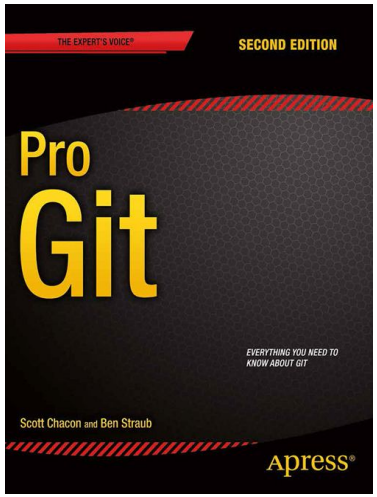
Git Command Line

Work without a GUI

Attribution

- Slides attributed to Dr. Cyndi Rader. The original version can be found at http://eecs.mines.edu/Courses/csci306/ChapterNotes.html
- Figures and info for this lecture come primarily from Pro Git, available: http://www.git-scm.com/book/en/v2
- Material is used in accordance with the Creative Commons Attribution Non Commercial Share Alike 3.0 license.



Book available online for free at: http://git-scm.com/book/en/v2

Setting up Git

Installation and configuration

Install Git

- Instructions: http://git-scm.com/book/en/v2/Getting-Started-Installing-Git

- Start Git and do basic configuration
  - $ git config --global user.name "Your name"
  - $ git config --global user.email john.doe@example.com
  - $ git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession –noPlugin"

  - $ git config --list (or git config –l)

Simple command line

- Open Git Bash terminal
- Where am I??
  - $ pwd – print working directory
- How do I move around??
  - $ cd – change directory
  - $ cd .. – change to parent directory
  - $ cd ~/Documents # My Documents directory in Win
- What files are there?
  - $ ls – list
  - $ ls - l – list with details

Git Bash command line

- Use up/down arrow to select/repeat commands
- Click on icon in top left corner to change properties of the command window OR edit (e.g. copy, paste)
  - Ins: paste
  - Shift-Ins: copy-paste selection within Git Bash window
- May be able to use tab for completion

- If output of a command (e.g. git log) takes up more than one screen
  - space: scroll down next page
  - q: return to the command line

Getting started using Git command line

NetBeans project repository

Initialising repository

- Create a Java Project with NetBeans (mine is GitIntro)
- $ cd into project directory
- Create a class (mine is HelloGit)
- $ git init

```
$ cd ~/Documents/NetBeansProjects

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects
$ cd GitIntro

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects/GitIntro
$ ls -l
total 9
-rw-r--r-- 1 hannutam 1049089 3609 Jan 21 15:54 build.xml
-rw-r--r-- 1 hannutam 1049089   85 Jan 21 15:54 manifest.mf
drwxr-xr-x 1 hannutam 1049089    0 Jan 21 15:54 nbproject/
drwxr-xr-x 1 hannutam 1049089    0 Jan 21 15:55 src/

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects/GitIntro
$ git init
Initialized empty Git repository in C:/Users/hannutam/Documents/NetBeansProjects
/GitIntro/.git/
```

.gitignore

- It's important to tell Git what files you do *not* want to track
- Temp files, executable files, etc. do not need version control (and can cause major issues when merging!)
- https://help.github.com/articles/ignoring-files
- Place .gitignore in the root of repo

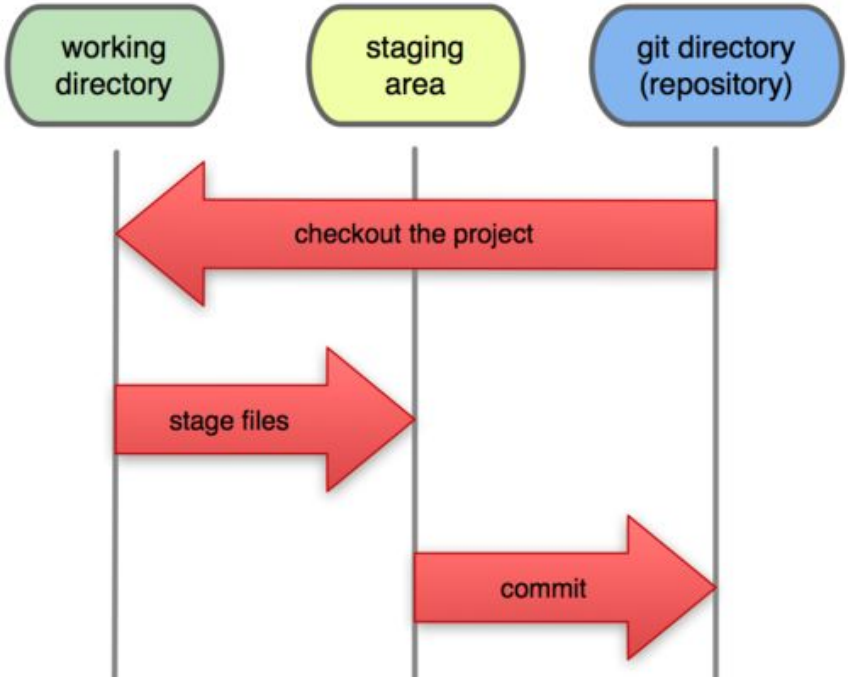
.gitignore – Example for NetBeans project

- # Class Files
- *.class
- # Package Files
- *.jar
- *.war
- *.ear
- # NetBeans specific
- nbproject/private/
- build/
- nbbuild/
- dist/
- nbdist/
- nbactions.xml
- .nb-gradle/

More useful gitignores:
https://github.com/github/gitignore/tree/master/Global

Note for distributed projects:
You still have some NetBeans specific files, which may not be needed by developers using other IDE (e.g. Eclipse)

Local Operations revisited



Initializing	Putting a directory under version control	$ cd <dir> $ git init
Staging	Telling Git what files to include in the next commit	$ git add
Committing	Storing the staged snapshot	$ git commit
Checking out	Retrieving a stored snapshot to working directory	$ git checkout

What's the status?

$ git status

```
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        build.xml
        manifest.mf
        nbproject/
        src/

nothing added to commit but untracked files present (use "git add" to track)
```

- Nothing tracked yet.
- Get used to reading helpful messages

Let's track a file

`$ git add` – tells git to *track* a file

```
$ git add src/*

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects/GitIntro (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   src/HelloGit.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        build.xml
        manifest.mf
        nbproject/
```

Stage all files

$ git add

```
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory.

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects/GitIntro (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   build.xml
        new file:   manifest.mf
        new file:   nbproject/build-impl.xml
        new file:   nbproject/genfiles.properties
        new file:   nbproject/project.properties
        new file:   nbproject/project.xml
        new file:   src/HelloGit.java
```

- Note use of . in git add
- Often desirable to have no untracked files

Commit the staged files

$ git commit

```
$ git commit -m "Initial project version"
[master (root-commit) 265148d] Initial project version
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory.
 8 files changed, 1621 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 build.xml
 create mode 100644 manifest.mf
 create mode 100644 nbproject/build-impl.xml
 create mode 100644 nbproject/genfiles.properties
 create mode 100644 nbproject/project.properties
 create mode 100644 nbproject/project.xml
 create mode 100644 src/HelloGit.java

hannutam@ETY-B227-HANNUT MINGW64 ~/Documents/NetBeansProjects/GitIntro (master)
```

- When you commit, you must provide a comment
  - if you forget, Git will open a text editor so you can write one
- If you make a typo in the message: $ git commit --amend

What if you change a file?

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/HelloGit.java

no changes added to commit (use "git add" and/or "git commit -a")
```

Notice more helpful hints Git provides.
You could add to the staging area, OR add & commit in one step.

```
$ git commit -am "Saying hello to GIT"
[master 8a996be] Saying hello to GIT
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Be careful if you add to the staging area and then make more changes – the file can appear as both staged and unstaged.

You made some changes – but what did you do?

`$ git diff`

```
$ git diff
diff --git a/src/HelloGit.java b/src/HelloGit.java
index edeff09..78ad014 100644
--- a/src/HelloGit.java
+++ b/src/HelloGit.java
@@ -3,6 +3,7 @@ public class HelloGit {

        public static void main(String[] args) {
                System.out.println("Hello Git!");
+               System.out.println("It's good to know you!");

        }
```

`git diff` command compares your working directory with your staging area. These are the changes that are not yet staged.

Comparing in many ways

`$ git diff`	Working directory vs. stage
`$ git diff --staged`	Stage vs. latest commit
`$ git diff HEAD`	Working directory vs. latest commit Note: HEAD points to the latest commit

Note the dash convention for command options:

- `-` One dash for one letter
- `--` Two dashes for longer

Some options have both versions:

```
$ git commit --message
$ git commit -m
```

What if I remove a file?

A new file added not committed

```
$ git add src/HelloWorld.java

crader_a@EECS-LAPTOP-02 /c/csm_classes/cs306/JavaCode/Practice/GitIntro (master)

$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   src/HelloWorld.java
```

Now I remove HelloWorld.java directly from the directory

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   src/HelloWorld.java

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    src/HelloWorld.java
```

removal, continued

$ git rm [file]

```
$ git rm src/HelloWorld.java
rm 'src/HelloWorld.java'

crader_a@EECS-LAPTOP-02 /c/csm_classes/cs306/JavaCode/Practice/GitIntro (master)

$ git status
On branch master
nothing to commit, working directory clean
```

- This removes the file from being tracked
- If you've already committed, the file is still in the database
- If you delete a file from within NetBeans, it does the `git rm` for you
    - Git support in NetBeans since v. 7.4

Viewing commit history

$ git log

```
$ git log
commit db07dba2ea40c7fcaf3f6cba5bac957621052341
Author: Cyndi Rader <crader@mines.edu>
Date:   Mon Jun 9 17:24:43 2014 -0600

    Nice message to git

commit 8a996be50f3d433b267d944c5062c1754034b4a7
Author: Cyndi Rader <crader@mines.edu>
Date:   Mon Jun 9 17:21:05 2014 -0600

    Saying hello to GIT

commit eaf901e7da8f5fb7afd0230aaa2c3dda44dd7d98
Author: Cyndi Rader <crader@mines.edu>
Date:   Mon Jun 9 17:19:13 2014 -0600
```

There are many useful options for git log, see next slide.

git log options

- Can specify a format, such as:
  - $ git log --oneline
  - MORE options, see documentation
- Can filter, such as:
  - $ git log --since=2.weeks
  - includes filters like –since, --after, --until, --before, --author etc
- Can redirect output to file, such as:
  - $ git log >> gitlog.txt

Undoing changes

Enter the most confusing part of Git !

Undoing changes on file level

- The output of git status helps with simple file level undos
- Unstaging a staged file
  - $ git reset [commit] [file]
  - $ git reset HEAD main.java
- Undoing changes in a file in the working directory
  - $ git checkout [commit] [file]
  - $ git checkout HEAD main.java
  - If [commit] is omitted, copies the file from the stage
  - If [file] is omitted, all the files are updated to match the commit
  - Danger: overwrites the file(s) and you will lose all changes you have made!

Undoing changes on commit level

- Redo the latest commit
  - $ git commit --amend
  - Combines all staged changes with the latest commit and replaces the latest commit with the resulting snapshot
  - E.g. to add some files or change the commit message
- Undo a commit by creating a *new* commit
  - $ git revert [commit]
  - This is a safe way to undo changes, as it has no chance of re-writing the commit history

Undoing changes on commit level

- Undo commits by deleting them
  (move the tip of a branch and HEAD to a different commit)
  - $ git reset [commit] --option
    - --soft – The stage and working directory are not altered
    - --mixed – The stage is updated to match the commit, but the working directory is not affected (default option).
    - --hard – The stage and the working directory are both updated to match the specified commit

- Very Important Rule:
  Undo changes only in the local repo, never in the one shared with other developers!

More information on undoing changes

- Basic
  - https://www.atlassian.com/git/tutorials/undoing-changes/
  - http://git-scm.com/book/en/v2/Git-Basics-Undoing-Things
- Advanced
  - https://www.atlassian.com/git/tutorials/resetting-checking-out-and-reverting/
  - http://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified

Summaries

Local repo and Git command line

Summary of getting started (local repo)

- Create a new Java Project in NetBeans
- Bring up Git Bash
- cd to your directory
- git init
- create a .gitignore
- add .gitignore (git .gitignore)
- commit (git commit –m "Initial commit with NetBeans setup")
- …
- create a new class
- add that class (git add src/*)
- commit (git commit –m "Add …")

Summary of basic git commands

- git init (creates local .git repo)
- git add src/* (tracks all files in source directory)
- git commit -m "Initial commit" (adds files to repo)
- git status (see what files have been modified, etc.)
- git diff (see what changes you've made to files)
- git log (see list of commits)
- git checkout (check out files, commits, or branches)
- git revert (undo commit by appending a new commit)
- git reset (remove commit, undo changes in the staging area or in the working directory)