Git for SE101
Overview

- What is Git?
- Creating a bare Git repository
- Creating a working copy of the repository
- Committing files to your repository
- Working remotely
- Reverting
- Submitting your work for se101
What is Git?

Git is a distributed version control system enabling you to:

- Easily track the changes you make to your code
- Easily collaborate with others when you write code

Useful links:

Creating a Git repository

Create and change to the directory where we will put our repository

```bash
$ cd
$ mkdir -p ~/se101/git/se101.git
$ cd ~/se101/git/se101.git
```

Next, create the bare repository. A bare repository is a central repository that you will not access directly:

```bash
$ git init --bare
```

A bare Git repository will now be created at `/home/userid/se101/git/se101.git`. Try typing `ls` to see what was created.

```bash
$ ls
```
Cloning a working copy

Change back to your ~/se101/git folder:

$ cd ~/se101/git/

Next, we want to create a working copy of our empty repository:

$ git clone se101.git workspace

This creates a folder workspace where we will put all of our labs and projects. Change into the new folder:

$ cd workspace

Create a new folder for the first problem in assignment 1 (Problem 2.17):

$ mkdir A1_2_17

$ cd A1_2_17
Adding a file

Create a file

$ touch numbers.java

Check the git status. It shows untracked changes.

$ git status

Now we need to add the file to the repository:

$ git add numbers.java

Check the git status. It shows our changes staged to be committed:

$ git status
Git statuses

• Files in the git directory can be in one of four states in relation to Git

  • **Untracked**: The file has not been added to the repository

  • **New**: The file has been added to the staging area of the repository, but has not yet been committed to the repository

  • **Unmodified**: The current version of the file has been committed to the repository

  • **Modified**: The file has been committed to the repository, but has been modified since it was last committed
Commit the file to your cloned copy. “Initial commit” is a comment to help you identify this version. The comments are mandatory and if you don’t specify one, git will open a text editor for you, where you will be required to write one:

$ git commit -m "Initial commit"

We want to push our changes back to the bare repository so we can check them out elsewhere

$ git push origin master
Committing changes

Edit the file you just committed. Try:

```
$ echo "//testing" >> numbers.java
$ git status
```

We need to add the file again to commit our changes!

```
$ git add numbers.java
```

Check the status! You should see “modified”:

```
$ git status
```

Commit the file to your cloned copy and push it to the master!

```
$ git commit -m "added test comment"
$ git push
```
Git workflow

• Every time you create a new file or change an existing file, you have to use four commands:
  • `git pull`: make sure your working copy is up to date before doing work to avoid conflicts
  • `git add`: to add the files you created/changed
  • `git commit -m "message"`: to commit the changes to your cloned repository
  • `git push`: to push the changes to the master

• Always remember to check the status to make sure your changes are committed
Viewing commit history

Check the history for the current project

$ git log

It will show your revision history with hashes to identify each commit, the author, date, and the messages you type.
Install Git if you don’t have it yet:

$ sudo apt-get install git

Change to the directory you want to clone to

$ cd whatever_path_you_want

Remember, this is your computer, so you can put this wherever you want. Somewhere in your home folder would make sense, like ~/se101/

Clone the master:

$ clone userid@tux.cs.drexel.edu:~/se101/git/se101.git

workspace

** Always remember to “git pull” to avoid conflicts **
Working off campus (Windows)

• Install Git if you don’t have it yet: http://git-scm.com/download

• Make sure you choose: checkout as-is and commit unix-style during install.

• Now run Git Bash from your start menu and you have a bash shell you can work in.

• By default, you will be in your /c/User/Username folder. Changing to your My Documents folder would be prudent:

$ cd Documents

• Now, you can follow the Unix instructions (skipping the installation step)
Working off campus (OSX)

- Install Git if you don’t have it yet: [http://git-scm.com/download](http://git-scm.com/download)
- Start a new terminal window (Apple-Space and then type Terminal)
- You will be in your /users/userid folder on your computer. It would make sense to change to your documents folder

  $ cd Documents

- Now, you can follow the Unix instructions (skipping the installation step)
Git repository hierarchy

Bare repository
~/se101/git/se101.git
on tux

Working copy (local)
~/se101/git/workspace
on Tux

Working copy (remote)
In whatever folder you choose
On your laptop or computer at home

***Remember to use push and pull commands to keep working copies synchronized***
Advanced Git Techniques

• Reverting committed changes
  • If you committed something by accident and you want to undo your changes

• Reverting uncommitted changes
  • If you changed something that you haven’t committed but want the latest version in the repository back.
Reverting a commit

Suppose we have made a mistake and we want to revert our last commit. The latest commit is the HEAD commit. We can revert it using

```
$ git revert -n HEAD
```

If we want to revert a specific revision, we try:

```
$ git revert -n [REVISION_ID]
```

Where [REVISION_ID] is the first few characters from the unique identifier listed in the log (for example 0071e80 was my latest commit)

The –n switch tells git not to immediately commit it.
Suppose we have made a mistake, but we haven’t committed it yet. We can undo local changes all the changes we have made since our last commit using two different commands:

Revert all uncommitted changes:

$ git reset --hard HEAD

Revert a specific file:

$ git checkout test_file
Using Eclipse

- Set your Eclipse workspace to the path of your cloned working copy of the repository:
  /home(userid)/se101/git/workspace
- Go to File->Switch workspace if not prompted.
Using Eclipse

• Create a new Java Project for each exercise:

  • For assignment 1, call your projects:
    • A1_2_17 for Exercise 2.17 from assignment 1,
    • A1_2_20 for Exercise 2.30 from assignment 1,
    • And so on for all remaining assignments.

  • For lab 2, call your projects:
    • L2_1 for the first exercise from lab 1,
    • L2_2 for the second exercise from lab 2,
    • And so on for all remaining labs.

*** If you do not name your labs and assignments correctly, we will not grade them! ***
Starting the first assignment

- A folder will be created for each of the projects you create in Eclipse.
- You will have five different folders/projects:
- Each folder will contain a bin folder with your .class files in it and a src folder with your .java files in it
- Always add and commit your .java files

$ git add *.java
UML Diagrams

- Some assignments require UML diagrams.
- You can create UML diagrams in any program of your choosing (ArgoUML, Visio, ObjectAid, etc.)
- UML diagrams should be put in the doc subfolder for the corresponding exercise.
- For example, the UML diagram for assignment 1, problem 2.17 would go in A1_2_17/doc/
- Always submit a .png image of your UML diagram
- Always add and commit your UML files
  
  $ git add doc/*