Git Introduction

Version control with Git

Basic Intro to Git

- We will:
  - Discuss how Git differs from Subversion
  - Discuss the basic Git model
  - Pull/clone files from a repository on github
  - Edit files in your own local Git repo
  - Push files to a repo on github

Git Resources

- At the command line: (where verb = config, add, commit, etc.)

  $ git help <verb>

  $ git <verb> --help

  $ man git-<verb>

- Free on-line book: http://git-scm.com/book
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Reference page for Git: http://gitref.org/index.html
- Git website: http://git-scm.com/

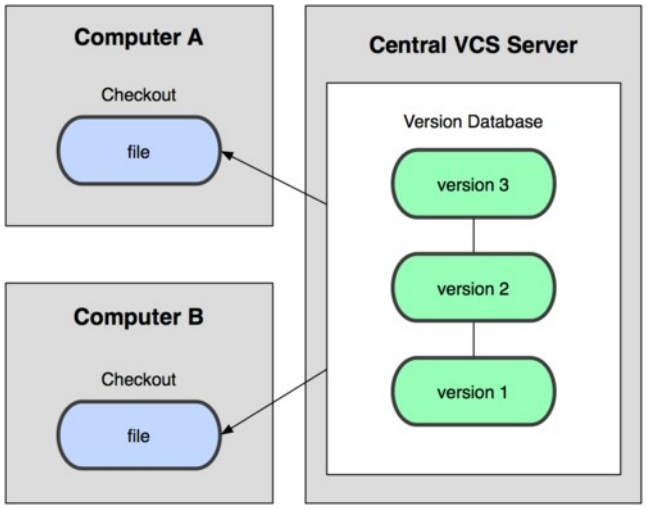
- Git for Computer Scientists

Git History

- Came out of Linux development community
- Linus Torvalds, 2005
- Initial goals:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects like Linux efficiently

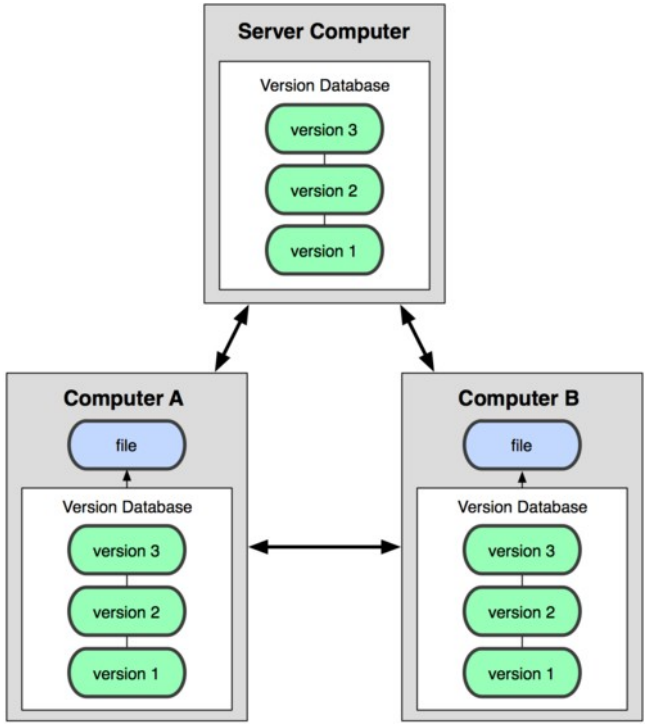
Git uses a distributed model

Centralized Model



(CVS, Subversion, Perforce)

Distributed Model

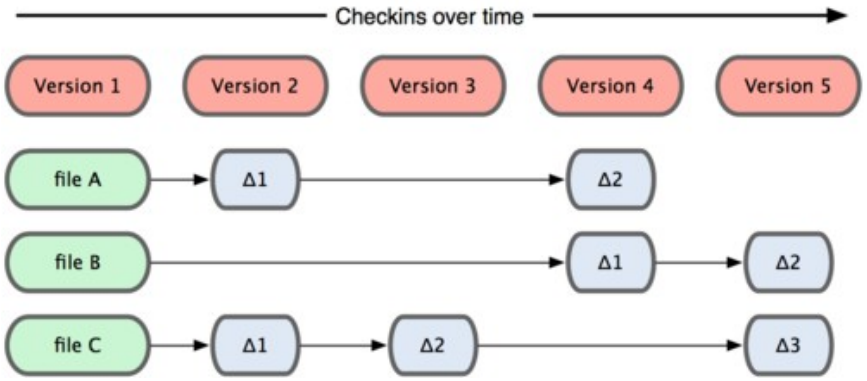


(Git, Mercurial)

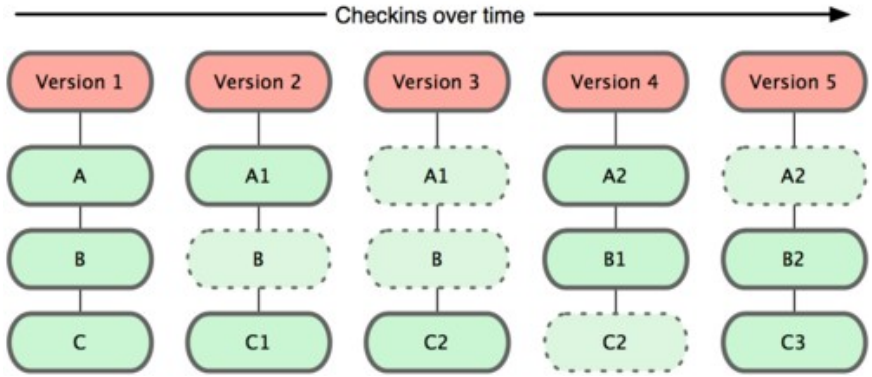
Result: Many operations are local

Git takes snapshots

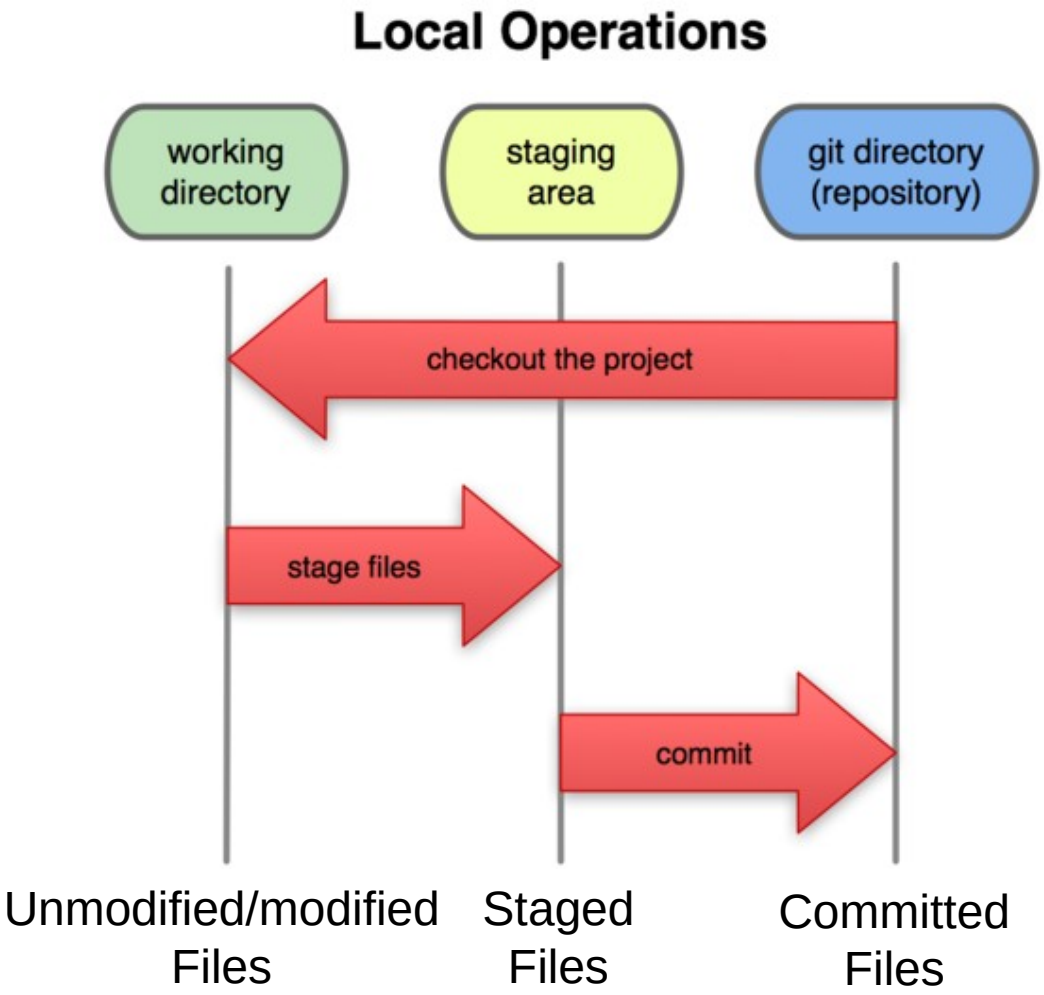
Subversion



Git

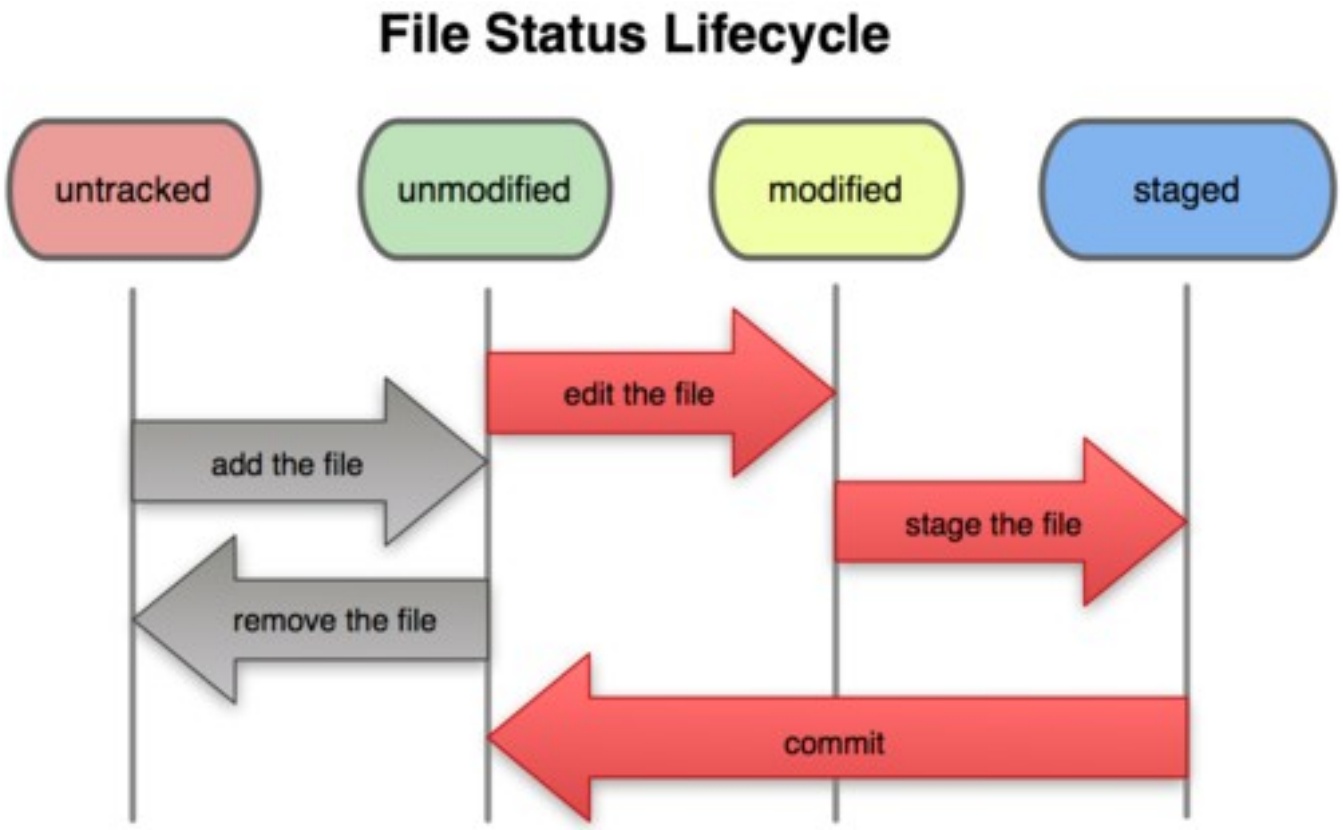


A Local Git project has three areas



Note: working directory sometimes called the “working tree”, staging area sometimes called the “index”.

Git file lifecycle



Basic Workflow

Basic Git workflow:

1. **Modify** files in your working directory.
2. **Stage** files, adding snapshots of them to your staging area.
3. Do a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

- Notes:
  - If a particular version of a file is in the **git directory**, it's considered **committed**.
  - If it's modified but has been added to the **staging area**, it is **staged**.
  - If it was **changed** since it was checked out but has not been staged, it is **modified**.

Aside: So what is github?

- GitHub.com is a site for online storage of Git repositories.
- Many open source projects use it, such as the Linux kernel.
- You can get free space for open source projects or you can pay for private projects.

Question: Do I have to use github to use Git?

Answer: No!

- you can use Git completely locally for your own purposes, or
- you or someone else could set up a server to share files, or
- you could share a repo with users on the same file

Install git

Installing on Linux

Check:-

$ git –version

$ sudo apt install git-all

Get ready to use Git!

1. Set the name and email for Git to use when you commit:

```
$ git config --global user.name “Abcd”
$ git config --global user.email “abc@xyz.com”
```

1. You can call **`git config --list`** to verify these are set.
2. These will be set globally for all Git projects you work with.
3. $ git <verb> --help

or

$ git help <verb>

Create a local copy of a repo

Two common scenarios:
(only do one of these)

To **clone an already existing repo** to your current directory:

```
$ git clone <url>
[local dir name]
```

This will create a directory named *local dir name*, containing a working copy of the files from the repo, and a **.git** directory (used to hold the staging area and your actual repo)

To **create a Git repo** in your current directory:

```
$ git init
```

This will create a **.git** directory in your current directory.

Then you can commit files in that directory into the repo:

```
$ git add file1.java
$ git commit –m “initial project version”
```

Git commands

command	description
`git clone` ***`url [dir]`***	copy a git repository so you can add to it
`git add` ***`files`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init`, `reset`, `branch`, `checkout`, `merge`, `log`, `tag`	

Committing files

- The first time we ask a file to be tracked, *and* **every time before we commit a file** we must add it to the staging area:

```
$ git add hello.html
```

This takes a snapshot of these files at this point in time and adds it to the staging area.

- To move staged changes into the repo we commit:

```
$ git commit –m “added some lines”
```

Note: To unstage a change on a file before you have committed it:

```
$ git reset HEAD  --  filename
```

Note: To unmodify a modified file:

```
$ git checkout  --  filename
```

Status and Diff

- To view the **status** of your files in the working directory and staging area:

```
$ git status
```
or
```
$ git status –s
```
(`-s` shows a short one line version similar to svn)

- To see what is modified but unstaged:

```
$ git diff
```

- To see staged changes:

```
$ git diff --cached
```

Getting Started

1. Installation
2. Find/open command line
3. Try a few git commands

Installation

- **Windows:-**

  open web browser git-scm.com

Download and follow the instructions

- **Linux :-**
- wget http://kernel.org/pub/software/scm/git/git-1.7.6.tar.bz2
- tar xvfj git-1.7.6.tar.bz2
- cd git-1.7.6
- ./configure
- make
- make install

Git Commands

- Open git-bash
- ~ git - -version
- ~ git config - -global user.name “your name”
- ~ git config - -global user.email “your@emailaddress”
- ~ git config –list → check if set
- ~ mkdir gittest → create a directory in your pwd
- ~ cd gittest
- ~ mkdir hello world
- ~ cd hello world
- ~ pwd → check your present working directory
- ~ git init →initializes git in your computer
- ~ touch index.html → open the html file and add some content

Cntd…

- ~ git add index.html → staging starts
- ~ git status → check the status now
- ~ git commit –m ‘my first commit’
- ~ git checkout -- . →commit the file to your local repo
- Now clone from the following and open it in a web browser
- ~ git clone https://github.com/LearnWebCode/welcome-to-git.git

Git Commands cntd..

$ git init

$ git mkdir gittest

$ git cd gittest

$ git branch add

$ git branch

Suppose we add one more line at the end of test.txt, which is not staged till now. Then, ‘git checkout’ command can be used to remove the changes.

```
soma@soma-WIV58425E-0002:~/gittest$ gedit test.txt
soma@soma-WIV58425E-0002:~/gittest$ git status test.txt
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
soma@soma-WIV58425E-0002:~/gittest$ git diff
diff --git a/test.txt b/test.txt
index 8eb4d91..a5b90d3 100644
--- a/test.txt
+++ b/test.txt
@@ -7,4 +7,4 @@ second time added
 lines
 after commit
 added lines
-
+checkout command example
soma@soma-WIV58425E-0002:~/gittest$ git checkout test.txt
Updated 1 path from the index
soma@soma-WIV58425E-0002:~/gittest$ git status test.txt
On branch master
nothing to commit, working tree clean
soma@soma-WIV58425E-0002:~/gittest$ gedit test.txt
```

The git checkout command operates upon three distinct entities: files, commits, and branches.

```
soma@soma-WIV58425E-0002:~$ git status hello.c
On branch master

No commits yet

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.c

soma@soma-WIV58425E-0002:~$ git checkout -b test
Switched to a new branch 'test'
soma@soma-WIV58425E-0002:~$ git status hello.c
On branch test

No commits yet

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.c
```

```
soma@soma-WIV58425E-0002:~$ git diff
diff --git a/hello.c b/hello.c
index 1192013..dd15bc6 100644
--- a/hello.c
+++ b/hello.c
@@ -3,5 +3,6 @@
 int main()
 {
     printf("Hello world\n");
+    printf("now in git\n");
     return 0;
 }
soma@soma-WIV58425E-0002:~$ git branch
soma@soma-WIV58425E-0002:~$ git commit hello.c
warning: could not open directory '.dbus/': Permission denied
Aborting commit due to empty commit message.
soma@soma-WIV58425E-0002:~$ git commit hello.c -m "first commit in hello.c"
[test (root-commit) 157627f] first commit in hello.c
 1 file changed, 8 insertions(+)
 create mode 100644 hello.c
soma@soma-WIV58425E-0002:~$ git branch
* test
soma@soma-WIV58425E-0002:~$ git checkout -b master
Switched to a new branch 'master'
soma@soma-WIV58425E-0002:~$ git branch
* master
  test
```

Branching

To create a branch called experimental:

- **`$ git branch experimental`**

To list all branches: (* shows which one you are currently on)

- **`$ git branch`**

To switch to the experimental branch:

- **`$ git checkout experimental`**

Later on, changes between the two branches differ, to merge changes from experimental into the master:

- **`$ git checkout master`**
- **`$ git merge experimental`**

Note: **`git log --graph`** can be useful for showing

```
soma@soma-WIV58425E-0002:~/gittest$ git branch add
soma@soma-WIV58425E-0002:~/gittest$ git branch
  add
  checkout
* master
  test
soma@soma-WIV58425E-0002:~/gittest$ git branch -d add
Deleted branch add (was 3083198).
soma@soma-WIV58425E-0002:~/gittest$ git branch
  checkout
* master
  test
soma@soma-WIV58425E-0002:~/gittest$ git branch add
soma@soma-WIV58425E-0002:~/gittest$ git branch
  add
  checkout
* master
  test
soma@soma-WIV58425E-0002:~/gittest$ git checkout add
Switched to branch 'add'
soma@soma-WIV58425E-0002:~/gittest$ git branch
* add
  checkout
  master
  test
soma@soma-WIV58425E-0002:~/gittest$ gedit test.txt
soma@soma-WIV58425E-0002:~/gittest$ git commit -am test.txt
[add f47f47e] test.txt
 1 file changed, 1 insertion(+)
soma@soma-WIV58425E-0002:~/gittest$ git checkout master
```

```
soma@soma-WIV58425E-0002:~/gittest$ git merge add
Updating 3083198..f47f47e
Fast-forward
 test.txt | 1 +
 1 file changed, 1 insertion(+)
```

Pulling and Pushing

Good practice:

1. **Add** and **Commit** your changes to your local repo
2. **Pull** from remote repo to get most recent changes (fix conflicts if necessary, add and commit them to your local repo)
3. **Push** your changes to the remote repo

Pulling and Pushing

To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:

```
$ git pull origin master
```

To push your changes from your local repo to the remote repo:

```
$ git push origin master
```

Notes: **`origin`** = an alias for the URL you cloned from

`master` = the remote branch you are pulling from/pushing to, (the local branch you are pulling to/pushing from is your current branch)

Note: On attu you will get a Gtk-warning, you can ignore this.

Pulling and Pushing

```
soma@soma-WIV58425E-0002:~/pushcmd2$ cd sample
soma@soma-WIV58425E-0002:~/pushcmd2/sample$ gedit test2.txt
soma@soma-WIV58425E-0002:~/pushcmd2/sample$ git commit test2.txt -m "push commit 1"
error: pathspec 'test2.txt' did not match any file(s) known to git
soma@soma-WIV58425E-0002:~/pushcmd2/sample$ git add test2.txt
soma@soma-WIV58425E-0002:~/pushcmd2/sample$ git commit test2.txt -m "push commit 1"
[master 9a465cf] push commit 1
 1 file changed, 2 insertions(+)
 create mode 100644 test2.txt
```

Pulling and Pushing

```
soma@soma-WIV58425E-0002:~/pushcmd2$ git remote rm origin
soma@soma-WIV58425E-0002:~/pushcmd2$ git remote add origin git@github.com:soma-g/sample.git
soma@soma-WIV58425E-0002:~/pushcmd2$ git push -u origin master
```

```
soma@soma-WIV58425E-0002:~/pushcmd2/sample$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:soma-g/sample.git
   59d53fc..9a465cf  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

SVN vs. Git

- SVN:
  - central repository approach – the main repository is the only “true” source, only the main repository has the complete file history
  - Users check out local copies of the current version
- Git:
  - Distributed repository approach – every checkout of the repository is a full fledged repository, complete with history
  - Greater redundancy and speed
  - Branching and merging repositories is more heavily used as a result

Some useful links

How to fix github permission denied publickey fatal could not read from remote repository?

https://www.youtube.com/watch?v=lV5mrUYsucU