

GIT: A BRIEF PRIMER

Stephen Sekula^{1,2}

¹SNOLAB, Lively, ON, Canada

²Queen's University, Kingston, ON, Canada

May 10, 2024



Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

Git is a Revision Control System

- ▶ A *Revision Control System* (RCS) is any system that allows you to manage versions of electronic files in a way that permits the resolution of conflicting/complementary changes when more than one contributor is active.
- ▶ There are many older systems than Git. I grew up on *CVS* (Concurrent Versions System) and then («SHUDDER») *Subversion*. Git is my favourite.
- ▶ It was developed in 2005 when Linus Torvalds, the creator and lead maintainer of the LINUX operating system kernel, needed to replace the *Bitkeeper* RCS, which had revoked its free license. Linus turned over management of Git to Junio Hamano before its formal version 1.0 release. Like LINUX, Git is an open-source project.
- ▶ The documentation for Git claims Linus named it after himself (just like LINUX), using the British slang word for an unpleasant person. On its best day, documentation suggests Git stands for "Global Information Tracker".
- ▶ Git's strength is the ability to work locally on an entire copy of a project, asynchronously with many other people, and still resolve conflict.



Git is a Revision Control System

- ▶ A *Revision Control System* (RCS) is any system that allows you to manage versions of electronic files in a way that permits the resolution of conflicting/complementary changes when more than one contributor is active.
- ▶ There are many older systems than Git. I grew up on *CVS* (Concurrent Versions System) and then («SHUDDER») *Subversion*. Git is my favourite.
- ▶ It was developed in 2005 when Linus Torvalds, the creator and lead maintainer of the LINUX operating system kernel, needed to replace the *Bitkeeper* RCS, which had revoked its free license. Linus turned over management of Git to Junio Hamano before its formal version 1.0 release. Like LINUX, Git is an open-source project.
- ▶ The documentation for Git claims Linus named it after himself (just like LINUX), using the British slang word for an unpleasant person. On its best day, documentation suggests Git stands for "Global Information Tracker".
- ▶ Git's strength is the ability to work locally on an entire copy of a project, asynchronously with many other people, and still resolve conflict.



Git is a Revision Control System

- ▶ A *Revision Control System* (RCS) is any system that allows you to manage versions of electronic files in a way that permits the resolution of conflicting/complementary changes when more than one contributor is active.
- ▶ There are many older systems than Git. I grew up on *CVS* (Concurrent Versions System) and then («SHUDDER») *Subversion*. Git is my favourite.
- ▶ It was developed in 2005 when Linus Torvalds, the creator and lead maintainer of the LINUX operating system kernel, needed to replace the *Bitkeeper* RCS, which had revoked its free license. Linus turned over management of Git to Junio Hamano before its formal version 1.0 release. Like LINUX, Git is an open-source project.
- ▶ The documentation for Git claims Linus named it after himself (just like LINUX), using the British slang word for an unpleasant person. On its best day, documentation suggests Git stands for "Global Information Tracker".
- ▶ Git's strength is the ability to work locally on an entire copy of a project, asynchronously with many other people, and still resolve conflict.



Git is a Revision Control System

- ▶ A *Revision Control System* (RCS) is any system that allows you to manage versions of electronic files in a way that permits the resolution of conflicting/complementary changes when more than one contributor is active.
- ▶ There are many older systems than Git. I grew up on *CVS* (Concurrent Versions System) and then («SHUDDER») *Subversion*. Git is my favourite.
- ▶ It was developed in 2005 when Linus Torvalds, the creator and lead maintainer of the LINUX operating system kernel, needed to replace the *Bitkeeper* RCS, which had revoked its free license. Linus turned over management of Git to Junio Hamano before its formal version 1.0 release. Like LINUX, Git is an open-source project.
- ▶ The documentation for Git claims Linus named it after himself (just like LINUX), using the British slang word for an unpleasant person. On its best day, documentation suggests Git stands for "Global Information Tracker".
- ▶ Git's strength is the ability to work locally on an entire copy of a project, asynchronously with many other people, and still resolve conflict.



Git is a Revision Control System

- ▶ A *Revision Control System* (RCS) is any system that allows you to manage versions of electronic files in a way that permits the resolution of conflicting/complementary changes when more than one contributor is active.
- ▶ There are many older systems than Git. I grew up on *CVS* (Concurrent Versions System) and then («SHUDDER») *Subversion*. Git is my favourite.
- ▶ It was developed in 2005 when Linus Torvalds, the creator and lead maintainer of the LINUX operating system kernel, needed to replace the *Bitkeeper* RCS, which had revoked its free license. Linus turned over management of Git to Junio Hamano before its formal version 1.0 release. Like LINUX, Git is an open-source project.
- ▶ The documentation for Git claims Linus named it after himself (just like LINUX), using the British slang word for an unpleasant person. On its best day, documentation suggests Git stands for "Global Information Tracker".
- ▶ Git's strength is the ability to work locally on an entire copy of a project, asynchronously with many other people, and still resolve conflict.



What is the idea of Git? (I)

- ▶ Let's consider a simple example: co-authoring a scientific or technical paper.
- ▶ Let's say you have three collaborators — Amit, **Blaise**, and Chris — working together to write up scientific results.
- ▶ The work begins with Amit creating a file to hold the paper and adding a first paragraph of text to the paper. Let's call this *revision 1* of the document.
- ▶ **Blaise** and Chris then begin separately working on revision 1 of the document to add their own content.

```

REVISION 1
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu dui vivamus arcu felis bibendum. Turpis egestas sed
4 tempus urna et pharetra. Posuere urna nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Urna porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Nectus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.

```

What is the idea of Git? (I)

- ▶ Let's consider a simple example: co-authoring a scientific or technical paper.
- ▶ Let's say you have three collaborators — Amit, **Blaise**, and Chris — working together to write up scientific results.
- ▶ The work begins with Amit creating a file to hold the paper and adding a first paragraph of text to the paper. Let's call this *revision 1* of the document.
- ▶ **Blaise** and Chris then begin separately working on revision 1 of the document to add their own content.

```

REVISION 1
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu du vivamus arcu felis bibendum. Turpis egestas sed
4 tempus urna et pharetra. Posuere urna nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Urna porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Nisi et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.

```

What is the idea of Git? (I)

- ▶ Let's consider a simple example: co-authoring a scientific or technical paper.
- ▶ Let's say you have three collaborators — Amit, **Blaise**, and Chris — working together to write up scientific results.
- ▶ The work begins with Amit creating a file to hold the paper and adding a first paragraph of text to the paper. Let's call this *revision 1* of the document.
- ▶ **Blaise** and Chris then begin separately working on revision 1 of the document to add their own content.

```
REVISION 1
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu dui vivamus arcu felis bibendum. Turpis egestas sed
4 tempus urna et pharetra. Posuere urna nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Urna porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Nisi et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
```

What is the idea of Git? (I)

- ▶ Let's consider a simple example: co-authoring a scientific or technical paper.
- ▶ Let's say you have three collaborators — **Amit**, **Blaise**, and Chris — working together to write up scientific results.
- ▶ The work begins with **Amit** creating a file to hold the paper and adding a first paragraph of text to the paper. Let's call this *revision 1* of the document.
- ▶ **Blaise** and Chris then begin separately working on revision 1 of the document to add their own content.

```

REVISION 1
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu du vivamus arcu felis bibendum. Turpis egestas sed
4 tempus urna et pharetra. Posuere urna nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Urna porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Nisi et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.

```

What is the idea of Git? (II)

REVISION 2

```

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 lobore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu dui vivamus arcu felis bibendum. Turpis egestas sed
4 tempus uma et pharetra. Posuere uma nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Uma porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Netus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
13
14 At augue eget arcu dictum. Dignissim sodales ut eu sem integer. Ornare lectus sit amet
15 est placerat in. Lectus urna dui convallis convallis tellus. Ornare aenean euismod
16 elementum nisi quis eleifend quam adipiscing. Vulputate dignissim suspendisse in est
17 ante in. Eget est lorem ipsum dolor sit amet consectetur adipiscing. Vitae justo eget
18 magna fermentum iaculis eu. Mi in nulla posuere sollicitudin. Sodales ut etiam sit amet.
19 Bibendum est ultricies integer quis auctor elit sed vulputate mi.

```

Blaise adds a second paragraph to the document and generates a *revision 2* of the document.

What is the idea of Git? (II)

REVISION 2

```

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu duī vivamus arcu felis bibendum. Turpis egestas sed
4 tempus uma et pharetra. Posuere uma nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Uma portitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Netus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet duī accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
13
14 At augue eget arcu dictum. Dignissim sodales ut eu sem integer. Ornare lectus sit amet
15 est placerat in. Lectus urna duis convallis convallis tellus. Ornare aenean euismod
16 elementum nisi quis eleifend quam adipiscing. Vulputate dignissim suspendisse in est
17 ante in. Eget est lorem ipsum dolor sit amet consectetur adipiscing. Vitae justo eget
18 magna fermentum iaculis eu. Mi in nulla posuere sollicitudin. Sodales ut etiam sit amet.
19 Bibendum est ultricies integer quis auctor elit sed vulputate ni.

```

REVISION 3

```

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultricies mi
3 quis hendrerit dolor magna eget. Arcu duī vivamus arcu felis bibendum. Turpis egestas sed
4 tempus uma et pharetra. Posuere uma nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Uma portitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Netus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet duī accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
13
14 Dolor magna eget est lorem ipsum dolor. Eget dolor morbi non arcu risus quis varius quam
15 quisque. Vulputate dignissim suspendisse in est. Neque aliquam vestibulum morbi blandit. Ut
16 eu sem integer vitae. Aenean vel elit scelerisque mauris pellentesque. In nibh mauris cursus.
17 mattis molestie a iaculis at erat. Tincidunt ornare massa eget egestas purus viverra.
18 accumsan. Mi bibendum neque egestas congue quisque egestas diam. Ultricies mi eget
19 mauris pharetra et. Eu lobortis elementum nibh tellus. Varius morbi enim nunc faucibus.
20 Phasellus faucibus scelerisque eleifend donec pretium.

```

Blaise adds a second paragraph to the document and generates a *revision 2* of the document. Independent of that, Chris adds their own second paragraph.

What is the idea of Git? (II)

REVISION 2

```

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultrices mi
3 quis hendrerit dolor magna eget. Arcu dul vivamus arcu felis bibendum. Turpis egestas sed
4 tempus uma et pharetra. Posuere uma nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Uma porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Netus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
13
14 At augue eget arcu dictum. Dignissim sodales ut eu sem integer. Ornare lectus sit amet
15 est placerat in. Lectus urna dui convallis convallis tellus. Ornare aenean euismod
16 elementum nisi quis eleifend quam adipiscing. Vulputate dignissim suspendisse in est
17 ante in. Eget est lorem ipsum dolor sit amet consectetur adipiscing. Vitae justo eget
18 magna fermentum iaculis eu. Mi in nulla posuere sollicitudin. Sodales ut etiam sit amet.
19 Bibendum est ultrices integer quis auctor elit sed vulputate mi.

```

REVISION 3

```

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
2 labore et dolore magna aliqua. Quis commodo odio aenean sed adipiscing. Massa ultrices mi
3 quis hendrerit dolor magna eget. Arcu dul vivamus arcu felis bibendum. Turpis egestas sed
4 tempus uma et pharetra. Posuere uma nec tincidunt praesent semper feugiat. Libero id
5 faucibus nisi tincidunt. Uma porttitor rhoncus dolor purus non enim. Platea dictumst
6 vestibulum rhoncus est pellentesque elit. Hac habitasse platea dictumst vestibulum rhoncus
7 est pellentesque. Netus et malesuada fames ac turpis egestas. Scelerisque varius morbi enim
8 nunc faucibus a pellentesque. Ut tellus elementum sagittis vitae et leo. Lectus mauris ultrices
9 eros in cursus turpis. Commodo elit at imperdiet dui accumsan sit amet. Quis viverra nibh
10 cras pulvinar. Magna fermentum iaculis eu non diam phasellus vestibulum lorem sed. Lacus
11 laoreet non curabitur gravida arcu ac tortor dignissim. Sed egestas egestas fringilla phasellus
12 faucibus scelerisque eleifend.
13
14 Dolor magna eget est lorem ipsum dolor. Eget dolor morbi non arcu risus quis varius quam
15 quisque. Vulputate dignissim suspendisse in est. Neque aliquam vestibulum morbi blandit. Ut
16 eu sem integer vitae. Aenean vel elit scelerisque mauris pellentesque. In nibh mauris cursus
17 mattis molestie a iaculis at erat. Tincidunt ornare massa eget egestas purus viverra
18 accumsan. Mi bibendum neque egestas congue quisque egestas diam. Ultrices mi eget
19 mauris pharetra et. Eu lobortis elementum nibh tellus. Varius morbi enim nunc faucibus.
20 Phasellus faucibus scelerisque eleifend donec pretium.

```

Blaise adds a second paragraph to the document and generates a *revision 2* of the document. Independent of that, Chris adds their own second paragraph.

All three now wish to bring their contributions together into a single version of the document (*revision 4*). How can they manage this without a bunch of copy-and-paste in a fourth copy of the document? This is the situation where Git excels.

Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

How Do I Get Git? (Windows)



git for windows
VERSION 2.45.0

FAQ REPOSITORY MAILING LIST

We bring the awesome
Git SCM to Windows

Download Contribute

Tools & Features

Git for Windows focuses on offering a lightweight, native set of tools that bring the full feature set of the Git SCM to Windows while providing appropriate user interfaces for experienced Git users and novices alike.

Git BASH

Git for Windows provides a BASH emulation used to run Git from the command line. *NIX users should feel right at home, as the BASH emulation behaves just like the "git" command in LINUX and UNIX environments.

Git GUI

As Windows users commonly expect graphical user interfaces, Git for Windows also provides the Git GUI, a powerful alternative to Git BASH, offering a graphical version of just about every Git command line function, as well as comprehensive visual diff tools.

Shell Integration

Simply right-click on a folder in Windows Explorer to access the BASH or GUI.

```
MINGW64/c/Users/me/git
me@work MINGW64 ~
$ git clone https://github.com/git-for-windows/git
Cloning into 'git'...
remote: Enumerating objects: 500937, done.
remote: Counting objects: 100% (3486/3486), done.
remote: Compressing objects: 100% (1415/1415), done.
remote: Total 500937 (delta 2494), reused 2917 (delta 2071), pack-reused 497451
Receiving objects: 100% (500937/500937), 221.14 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (362274/362274), done.
Updating files: 100% (4031/4031), done.

me@work MINGW64 ~
$ cd git

me@work MINGW64 ~/git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

me@work MINGW64 ~/git (main)
$ |
```

Download from gitforwindows.org

How Do I Get Git? (Mac)



Xcode

Xcode 15 SwiftUI SwiftUI Resources [Download](#)

Xcode 15

Xcode 15 enables you to develop, test, and distribute apps for all Apple platforms. Code and design your apps faster with enhanced code completion, interactive previews, and live animations. Use Git staging to craft your next commit without leaving your code. Explore and diagnose your test results with redesigned test reports with video recording. And start deploying seamlessly to TestFlight and the App Store from Xcode Cloud. Creating amazing apps has never been easier.

Download for macOS

There are several options for installing Git on macOS. Note that any non-source distributions are provided by third parties, and may not be up to date with the latest source release.

Choose one of the following options for installing Git on macOS:

Homebrew

Install [homebrew](#) if you don't already have it, then:

```
$ brew install git
```

MacPorts

Install [MacPorts](#) if you don't already have it, then:

```
$ sudo port install git
```

Xcode

Apple ships a binary package of Git with [Xcode](#).

Binary installer

Tim Harper provides an [installer](#) for Git. The latest version is 2.33.0, which was released over 2 years ago, on 2021-08-30.

Building from Source

If you prefer to build from source, you can find tarballs [on kernel.org](#). The latest version is 2.45.0.

Installing git-gui

If you would like to install [git-gui](#) and [gitk](#), git's commit GUI and interactive history browser, you can do so using [homebrew](#)

```
$ brew install git-gui
```

Install Mac's Xcode package from the App Store

How Do I Get Git? (Linux)

Download for Linux and Unix

It is easiest to install Git on Linux using the preferred package manager of your Linux distribution. If you prefer to build from source, you can find tarballs on kernel.org. The latest version is 2.45.0.

Debian/Ubuntu

For the latest stable version for your release of Debian/Ubuntu

```
# apt-get install git
```

For Ubuntu, this PPA provides the latest stable upstream Git version

```
# add-apt-repository ppa:git-core/ppa # apt update; apt install git
```

Fedora

```
# yum install git (up to Fedora 21)
```

```
# dnf install git (Fedora 22 and later)
```

Gentoo

```
# emerge --ask --verbose dev-vcs/git
```

Arch Linux

```
# pacman -S git
```

openSUSE

```
# zypper install git
```

Mageia

```
# urpmi git
```

Nix/NixOS

```
# nix-env -i git
```

FreeBSD

```
# pkg install git
```

Solaris 9/10/11 (OpenCSW)

```
# pkgutil -i git
```

Solaris 11 Express

```
# pkg install developer/versioning/git
```

OpenBSD

```
# pkg_add git
```

Alpine

```
# apk add git
```

Red Hat Enterprise Linux, Oracle Linux, CentOS, Scientific Linux, et al.

RHEL and derivatives typically ship older versions of git. You can [download a tarball](#) and build from source, or use a 3rd-party repository such as the [IUS Community Project](#) to obtain a more recent version of git.

Slitaz

```
# tazpkg get-install git
```

```

• (base) sjsekula@papa:/home/sjsekula/Documents/slides-trove$ 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 restore <file>..." to discard changes in working directory)
    modified:   20240510-EIEIOO-Git-Workshop/20240510-EIEIOO-Git-Workshop.tex
    modified:   20240510-EIEIOO-Git-Workshop/src/slides.tex

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2019-06-19-smu-atlas/#units.py#
    2019-06-19-smu-atlas/.ipynb_checkpoints/
    20190515-smu_atlas_weekly/
    2020-07-02-XFitter-Talk/src/slides-backup.tex
    2020-11-12-EIC-Charm-Jets-PID/
    20240429-HALO-unfolding.pdf
    20240510-EIEIOO-Git-Workshop.pdf
    20240510-EIEIOO-Git-Workshop/Images/
    Images/logo-BNL-black.svg
    cover-CCDIS_e10_p275_j24_eta16_3d-2.png
    phys1303/src/extendedbodies-problems.tex
    phys3305/Images/Fluorodeoxyglucose_18-F_skeletal.svg

no changes added to commit (use "git add" and/or "git commit -a")
• (base) sjsekula@papa:/home/sjsekula/Documents/slides-trove$ ls 20190515-smu_atlas_weekly/
20190515-smu_atlas_weekly.tex  20190515-smu_atlas_weekly.tex~  bib  Images  src
• (base) sjsekula@papa:/home/sjsekula/Documents/slides-trove$ git add 20190515-smu_atlas_weekly
• (base) sjsekula@papa:/home/sjsekula/Documents/slides-trove$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   20190515-smu_atlas_weekly/20190515-smu_atlas_weekly.tex
    new file:   20190515-smu_atlas_weekly/Images/Hbb-2019.jpg
    new file:   20190515-smu_atlas_weekly/Images/TW-retuning.jpg
    new file:   20190515-smu_atlas_weekly/Images/hfsfweb.png
    new file:   20190515-smu_atlas_weekly/bib/sources.bib
    new file:   20190515-smu_atlas_weekly/src/backup.tex
    new file:   20190515-smu_atlas_weekly/src/slides.tex
    new file:   20190515-smu_atlas_weekly/src/title.tex

```

Install using your Linux distribution's package manager

VSCode: Universal Open-Source Code Development Platform

The screenshot displays the Visual Studio Code interface. On the left, the 'EXTENSIONS MARKETPLACE' sidebar shows a list of installed and available extensions, including Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, Vetur, and C#. The main editor area shows a file named 'serviceWorker.js' with the following code:

```

src > .\serviceWorker.js > register > @window.addEventListener('load') callback
checkValidServiceWorker(swUrl, config);
// Add some additional logging to localhost, pr
// service worker/PWA documentation.
navigator.serviceWorker.ready.then(() => {
  @product
  @productSub
  @removeSiteSpecificTrackingException
  @removeWebWideTrackingException
  @requestMediaKeySystemAccess
  @sendBeacon
  @serviceworker (property) Navigator.serviceMorker...
  @storage
  @storeSiteSpecificTrackingException
  @storeWebWideTrackingException
  @userAgent
})
@vendor
function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {

```

The terminal window at the bottom shows the command 'node' and the output:

```

You can now view create-react-app in the browser.
Local: http://localhost:3000/
On Your Network: http://10.211.55.3:3000/
Note that the development build is not optimized.

```

Download and install from <https://code.visualstudio.com/>

Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

Finding Help on the Command Line

The `git` command can be followed by a second command to execute a task. This includes asking for help: `git help`. For example,

```
> git help
```

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: `git help tutorial`)

```
clone      Clone a repository into a new directory
init       Create an empty Git repository or reinitialize an existing one
```

work on the current change (see also: `git help everyday`)

```
add        Add file contents to the index
mv         Move or rename a file, a directory, or a symlink
restore    Restore working tree files
rm         Remove files from the working tree and from the index
```

Getting a Repository for Practice

We want to begin by getting an existing project and *cloning* it our own computer. Let's exercise getting help on commands:

```
> git help clone
```

NAME

```
git-clone - Clone a repository into a new directory
```

SYNOPSIS

```
git clone [--template=<template_directory>]
  [-l] [-s] [--no-hardlinks] [-q] [-n] [--bare] [--mirror]
  [-o <name>] [-b <name>] [-u <upload-pack>] [--reference <repository>]
  [--dissociate] [--separate-git-dir <git dir>]
  [--depth <depth>] [--[no-]single-branch] [--no-tags]
  [--recurse-submodules[=<pathspec>]] [--[no-]shallow-submodules]
  [--[no-]remote-submodules] [--jobs <n>] [--sparse] [--[no-]reject-shallow]
  [--filter=<filter>] [--] <repository>
  [<directory>]
```

DESCRIPTION

Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using `git branch --remotes`), and creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

That's a lot of information. Let's boil it down to the bare minimum:

```
git clone <<PROJECT URL>>
```

Cloning a Specific Repository

```
git clone https://github.com/stephensekula/Git-Tutorial-EIEIOO.git
```


Repository Structure: What Am I Looking At Here?

```
> cd Git-Tutorial-EIEIOO/  
> ls -l  
total 8  
-rw-r--r-- 1 ssekula ssekula 1071 May  7 12:10 LICENSE  
-rw-r--r-- 1 ssekula ssekula   95 May  7 12:10 README.md
```

When you checkout a project, by default you see (and are working on) what is known as *the main branch* of the project. In the main branch, we see two files: a `LICENSE` file and a `README.md` file. The latter is meant to serve as the "user instructions" for any project and is written in a simple text-based formatting language called *Markdown* (md), which web browsers can interpret and format as nice documents.

Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

Change A File

Let's begin by changing one of the files.

- ▶ Open the `README.md` file in an editor (e.g. VSCode, TextEdit (Mac), Notepad (Windows), Emacs (**NIX), VI (**NIX), gedit (GNOME Desktop on Linux), etc.).
- ▶ Add some text to the bottom of the file, e.g.

```
Ubuntu > home > ssekula > Documents > Git-Tutorial-EIEIOO > ⓘ README.md > [abc] # Git-Tutorial-EIEIOO
1  # Git-Tutorial-EIEIOO
2  A playground for participants in EIEIOO and other workshop environments.
3
4  WOW! Look at this AMAZING text I added. I am such a wordsmith.
5  [ ]
```

- ▶ Cool. Cool. Cool. What does this have to do with Git?

Change A File

Let's begin by changing one of the files.

- ▶ Open the `README.md` file in an editor (e.g. VSCode, TextEdit (Mac), Notepad (Windows), Emacs (**NIX), VI (**NIX), gedit (GNOME Desktop on Linux), etc.).
- ▶ Add some text to the bottom of the file, e.g.

```
Ubuntu > home > ssekula > Documents > Git-Tutorial-EIEIOO > ⓘ README.md > [abc] # Git-Tutorial-EIEIOO
1  # Git-Tutorial-EIEIOO
2  A playground for participants in EIEIOO and other workshop environments.
3
4  WOW! Look at this AMAZING text I added. I am such a wordsmith.
5  [ ]
```

- ▶ Cool. Cool. Cool. What does this have to do with Git?

Check the Project Status

Git is already tracking changes to files known to the project (e.g., LICENSE and README.md) the moment you save your changes to the file. You can see that Git is aware of changes using the `git status` command:

```
> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

Add A File

Our recent file editing has resulted in changes that Git recognizes. However, those changes are not automatically stored. We have to tell Git to *add files that have been changed* and then to *commit those changes to the project* so we can manage them (e.g., back them out if we don't like them).

To do this we use

```
git add <<FILENAME>>
```

For example,

```
git add README.md
```

Add A File

Our recent file editing has resulted in changes that Git recognizes. However, those changes are not automatically stored. We have to tell Git to *add files that have been changed* and then to *commit those changes to the project* so we can manage them (e.g., back them out if we don't like them).

To do this we use

```
git add <<FILENAME>>
```

For example,

```
git add README.md
```

BONUS: Run `git status`. What is different now that you have run `git add`?

Commit A Change

Even though you have added the changed files, the changes themselves are not recorded in the repository. To do this, you need to *commit your changes*. It is this step that forever emblazons what you did in the history of the project . . . at least, *in your local copy of the project*.

Committing comes with two minimal actions: executing commit and recording a log file message explaining what you did. I like to do this in one line:

```
> git commit -m "I added to the README.md file explaining my excellent prose."  
[main de69555] I added to the README.md file explaining my excellent prose.  
1 file changed, 2 insertions(+)
```

Commit A Change

Even though you have added the changed files, the changes themselves are not recorded in the repository. To do this, you need to *commit your changes*. It is this step that forever emblazons what you did in the history of the project . . . at least, *in your local copy of the project*.

Committing comes with two minimal actions: executing commit and recording a log file message explaining what you did. I like to do this in one line:

```
> git commit -m "I added to the README.md file explaining my excellent prose."  
[main de69555] I added to the README.md file explaining my excellent prose.  
 1 file changed, 2 insertions(+)
```

If you use the one-line approach, *try to keep your message to 50 characters or less*. If you need more space, run `git commit` without the `-m` option and use the editor window that opens to write (a) a short one-line (50 character) title and below that (b) a list of your changes.

Commit A Change

Even though you have added the changed files, the changes themselves are not recorded in the repository. To do this, you need to *commit your changes*. It is this step that forever emblazons what you did in the history of the project . . . at least, *in your local copy of the project*.

Committing comes with two minimal actions: executing commit and recording a log file message explaining what you did. I like to do this in one line:

```
> git commit -m "I added to the README.md file explaining my excellent prose."  
[main de69555] I added to the README.md file explaining my excellent prose.  
1 file changed, 2 insertions(+)
```

If you use the one-line approach, *try to keep your message to 50 characters or less*. If you need more space, run `git commit` without the `-m` option and use the editor window that opens to write (a) a short one-line (50 character) title and below that (b) a list of your changes.

Each commit is assigned a unique identifier (e.g., `de695553ab207d2d644464e1cbe202d70c7d5a07` (long form) or `de69555` (short form)). This is how you can select which commits (from someone else) you want to apply to your copy of the project. (reflect on the Amit, Blaise, and Chris problem)

See Your Change

You can check the log associated with the project to see that your change has been recorded.

```
> git log
commit de695553ab207d2d644464e1cbe202d70c7d5a07 (HEAD -> main)
Author: Stephen Jacob Sekula <stephen.sekula@snolab.ca>
Date: Thu May 9 14:04:11 2024 -0400

    I added to the README.md file explaining my excellent prose.

commit 2790b4f75893c248862b52f6f2508baded11eeea (origin/main, origin/HEAD)
Author: Stephen Sekula <stephensekula@users.noreply.github.com>
Date: Wed May 1 14:39:01 2024 -0400

    Initial commit
```

See Your Change

You can check the log associated with the project to see that your change has been recorded.

```
> git log
commit de695553ab207d2d644464e1cbe202d70c7d5a07 (HEAD -> main)
Author: Stephen Jacob Sekula <stephen.sekula@snolab.ca>
Date: Thu May 9 14:04:11 2024 -0400

    I added to the README.md file explaining my excellent prose.

commit 2790b4f75893c248862b52f6f2508baded11eeea (origin/main, origin/HEAD)
Author: Stephen Sekula <stephensekula@users.noreply.github.com>
Date: Wed May 1 14:39:01 2024 -0400

    Initial commit
```

What is the HEAD? This refers to the current branch's latest commit. We are in the `main` branch, and this commit represents its HEAD.

Push Your Change

But wait! It's true that your *local clone of this project* knows about the new change, but what about all your non-local (remote) collaborators? How do they pick up this change? You have to *push your commits to the original project*. In this case, my project was stored on the site Github, so we need to push this back there. This is known as pushing changes to the remote repository (the "remote"):

```
> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 454 bytes | 454.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:stephensekula/Git-Tutorial-EIEIOO.git
 2790b4f..de69555  main -> main
```

Push Your Change

But wait! It's true that your *local clone of this project* knows about the new change, but what about all your non-local (remote) collaborators? How do they pick up this change? You have to *push your commits to the original project*. In this case, my project was stored on the site Github, so we need to push this back there. This is known as pushing changes to the remote repository (the "remote"):

```
> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 454 bytes | 454.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:stephensekula/Git-Tutorial-EIEIOO.git
 2790b4f..de69555  main -> main
```

Nota bene: unless you (a) have an account on the remote system and (b) are recognized as a developer with permission to push changes, you cannot push to a remote. Pulling is generally freely available (anyone can take); pushing is limited to the development team (few can give).

Push Your Change

But wait! It's true that your *local clone of this project knows about the new change*, but what about all your non-local (remote) collaborators? How do they pick up this change? You have to *push your commits to the original project*. In this case, my project was stored on the site Github, so we need to push this back there. This is known as pushing changes to the remote repository (the "remote"):

```
> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 454 bytes | 454.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:stephensekula/Git-Tutorial-EIEIOO.git
 2790b4f..de69555  main -> main
```

Nota bene: unless you (a) have an account on the remote system and (b) are recognized as a developer with permission to push changes, you cannot push to a remote. Pulling is generally freely available (anyone can take); pushing is limited to the development team (few can give).

By default, `git push` assumes you want to push changes to the remote listed at the top of the information provided by `git remote -v`. Try it and see what you learn.

Pull Others Changes

Let's say I go ahead and edit the `README.md` file one more time:

```
# Git-Tutorial-EIEIOO
A playground for participants in EIEIOO and other workshop environments.

WOW! Look at this AMAZING text I added. I am such a wordsmith.

This third line is clearly superior. All other third lines are a lie.
```

I then add, commit, and push my changes. If someone else has pushed *their changes* in the meantime, this happens:

```
> git add README.md
> git commit -m "A clearly best third line ever in a README.md file!"
[main 2373be4] A clearly best third line ever in a README.md file!
 1 file changed, 2 insertions(+)
> git push
To github.com:stephensekula/Git-Tutorial-EIEIOO.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'github.com:stephensekula/Git-Tutorial-EIEIOO.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Best practice: before you try to push your changes, pull from the remote repository (early and often when working on any branch) to keep up with changes. Then push your changes. This often avoids conflict.

The Git Four-Step

The habit you want to build as you develop a project is to execute periodically the "Git Four-Step":

- ▶ `git pull`
- ▶ `git add «FILES»`
- ▶ `git commit «MESSAGE»`
- ▶ `git push`

Resolve Conflicts I

In the previous example, we will encounter our first conflict in the development team. Two of us have added a third line to the README.md file. If I run a pull command:

```
> git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Resolve Conflicts I

In the previous example, we will encounter our first conflict in the development team. Two of us have added a third line to the README.md file. If I run a pull command:

```
> git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

CONFLICT! What has happened as a result of being in this state?

Resolve Conflicts I

In the previous example, we will encounter our first conflict in the development team. Two of us have added a third line to the README.md file. If I run a pull command:

```
> git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

CONFLICT! What has happened as a result of being in this state?

The file with the conflict (README.md) has been modified to contain all the options. You now have to edit the file and resolve those conflicts, either manually (deleting some things, keeping others) or using a tool built into your editor (VSCode provides this ability through plugins).

Resolve Conflicts II

```

1 # Git-Tutorial-EIEIOO
2 A playground for participants in EIEIOO and other workshop environments.
3
4 WOW! Look at this AMAZING text I added. I am such a wordsmith.
5
6 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
7 <<<<<< HEAD (Current Change)
8 This third line is clearly superior. All other third lines are a lie.
9 =====
10 What a nifty third line. Best third line ever in a README file.
11 >>>>>> c4d0b622ba57c9393ceab9336b8bced594f1dbe9 (Incoming Change)
12

```

RESOLVED!

```

1 # Git-Tutorial-EIEIOO
2 A playground for participants in EIEIOO and other workshop environments.
3
4 WOW! Look at this AMAZING text I added. I am such a wordsmith.
5
6 This third line is clearly superior. All other third lines are a lie.
7 What a nifty third line. Best third line ever in a README file.
8

```

I use VSCode's interface as an example. We see highlighted above (left) the two conflicting line choices. VSCode allows you to choose your change (current change), the original one from the remote version of the main branch (incoming change), or to accept both. I accepted both, and the resulting file is shown right. Now you can `git add`, `git commit`, and then `git push`.

Outline

What is Git?

Git'ting Git

Git'ting Started


Git'ting Going

Git'ting More Advanced

Next Steps

Visualizing a Project with Branches

What is a *branch*? It's just a term for how the project contents can be developed in multiple, parallel (and sometimes divergent) ways. There are tools to visualize a project and all of its branches, like *Git Graph* for VSCode:

Graph	Description	Date	Author	Commit
	<div style="display: flex; align-items: center;"> ○ <div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-right: 5px;"> 🔗 main </div> origin <div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-left: 10px;"> 🔗 origin/HEAD </div> OK. This one is jus... </div>	9 May 2024 19:...	Stephen Jacob Sekula	2652a97c
	Add some space and a weird request	9 May 2024 19:...	Stephen Jacob Sekula	f1b13c22
	This is a great sentence to have in the README	9 May 2024 19:...	Stephen Jacob Sekula	fb25e227
	<div style="display: flex; align-items: center;"> 🔗 <div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-right: 5px;"> 🔗 origin/develop/better-readme </div> Add another random se... </div>	9 May 2024 19:...	Stephen Jacob Sekula	0a00a85d
	Add a random sentence to the file	9 May 2024 19:...	Stephen Jacob Sekula	f5857587
	Merging two great third lines into one file	9 May 2024 18:...	Stephen Jacob Sekula	c98e1354
	A clearly best third line ever in a README.md file!	9 May 2024 14:...	Stephen Jacob Sekula	2373be4c
	Added best third line ever!	9 May 2024 14:...	Stephen Jacob Sekula	c4d0b622
	I added to the README.md file explaining my excellent pr...	9 May 2024 14:...	Stephen Jacob Sekula	de695553
	Initial commit	1 May 2024 14:...	Stephen Sekula	2790b4f7

Creating A Branch

You can list branches in a project:

```
> git branch -l
* main
```

This is boring! Let's create a second branch beside the main one to keep developing our awesome README.md file:

```
> git checkout -b develop/better-readme
Switched to a new branch 'develop/better-readme'
```

Normally, the `checkout` command is how you select other branches to work on. Executed this way, the command (a) creates the branch with your chosen name and (b) then checks it out for you to develop. You are now developing on this branch, and any changes you make here do not affect the main branch.

Merging Changes Into Your Branch

While you are working on your great branch, somebody else might be making changes to the main branch. In many cases, you may wish to pull those changes into your branch. You can *merge* in these changes as follows:

```
> git checkout main
> git pull
> git checkout develop/better-
  readme
> git merge main
```

Merging Changes Into Your Branch

While you are working on your great branch, somebody else might be making changes to the main branch. In many cases, you may wish to pull those changes into your branch. You can *merge* in these changes as follows:

```
> git checkout main
> git pull
> git checkout develop/better-readme
> git merge main
```

We see the graph change from what it was before, as the development branch (blue) now contains the separate changes made to the main branch (pink) and the two come back together again, even though they remain independent paths in the project. They are "harmonized".

Graph	Description	Date	Author	Commit
	develop/better-readme <i>origin</i> Merged in the ma...	10 May 2024 08...	Stephen Jacob Sekula	ccd50d03
	Merge branch 'main' into develop/better-readme	10 May 2024 08...	Stephen Jacob Sekula	7afd8470
	main <i>origin</i> origin/HEAD OK. This one is just str...	9 May 2024 19...	Stephen Jacob Sekula	2652a97c
	Add some space and a weird request	9 May 2024 19...	Stephen Jacob Sekula	f1b13c22
	This is a great sentence to have in the README	9 May 2024 19...	Stephen Jacob Sekula	fb25e227
	Add another random sentence to the file	9 May 2024 19...	Stephen Jacob Sekula	0a00a85d
	Add a random sentence to the file	9 May 2024 19...	Stephen Jacob Sekula	f5857587
	Merging two great third lines into one file	9 May 2024 18...	Stephen Jacob Sekula	c98e1354
	A clearly best third line ever in a README.md file!	9 May 2024 14...	Stephen Jacob Sekula	2373be4c
	Added best third line ever!	9 May 2024 14...	Stephen Jacob Sekula	c4d0b622
	I added to the README.md file explaining my excellent pr...	9 May 2024 14...	Stephen Jacob Sekula	de695553
	Initial commit	1 May 2024 14...	Stephen Sekula	2790b4f7

Cherry-Picking Merges Into Your Branch

You may not want to merge in every commit in the main branch. If you just want one of them, you can *cherry pick* that specific commit. You need only know the long or short code for that commit.

```
> git cherry-pick 3c5b250
Auto-merging README.md
[develop/better-readme 1f42be3] How can space
    have a colour?
Date: Fri May 10 08:22:08 2024 -0400
1 file changed, 1 insertion(+), 2 deletions(-)
```

Cherry-Picking Merges Into Your Branch

You may not want to merge in every commit in the main branch. If you just want one of them, you can *cherry pick* that specific commit. You need only know the long or short code for that commit. This ability is why people often advocate for *atomic commits* — the smallest possible self-consistent commit so that others can cherry pick specific changes or features and ignore others.

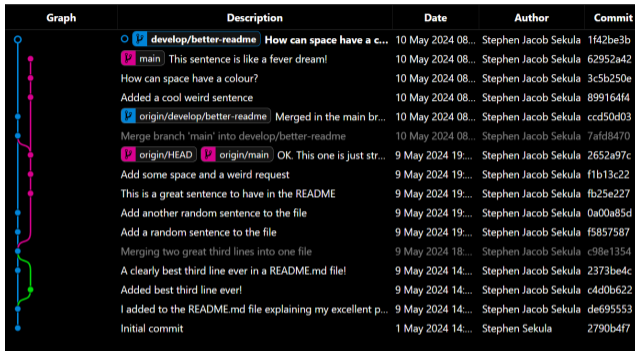
```
> git cherry-pick 3c5b250
Auto-merging README.md
[develop/better-readme 1f42be3] How can space
    have a colour?
Date: Fri May 10 08:22:08 2024 -0400
1 file changed, 1 insertion(+), 2 deletions(-)
```






















Cherry-Picking Merges Into Your Branch

You may not want to merge in every commit in the main branch. If you just want one of them, you can *cherry pick* that specific commit. You need only know the long or short code for that commit. This ability is why people often advocate for *atomic commits* — the smallest possible self-consistent commit so that others can cherry pick specific changes or features and ignore others.

In the example at the right, I cherry-picked one of three commits from the main branch and brought it into the development branch:

```
> git cherry-pick 3c5b250
Auto-merging README.md
[develop/better-readme 1f42be3] How can space
have a colour?
Date: Fri May 10 08:22:08 2024 -0400
1 file changed, 1 insertion(+), 2 deletions(-)
```



Graph	Description	Date	Author	Commit
	 <code>develop/better-readme</code> How can space have a c...	10 May 2024 08...	Stephen Jacob Sekula	1f42be3b
	 <code>main</code> This sentence is like a fever dream!	10 May 2024 08...	Stephen Jacob Sekula	62952a42
	How can space have a colour?	10 May 2024 08...	Stephen Jacob Sekula	3c5b250e
	Added a cool weird sentence	10 May 2024 08...	Stephen Jacob Sekula	899164f4
	 <code>origin/develop/better-readme</code> Merged in the main br...	10 May 2024 08...	Stephen Jacob Sekula	ccd50d03
	Merge branch 'main' into develop/better-readme	10 May 2024 08...	Stephen Jacob Sekula	7afd8470
	 <code>origin/HEAD</code>  <code>origin/main</code> OK. This one is just str...	9 May 2024 19...	Stephen Jacob Sekula	2652a97c
	Add some space and a weird request	9 May 2024 19...	Stephen Jacob Sekula	f1b13c22
	This is a great sentence to have in the README	9 May 2024 19...	Stephen Jacob Sekula	fb25e227
	Add another random sentence to the file	9 May 2024 19...	Stephen Jacob Sekula	0a00a85d
	Add a random sentence to the file	9 May 2024 19...	Stephen Jacob Sekula	f5857587
	Merging two great third lines into one file	9 May 2024 18...	Stephen Jacob Sekula	c98e1354
	A clearly best third line ever in a README.md file!	9 May 2024 14...	Stephen Jacob Sekula	2373be4c
	Added best third line ever!	9 May 2024 14...	Stephen Jacob Sekula	c4d0b622
	I added to the README.md file explaining my excellent p...	9 May 2024 14...	Stephen Jacob Sekula	de695553
	Initial commit	1 May 2024 14...	Stephen Sekula	2790b4f7

Merging Your Branch Into the Main Branch

This is basically the same as merging changes from main into your development branch ...just in reverse!

```
> git checkout main  
> git merge develop/better-readme
```

Tagging An Important Milestone

A *tag* is a string or number associated with a snapshot of the code at a particular state. It could be associated with a moment in time along the development of a branch,

You can tag your branch (or main) at a given state by executing

```
git tag <<STRING>>
```

For example

```
git tag v0.0.1
```


Tagging An Important Milestone

A *tag* is a string or number associated with a snapshot of the code at a particular state. It could be associated with a moment in time along the development of a branch, or it could be associated with a selected series of commits that are cherry-picked into a dedicated branch.

You can tag your branch (or main) at a given state by executing

```
git tag <<STRING>>
```

For example

```
git tag v0.0.1
```

Tagging An Important Milestone

A *tag* is a string or number associated with a snapshot of the code at a particular state. It could be associated with a moment in time along the development of a branch, or it could be associated with a selected series of commits that are cherry-picked into a dedicated branch. "Windows 11" or "macOS 14 Sonoma" or "Ubuntu 22.04" are tags (of a kind) and represent the state of a suite of software at some time.

You can tag your branch (or main) at a given state by executing

```
git tag <<STRING>>
```

For example

```
git tag v0.0.1
```

Outline

What is Git?

Git'ting Git

Git'ting Started

Git'ting Going

Git'ting More Advanced

Next Steps

Review and Outlook

- ▶ We have learned a little bit about the history and purpose of Git as a revision control system.
- ▶ We have seen ways to get Git installed on your system.
- ▶ We have learned about some of the basic commands (clone, add, commit, pull, push, status) that are routinely used together to manage changes to a project.
- ▶ We have seen some steps toward advanced usage (branch, merge, cherry-pick).
- ▶ Your homework: the best learning occurs when you have a goal and a purpose for a tool. How would *you* use Git to manage a project or process that is important to you?

Review and Outlook

- ▶ We have learned a little bit about the history and purpose of Git as a revision control system.
- ▶ We have seen ways to get Git installed on your system.
- ▶ We have learned about some of the basic commands (clone, add, commit, pull, push, status) that are routinely used together to manage changes to a project.
- ▶ We have seen some steps toward advanced usage (branch, merge, cherry-pick).
- ▶ Your homework: the best learning occurs when you have a goal and a purpose for a tool. How would *you* use Git to manage a project or process that is important to you?

Review and Outlook

- ▶ We have learned a little bit about the history and purpose of Git as a revision control system.
- ▶ We have seen ways to get Git installed on your system.
- ▶ We have learned about some of the basic commands (clone, add, commit, pull, push, status) that are routinely used together to manage changes to a project.
- ▶ We have seen some steps toward advanced usage (branch, merge, cherry-pick).
- ▶ Your homework: the best learning occurs when you have a goal and a purpose for a tool. How would *you* use Git to manage a project or process that is important to you?

Review and Outlook

- ▶ We have learned a little bit about the history and purpose of Git as a revision control system.
- ▶ We have seen ways to get Git installed on your system.
- ▶ We have learned about some of the basic commands (clone, add, commit, pull, push, status) that are routinely used together to manage changes to a project.
- ▶ We have seen some steps toward advanced usage (branch, merge, cherry-pick).
- ▶ Your homework: the best learning occurs when you have a goal and a purpose for a tool. How would *you* use Git to manage a project or process that is important to you?

Review and Outlook

- ▶ We have learned a little bit about the history and purpose of Git as a revision control system.
- ▶ We have seen ways to get Git installed on your system.
- ▶ We have learned about some of the basic commands (clone, add, commit, pull, push, status) that are routinely used together to manage changes to a project.
- ▶ We have seen some steps toward advanced usage (branch, merge, cherry-pick).
- ▶ Your homework: the best learning occurs when you have a goal and a purpose for a tool. How would *you* use Git to manage a project or process that is important to you?

References I