Git: How to Setup and Use it on the SEAS Machines

What is Git and why use it?

Git, a tool widely used in industry, is an open source distributed version control system (a platform for multiple collaborators to work on the same project at the same time). Each time a user decides to share a change with his collaborators, the user simply uses Git to make those changes available to his collaborators on a shared file system. By sharing files this way, recovery and version definition becomes reliable, responsive, and simple. In this tutorial, your shared file system (called a *repository*) and you will be communicating with it using Git. There will also be a *remote repository* hosted by GitHub.com. Because version control systems (VCS) are so common in industry, it is important that new users master the terminology associated with this technology. For this reason, this tutorial also includes definitions of common VCS terminology.

Prerequisites

This tutorial assumes the following:

- You have a SEAS account and are in front of a SEAS linux machine
- You have a basic understanding of Unix file system navigation from command line (More information on command line navigation can be found via Google)
- You have an account on repository hosting site GitHub (*https://github.com/signup/free*)
- You have an SSH key associated with your GitHub account. If not, it is easy to set up; follow the instructions on this website: *http://help.github.com/linux-set-up-git*

Setting up Git on your SEAS Machine

Making a New Repository

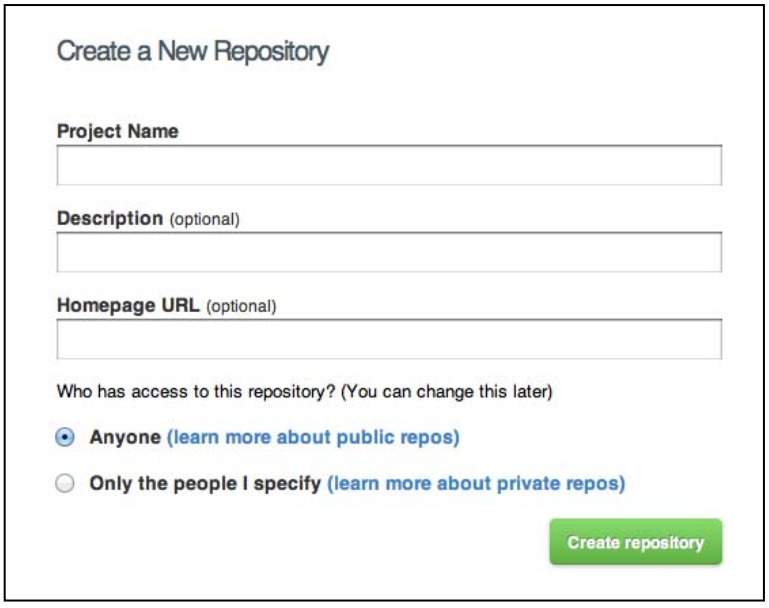
If you have already created a repository for collaboration or you want to work on an existing repository, skip this section.

1. **Log in to your GitHub account from a SEAS Linux machine**. Login to GitHub at *https://github.com/login*

2. An account dashboard will appear. Click on the "**New repository**" button in the "Your Repositories" area.

/wərd/ **`Repository or Repo`**: A place to store files. In this set up of version control, we will be using a local repository (the folder on your computer) and a remote repository (on the GitHub account) to prepare and share files.

3. In the new page that appears, **fill in the information about your new repository**.



Project name *:* The name of the repository *(ex. hello world)*

Description (optional)*:* A brief description about the repository *(ex. "this is a friendly project")*

***Homepage URL (optional*)**: URL for website related to the project *(ex. if this is a repository shared by a class, the URL points to the course syllabus)*

*By default, under "**who has access to this repository**", "**Anyone**" is selected. To restrict access to specific collaborators and get more features, you must upgrade your account to a paid account.*

4. When you have finished filling out this form, click on the "**Create repository**" button.

5. **Open a terminal window.**

6. **Set up your Git username and email address** by running the following commands at the terminal command line

```
git config --global user.name "Your Name"
```
→ This configures your name as you want it to appear to your collaborators

```
git config --global user.email puneetv@seas.upenn.edu
```
→ This configures the email address associated with your Github account with your computer. Use the e-mail you used to sign up for Github.

7. **Make a folder for your project.** Navigate into that folder using command line navigation.

8. **Initialize Git**. Type the following command in order to use git commands with in the folder you just navigated to:

```
git init
```

9. Now, you are going to **establish a connection to the GitHub repo and test your ability to communicate with it**. Don't worry about learning the commands below at the moment. You will see the commands for add, commit, and push explained in greater detail in the "Start Collaborating" section.

    a. Start by making a sample text file (.txt).

    b. To add this file to "staged for commit," enter the following command, replacing `samplefile.txt` with the name of the file you created:

    ```
    git add samplefile.txt
    ```

    c. Now, commit your changes to the repository on your local machine and prepare to share them with collaborators. Also, make a note to go with your changes to the repo. Do so by running the following command:

    ```
    git commit –m "commit message"
    ```

    d. Set your project folder to communicate with Github by running the following command:

```
git remote add origin
git@github.com:USERNAME/NAMEOFPROJECT.git
```

Important Note: Any spaces in the name of your project will be turned to dashes ("-") in the git address.

e. Finally, run the following command to push your first committed file to github and to establish the master branch of your project:

```
git push –u origin master
```

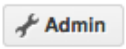
Important Note: If you get an error reporting `permission denied` from the terminal, it means your SSH key is not properly set up. Please see the SSH key setup guide from GitHub, and then return to this step: *http://help.github.com/linux-set-up-git*

10. **Add collaborators to your repo.** If you made it to this step, congratulations! You are now ready to share your repo with your collaborators.

    a. Login to GitHub, if you have logged out *https://github.com/login*

    b. Navigate to your GitHub account Dashboard, find the "Your Repositories" area, and click on the name of the repository you wish to share.

    c. Click on Admin icon in the upper right hand corner of the new page that appears.



    d. In the left side panel, select "Collaborators."

    e. In the new panel that appears, the GitHub usernames of your collaborators. Your collaborators will see an invitation to join your repository when they next login to their GitHub account.

Collaborate Using an Existing Repo

1. **Log in to your GitHub account from a SEAS Linux machine**. Login to GitHub at *https://github.com/login*

2. **Click on the notifications icon** in the upper right of the account dashboard page that appears.

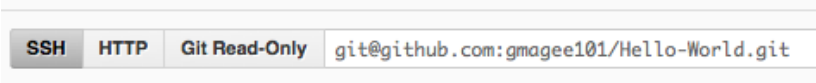
3. In the new page that appears, you should **see an invitation to join an existing repository.**

Important Note: If you do not see an invitation, contact your collaborator that set up the repo and request that they add your GitHub username as a collaborator on the repo.

4. **Click on the name of the repo** in the invitation. You will see the full information for the repo, including the information you will need to access it on your local machine in the new page that appears.

5. **Clone the repo**. Cloning the repo will copy all of the files currently in the repo to your computer and give you the ability to communicate with the repo.

    a. At the top of the new that appears, make sure that repository information is set o SSH.



    b. Copy the repository information in the text box to the right of the SSH button.

    c. Open a terminal window and navigate to the directory where you want to store files you are collaborating on with others.

    d. Run the following command, replacing "repo_info" with the information you copied off the GitHub website:

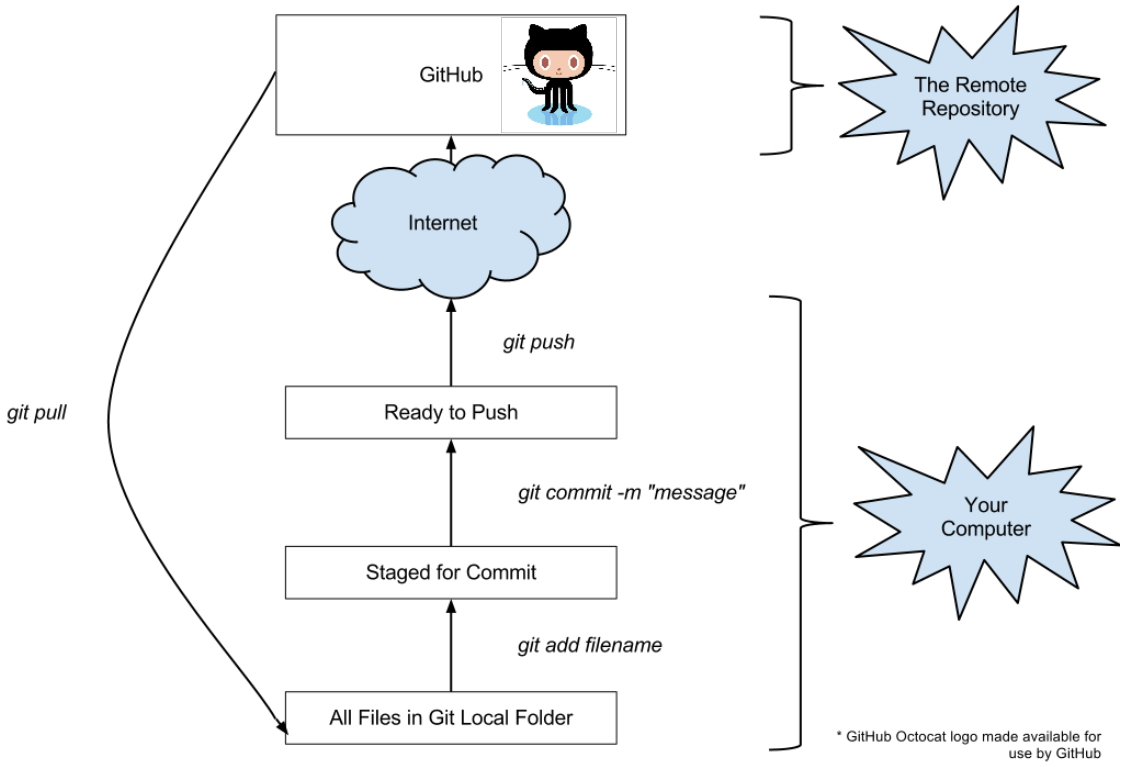
```
git clone repo_info
```

Important Note: If you get an error reporting `permission denied` from the terminal, it means that your SSH key is not properly set up. Please see the SSH key setup guide from GitHub, and then return to this step: *http://help.github.com/linux-set-up-git*

Start Collaborating

Once your local repository is set up and communicating with the remote repository, you are ready to add to and receive changes from the remote repository. This section focuses on the pattern of commands you will need to share files with your collaborators. In general, it is best to frequently send and receive files from the remote repository to minimize conflicts between the files that you and your collaborators are working on, as well as share the most recent code you have written. In general, you would want to run the commands below after finishing a single method or unittest.

The next set of instructions will explain this diagram, which outlines what commands to run to communicate with the remote repository:



The cycle of adding and receiving changes to shared files

1. **Make sure you are in your project directory.** You will need to go to the local folder you made in the previous section to continue.

2. **Get the most recent version of the shared files**. To get the most recent version of the remote files **(why)**, you will need to do a pull. You can do this by running the command:

```
git pull
```

/wərd/ **`Pull`**: Getting the most recent changes to the shared files and applying them to the local files on your machine.

Important Note: If your local copy of a file and the remote copy of a file both contain new additions, git will ask you to manually merge the files before the pull can be successfully completed. For more in merging conflicting files in git, see the git manual: http://schacon.github.com/git/user-manual.html - resolving-a-merge

3. **Choose which files you want to share with your collaborators.** Now, you will need to choose which files in your local folder you would like to put on the shared repository. To choose a file to put on the repository, type the command below, followed by the name of the file you wish to share.

```
git add myFileToShare.txt
```

/wərd/ **`Add`**: Adding a file you wish to share with your collaborators to a bundle that will eventually be committed to your local repository and later pushed to the hosted repository. This bundle is usually called "staged for commit."

4. **Check which files are going to be shared.** Git provides an easy way for you to check which files in the folder are not going to be shared, which files have been bundled for sharing with collaborators, and which bundles are complete and ready to go to the remote repository. You can see the state of your files as seen by git by running:

```
git status
```

Important Note: You may run this command at any time. In fact, it is good practice to run this command at just about any phase of working with Git.

5. **Complete Your Bundle of Files**. Once you have selected all of the files that you will need to share with your collaborators, you will need to complete the bundle. To complete the bundle, run the command below, followed by a short message (about one line) that describes the small change that you are sharing.

```
git commit –m "updated unittest for fibonnaci method"
```

/wərd/ **`Commit`**: The process of completing the bundle and adding a message to your collaborators so they know what changes to look for.

6. **Send your files to remote repository.** Run the command below to send your files to the remote repository.

```
git push
```

/wərd/ **`Push`**: Sending your committed changes to the remote repository so that your collaborators can view them.

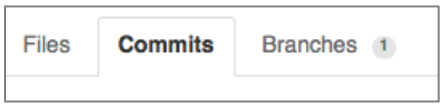
Important Note: If your local copy of a file in your commit and the remote copy of a file both contain new additions, git will ask you to manually merge the files before the push can be successfully completed. For more in merging conflicting files in git, see the git manual: http://schacon.github.com/git/user-manual.html#resolving-a-merge

7. **Go to the remote repository and verify that your changes were uploaded.**

   a. Log in to your GitHub account at *https://github.com/login*

   b. In the account dashboard window that appears, click on the name of the repository you are working on from the "Your Repositories" section.

   c. In the new window that appears, select "Commits" from the menu at the top of the page.



   d. In the log that appears below, you should be able to see a log of who has contributed to the project along with their commit notes about their contributions.

8. **Repeat**. Because you should be adding small changes to the remote repository frequently, you should make an effort to learn this idiomatic usage of Git:

   pull – add – commit – push

More Information:

For more information on Git and to find out more about Git features, read:

The Git Tutorial:
http://www.vogella.de/articles/Git/article.html

ProGit by Scott Chacon
ISBN: 1430218339