

Version Control Using Git

Justin Baker

June 15, 2020

Summer @ICERM

Motivating Version Control

Version control stores a history of changes made to a file system.

Frictionless Context Switching

- Reference changes
- Undo changes
- Combine changes

(programming, graphic design, audio design)

What Would We Want?

- Minimum Footprint Maximum Speed (Efficient for Large Data Sizes)
- Collaborations (Fully Distributed [DVCS])
- Any Change Any Time (Highly Nonlinear Development)
- Wide Software Integration (MATLAB, VSCode, Eclipse, ...)

More?

- Minimize Data Loss (Nearly All Operations are Local)
- Detectable Data Loss (Checksumming)

Downloading Git

`https://git-scm.com/downloads`

Integrate into your favorite command line (native Git Bash)

Use a GUI (native Git GUI)

Learn specific software integration

- MATLAB
- VSCode
- Eclipse

Starting a Repository

Initializing Repository

Starting From Scratch



A screenshot of a MINGW64 terminal window. The title bar shows the path "MINGW64:/c/Users/Justin/example". The terminal content shows the prompt "Justin@DESKTOP-E44VT5L MINGW64 ~" followed by the command "\$ mkdir example".

```
Justin@DESKTOP-E44VT5L MINGW64 ~  
$ mkdir example
```

Initializing Repository

Starting From Scratch

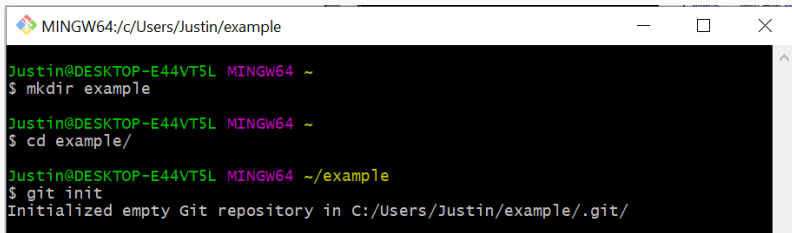


A screenshot of a Windows terminal window titled "MINGW64:/c/Users/Justin/example". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal background is black with green text. It shows two commands being executed: first, "mkdir example" and second, "cd example/". The prompt "Justin@DESKTOP-E44VT5L MINGW64 ~" is visible before each command.

```
Justin@DESKTOP-E44VT5L MINGW64 ~  
$ mkdir example  
  
Justin@DESKTOP-E44VT5L MINGW64 ~  
$ cd example/
```

Initializing Repository

Starting From Scratch



```
MINGW64:/c/Users/Justin/example

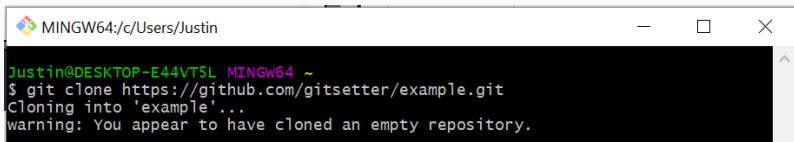
Justin@DESKTOP-E44VT5L MINGW64 ~
$ mkdir example

Justin@DESKTOP-E44VT5L MINGW64 ~
$ cd example/

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ git init
Initialized empty Git repository in C:/Users/Justin/example/.git/
```


Initializing Repository

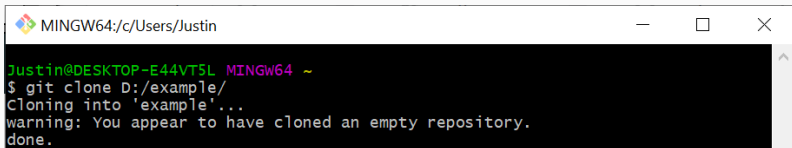
Starting From Existing Repository

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Justin'. The terminal shows a user prompt 'Justin@DESKTOP-E44VT5L MINGW64 ~' followed by the command '\$ git clone https://github.com/gitsetter/example.git'. The output shows 'Cloning into 'example'...' and a warning message: 'warning: You appear to have cloned an empty repository.'

```
MINGW64:/c/Users/Justin  
Justin@DESKTOP-E44VT5L MINGW64 ~  
$ git clone https://github.com/gitsetter/example.git  
Cloning into 'example'...  
warning: You appear to have cloned an empty repository.
```

Initializing Repository

Starting From Existing Repository

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Justin'. The terminal shows a user named Justin at a desktop named DESKTOP-E44VT5L, using the MINGW64 shell. The user has entered the command 'git clone D:/example/'. The output shows 'Cloning into 'example'...' followed by a warning: 'warning: You appear to have cloned an empty repository.' and finally 'done.'.

```
MINGW64:/c/Users/Justin  
Justin@DESKTOP-E44VT5L MINGW64 ~  
$ git clone D:/example/  
Cloning into 'example'...  
warning: You appear to have cloned an empty repository.  
done.
```

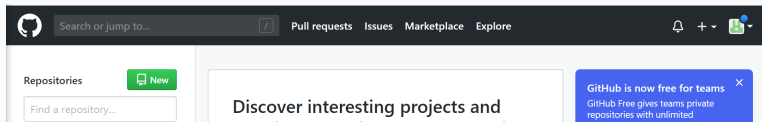
Initializing Repository

Starting From No Repository

- Make Directory (optional)
- Navigate into Directory
- `git init`

Starting From Existing Repository

- `git clone [path]`



The screenshot shows the GitHub homepage. At the top is a dark navigation bar with the GitHub logo, a search bar labeled "Search or jump to...", and links for "Pull requests", "Issues", "Marketplace", and "Explore". On the right of the bar are icons for notifications, a plus sign, and a user profile. Below the navigation bar, the left sidebar features the "Repositories" section with a "New" button and a search input labeled "Find a repository...". The main content area has a heading "Discover interesting projects and" followed by a blue promotional banner that reads "GitHub is now free for teams" and "GitHub Free gives teams private repositories with unlimited".

Search or jump to...

Pull requests Issues Marketplace Explore

Repositories

New

Find a repository...

Discover interesting projects and

GitHub is now free for teams

GitHub Free gives teams private repositories with unlimited

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner

Repository name *



gitsetter ▾

/

Great repository names are short and memorable. Need inspiration? How about **silver-system**?

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**


This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾




Add a license: **None** ▾



Create repository



[Pull requests](#)
[Issues](#)
[Marketplace](#)
[Explore](#)

[gitsetter / example](#)
Private

Unwatch 1
Star 0
Fork 0







[Code](#)
[Issues 0](#)
[Pull requests 0](#)
[Actions](#)
[Projects 0](#)
[Security 0](#)
[Insights](#)
[Settings](#)



No description, website, or topics provided.

[Manage topics](#)
[Edit](#)

8 commits
2 branches
0 packages
2 releases
MIT

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

 gitsetter merged	Latest commit b5ff869 2 hours ago	
 .gitignore	MIT LICENSE, sparse README	4 days ago
 LICENSE	MIT LICENSE, sparse README	4 days ago
 README.md	MIT LICENSE, sparse README	4 days ago
 main.m	merged	2 hours ago
 test.py	Corrected: Syntax Error	2 hours ago

 README.md
 

S@I Version Control Example

Credential Settings

`https://help.github.com/en/github/using-git/
caching-your-github-password-in-git`

Essential Files

- README.md (explain your project)

Coronavirus (Covid-19) Data in the United States

NEW: We are publishing the data behind our [excess deaths tracker](#) in order to provide researchers and the public with a better record of the true toll of the pandemic. This data is compiled from official national and municipal data for 24 countries. See the data and documentation in the [excess-deaths/](#) directory.

[[U.S. Data \(Raw CSV\)](#) | [U.S. State-Level Data \(Raw CSV\)](#) | [U.S. County-Level Data \(Raw CSV\)](#)]

The New York Times is releasing a series of data files with cumulative counts of coronavirus cases in the United States, at the state and county level, over time. We are compiling this time series data from state and local governments and health departments in an attempt to provide a complete record of the ongoing outbreak.

Since late January, The Times has tracked cases of coronavirus in real time as they were identified after testing. Because of the widespread shortage of testing, however, the data is necessarily limited in the picture it presents of the outbreak.

We have used this data to power our [maps](#) and [reporting](#) tracking the outbreak, and it is now being made available to the public in response to requests from researchers, scientists and government officials who would like access to the data to better understand the outbreak.

The data begins with the first reported coronavirus case in Washington State on Jan. 21, 2020. We will publish regular updates to the data in this repository.

- README.md (explain your project)
- LICENSE (<https://choosealicense.com/>)

MIT License

Copyright (c) [year] [fullname]

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Essentials

- README.md (explain your project)
- LICENSE (<https://choosealicense.com/>)
- .gitignore

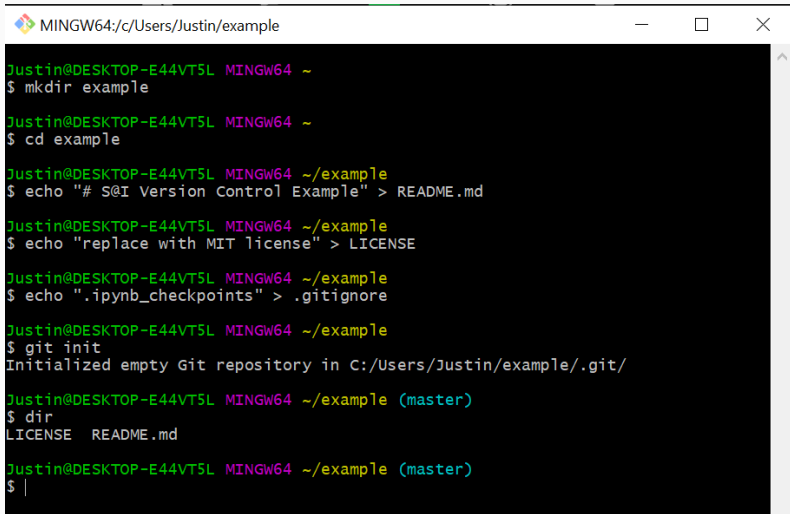
```
# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py
```

Initialization Example

Example



```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~
$ mkdir example

Justin@DESKTOP-E44VT5L MINGW64 ~
$ cd example

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo "# S@I Version Control Example" > README.md

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo "replace with MIT license" > LICENSE

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo ".ipynb_checkpoints" > .gitignore

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ git init
Initialized empty Git repository in C:/Users/Justin/example/.git/

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ dir
LICENSE  README.md

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ |
```

Example

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~
$ mkdir example

Justin@DESKTOP-E44VT5L MINGW64 ~
$ cd example

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo "# S@I Version Control Example" > README.md

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo "replace with MIT license" > LICENSE

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ echo ".ipynb_checkpoints" > .gitignore

Justin@DESKTOP-E44VT5L MINGW64 ~/example
$ git init
Initialized empty Git repository in C:/Users/Justin/example/.git/

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ dir
LICENSE  README.md

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ |

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ dir -a
.  ..  .git  .gitignore  LICENSE  README.md
```

Repository States

Modified Repository

git status

MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)

\$ git status

On branch master

No commits yet

Untracked files:

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

.gitignore

LICENSE

README.md

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

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)

\$ |

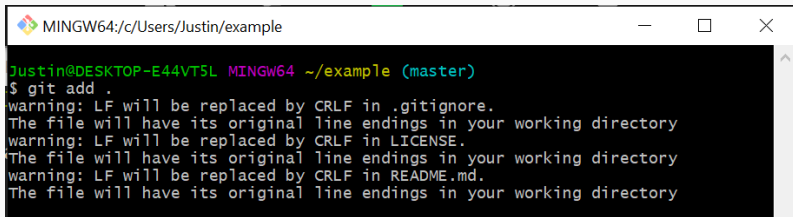
Staged Repository

`git add <filename/arg>`

Basig arg syntax

. - all files

.m - all files of type .m (.py, *.c, *.txt, ...)

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Justin/example'. The terminal shows the command 'git add .' being executed. The output consists of four lines of warnings, each indicating that a file's line endings will be converted from LF to CRLF. The files mentioned are .gitignore, LICENSE, and README.md. The prompt shows the user is in the directory ~/example on the master branch.

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in LICENSE.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
```

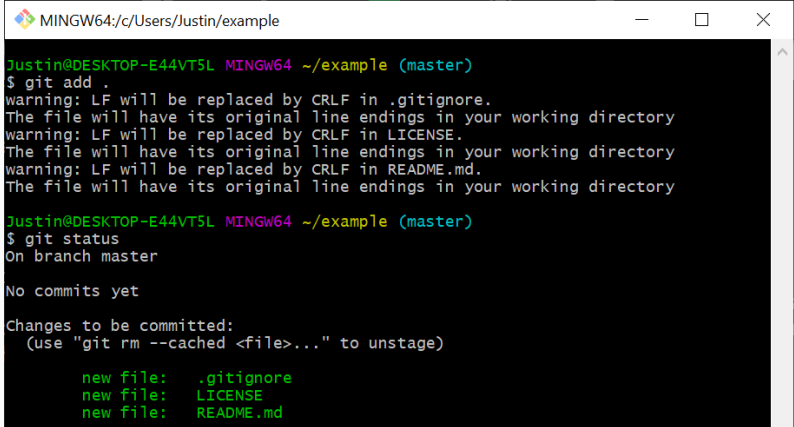
Staged Repository

`git add <filename/arg>`

Basig arg syntax

`.` - all files

`*.m` - all files of type `.m` (`*.py`, `*.c`, `*.txt`, ...)

A screenshot of a terminal window titled 'MINGW64:/c/Users/Justin/example'. The terminal shows the execution of 'git add .' which results in three warnings about line endings being replaced by CRLF in .gitignore, LICENSE, and README.md. Subsequently, 'git status' is run, showing 'On branch master' and 'No commits yet'. It then lists 'Changes to be committed:' as three new files: .gitignore, LICENSE, and README.md.

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in LICENSE.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git status
On branch master

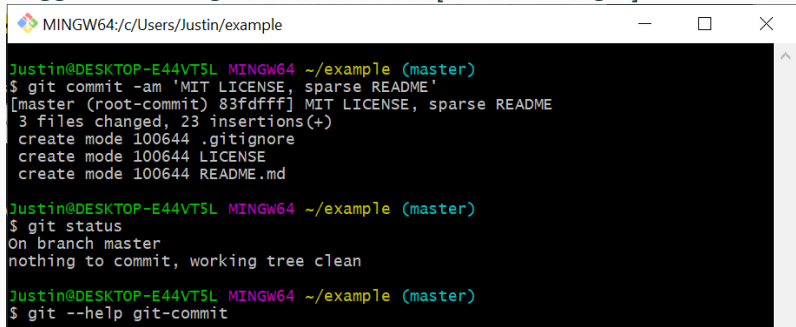
No commits yet

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

        new file:   .gitignore
        new file:   LICENSE
        new file:   README.md
```

Committed Changes

Suggested Use: `git commit -am '[outline changes]'`

A terminal window titled 'MINGW64:/c/Users/Justin/example' with standard window controls. The terminal shows a sequence of git commands and their outputs. The first command is 'git commit -am 'MIT LICENSE, sparse README'', which creates three files and commits them. The second command is 'git status', which shows the working tree is clean. The third command is 'git --help git-commit', which is partially visible.

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git commit -am 'MIT LICENSE, sparse README'
[master (root-commit) 83fdfff] MIT LICENSE, sparse README
3 files changed, 23 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
create mode 100644 README.md

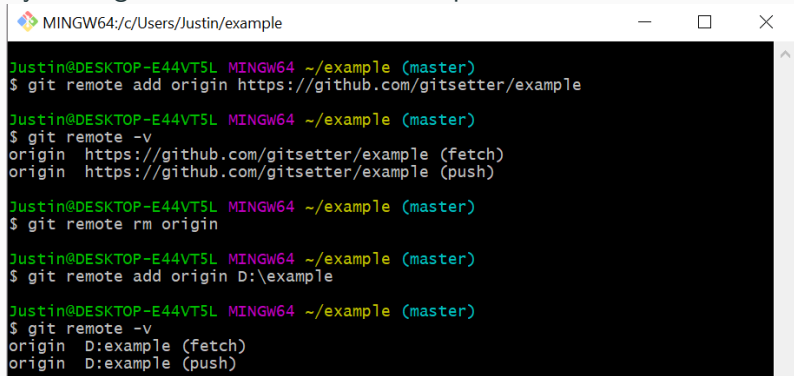
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git status
On branch master
nothing to commit, working tree clean

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git --help git-commit
```

More Options: `git --help git-commit`

Remotes

Syntax: `git remote add <name> <path>`

A terminal window titled 'MINGW64:/c/Users/Justin/example' with standard window controls. The terminal shows a series of git commands and their outputs. The prompt is 'Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)'. The commands and outputs are: 1. '\$ git remote add origin https://github.com/gitsetter/example' with no output. 2. '\$ git remote -v' with output 'origin https://github.com/gitsetter/example (fetch)' and 'origin https://github.com/gitsetter/example (push)'. 3. '\$ git remote rm origin' with no output. 4. '\$ git remote add origin D:\example' with no output. 5. '\$ git remote -v' with output 'origin D:example (fetch)' and 'origin D:example (push)'.

```
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git remote add origin https://github.com/gitsetter/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git remote -v
origin https://github.com/gitsetter/example (fetch)
origin https://github.com/gitsetter/example (push)

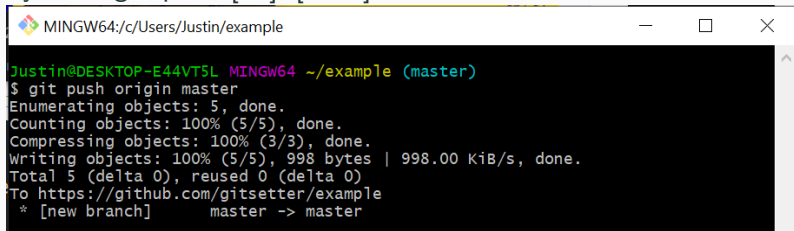
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git remote rm origin

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git remote add origin D:\example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git remote -v
origin D:example (fetch)
origin D:example (push)
```

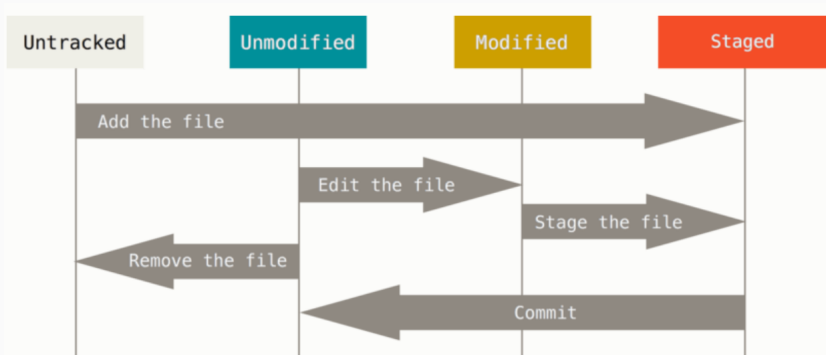
Pushing to Remote

Syntax: `git push [to] [from]`

A screenshot of a Windows terminal window titled "MINGW64:/c/Users/Justin/example". The terminal shows the execution of the command "git push origin master". The output indicates a successful push: "Enumerating objects: 5, done.", "Counting objects: 100% (5/5), done.", "Compressing objects: 100% (3/3), done.", "Writing objects: 100% (5/5), 998 bytes | 998.00 KiB/s, done.", "Total 5 (delta 0), reused 0 (delta 0)", and "To https://github.com/gitsetter/example". A final line shows "* [new branch] master -> master".

```
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 998 bytes | 998.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/gitsetter/example
* [new branch] master -> master
```

Summary



Branches

Commits are *snapshots* of the project.

Under the Hood

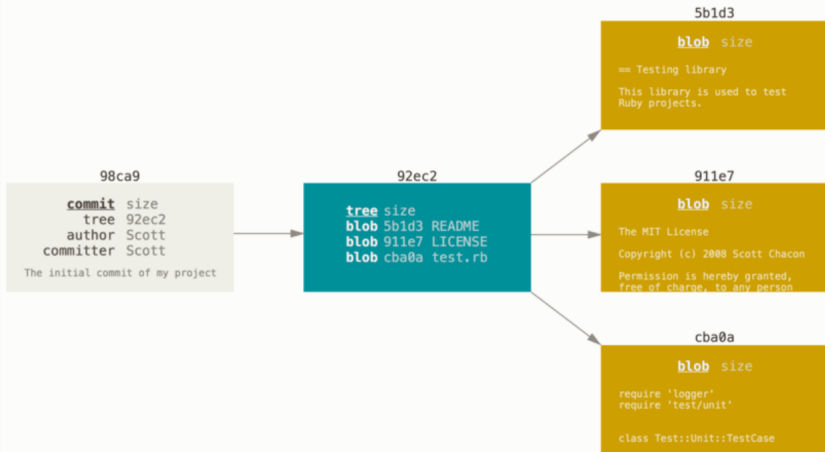
Commits are *snapshots* of the project.

Snapshots are *trees* with reference pointers to changes.

Under the Hood

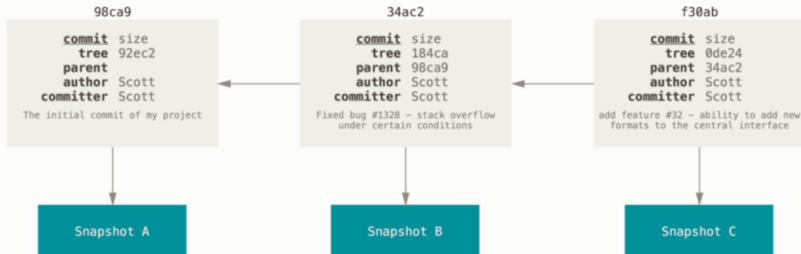
Commits are *snapshots* of the project.

Snapshots are *trees* with reference pointers to changes.



Under the Hood

The *repository* stores these series of snapshots.

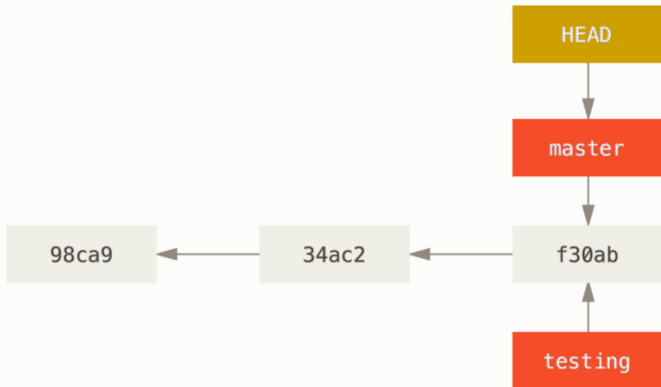


Under the Hood

Branches point to commits.

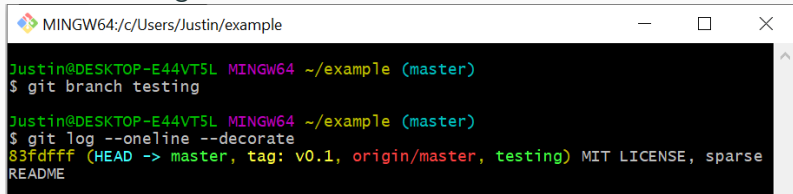
The *HEAD* points to the current branch you are on.

master is the default branch.



Branching

New branch: `git branch <name>`.



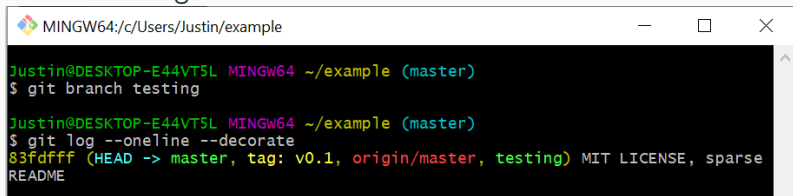
A screenshot of a terminal window titled "MINGW64:/c/Users/Justin/example". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal shows the following commands and output:

```
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git branch testing

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git log --oneline --decorate
83fdfff (HEAD -> master, tag: v0.1, origin/master, testing) MIT LICENSE, sparse
README
```

Branching

New branch: `git branch <name>`.

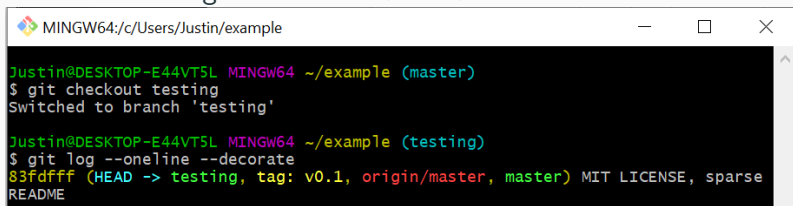


```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git branch testing

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git log --oneline --decorate
83fdfff (HEAD -> master, tag: v0.1, origin/master, testing) MIT LICENSE, sparse
README
```

Switch branch: `git checkout <name>`

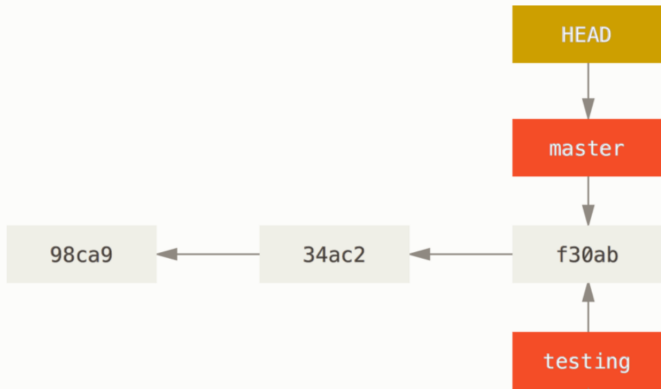


```
MINGW64:/c/Users/Justin/example

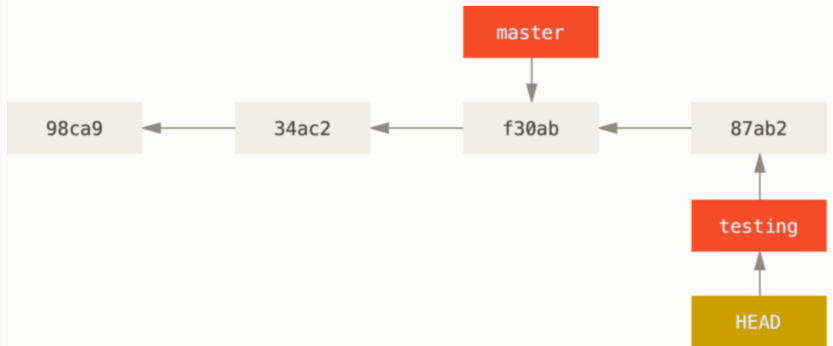
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git checkout testing
Switched to branch 'testing'

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git log --oneline --decorate
83fdfff (HEAD -> testing, tag: v0.1, origin/master, master) MIT LICENSE, sparse
README
```

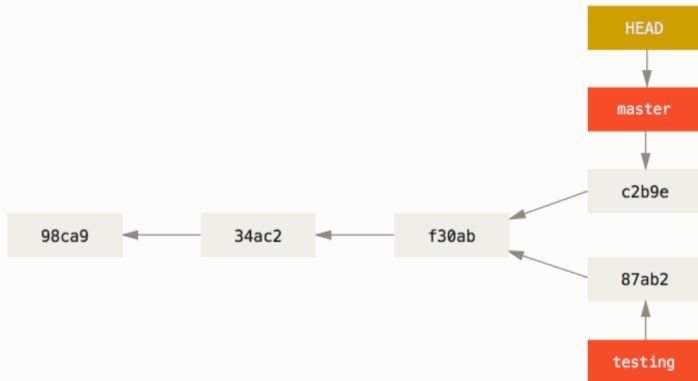
Branch Commit



Branch Commit

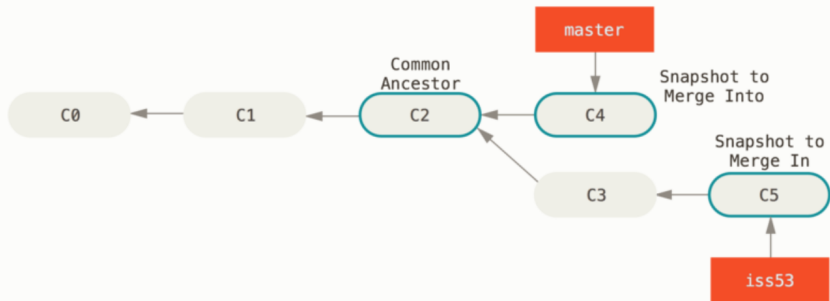


Branch Commit

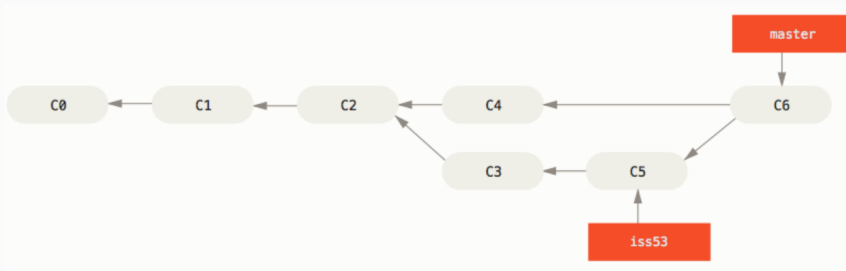


Merging

Basic Merge



Basic Merge



Basic Merge

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ echo 'print('Hello world')' > test.py

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git add .
warning: LF will be replaced by CRLF in test.py.
The file will have its original line endings in your working directory

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git commit -am 'Python Test File'
[testing 3f129c4] Python Test File
1 file changed, 1 insertion(+)
create mode 100644 test.py

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git push origin master
Everything up-to-date

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git push origin testing
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 300.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'testing' on GitHub by visiting:
remote:   https://github.com/gitsetter/example/pull/new/testing
remote:
To https://github.com/gitsetter/example
 * [new branch]      testing -> testing

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ |
```

Basic Merge

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git checkout master
Switched to branch 'master'

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ echo 'disp('Hello World')' > main.m

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git add .
warning: LF will be replaced by CRLF in main.m.
The file will have its original line endings in your working directory
g
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git commit -am 'Main Matlab File'
[master 5224acd] Main Matlab File
1 file changed, 1 insertion(+)
create mode 100644 main.m

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 292.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/gitsetter/example
83fdfff..5224acd master -> master

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ |
```

Basic Merge

git merge <branch>

```
MINGW64/c/Users/Justin/example
Justin@DESKTOP-E44VTSL MINGW64 ~/example (master)
$ git merge testing
hint: Waiting for your editor to close the file... Error detected while processing
/c/Users/Justin/.vimrc:
line 10:
E185: Cannot find color scheme 'gruvbox'
line 12:
E117: Unknown function: plug#begin
line 13:
E492: Not an editor command: Plug 'vim-latex/vim-latex'
line 14:
E117: Unknown function: plug#end
Press ENTER or type command to continue
Merge made by the 'recursive' strategy.
 test.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test.py

Justin@DESKTOP-E44VTSL MINGW64 ~/example (master)
$ git status
On branch master
nothing to commit, working tree clean

Justin@DESKTOP-E44VTSL MINGW64 ~/example (master)
$ git log
commit 447e9e21ddeb130bf290f66236c1611e7ab8618 (HEAD -> master)
Merge: 5224acd 3f129c4
Author: Justin Baker <justinbaker006@gmail.com>
Date: Mon Jun 15 07:42:01 2020 -0600

    Merge branch 'testing'

    Combining Matlab and Python implementations.

commit 5224acd47dee7d4a1a19230f99ac401047ff31c (origin/master)
Author: Justin Baker <justinbaker006@gmail.com>
Date: Mon Jun 15 07:41:20 2020 -0600

    Main Matlab File

commit 3f129c4f35ec1b4f2e4b05a4bf0221ba44c760c7 (origin/testing, testing)
Author: Justin Baker <justinbaker006@gmail.com>
Date: Mon Jun 15 07:39:18 2020 -0600

    Python Test File

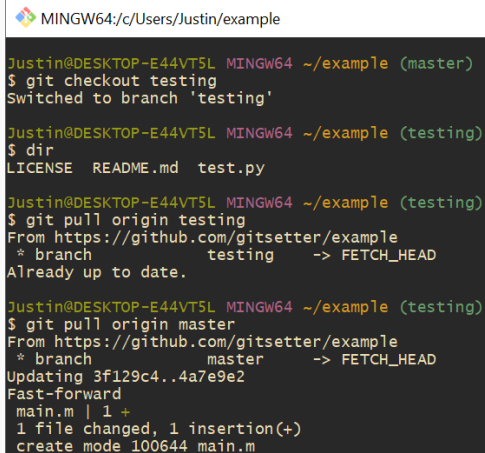
commit 83fdfff72c228c60f735cd4d4b40cb92a08a8bcb (tag: v0.1)
Author: Justin Baker <justinbaker006@gmail.com>
Date: Thu Jun 11 08:33:29 2020 -0600

    MIT LICENSE, sparse README

Justin@DESKTOP-E44VTSL MINGW64 ~/example (master)
$ git push origin master
```

Basic Merge

Fast-forwarding testing branch



A terminal window titled 'MINGW64:/c/Users/Justin/example' with standard window controls. The terminal shows a sequence of git commands: checkout to 'testing', listing files, and two pull operations. The first pull from 'testing' shows 'Already up to date.' The second pull from 'master' shows a 'Fast-forward' merge of 'main.m' with one file change and one insertion.

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git checkout testing
Switched to branch 'testing'

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ dir
LICENSE  README.md  test.py

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git pull origin testing
From https://github.com/gitsetter/example
* branch      testing      -> FETCH_HEAD
Already up to date.

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git pull origin master
From https://github.com/gitsetter/example
* branch      master      -> FETCH_HEAD
Updating 3f129c4..4a7e9e2
Fast-forward
 main.m | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 main.m
```


Merge Conflicts

Conflicts occur if the same line is changed in two different ways.

```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git checkout master
Switched to branch 'master'

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git merge testing
Auto-merging main.m
CONFLICT (content): Merge conflict in main.m
Automatic merge failed; fix conflicts and then commit the result.

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   main.m

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

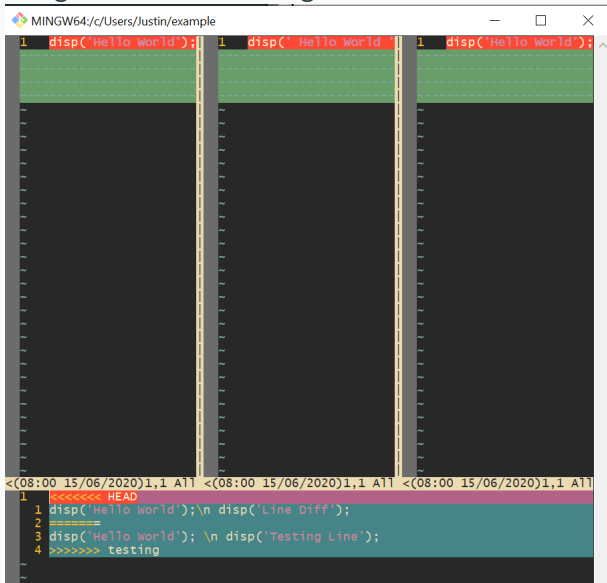
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffmerge ecme
rge p4merge araxis bc codecompare emerge vimdiff
Merging:
main.m

Normal merge conflict for 'main.m':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

Merge Conflicts

Mergetool interface using vimdiff.



The screenshot shows the vimdiff mergetool interface in a MINGW64 terminal window. The window title is "MINGW64:/c/Users/Justin/example". The interface displays three panels side-by-side, each showing a diff of code. The top panel shows a diff of a single line: `disp('Hello world');`. The bottom panel shows a diff of four lines, with the first line being the same as the top panel, and the subsequent lines being `\n disp('Line Diff');`, `====`, `\n disp('Testing Line');`, and `>>>>> testing`. The status bar at the bottom of each panel shows the commit hash, date, time, and file path: `<(08:00 15/06/2020)1,1 A11`.

```
1 disp('Hello world');
```

```
1 disp('Hello world');
```

```
1 disp('Hello world');
```

```
<(08:00 15/06/2020)1,1 A11 <(08:00 15/06/2020)1,1 A11 <(08:00 15/06/2020)1,1 A11
```

```
1 <<<<<< HEAD
```

```
1 disp('Hello world');\n disp('Line Diff');
```

```
2 =====
```

```
3 disp('Hello world'); \n disp('Testing Line');
```

```
4 >>>>>> testing
```

Merge Conflicts

```
MINGW64/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (testing)
$ git checkout master
Switched to branch 'master'

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git merge testing
Auto-merging main.m
CONFLICT (content): Merge conflict in main.m
Automatic merge failed; fix conflicts and then commit the result.

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERCING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   main.m

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

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERCING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecme
rge p4merge araxis bc codecompare emerge vimdiff
Merging:
main.m

Normal merge conflict for 'main.m':
{local}: modified file
{remote}: modified file
Hit return to start merge resolution tool (vimdiff):

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERCING)
$ git merge testing
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master|MERCING)
$ git commit -am 'merged'
[master b5ff869] merged
```

Improved Usage

Tags

Tags are shortcuts to significant commits.

```
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git tag v0.2

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git tag -l
v0.1
v0.2

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git push origin --tags
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/gitsetter/example
* [new tag]          v0.2 -> v0.2
```

Tags are shortcuts to significant commits.

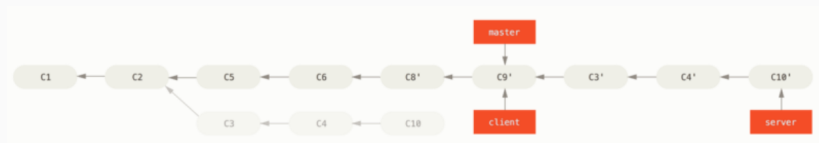
```
MINGW64:/c/Users/Justin/example
Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git show v0.1
commit 83fdfff72c228c60f735cd4d4b40cb92a08a8bcb (HEAD -> master, tag: v0.1, origin/master)
Author: Justin Baker <justinbaker006@gmail.com>
Date: Thu Jun 11 08:33:29 2020 -0600

    MIT LICENSE, sparse README

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..763513e
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+.ipynb_checkpoints
diff --git a/LICENSE b/LICENSE
new file mode 100644
index 0000000..8aa2645
--- /dev/null
+++ b/LICENSE
@@ -0,0 +1,21 @@
+MIT License
+
+Copyright (c) [year] [fullname]
+
+Permission is hereby granted, free of charge, to any person obtaining a copy
+of this software and associated documentation files (the "Software"), to deal
+in the Software without restriction, including without limitation the rights
+to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

Rebasing

Rebasing is linear merging (pros and cons)
git rebase master testing



Working with Others (Must Read)

<https://git-scm.com/book/en/v2/>

Distributed-Git-Contributing-to-a-Project

Submodules

<https://git-scm.com/book/en/v2/Git-Tools-Submodules>

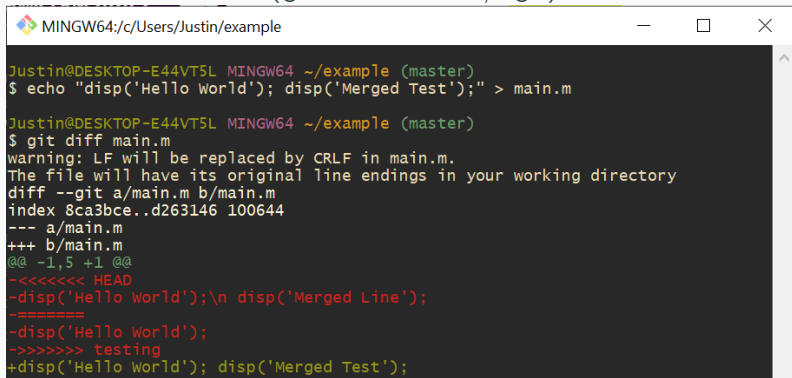
Bundling

<https://git-scm.com/book/en/v2/Git-Tools-Bundling>

Trouble Shooting Tools

Exploring Differences

View File Differences (git diff <filename/arg>)



A screenshot of a terminal window titled "MINGW64:/c/Users/Justin/example". The terminal shows a sequence of commands and their outputs. First, the user runs `echo "disp('Hello World'); disp('Merged Test');" > main.m`. Then, they run `git diff main.m`, which produces a diff output. The output includes a warning about CRLF line endings, the file path `a/main.m`, and a comparison between the index and the working directory. The diff shows a deletion of a line containing `disp('Hello World');` and an addition of a line containing `disp('Hello World');` followed by `disp('Merged Test');`.

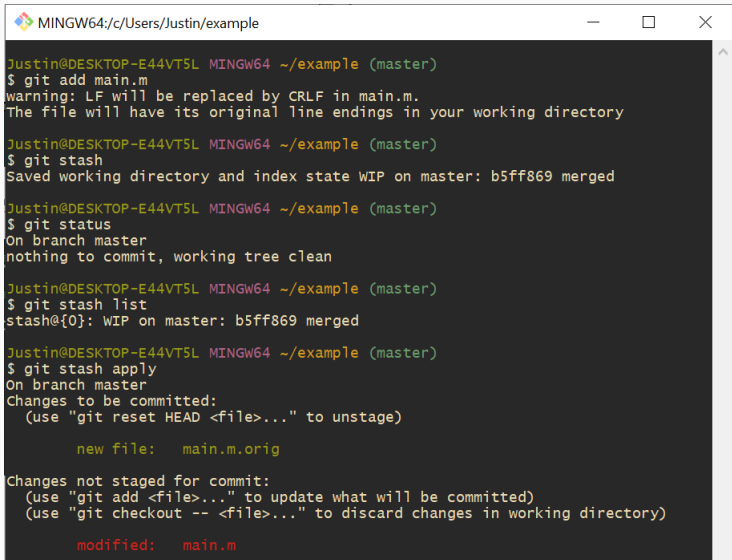
```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ echo "disp('Hello World'); disp('Merged Test');" > main.m

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git diff main.m
warning: LF will be replaced by CRLF in main.m.
The file will have its original line endings in your working directory
diff --git a/main.m b/main.m
index 8ca3bce..d263146 100644
--- a/main.m
+++ b/main.m
@@ -1,5 +1 @@
-<<<<<<< HEAD
-disp('Hello World');\n disp('Merged Line');
-=====
-disp('Hello World');
->>>>>> testing
+disp('Hello World'); disp('Merged Test');
```

Stashing

A stash is a temporary storage of file changes.



```
MINGW64:/c/Users/Justin/example

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git add main.m
warning: LF will be replaced by CRLF in main.m.
The file will have its original line endings in your working directory

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git stash
Saved working directory and index state WIP on master: b5ff869 merged

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git status
On branch master
nothing to commit, working tree clean

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git stash list
stash@{0}: WIP on master: b5ff869 merged

Justin@DESKTOP-E44VT5L MINGW64 ~/example (master)
$ git stash apply
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   main.m.orig

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:   main.m
```

Advanced Merging

https:

`//git-scm.com/book/en/v2/Git-Tools-Advanced-Merging`

Reset

https:

`//git-scm.com/book/en/v2/Git-Tools-Reset-Demystified`