

Introduction to Git and Github

Computing in Optimization and Statistics: Lecture 1
Jackie Baek

MIT

January 10, 2017

What is git and GitHub?

- ▶ **git** is a *version control system*.
 - ▶ Other version control systems include mercurial, svn, perforce.
 - ▶ git is modern (2005) and most popular.
- ▶ **GitHub** is a service that allows you to host projects using git.

What is a version control system?

- ▶ Software that stores "snapshots" of a project over time.
- ▶ Can be used for projects big or small, long-term or short-term.

Why should I learn it?

- ▶ Everyone uses it.
 - ▶ We'll be using it in this class.
- ▶ Backup (in the cloud).
- ▶ Versioning with fine granularity.
- ▶ Collaboration.
 - ▶ But useful even when working by yourself.

Why should I learn it?

- ▶ Everyone uses it.
 - ▶ We'll be using it in this class.
- ▶ Backup (in the cloud).
- ▶ Versioning with fine granularity.
- ▶ Collaboration.
 - ▶ But useful even when working by yourself.

Can't we just use Dropbox?

Why should I learn it?

- ▶ Everyone uses it.
 - ▶ We'll be using it in this class.
- ▶ Backup (in the cloud).
- ▶ Versioning with fine granularity.
- ▶ Collaboration.
 - ▶ But useful even when working by yourself.

Can't we just use Dropbox?

- ▶ git gives finer granularity: files vs. lines within a file.
- ▶ This granularity is essential when writing code.

Terminology

- ▶ **repository (repo)**: the project that contains all files.

Terminology

- ▶ **repository (repo):** the project that contains all files.
- ▶ **commit:** one snapshot of the repository.

Terminology

- ▶ **repository (repo):** the project that contains all files.
- ▶ **commit:** one snapshot of the repository.
 - ▶ **log:** list of all commits.

Terminology

- ▶ **repository (repo):** the project that contains all files.
- ▶ **commit:** one snapshot of the repository.
 - ▶ **log:** list of all commits.
 - ▶ **HEAD:** the currently checked out commit.

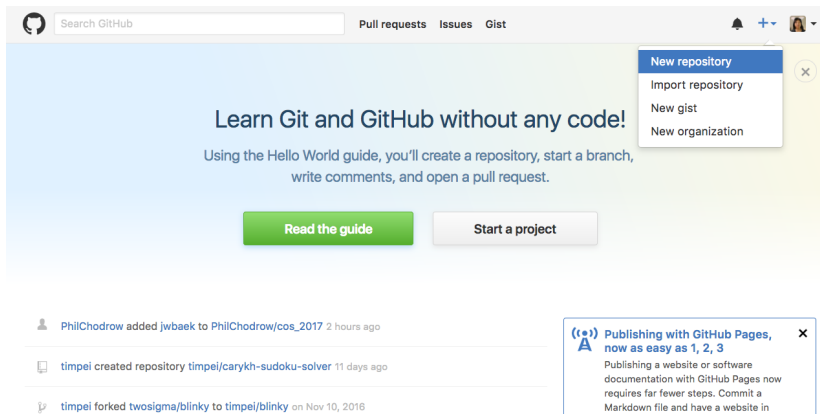
Terminology

- ▶ **repository (repo):** the project that contains all files.
- ▶ **commit:** one snapshot of the repository.
 - ▶ **log:** list of all commits.
 - ▶ **HEAD:** the currently checked out commit.
- ▶ **local:** repository sitting on your local machine.
- ▶ **remote:** repository sitting on a remote server (i.e. GitHub).

Terminology

- ▶ **repository (repo):** the project that contains all files.
- ▶ **commit:** one snapshot of the repository.
 - ▶ **log:** list of all commits.
 - ▶ **HEAD:** the currently checked out commit.
- ▶ **local:** repository sitting on your local machine.
- ▶ **remote:** repository sitting on a remote server (i.e. GitHub).
- ▶ **pull:** grab changes from remote to local.
- ▶ **push:** update remote with local changes.

Creating a new repository



The screenshot shows the GitHub homepage. At the top, there is a search bar with the text "Search GitHub", and navigation links for "Pull requests", "Issues", and "Gist". On the right side, there are icons for a notification bell, a plus sign, and a user profile. A dropdown menu is open, showing options: "New repository" (highlighted in blue), "Import repository", "New gist", and "New organization".

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)

Recent Activity:

- PhilChodrow added [jwbaek](#) to [PhilChodrow/cos_2017](#) 2 hours ago
- [timpei](#) created repository [timpei/carykh-sudoku-solver](#) 11 days ago
- [timpei](#) forked [twosigma/blinky](#) to [timpei/blinky](#) on Nov 10, 2016

Advertisement: Publishing with GitHub Pages, now as easy as 1, 2, 3. Publishing a website or software documentation with GitHub Pages now requires far fewer steps. Commit a Markdown file and have a website in

Creating a new repository

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner





Repository name

playground ✓

Great repository names are short and memorable. Need inspiration? How about **probable-meme**.

Description (optional)

playground for Computing in Optimization and Statistics

-  **Public**
Anyone can see this repository. You choose who can commit.
-  **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ ⓘ

Create repository

Creating a new repository

The screenshot shows the GitHub interface for a repository named 'playground' by user 'jwbaek'. At the top, there is a search bar and navigation links for 'Pull requests', 'Issues', and 'Gist'. The repository name 'jwbaek / playground' is displayed, along with statistics: 1 Unwatch, 1 Star, and 0 Forks. Below this, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The repository description is 'playground for Computing in Optimization and Statistics', with an 'Edit' button. A summary bar shows '1 commit', '1 branch', '0 releases', and '1 contributor'. Action buttons include 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A commit history table shows an 'Initial commit' by 'jwbaek' with a 'README.md' file, committed 'a minute ago'. The main content area shows the 'README.md' file content, which includes the title 'playground' and the same repository description.

This repository Search

Pull requests Issues Gist

jwbaek / playground

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

playground for Computing in Optimization and Statistics Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

jwbaek Initial commit	Latest commit 0cdaa10 a minute ago
README.md Initial commit	a minute ago

README.md

playground

playground for Computing in Optimization and Statistics

Cloning a repository

```
$ git clone <URL>
```

- ▶ Go to any repository and copy the URL
- ▶ This will create a new directory with the same name as the repository name and clone the repo there.

```
$ git clone https://github.com/jwbaek/playground
```


Cloning a repository

```
$ git clone <URL>
```

- ▶ Go to any repository and copy the URL
- ▶ This will create a new directory with the same name as the repository name and clone the repo there.

```
$ git clone https://github.com/jwbaek/playground
```

```
$ git config --global core.editor "nano"
```

Let's make some changes

- ▶ Create a new file called `new_file.txt`
 - ▶ Add "This is a new file"
- ▶ Modify `existing_file.txt`
 - ▶ interesting → uninteresting

Let's make some changes

- ▶ Create a new file called `new_file.txt`
 - ▶ Add "This is a new file"
- ▶ Modify `existing_file.txt`
 - ▶ `interesting` → `uninteresting`

```
$ cd playground
$ nano new_file.txt
  This is a new file
$ nano existing_file.txt
  interesting -> uninteresting
```

See what changed

```
$ git diff
```

- ▶ Shows what changed since the last commit

Checking the status of our files

```
$ git status
```

On branch master

Your branch is up-to-date with 'origin/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: existing_file.txt

Untracked files:

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

new_file.txt

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

File states

- ▶ Git will notice any file in the directory of the repository.

File states

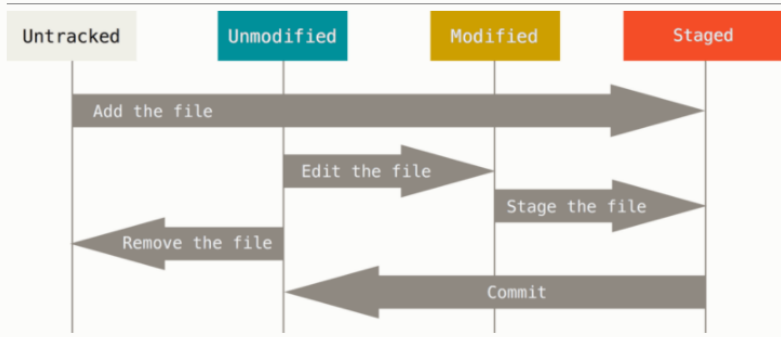
- ▶ Git will notice any file in the directory of the repository.
- ▶ A file is either **untracked** or **tracked**.

File states

- ▶ Git will notice any file in the directory of the repository.
- ▶ A file is either **untracked** or **tracked**.
- ▶ A tracked file may be:
 1. **Unmodified:** No changes since the last commit.
 2. **Modified:** Changes have been made to it since the last commit.
 3. **Staged:** Changes will be committed in the next commit.

File states

- ▶ Git will notice any file in the directory of the repository.
- ▶ A file is either **untracked** or **tracked**.
- ▶ A tracked file may be:
 1. **Unmodified**: No changes since the last commit.
 2. **Modified**: Changes have been made to it since the last commit.
 3. **Staged**: Changes will be committed in the next commit.



Staging files

```
$ git add <filepath>
```

- ▶ Any *untracked* or *modified* file that is added will be *staged*.
- ▶ Each such file will be included in the next commit.

Staging files

```
$ git add <filepath>
```

- ▶ Any *untracked* or *modified* file that is added will be *staged*.
- ▶ Each such file will be included in the next commit.

```
$ git add new_file.txt  
$ git add existing_file.txt
```

Use git add to either:

- ▶ Add a new file to the repository (untracked → staged)
- ▶ Record a change that you made to an existing file (modified → staged)

git commit

```
$ git commit -m <commit message>
```

- ▶ This creates a new snapshot of our repository with all changes that we have staged.

git commit

```
$ git commit -m <commit message>
```

- ▶ This creates a new snapshot of our repository with all changes that we have staged.

```
$ git commit -m "Added new interesting file."
```

- ▶ This new snapshot (commit) is saved in our local repository.
- ▶ This *does not* push our changes to the remote repository (GitHub).

git log

```
$ git log
```

```
commit ca82a6dff817ec66f44342007202690a93763949
```

```
Author: Jackie Baek <baek@mit.edu>
```

```
Date: Mon Mar 17 21:52:11 2008 -0700
```

```
    this is my commit message
```

```
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
```

```
Author: Scott Chacon <schacon@gee-mail.com>
```

```
Date: Sat Mar 15 16:40:33 2008 -0700
```

```
    removed unnecessary test
```

```
commit a11bef06a3f659402fe7563abf99ad00de2209e6
```

```
Author: Scott Chacon <schacon@gee-mail.com>
```

```
Date: Sat Mar 15 10:31:28 2008 -0700
```

```
    first commit
```

Interacting with remote

```
$ git push
```

- ▶ Update remote repository with local commits.

```
$ git pull
```

- ▶ Updates local repository with remote commits.

Merging

Merging

- ▶ When we 'git pull', git fetches the remote repository from GitHub and *merges* the new remote updates with our local repository.
- ▶ Even if both remote and local modified the same file, git is *usually* able to correctly merge the two copies.
- ▶ We get a **merge conflict** if both parties modified the *same parts of the same file*.

Merging

```
$ git pull
```

```
remote: Counting objects: 3, done.  
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.  
https://github.com/jwbaek/playground  
    50c8ec4..0c13bac  master    -> origin/master  
Auto-merging existing_file.txt  
CONFLICT (content): Merge conflict in existing_file.txt  
Automatic merge failed; fix conflicts and then  
commit the result.
```

Resolving Merge Conflicts

```
$ cat existing_file.txt
```

Resolving Merge Conflicts

```
$ cat existing_file.txt
```

What an

```
<<<<<< HEAD
```

```
uninteresting
```

```
=====
```

```
fun
```

```
>>>>>> 0c13bac86a172ae60766d615f92d2b01d7bf131d
```

```
document!
```

- ▶ The markers <<<<<<, =====, >>>>>> indicate the conflict.
- ▶ The section in between the first two markers is your local change (HEAD), while the bottom section indicates the update from remote.
- ▶ Must resolve conflict manually by editing the file, making sure to get rid of the conflict markers.

Resolving Merge Conflicts

```
$ cat existing_file.txt
```

What an

```
<<<<<<< HEAD
```

```
uninteresting
```

```
=====
```

```
fun
```

```
>>>>>>> 0c13bac86a172ae60766d615f92d2b01d7bf131d
```

```
document!
```

- ▶ The markers <<<<<<<, =====, >>>>>>> indicate the conflict.
- ▶ The section in between the first two markers is your local change (HEAD), while the bottom section indicates the update from remote.
- ▶ Must resolve conflict manually by editing the file, making sure to get rid of the conflict markers.

```
$ nano existing_file.txt
```

Resolving Merge Conflicts

- ▶ After resolving conflicts, we must add the file for staging and commit again.
- ▶ Git will automatically create a commit message: "Merge branch 'master' of <https://github.com/jwbaek/playground>"

Resolving Merge Conflicts

- ▶ After resolving conflicts, we must add the file for staging and commit again.
- ▶ Git will automatically create a commit message: "Merge branch 'master' of <https://github.com/jwbaek/playground>"

```
$ git add existing_file.txt  
$ git commit
```

- ▶ At this point, we can push.

Typical Workflow

Fetch remote changes.

```
$ git pull
```

(If there are any conflicts, resolve them and commit.

```
$ git add <conflicted files>
```

```
$ git commit )
```

Make changes

Stage modified and new files.

```
$ git add <files>
```

Commit changes.

```
$ git commit -m "this is my commit message"
```

Push local changes to remote.

```
$ git push
```


Useful tips

- ▶ Google is your friend. (e.g. "How to undo merge in git".)
- ▶ Almost anything can be undone, as long as it is committed.
- ▶ Commit often, pull often.
- ▶ Each command has many options.
 - ▶ Use 'git <verb> --help' for documentation.

Thank you!