Our department runs a git server (`git.cs.slu.edu`) for students to store and submit their coursework and projects. This course, Operating Systems, requires the use of git to submit coursework. This document provides some brief technical information about initial setup of git. **This guide assumes you are working on one of the CS department Linux machines**. It should be possible to access git through other methods, such as a git client on your personal machine, but these are not officially supported.

Git access

Initial setup There are some one-time steps for setting up your usage of git (for any repository):

1. Let git know your name
   `git config --global user.name "Jane Smith"`
2. Let git know your email
   `git config --global user.email "jane.smith@slu.edu"`
3. Set your preferred editor (you can also use vi, emacs, etc.)
   `git config --global core.editor "nano"`

Now to set up the directory for this specific course. Note that the semester may need to be changed.

4. Create the initial project repository.
   `git clone git@git.cs.slu.edu:courses/spring19/csci_3500/username`

where "username" is the account name you use to login to the department Linux resources.

Adding new files to the repository Before you can upload files to git you must first tell the system what files you want to track. You may add multiple files in the same command. For example:

```
git add assignment.c responses.txt
```

You don't need to run the `add` command on files you initially got from the repository.

Fetching changes from the repository Other authorized users (such as a group-work partner or your instructor) can make and upload changes to your git repository as well as yourself. Individual course repositories are only accessible by you and the instructor, but group repositories (that you might find in Capstone or Software Engineering) can have many active participants.

```
git pull
```

Uploading files to the repository Changes to your files must be *staged* before they can be uploaded. This is done by specifying what modified files you want to `commit`, and any files not marked as such will not be sent to the repository. After committing, the files are actually sent with the `push` command. When committing you are required to provide a short message explaining the purpose of the commit. Descriptive messages make it easier to understand your contributions to a project and for others to understand what changes are taking place.

You should always `pull` changes from the repository before you upload new changes. This gives you the opportunity to fix any conflicts that may occur.

```
git pull
git commit assignment.c responses.txt -m ‘‘Explanation of changes’’
git push origin master
```

You can selectively upload changes by only committing those files you want to send at the current time. Alternatively, you can commit all changes in a given directory with the `-a` flag.

```
git pull
git commit -a -m ‘‘Finished part 1!’’
git push origin master
```

Verifying your changes have been sent To verify that your local changes have been successfully sent to SLU's repository you can use the `status` command.

```
git status
```

If git says that your branch is *ahead* of origin/master, this means you have local changes that have not been successfully uploaded. If you are *behind* the origin/master then there are remote changes in the repository that you have not downloaded with a `pull` command. If neither of these is the case then your local files are the same as the remote files on the repository.

```
# On branch master
nothing to commit, working directory clean
```

Managing merge conflicts Occasionally, two users in a large repository will modify the same files. Normally these users don't happen to modify the exact same parts of a file and git is very good at *merging* everyone's work automatically. When two commits conflict in a way that git can't combine them automatically a *merge conflict* has occurred and a human must intervene to reconcile everyone's changes.

Handling merge conflicts is beyond the scope of this guide, but many tutorials can be found through your favorite search engine.

Accessing from outside SLU's Linux environment Users are not allowed to log in directly to the git server. Instead, the server uses a ssh public/private keypair. We have set up your CS department accounts to create such a keypair in your home directory so that your account can successfully interact with the git server.

If you wish to connect from some other machine, you will be responsible for copying your private key from hopper, and placing into the appropriate ssh directory on the computer of your choice. How to do this depends on the OS you are using, but the files you will need from hopper can be found at `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`.

Learning more about git There are a variety of good information sources online about git. For example `https://www.atlassian.com/git/tutorials`