTE: PATH_PARTS does not exist this week. It will next week.
- [ ] Make a commit for nav.php
- [ ] Create a file named footer.php, you should have similar content as I do: footer.php
- [ ] Make a commit for footer.php
- [ ] Create a file named index.php, you should have the same content as I do: index.php
- [ ] Make a commit for index.php
- [ ] Create a file named about.php, same as index.php just change the name and header.
- [ ] Make a commit for about.php
- [ ] Create a file named contact.php, same as index.php just change the name and header.
- [ ] Make a commit for contact.php

- ☐ Create a file named admin.php, same as index.php just change the name and header.
- ☐ Make a commit for admin.php

CSS templates

- ☐ Create a folder named css.
- ☐ Create a file named custom.css in the css folder, This should just be basic layout rules using a grid or flex layout. You may modify from my sample CS 148 lab 1 (I used a grid).
- ☐ Make a commit for custom.css
- ☐ Create a file named tablet.css in the css folder. Notice in my sample above that I do not repeat css rules that are in custom.
- ☐ Make a commit for tablet.css
- ☐ Create a file named phone.css in the css folder.
- ☐ Make a commit for phone.css
- ☐ Be sure you are not repeating CSS rules.
- ☐ Quality of your css.

- ☐ You need to have a minimum of 12 commits in your git history (one for each file)
- ☐ Make sure you have pulled to the live site.

Standard Requirements

- ☐ Main goals of assignment not met.
- ☐ Update your main index.php file with all the required links in the correct place. My Sample
- ☐ Do not use any banned elements (div, span, br).
- ☐ Be sure your text is still legible with and without the Color blindness tool.
- ☐ All files must use proper coding guidelines.
- ☐ Make sure paragraph, lists, forms text is not centered. Centering boxes is ok.
- ☐ (5% of your grade) Proper use of version control (commits on a regular basis, commits small with

good message about what was accomplished with this commit) jigsaw.w3.org/css-validator/

- ☐ (5% of your grade) Make sure your files all pass w3c HTML validation: validator.w3.org/
- ☐ (5% of your grade) Make sure your files all pass w3c CSS validation: jigsaw.w3.org/css-validator/
- ☐ sFTP your files to your silk account which is our web server
- ☐ Submit your assignment only once in blackboard. See next direction first.
- ☐ Copy this code as a text submission in blackboard, look for the Write Submission button. You will need to click on the html button once you are in the editor and paste this code there:

```
<a target="_blank"
href="//lcarrera.w3.uvm.edu/cs148/live-lab1">
http://lcarrera.w3.uvm.edu/cs148/live-lab1</a>
```

NOTE:

This is how you will set up every project EXCEPT you will not have to do the ssh keys as they are already set up on your computer, GitHub and the server. I recomend setting up a new project named delete and go through all the steps above (except keys) the delete everything. Repeat till you remember it.