This document describes the basic workflow we will use in CS 135.

BASIC UTILITIES
This course makes use of three basic systems:
* Python 3.
* Git.
* Emacs, or an equivalent editor.
Each of these systems supports the CS135 workflow. Let's see how.

ACCOUNTS AND THE SERVER.
Everyone has an account on our CS servers. Typically, the account name is your OIT 'unix' account name, prepended with your class year. You have been given a random password. Please make sure you change this to something you can remember. In this document, I'll represent this account with '20xyz'.

We have several servers that support different types of services. For our use, gala is a server that has all filesystems available all the time (individual machines in our labs, for example, only have your files available when you’re sitting in front the console). The full name of this server is gala.cs.williams.edu. It is visible inside and outside the college.

OVERVIEW OF GIT.
Git is a flexible system for sharing data (files, handouts, code, etc.). These data are organized in 'repositories'. When you want to make use of a repository for the first time in a new environment (our lab, your laptop, an individual OIT machine), you must 'clone' it. The clone looks like this:
```
git clone ssh://18xyz@gala.cs.williams.edu/remote.git local
```
You will have to provide your CS password to verify your identity.
It makes a full copy of the 'remote' git repository in your 'local' directory. Once you have cloned the repository, you need never clone it in this environment again. Some repositories are shared among all class members; others are private and are shared only between you the the CS135 staff.

When you start a work session, you typically want to make sure you update your local repository with changes that might have occurred while you were away. This is called 'pulling' the repository. It’s simple:
```
git pull
```
You’ll have to provide a password.

While you work, you’ll modify and create files. Occasionally, you’ll feel the project has made some progress. You’ll want to add and commit any files that you have changed. If, for example, you’ve made modifications to 'election.py' and ‘README’, you’ll probably use commands like
```
git add election.py
```
```
git add README
```
```
git commit -m 'Added vote total calculations.'
```
These changes are entered permanently into the repository, and the message you specified (with ‘-m’) is added to the log file. Git prints the log with:
```
git log
```
None of these commands requires a password because they’re operations on the local repository.

To see what files are currently included in your commits, you can use
```
git status
```
which shows the state of all the important files in your directory structure. If you have an important file that is not being tracked by git, you should consider adding and committing the file.

As you make changes locally, your repository holds work not stored in the global repository. *Whenever* you are done working, you should perform a commit (as above), and then ‘push’ your new work up to the server’s version of the repository. This is accomplished with a command like
```
git push
```
This causes all the commits (changes) in your local repository to be applied to the global repository. It is important to remember that all the different
versions of your data that you’ve ever committed are remembered. If you think you made a mistake, you can always ‘check out’ an older version. The only way, however, that you can keep track of these older versions is if you
1. pull at the beginning of a work session,
2. commit often, and
3. push at the end of the work session.
When you push changes up to the server those changes will be transferred to other environments you may use when you perform a ‘pull’ in those locations.

ESTABLISHING YOUR GIT IDENTITY.
As you commit to repository changes, git tags all your changes with your identity. It is helpful if you establish your identity the first time you use a new environment. These commands would establish this for ‘Joe Cool’:

   git config --global user.name "Joseph Cool"
   git config --global user.email "jjc9@williams.edu"
   git config --global push.default simple
   git config --global core.editor emacs

You should, of course, use your own identity! Try to avoid being obscure: this identity is used by graders to understand who authored an assignment.

THE CS135 SHARED REPOSITORY.
We maintain a read-only shared repository for this class. It allows lecture materials to shared quickly and accurately. You’ll find lecture notes, lab handouts, examples, and documents like this in the ‘shared repository.’
The remote repository is cloned with:

   git clone ssh://20xyz@gala.cs.williams.edu/˜cs135/shared.git ~/cs135/shared

This makes a new copy of the shared resources in a subdirectory of the current directory, called ‘shared’. You should occasionally do a

git pull
when you’re sitting in the shared directory. This will bring your local copy of shared files up-to-date. It is unwise to make changes to files in this directory since git will become concerned that these changes might need to be shared. To avoid this possibility, it’s not possible to push the local version of the repository.

PRIVATE CS135 WORK REPOSITORIES.
Whenever we start working on a new lab in a new environment, say ‘lab3’, we clone our private repository from the server. (I suggest this repository be stored in your local cs135 directory you constructed at the beginning of the semester.) This gives us access to starter files and handouts for that lab:

   git clone ssh://20xyz@gala.cs.williams.edu/˜cs135/20xyz/lab3.git ~/cs135/lab3

Notice that your name appears twice in this command. If you perform work in more than one environment (say your laptop and our lab), you’ll need to clone (exactly once!) in each location.

From this point on, your workflow consistently looks like:

1. pull at the beginning of a work session:
   
git pull

2. commit one or more times:
   
git add <changed files>
   
git commit -m ’<useful* comments about changes>’

3. push at the end of the work session:
   
git push

When your assignments are graded, Duane or a TA will update your repository with a grade file, which you can pull back to your private space.

Some homework assignments may involve communication through a private repository, as well.

   * * *