

GIT

Exercises

Exercise 1 – GitHub Setup



Overview

In this exercise, you will setup Github for Windows (or Mac).

Objectives

At the conclusion of this exercise, you should be able to:

- ✓ Install GitHub for Windows (or Mac)
- ✓ Use the Git command line
- ✓ Create an initial repository

Step 1: Preparation



In this exercise, you will obtain and install a GitHub environment. Begin by downloading the GitHub implementation for your platform.

Either Windows:

<https://windows.github.com/>

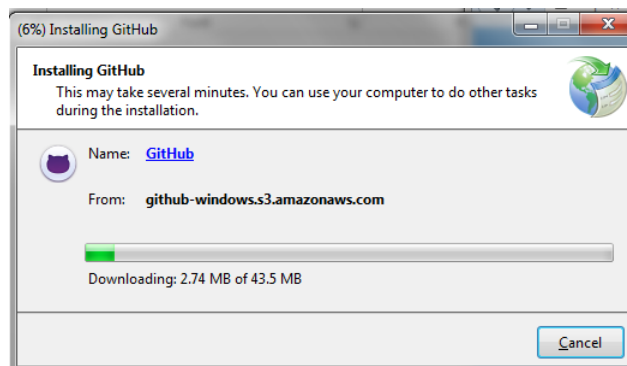
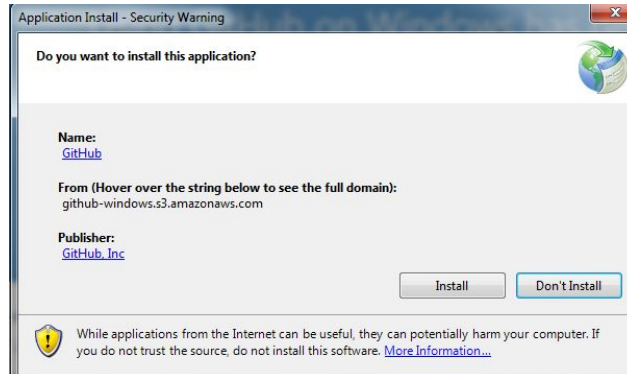
Or Mac:

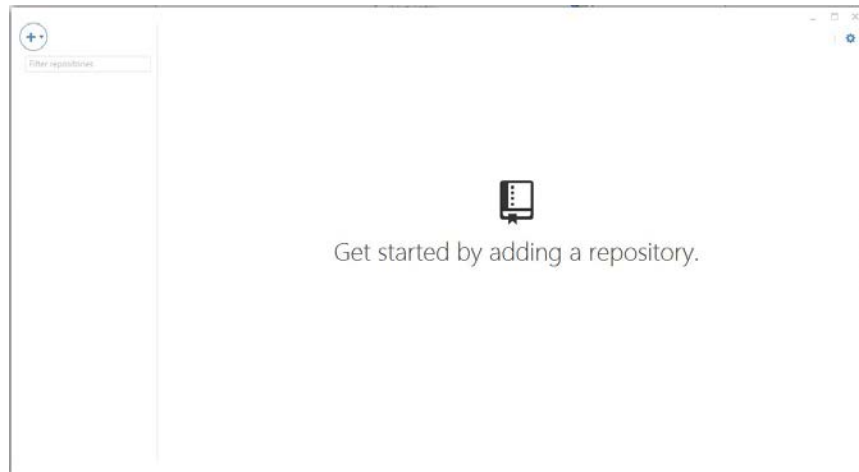
<https://mac.github.com/>



Step 2: Install GitHub

Run the installer, accepting the defaults. (Windows screens are shown, Mac will differ slightly)





Close the GitHub for Windows (or Mac)

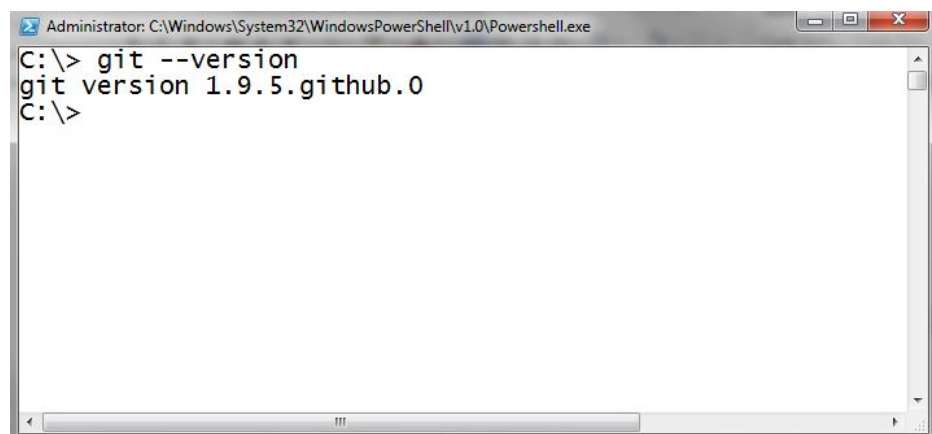
Step 3: Use Git Shell

From the Start menu (or applications) open a GIT shell.

- The GIT shell is a normal shell window with the environment set to include GIT.

Verify the version of Git with the following command

```
git --version
```



Step 4: Set common GIT Options

Git has many core options that change the behavior of Git. Display all the settings for Git:

```
git config --list
```

The first thing you should do when you install Git is to set your user name and e-mail address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@empl.com
```

```
git config --list
```

Step 5: Create a local Git Repository

In your GIT shell change to c:\temp and run the following command to create a project folder:

```
mkdir c:\temp\gitclass
```

Change into that folder:

```
cd c:\temp\gitclass
```

Initialize the git repository for our project directory with the following command:

```
git init
```

```
posh~git - gitclass [master]
C:\>
C:\> cd \temp
C:\temp> mkdir gitclass

Directory: C:\temp

Mode                LastWriteTime         Length Name
----                -
d-----          3/9/2015   7:50 PM             gitclass

C:\temp> cd gitclass
C:\temp\gitclass> git init
Initialized empty Git repository in C:/temp/gitclass/.git/
C:\temp\gitclass [master]>
```

Compared to SVN, the `git init` command is an incredibly easy way to create new version-controlled projects. Git doesn't require you to create a repository, import files, and check out a working copy. All you have to do is `cd` into your project folder and run `git init`, and you'll have a fully functional Git repository.

However, for most projects, `git init` only needs to be executed once to create a central repository—developers typically don't use `git init` to create their local repositories. Instead, they'll usually use `git clone` to copy an existing repository onto their local machine.

Exercise 2 – Managing Files in Git



Overview

In this Lab you will add and manage files in your "main – master" git repository. We will then clone the repository to mimic the actions developers would normally take.

Objectives

At the conclusion of this exercise, you should be able to:

- ✓ Manage files within the git repository
- ✓ Make changes and commit to the repository
- ✓ Clone a repository for developer actions

Step 1: Add content to the main repository



Open a text editor and create a file named file1.txt with a simple text message inside it :

file1.txt:

```
"Hello there my friends"
```

Save the file in the git project folder: `c:\temp\gitclass`

Open your git shell and do a `dir` in the project folder:

```
posh-git - gitclass [master]
C:\temp\gitclass [master]> dir
C:\temp\gitclass [master]>
C:\temp\gitclass [master]>
C:\temp\gitclass [master]> dir

Directory: C:\temp\gitclass

Mode                LastWriteTime         Length Name
----                -
-a---             3/9/2015   9:45 PM             24 file1.txt

C:\temp\gitclass [master +1 -0 -0 ]>
```

The master branch of the repository is active, and it shows 1 new file that is not in the repository.

Add the .txt file to the control of the repository:

```
git add *.txt
```

Commit the new file to the repository:

```
git commit -m "Initial file1 commit"
```

```
posh-git - gitclass [master]
C:\temp\gitclass [master]> dir
C:\temp\gitclass [master]>
C:\temp\gitclass [master]>
C:\temp\gitclass [master]> dir

Directory: C:\temp\gitclass

Mode                LastWriteTime         Length Name
----                -
-a---             3/9/2015   9:45 PM             24 file1.txt

C:\temp\gitclass [master +1 -0 -0 ]> git add *.txt
C:\temp\gitclass [master +1 -0 -0 ]> git commit -m "Initial file1 commit"
[master 3c563f4] Initial file1 commit
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
C:\temp\gitclass [master]>
```


Step 2: Cloning a repository

Developers will usually clone a main repository locally on their development platform. Changes will be made to their own repository and then occasionally pushed back to the main repository.

Open a new git shell window and change to the `c:\temp` folder

Run the command:

```
git clone c:\temp\gitclass c:\temp\gitclassclone
```

This will create an entire clone of the main repository. The first argument to the git clone will be a url or local path.

Take a look in the new `c:\temp\gitclassclone` folder and you should see our file.

Step 3: Updating a file in your repository

Files in your repository are monitored for changes. In our lab this repository is distinct from the "Main" repository that is under `c:\temp\gitclass`

Edit the contents of our copy of the file1.txt (the copy that is in the `c:\temp\gitclassclone` folder) to be:

```
Hello there my friends  
Version #2
```

Run the status command to see that file 1 is changed, but the change is not yet staged for a commit:

```
git status
```

Add file1.txt to the stage

```
git add *.txt
```

and then commit it to our repository:

```
git commit -m "Version 2"
```

Exercise 3 – Working with branches



Overview

In this lab you will learn to branch a repository.

Objectives

At the conclusion of this exercise, you should be able to:

- ✓ Create a branch to your repository
- ✓ Compare versions of files within your repository
- ✓ Push changes to your branch

Step 1: Create a new branch



For the following task, we will continue working in our "developer" copy of the repository. We will branch our code make a change and then commit it to the branch.

Within the c:\temp\gitclassclone project folder run the following command:

```
git branch version3
```

The branch was created, but your prompt will still show that you are in the master branch. Change to the version2 branch

```
git checkout version3
```

Your active branch is now version3. Open our file1.txt and change the file to reflect version 3:

```
Hello there my friends  
Version #3
```

Step 2: Commit changes to the branch

If you use the git status command you will see that one file is changes, none are staged and nothing is there commit. Run the following command to stage the change to our branch:

```
git add file1.txt
```

Commit the change to our active "version3" branch:

```
git commit -m "Version 3"
```

Step 3: Verify the different branches

Switch back to the master branch:

```
git checkout master
```

Look at the contents of the file. It should still be version 2.

Switch back to the version3 branch:

```
git checkout version3
```

Look at the contents of the file. It should be version 3 again. The git diff command can tell us all kinds of interesting things. Relevant to us now is the difference between two branches of a particular file. Run the following command:

```
git diff master..version3
```

You will see that the one line is removed and a replacement line was added. Change back to the master branch and then merge changes from version3:

```
git checkout master  
git merge version3
```

Exercise 4 – Practice



Overview

In this exercise, you will continue to use our main repository, but you will emulate a second developer wanting to make changes.

Objectives

At the conclusion of this exercise, you should be able to:

- ✓ Clone the master repository
- ✓ Create a branch in the new developer repository



Step 1: Create a new clone and make changes

On your own this time create a new clone of the main repository named `c:\temp\gitclassdeveloper2`.

In that new repository make a change to `file1.txt`. Change the text to contain "DEVELOPER 2: ..." these from

Create a branch in the `developer2` repository and then make another change that is to be committed to the new branch.

Exercise 5– GitHub



Overview

In this exercise you will create a GitHub account and repository, then synchronize your project with GitHub

Objectives

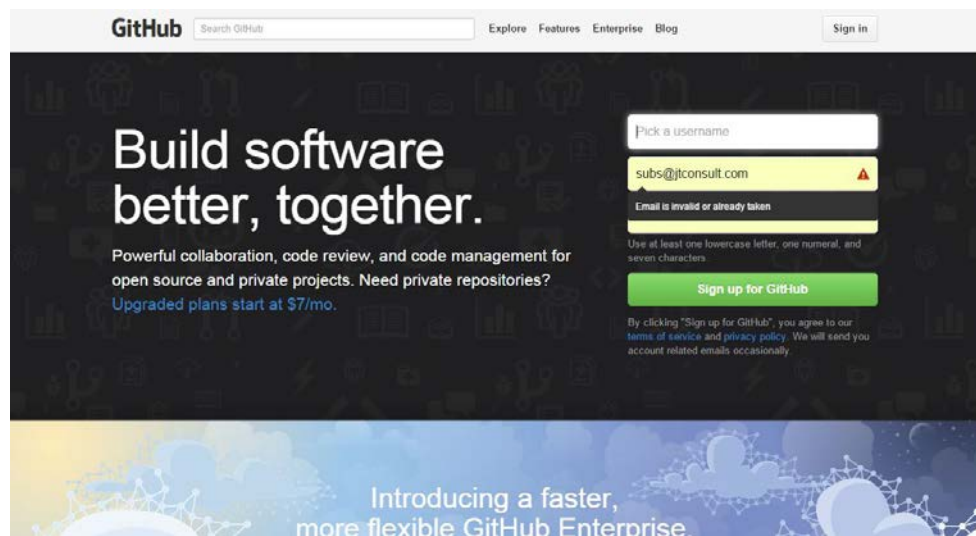
At the conclusion of this exercise, you should be able to:

- ✓ Create a Github repository
- ✓ synchronize your repository with Github



Step 1: Create a GitHub account

In a browser go to www.github.com and create an account.



Once you have your account login to the github web site.


Step 2: Change your local Git settings to match GitHub

Your local git settings need to be in synch with Github so your account will match. In your git shell within the c:\temp\gitclass project execute the following command, changing "Your Email" to .. your email.

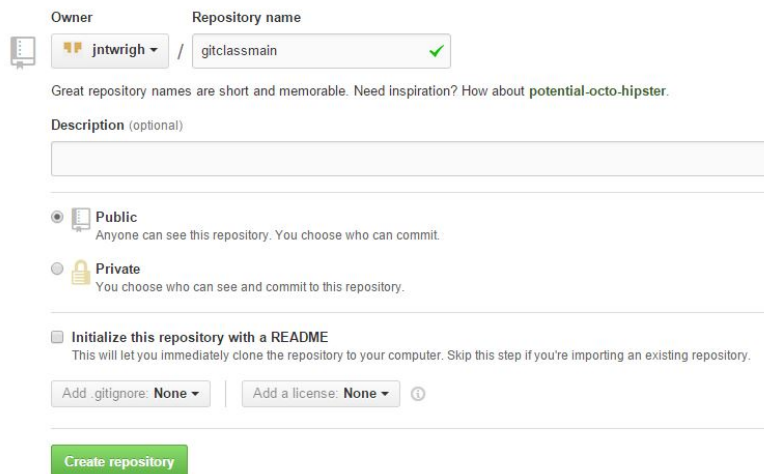
```
git config --global user.email "YOUR EMAIL"
```


Step 3: Create a GitHub Repository

Back in the browser go to: <https://github.com/> and log in if you are not already logged in.

Press the  (plus) button next to your username to create a new repository.

Name the repository gitclassmain -- leave the repository as Public and press the green Create repository button



Owner:  jntwrigh / Repository name: gitclassmain ✓

Great repository names are short and memorable. Need inspiration? How about [potential-octo-hipster](#).

Description (optional):

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** ⓘ

Create repository

Once you hit Create copy into your clipboard the link from the quick setup box. it will look something like:

```
https://github.com/USERNAME/gitclass.git
```

Step 4: Synchronize your local repository with GitHub

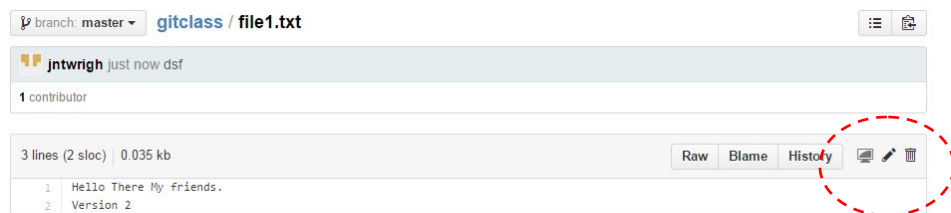
In your git shell within the c:\temp\gitclass project run the following command to push your project to a new remote repository: (Swap the URL with the URL you copied in the last step.

```
git remote add github https://github.com/USERNAME/gitclass.git
```

We have created a remote server reference named github that we can now push our project to. Run the following command to actually push your project up to github.

```
git push -u github master
```

In your web browser look at your github repository (You may have to refresh the screen, but you should see the file1.txt is up on github. Drill into file1.txt and view the contents.



Edit the file to say Edited on GitHub, then click on the Preview Changes link, add a version message and then commit the changes.

Back in your shell run the following command:

```
git pull github
```

Then view your file and you should see the changes from github.