

Introduction to git (and GitHub): A Distributed Version Control System

Mahidhar Tatineni

High-Performance Computing User Services Group
San Diego Supercomputer Center
University of California, San Diego

SDSC Summer Institute 2023

August 9, 2023

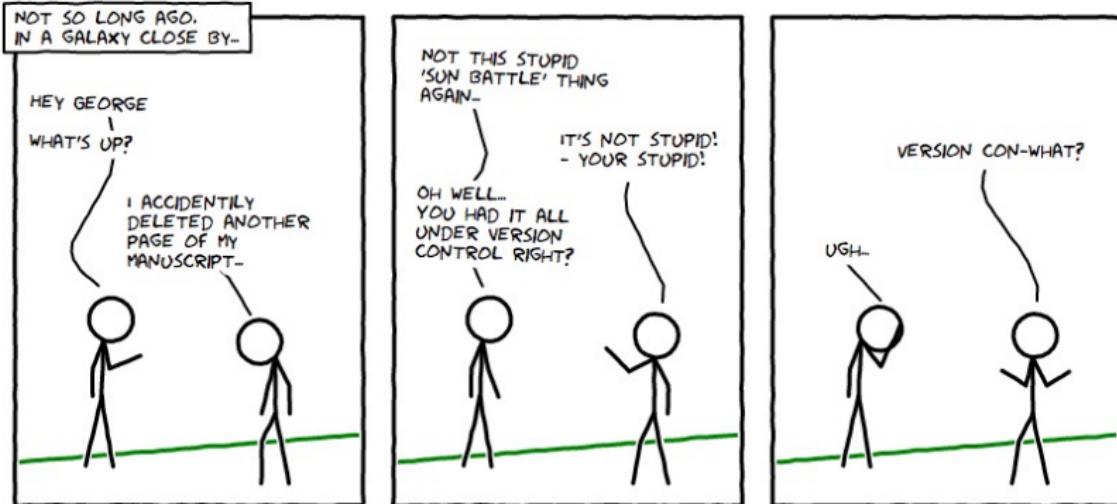
08:30AM - 10:00M PDT

Presentation based on talk by Martin Kandes at SI2021

Today's Session

- ▶ An Overview of Version Control
- ▶ About git
- ▶ Getting Started with git
- ▶ Working with git
- ▶ Getting Started with GitHub
- ▶ Summary and Conclusion
- ▶ Q&A

An Overview of Version Control



What is version control?

Version control is the process of recording and managing changes to a file or set of files over time so that you can recall specific versions later, if needed.

What do we version control?

The files we're typically interested in being version controlled are *software source code* files. However, you can version control almost any type of file.

What *should* we version control?

"In practice, everything that has been created manually should be put in version control, including programs, original field observations, and the source files for papers."

Best Practices for Scientific Computing
Wilson et al. 2012 (arXiv:1210.0530)

Have you used version control before?



Of course I have! Does this count?

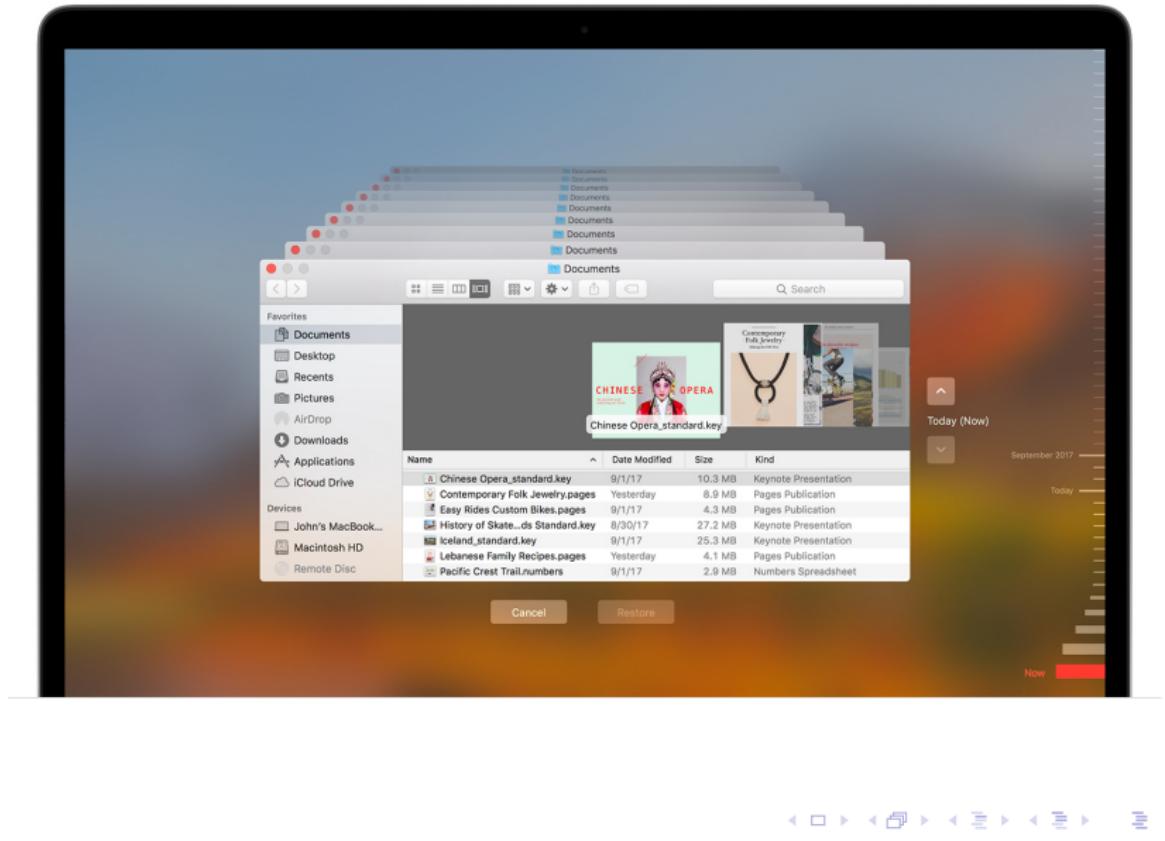
```
mkandes@castlebravo:~/Dropbox/documents/research/dissertation/code
mkandes@castlebravo:~$ ls
Desktop Downloads Dropbox Software ubuntu.def
mkandes@castlebravo:~$ cd ~/Dropbox/documents/research/dissertation/code
mkandes@castlebravo:~/Dropbox/documents/research/dissertation/code$ ls
gp3Drotate_v0060 gp3Drotate_v0073 gpse_v0.4.8.tar.gz gpse_v20120801
gp3Drotate_v0061 gpse gpse_v0.4.9.tar.gz gpse_v20120827
gp3Drotate_v0062 gpse_v0.2.7.tar.gz gpse_v0.5.1-1.tar.gz gpse_v20120828
gp3Drotate_v00625 gpse_v0.2.8.tar.gz gpse_v0.5.1.tar.gz gpse_v20120925
gp3Drotate_v0063 gpse_v0.2.9.tar.gz gpse_v0.5.3.tar.gz gpse_v20121115
gp3Drotate_v0064 gpse_v0.3.0.tar.gz gpse_v20120302 gpse_v20130122
gp3Drotate_v0065 gpse_v0.3.5.tar.gz gpse_v20120402 gpse_v20130522
gp3Drotate_v0066 gpse_v0.4.1.tar.gz gpse_v20120417 gpse_v20131218
gp3Drotate_v0068 gpse_v0.4.3.tar.gz gpse_v20120418 soa.tar.gz
gp3Drotate_v0069 gpse_v0.4.4.tar.gz gpse_v20120427 toa.tar.gz
gp3Drotate_v0071 gpse_v0.4.5.tar.gz gpse_v20120501
gp3Drotate_v00715 gpse_v0.4.6.tar.gz gpse_v20120614
gp3Drotate_v0072 gpse_v0.4.7.tar.gz gpse_v20120718
mkandes@castlebravo:~/Dropbox/documents/research/dissertation/code$
```

<https://github.com/mkandes/gpse>

What is a version control system?

- ▶ A *version control system* is a type of software tool that helps record and manage the changes you make to the file or set of files you want version controlled.
- ▶ Version control systems keeps track of *every* modification you make to the files being tracked in a special kind of database.
- ▶ If a mistake is made, you can then turn back the clock and compare earlier versions of the files to help fix the mistake.

Example: Apple's Time Machine



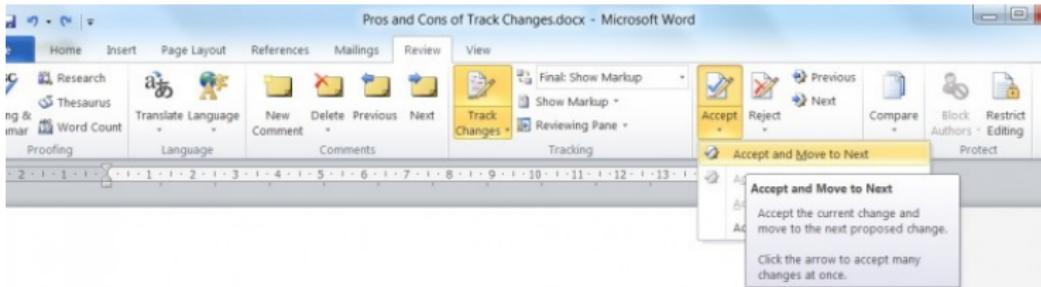
Example: Dropbox

The screenshot shows a browser window displaying the Dropbox website at <https://www.dropbox.com>. The user is signed in as Janey Smith. The main content is titled "Version history of 'Business overview.pptx' (follow renames)". It lists eight versions of the file, each with a thumbnail, the version number, the editor, the date edited, and the file size. Version 7 is the current version. Versions 1 through 6 are by Janey Smith (Work), and Version 1 is the oldest version added by her.

Version	Editor	Date	Size
Version 7 (current)	Edited by Janey Smith (Work)	1/8/2015 5:29 PM	3.54 MB
Version 6	Edited by Janey Smith (Work)	12/17/2014 9:42 AM	3.54 MB
Version 5	Edited by Janey Smith (Work)	12/17/2014 9:39 AM	3.54 MB
Version 4	Edited by Janey Smith (Work)	12/15/2014 2:38 PM	3.56 MB
Version 3	Edited by Janey Smith (Work)	12/5/2014 10:52 AM	3.16 MB
Version 2	Edited by Janey Smith (Work)	12/3/2014 3:12 PM	3.49 MB
Version 1 (oldest)	Added by Janey Smith (Work)	12/3/2014 12:30 PM	3.49 MB

At the bottom right are "Restore" and "Cancel" buttons.

Example: Microsoft Word's Track Changes



Some pros and cons of 'track-changes' feedback on work returned to students electronically

Despite the fact that 'track-changes' is normally used in one-to-one editing and feedback (for example on draft theses, dissertations, reports and so on) it seems likely that 'track-changes' feedback is already well on the way towards replacing 'handwritten comments on students' work in assessment in general. This short discussion is about using the 'track-changes' function in word-processing software to give students feedback when marking their work. This is normally when tutors use the 'track-changes' facilities to return to students their original word-processed assignments, duly edited with feedback comments which appear on-screen in another colour. The level of feedback can range from comments providing simple qualitative overall feedback on the whole document or on selected paragraphs or sentences, to very detailed feedback on individual words or phrases. This kind of feedback remains very valuable for large-scale work (essays, dissertations, long reports, drafts of articles for publication and so on).

The other side of 'track-changes' is where ~~deletions~~, additions, replaced words or phrases can be suggested, and the original author can accept or reject each change in turn, working

Formatted: Heading 1, Space Before 0 pt, After: 0 pt, Line spacing: single

Formatted: Font: (Default) Arial, 11

Comment [DA1]:

Formatted: Font: (Default) Arial, 11

Formatted: Font: (Default) Arial, 11
Highlight

Formatted: Font: (Default) Arial, 11

Formatted: Font: (Default) Arial, 11
pt, Italic

Formatted: Font: (Default) Arial, 11

Comment [DA2]:

Formatted: Font: (Default) Arial, 11

Example: Wikipedia's View History

Version control - Wikipedia - Mozilla Firefox

W Version control-Wikipedia X +

https://en.wikipedia.org/ Search

Most Visited

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Version control

From Wikipedia, the free encyclopedia

"Revision control system" redirects here. For the specific software implementation, see [Revision Control System](#).

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. (April 2011) (Learn how and when to remove this template message)

A component of software configuration management, **version control**, also known as **revision control** or **source control**,^[1] is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

The need for a logical way to organize and control revisions has existed for almost as long as writing has existed, but revision control became much more important, and complicated when the era of computing began. The numbering of book editions and of

Example: Wikipedia's View History

Version control: Revision history - Wikipedia - Mozilla Firefox

W Version control: Revision history X +

Most Visited

https://en.wikipedia.org/w/index.php?title=Version+control%3A+Revision+history&oldid=906311111

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Version control: Revision history

View logs for this page (view filter log)

Search for revisions

From year (and earlier): 2018 From month (and earlier):
all Tag filter: Show

For any version listed below, click on its date to view it.
For more help, see Help:Page history and Help>Edit summary.
External tools: Revision history statistics • Revision history search • Edits by user • Number of watchers • Page view statistics • Fix dead links

(cur) = difference from current version, (prev) = difference from preceding version,
m = minor edit, → = section edit, ← = automatic edit summary
(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 20:47, 26 July 2018 Elephanthunter (talk | contribs) . . (30,072 bytes) (-559) . . (Removing false claim that version control software is "essential" sourced from a company that made its revenue from version control software. It's very useful, but not essential.) (undo)
- (cur | prev) 20:58, 22 July 2018 InternetArchiveBot (talk | contribs) . . (30,631 bytes) (Revert to 20:47, 26 July 2018 [Elephanthunter]) (undo)

How does version control work? Record and Playback.

Version control systems start with a base version of a document and then simply record the changes you make each step of the way.

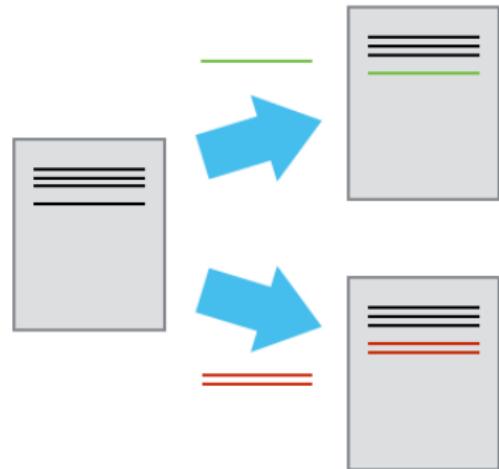


This allows you to rewind to start of the base document and playback each change you made, allowing you to recreate any version of the document in its recorded history.

How does version control work? Branching.

By recording the entire history of changes to a document and using the playback of those changes to recreate any version of it, the management and creation of new, independent versions or *branches* of the document itself becomes quite easy.

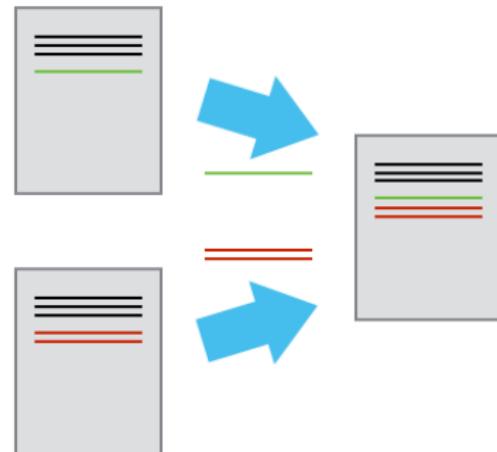
This *branching* helps facilitate the collaboration amongst multiple contributors to a single, shared document since each contributor simply needs to record their independent history of changes.



How does version control work? Merging.

When a contributor to a document wants to share their changes with their collaborators, they simply need to *merge* their changes into some common branch of the document shared by the team.

As long as there are no conflicts when the contributor's independent history of changes are played back on the common, shared branch of the document, the changes are *merged* and become a part of the shared history of the document.



Benefits of Using a Version Control System

- ▶ **Archiving:** You *must* regularly save the changes you make.
- ▶ **Reproducibility:** Creating a history of saved changes allows you to revert selected files or your entire project back to any previous state in its recorded history.
- ▶ **Collaboration:** You and a team of contributors can work on a project independently, but share your changes amongst one another as the project takes shape.
- ▶ **Accountability:** Compare changes, see who last modified something, and who introduced a problem and when.
- ▶ **Recoverability:** Each contributor has their own local, recent copy of the project and its complete history, making it highly unlikely you'll ever lose a significant portion of the project.



About git



<https://git-scm.com>

What is git?

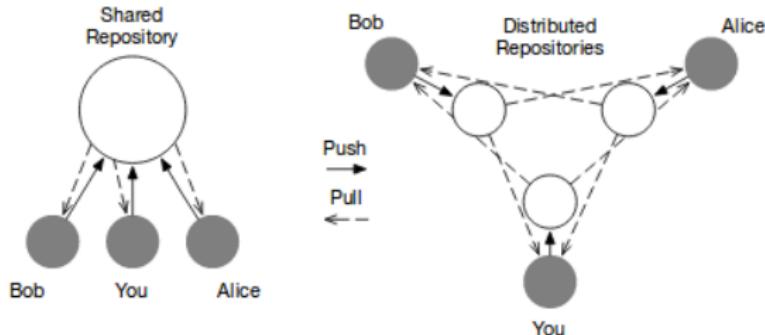
- ▶ git is a *distributed* version control system.
- ▶ git was designed to be *simple*, *fast*, and *fully-distributed*, with support for large projects and nonlinear development workflows.
- ▶ git was originally created in 2005 by Linus Torvalds and other Linux kernel developers when the free use of the proprietary version control system they had been using for kernel development was revoked by its copyright holder.

How is git different? Centralized vs. Distributed

In general, there are two models for version control systems:

- ▶ **Centralized:** Whenever you make a change and want to commit that change, you must send it back to a centralized *repository* server to be tracked. However, this makes sharing your changes with others who have access to the server immediate once the change is committed.
- ▶ **Distributed:** The process of committing changes and sharing them with others broken up into a two-step process. This allows you to commit changes in a *private repository* on your local computer, which eliminates the need to be connected to a network where the centralized server is accessible in order to commit a change.

How is git different? Private vs. Public Repositories



Gray circles are the private repositories; outlined circles are public repositories.

When using git, each contributor who is sharing changes with others on a project has at least two repositories:

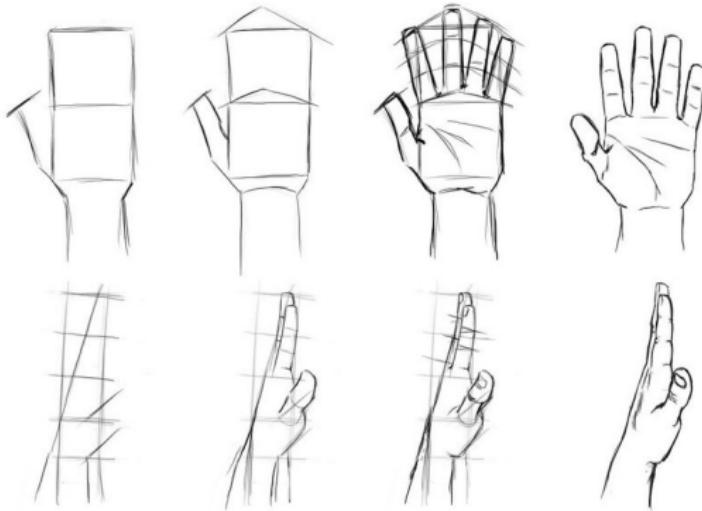
- ▶ a **private repository** is the one that exists on your local computer and is the one you make commits to;
- ▶ a **public repository** is the one that you use to share your changes with collaborators.

The git workflow

1. Update your private repository by fetching all recent changes committed by your team of collaborators from the shared public repository (or their individual public repositories).
2. Make changes and commit them to your private repository.
3. Review your committed changes.
4. Share your new changes with the team by pushing your committed changes back to the team's shared public repository (or your individual public repository; then submit a pull request from your public repository to the team's shared public repository).
5. Rinse and repeat.



Getting Started with git



Installing git on Linux

If you want to install git on a Linux-based system, you should be able to do so via your operating system's standard package management tool.

For example, on any RPM-based Linux distribution, such as Fedora, RHEL, or CentOS, you can use dnf:

```
$ sudo dnf install git-all
```

On any Debian-based distribution, such as Ubuntu, try apt:

```
$ sudo apt install git-all
```

Installing git on Mac OS X

There are several ways to install git on your Mac. Probably the easiest way is to install the Xcode Command Line Tools, which you can do by simply trying to run git from the Terminal:

```
$ git --version
```

If not already installed, you should be prompted to install it.

If the above option does not work or you need a more up-to-date version of git, you can always install it via a binary installer maintained by the git team, which is available for download at:

<https://git-scm.com/download/mac>

Installing git on Windows

There are also a few ways to install git under Windows operating systems. However, the official build is available for download on the git website. If you go to

<http://git-scm.com/download/win>

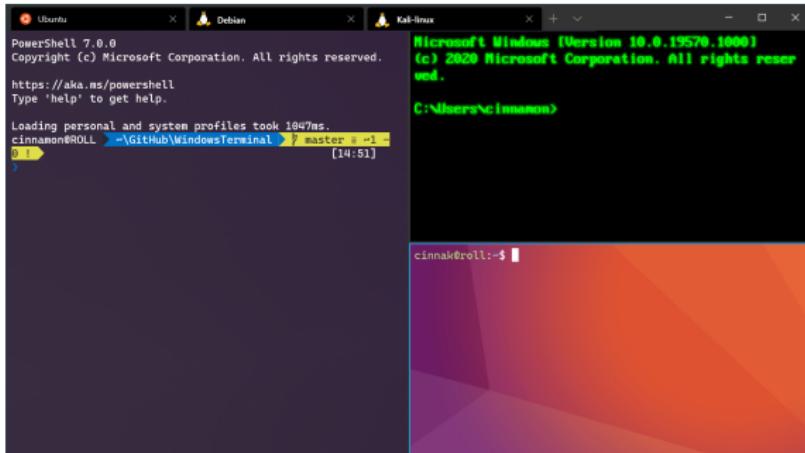
the download should start automatically.

Note: This is a project called “git for Windows”, which is separate from git itself. Please see

<https://gitforwindows.org>

for more information.

Windows Subsystem for Linux



The *Windows Subsystem for Linux* lets you run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot OS setup.

<https://docs.microsoft.com/en-us/windows/wsl>

Setting up git with git config

When we start using git for the first time on a new computer, there are always a few things we need to configure before we begin working with git.

The `git config` command lets you look up and set configuration variables that control all aspects of how git works.

Setting up git with git config

These configuration variables may be stored in a few different places on your system:

1. `/etc/gitconfig`: This file contains configuration variables applied to every user on the system and all of their repositories.
2. `~/.gitconfig`: This file contains configuration variables that apply only to you and **all** of your repositories on the system.
3. `~/path/to/your/repo/.git/config`: This file contains configuration variables that apply only to this repository.

Setting up your identity with git config

The first configuration variables you should set right after you install git for the first time are your `user.name` and `user.email` variables, which identify who you are and how to contact you.

```
$ git config --global user.name "Marty Kandes"  
$ git config --global user.email "mkandes@sdsc.edu"
```

These configuration variables are important because this information is included in every git commit you make in order to provide a history of who made what changes.

Note: The `--global` option used here places the values of these configuration variables in your personal `~/.gitconfig`.

Setting up your text editor with git config

After you've set up your identity, you can configure the default text editor that will be used by git when you need to edit a commit message, etc.

For example, if you want to use a different text editor, such as Emacs, you can type the following:

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

If not configured, git will use your system's default editor.

Setting up colors with git config

In addition to your identity and default text editor, you'll also probably want to make sure that you turn on color highlighting in git's user interface. You can do so with the following command:

```
$ git config --global color.ui "auto"
```

Checking your git config

To check the configuration variables you've set in your git config, you can run the command

```
$ git config --list
```

to list all of the variable values.

If you only need to check one variable in the configuration, you can always replace the `--list` option with the name of the configuration variable itself. For example, if you wanted to check the value of `user.email` you've set, you'd run the command:

```
$ git config user.email
```

Getting git help

If you ever need help while using git, you can access the comprehensive manual page (manpage) for any git command via the `git help` command:

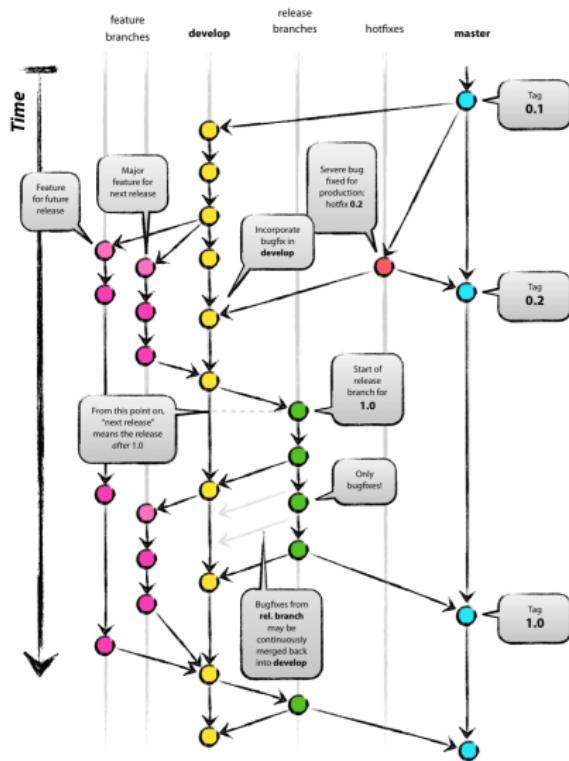
```
$ git help <command>
```

If you only need a quick reference to the options available for a given git command, you can use the `-h` or `--help` options appended to the command itself:

```
$ git <command> -h  
$ git <command> --help
```



Working with git



Creating your first git repository with git init

If you have a project on your local system that is not currently under version control and you want to start tracking it with git, you can create a git repository for that project by first going to that project's directory

```
$ cd /path/to/your/project
```

and then typing the command:

```
$ git init
```

This creates a new subdirectory named `.git` in the project directory where the repository will live. If you ever want to stop tracking the project with git, you simply need to delete this subdirectory.

Creating your first git repository with git init

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~$ ls
Desktop Downloads Dropbox Software VirtualBox VMs
mkandes@castlebravo:~$ cd ~/Desktop/
mkandes@castlebravo:~/Desktop$ ls
mkandes@castlebravo:~/Desktop$ mkdir planets
mkandes@castlebravo:~/Desktop$ cd planets/
mkandes@castlebravo:~/Desktop/planets$ ls -lahtr
total 8.0K
drwxr-xr-x 3 mkandes mkandes 4.0K Aug  5 19:14 ..
drwxrwxr-x 2 mkandes mkandes 4.0K Aug  5 19:14 .
mkandes@castlebravo:~/Desktop/planets$ git init
Initialized empty Git repository in /home/mkandes/Desktop/planets/.git/
mkandes@castlebravo:~/Desktop/planets$ ls -lahtr
total 12K
drwxr-xr-x 3 mkandes mkandes 4.0K Aug  5 19:14 ..
drwxrwxr-x 3 mkandes mkandes 4.0K Aug  5 19:14 .
drwxrwxr-x 7 mkandes mkandes 4.0K Aug  5 19:14 .git
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master

Initial commit

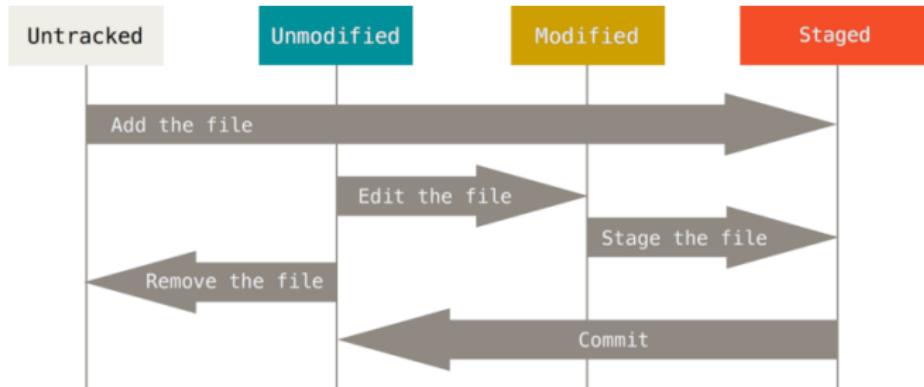
nothing to commit (create/copy files and use "git add" to track)
mkandes@castlebravo:~/Desktop/planets$
```

Checking the state your repository with git status

Each file in your repository's working directory may take on one of a few different states. The main tool you use to determine which files are in which state is with the command:

```
$ git status
```

Different states of a file in a git repository



There are two primary file states you will find in your repository:

1. **Tracked files** are all of the files that git already knows about and has under version control; they have secondary states, where they can be either *unmodified*, *modified*, or *staged*.
2. **Untracked files** are everything else — any files in your project's working directory not yet under version control.

Adding files to your git repository with git add

```
mkandes@castlebravo: ~/Desktop/planets

Initial commit

nothing to commit (create/copy files and use "git add" to track)
mkandes@castlebravo:~/Desktop/planets$ echo 'Cold and dry, but everything else i
s my favorite color.' > mars.txt
mkandes@castlebravo:~/Desktop/planets$ ls -lahtr
total 16K
drwxr-xr-x 3 mkandes mkandes 4.0K Aug  5 19:14 ..
drwxrwxr-x 7 mkandes mkandes 4.0K Aug  5 20:07 .git
-rw-rw-r-- 1 mkandes mkandes   56 Aug  5 20:08 mars.txt
drwxrwxr-x 3 mkandes mkandes 4.0K Aug  5 20:08 .
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master

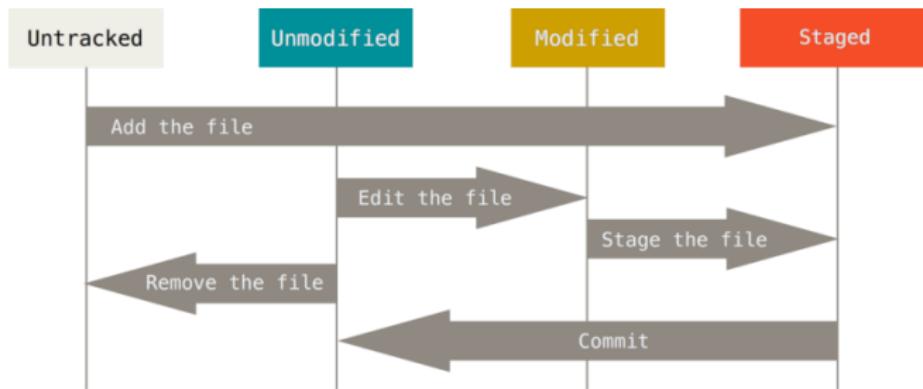
Initial commit

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

    mars.txt

nothing added to commit but untracked files present (use "git add" to track)
mkandes@castlebravo:~/Desktop/planets$
```

Adding files to your git repository with git add



To begin tracking a new file in your project's working directory, you must first *stage* the new file to be committed to the repository with the command:

```
$ git add <path/to/file/filename>
```

If you run `git status` again, you should see that the newly added file is now tracked and staged to be committed.

Adding files to your git repository with git add

```
mkandes@castlebravo: ~/Desktop/planets
drwxrwxr-x 3 mkandes mkandes 4.0K Aug  5 20:08 .
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master

Initial commit

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

    mars.txt

nothing added to commit but untracked files present (use "git add" to track)
mkandes@castlebravo:~/Desktop/planets$ git add mars.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master

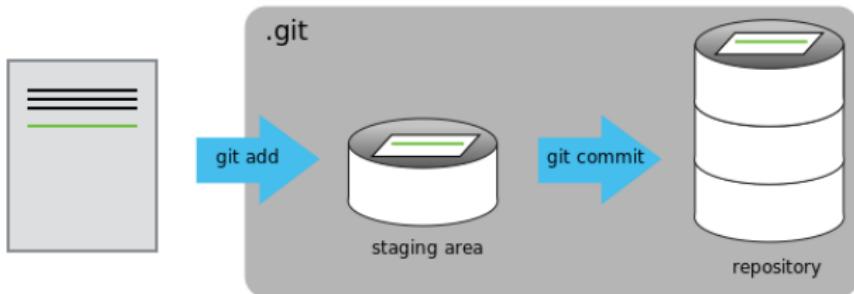
Initial commit

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

    new file:   mars.txt

mkandes@castlebravo:~/Desktop/planets$
```

Committing changes with git commit



Changes to files in your repository are tracked through commits, which you make with the

```
$ git commit
```

command. Prior to a commit, you need to stage the files whose changes you want to commit using the `git add` command.

Writing a commit message

Each commit requires a commit message. They provide the *context* about *why* a change was made. This may be to inform other collaborators working on the project about the change or, more often, to remind yourself why you made the change.

How you choose to format your commit messages is up to you and your team. But if you need some guidance, here are seven rules to writing a great git commit message:

<https://chris.beams.io/posts/git-commit>

Writing a commit message

Committing changes with git commit

```
mkandes@castlebravo: ~/Desktop/planets
On branch master

Initial commit

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

    new file:  mars.txt

mkandes@castlebravo:~/Desktop/planets$ git commit
[master (root-commit) d9995d3] Added my initial description of the planet Mars in mars.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mars.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$ git log
commit d9995d3e77f7c68f7a80b13bfca997008524ca3e
Author: Marty Kandes <mkandes@sdsc.edu>
Date:   Sun Aug 5 20:49:22 2018 -0700

    Added my initial description of the planet Mars in mars.txt
mkandes@castlebravo:~/Desktop/planets$
```

Viewing your commit history with git log

The power of git is that it tracks the changes to files in your project over time. You can use the

```
$ git log
```

command to view that history.

In the standard output of the git log command, you will see the commit ID of each commit, the author who made it, the date it was committed, and the commit message explaining the reason for the change(s).

By default, the git log command will list commit history in reverse chronological order.

Let's add some more files and make some changes ...

```
mkandes@castlebravo: ~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$ echo 'A pale blue dot, but it is home.' >
earth.txt
mkandes@castlebravo:~/Desktop/planets$ echo 'Not a planet.' > pluto.txt
mkandes@castlebravo:~/Desktop/planets$ echo 'Well, not the sunsets -- they are b
lue.' >> mars.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

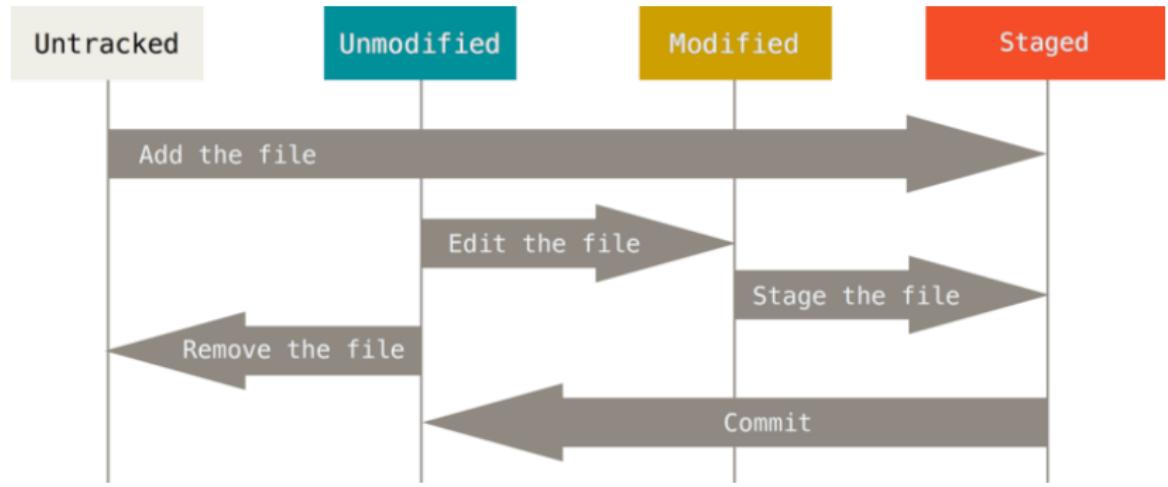
    modified:   mars.txt

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

    earth.txt
    pluto.txt

no changes added to commit (use "git add" and/or "git commit -a")
mkandes@castlebravo:~/Desktop/planets$
```

... then begin tracking the new files and stage the changes for the next commit ...



Staging new, untracked files and committing them ...

```
mkandes@castlebravo:~/Desktop/planets$ git add earth.txt pluto.txt
```

```
mkandes@castlebravo:~/Desktop/planets$ git commit -m "Added my initial descriptions of the planet Earth in earth.txt; also added my initial description of Pluto, even though it is only a minor planet now"
```

```
[master a1c283b] Added my initial descriptions of the planet Earth in earth.txt; also added my initial description of Pluto, even though it is only a minor planet now
```

```
2 files changed, 2 insertions(+)
```

```
create mode 100644 earth.txt
```

```
create mode 100644 pluto.txt
```

```
mkandes@castlebravo:~/Desktop/planets$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
      modified:   mars.txt
```

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

```
mkandes@castlebravo:~/Desktop/planets$ git log
```

```
commit a1c283b0eaebf7460ede1576c9525315aee1ae5a
```

```
Author: Marty Kandes <mkandes@sdsc.edu>
```

```
Date:  Sun Aug 5 22:32:43 2018 -0700
```

```
    Added my initial descriptions of the planet Earth in earth.txt; also added my initial description of Pluto, even though it is only a minor planet now
```

```
commit d9995d3e77f7c68f7a80b13bfca997008524ca3e
```

```
Author: Marty Kandes <mkandes@sdsc.edu>
```

```
Date:  Sun Aug 5 20:49:22 2018 -0700
```

```
    Added my initial description of the planet Mars in mars.txt
```

```
mkandes@castlebravo:~/Desktop/planets$
```



Comparing differences with git diff

While `git log` provides you with information about a commit — who made it, when they made it, and why they made it — sometimes looking at the specific changes made to the files themselves is what's more important.

You can use the

```
$ git diff
```

command to see the changes to any *modified* files in your working directory before you stage them to be committed.

Comparing differences with git diff

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

    modified:   mars.txt

no changes added to commit (use "git add" and/or "git commit -a")
mkandes@castlebravo:~/Desktop/planets$ git diff
diff --git a/mars.txt b/mars.txt
index 3f47692..ef4c79a 100644
--- a/mars.txt
+++ b/mars.txt
@@ -1 +1,2 @@
 Cold and dry, but everything else is my favorite color.
 +Well, not the sunsets -- they are blue.
mkandes@castlebravo:~/Desktop/planets$
```

Breaking down a git diff line-by-line

1. The first line tells us that git is comparing the old and new versions of the file with the `diff` command.
2. The second line tells us exactly which versions of the file are being compared.
3. The next two lines once again show the name of the file being changed.
4. The remaining lines show us the actual differences between the files and the lines on which they occur, with the `+` character in the first column showing where we added a new line of text.

After reviewing our change, it's time to commit it ...

```
mkandes@castlebravo: ~/Desktop/planets
diff --git a/mars.txt b/mars.txt
index 3f47692..ef4c79a 100644
--- a/mars.txt
+++ b/mars.txt
@@ -1 +1,2 @@
 Cold and dry, but everything else is my favorite color.
+Well, not the sunsets -- they are blue.
mkandes@castlebravo:~/Desktop/planets$ git add mars.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   mars.txt

mkandes@castlebravo:~/Desktop/planets$ git commit -m 'I forgot there is at least
one thing on Mars that is not my favorite color.'
[master 64b3df5] I forgot there is at least one thing on Mars that is not my fav
orite color.
 1 file changed, 1 insertion(+)
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$
```

Moving files and directories with git mv

Performing tasks such as reorganizing files under new directories, changing filenames, and so on requires that files or sometimes entire directories get moved from one place to another in your repository.

You can move a file (or directory) into another one with the command:

```
$ git mv <path/to/old/fname> <path/to/newfname>
```

git will stage the change for you after you call git mv. You must then call git commit afterwards to make the move permanent.

Moving files and directories with git mv

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ ls
earth.txt mars.txt pluto.txt
mkandes@castlebravo:~/Desktop/planets$ mkdir minor-planets
mkandes@castlebravo:~/Desktop/planets$ git mv pluto.txt minor-planets/pluto.txt
mkandes@castlebravo:~/Desktop/planets$ ls
earth.txt mars.txt minor-planets
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:   pluto.txt -> minor-planets/pluto.txt

mkandes@castlebravo:~/Desktop/planets$ git commit -m "Demoted pluto.txt to the new minor-planets subdirectory."
[master 40ee458] Demoted pluto.txt to the new minor-planets subdirectory.
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename pluto.txt => minor-planets/pluto.txt (100%)
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$ █
```

Removing files and directories with git rm

Files and directories in your repository sometimes outlive their usefulness. When this occurs, you can tell git to stop tracking changes to these files and remove them from the working tree of your repository — your copy of the repository at a particular point in its commit history; normally at the last commit.

To an remove an outdated file (or a directory) from the working tree of your repository, you use the command:

```
$ git rm -- <path/to/file/filename>
```

This **does not** remove the file from your repository's history; it only removes it from your working tree going forward

Like other git actions, you must run git commit after git rm to finalize the change.

Removing files and directories with git rm

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git rm -r -- minor-planets/
rm 'minor-planets/pluto.txt'
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    deleted:    minor-planets/pluto.txt

mkandes@castlebravo:~/Desktop/planets$ git commit -m "Stop tracking the minor planets."
[master 7bac3a3] Stop tracking the minor planets.
 1 file changed, 1 deletion(-)
 delete mode 100644 minor-planets/pluto.txt
mkandes@castlebravo:~/Desktop/planets$ git commit --amend -m "Stopped tracking the minor planets."
[master 3c8c1a7] Stopped tracking the minor planets.
Date: Sun Aug 5 23:49:40 2018 -0700
 1 file changed, 1 deletion(-)
 delete mode 100644 minor-planets/pluto.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$
```

Creating and switching between git branches

When you want to begin modifying the contents of your repository, but only want to do so to a limited subset of the files in it, you can begin using the *branching* and *merging* mechanisms of git.

To create a new *branch* in your repository, you use the command:

```
$ git branch <branch-name>
```

To then begin working on this new branch, you must switch over to this branch of this project using the command:

```
$ git checkout <branch-name>
```

The default branch in every git repository is the `master` branch.

Creating and switching between git branches

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git branch earth
mkandes@castlebravo:~/Desktop/planets$ git checkout earth
Switched to branch 'earth'
mkandes@castlebravo:~/Desktop/planets$ echo 'It is not flat.' >> earth.txt
mkandes@castlebravo:~/Desktop/planets$ vi earth.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch earth
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:   earth.txt

no changes added to commit (use "git add" and/or "git commit -a")
mkandes@castlebravo:~/Desktop/planets$ git add earth.txt
mkandes@castlebravo:~/Desktop/planets$ git commit -m "Created new branch for earth.txt; also added more Sagan to the first line of my description."
[earth 6d3b7a4] Created new branch for earth.txt; also added more Sagan to the first line of my description.
 1 file changed, 2 insertions(+), 1 deletion(-)
mkandes@castlebravo:~/Desktop/planets$ git log
commit 6d3b7a4a1649923fe294a3a40bd48abb276be9f0
Author: Marty Kandes <mkandes@sdsc.edu>
Date:   Mon Aug 6 00:05:17 2018 -0700
```

Creating and switching between git branches

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git branch
* earth
  master
mkandes@castlebravo:~/Desktop/planets$ git log -1
commit 6d3b7a4a1649923fe294a3a40bd48abb276be9f0
Author: Marty Kandes <mkandes@sdsc.edu>
Date:   Mon Aug 6 00:05:17 2018 -0700

    Created new branch for earth.txt; also added more Sagan to the first line of
    my description.
mkandes@castlebravo:~/Desktop/planets$ git checkout master
Switched to branch 'master'
mkandes@castlebravo:~/Desktop/planets$ git log -1
commit 3c8c1a7535887d7ffae18635a8c6761ec28059e8
Author: Marty Kandes <mkandes@sdsc.edu>
Date:   Sun Aug 5 23:49:40 2018 -0700

    Stopped tracking the minor planets.
mkandes@castlebravo:~/Desktop/planets$
```

Creating and switching between git branches

```
mkandes@castlebravo: ~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git branch
  earth
* master
mkandes@castlebravo:~/Desktop/planets$ echo 'It is more like an oblate spheroid.
' >> earth.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   earth.txt

no changes added to commit (use "git add" and/or "git commit -a")
mkandes@castlebravo:~/Desktop/planets$ git add earth.txt
mkandes@castlebravo:~/Desktop/planets$ git commit -m "Added a better description
of the shape of the Earth to earth.txt."
[master 48bbd18] Added a better description of the shape of the Earth to earth.t
xt.
 1 file changed, 1 insertion(+)
mkandes@castlebravo:~/Desktop/planets$
```

Merging commits between branches with git merge

Once you and/or your collaborators have made changes on a project in different branches, at some point you'll want to *merge* your changes together.

To *merge* the changes from different branches, you first need to *checkout* the branch you want the changes to be merged into.

```
$ git checkout <branch-name>
```

You then select the changes from another branch to be merged into this current branch.

```
$ git merge <branch-name-to-be-merged>
```

Note, however, if there is a *conflict* between the content the two branches, you must first resolve these conflicts prior to committing the merge.

Merging commits between branches with git merge

```
mkandes@castlebravo:~/Desktop/planets
mkandes@castlebravo:~/Desktop/planets$ git branch
  earth
* master
mkandes@castlebravo:~/Desktop/planets$ git merge earth
Auto-merging earth.txt
CONFLICT (content): Merge conflict in earth.txt
Automatic merge failed; fix conflicts and then commit the result.
mkandes@castlebravo:~/Desktop/planets$ cat earth.txt
<<<<< HEAD
A pale blue dot, but it is home.
It is more like an oblate spheroid.
=====
A pale blue dot suspended in a sunbeam., but it is home.
It is not flat.
>>>>> earth
mkandes@castlebravo:~/Desktop/planets$ git checkout earth
earth.txt: needs merge
error: you need to resolve your current index first
mkandes@castlebravo:~/Desktop/planets$ vi earth.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
```

Merging commits between branches with git merge

```
mkandes@castlebravo: ~/Desktop/planets
You have unmerged paths.
  (fix conflicts and run "git commit")

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

    both modified:  earth.txt

no changes added to commit (use "git add" and/or "git commit -a")
mkandes@castlebravo:~/Desktop/planets$ git add earth.txt
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

  modified:  earth.txt

mkandes@castlebravo:~/Desktop/planets$ git commit -m "Merged earth.txt from eart
h branch into master branch"
[master 38e66b9] Merged earth.txt from earth branch into master branch
mkandes@castlebravo:~/Desktop/planets$
```

Merging commits between branches with git merge

```
mkandes@castlebravo:~/Desktop/planets$ git status
modified:   earth.txt

mkandes@castlebravo:~/Desktop/planets$ git commit -m "Merged earth.txt from earth branch into master branch"
[master 38e66b9] Merged earth.txt from earth branch into master branch
mkandes@castlebravo:~/Desktop/planets$ git status
On branch master
nothing to commit, working directory clean
mkandes@castlebravo:~/Desktop/planets$ cat earth.txt
A pale blue dot suspended in a sunbeam, but it is home.
It is more like an oblate spheroid.
mkandes@castlebravo:~/Desktop/planets$ git checkout earth
Switched to branch 'earth'
mkandes@castlebravo:~/Desktop/planets$ cat earth.txt
A pale blue dot suspended in a sunbeam., but it is home.
It is not flat.
mkandes@castlebravo:~/Desktop/planets$ git checkout master
Switched to branch 'master'
mkandes@castlebravo:~/Desktop/planets$ git branch -d earth
Deleted branch earth (was 6d3b7a4).
mkandes@castlebravo:~/Desktop/planets$ git branch
* master
mkandes@castlebravo:~/Desktop/planets$
```



Getting Started with GitHub



<https://github.com>

Your GitHub Profile

mkanedes (Marty Kandes) - Mozilla Firefox

mkanedes (Marty Kandes) + GitHub, Inc. (US) https://github.com/mkanedes ... Search

Most Visited

Search or jump to... Pull requests Issues Marketplace Explore

Overview Repositories 8 Stars 0 Followers 3 Following 0

Popular repositories

naked-singularity
A repository of definition files for bootstrapping Singularity containers around the software applications, frameworks, and libraries you need to run on high-performance computing systems.

Shell ★ 3 6

zephyr
Provision cloud-based computing resources to augment existing, on-premise resources of an HTCondor batch system to meet user job demand.

Python ★ 1

factools
Forked from jdost321/factools
A repository of tools for GlideinWMS Factory Operations

Python

itpprp
Numerically approximate ground state solutions of the time-independent Gross-Pitaevskii equation in a polar coordinate system by using imaginary time propagation.

Fortran

gpse
Numerically approximate solutions of the time-dependent Gross-Pitaevskii equation in rotating frames of reference.

Fortran

condor_annex
There are many clouds like it, but this one is (not) mine.

Perl

48 contributions in the last year Contribution settings ▾

Add an authentication key to your account

The screenshot shows the GitHub Settings interface for the user 'mahidhar'. The left sidebar includes options like Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, and Sessions. The main content area is titled 'SSH keys' and contains a list of associated keys. A green button labeled 'New SSH key' is visible. One key listed is 'testkey', which is an SSH key with the SHA256 fingerprint: `SHA256:U0aNdD+YaLX030sMpLz+/wKYdofKPURw86CBNx/YQ8`. It was added on Aug 8, 2023, and is described as 'Last used within the last week — Read/write'. A 'Delete' button is located next to the key entry.

mahidhar (mahidhar)
Your personal account Switch to another account ▾

Go to your personal profile

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

testkey
SSH
SHA256:U0aNdD+YaLX030sMpLz+/wKYdofKPURw86CBNx/YQ8
Added on Aug 8, 2023
Last used within the last week — Read/write

Delete

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

Creating a New GitHub Repository

Create a New Repository - Mozilla Firefox

Create a New Repository X +

GitHub, Inc. (US) https://github.com/new ... Search

Most Visited

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

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

Owner mkandes / Repository name planets

Great repository names are short and memorable. Need inspiration? How about [didactic-umbrella](#).

Description (optional)

Example GitHub Repository for SDSC Summer Institute 2018

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

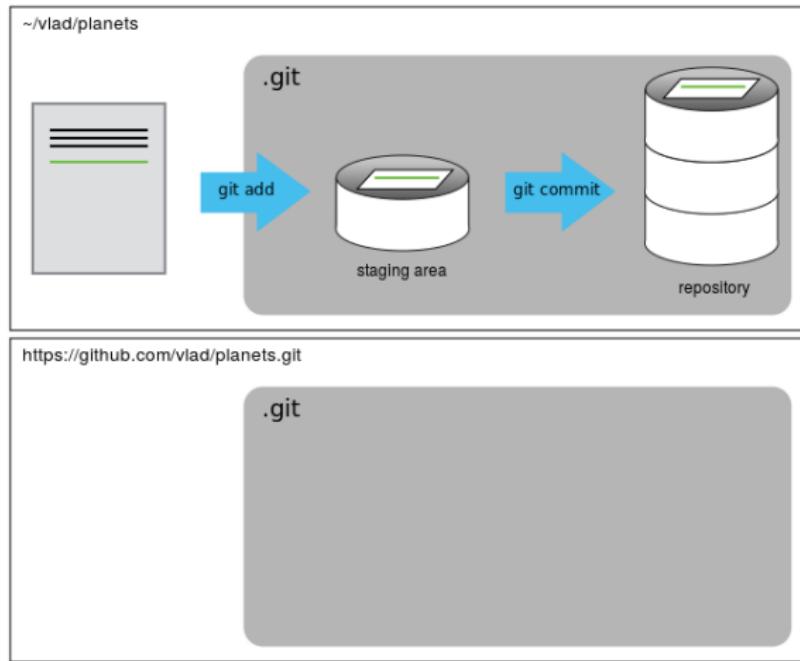
Private
You choose who can see and commit to this repository.

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

Add .gitignore: None Add a license: None

Create repository

Creating a New GitHub Repository



How to make your GitHub repository a remote

The screenshot shows a Firefox browser window with the URL <https://github.com/mkandes/planets>. The GitHub interface is visible, displaying the repository details for 'mkandes / planets'. The repository has 1 issue, 0 pull requests, 0 projects, 0 wiki pages, 0 insights, and 0 settings. A 'Quick setup — if you've done this kind of thing before' section provides instructions for cloning the repository using HTTPS or SSH, and recommends including README, LICENSE, and .gitignore files. Below this, a command-line section shows the steps to create a new repository: echo "# planets" >> README.md, git init, git add README.md, git commit -m "first commit", git remote add origin https://github.com/mkandes/planets.git, and git push -u origin master. Another section shows how to push an existing repository from the command line: git remote add origin https://github.com/mkandes/planets.git and git push -u origin master. The final section shows how to import code from another repository, with a note that it can be initialized from Subversion, Mercurial, or TFS projects.

mkandes/planets - Mozilla Firefox

mkandes/planets

GitHub, Inc. (US) https://github.com/mkandes/planets

Search

Most Visited

Search or jump to...

Pull requests Issues Marketplace Explore

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** <https://github.com/mkandes/planets.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# planets" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/mkandes/planets.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/mkandes/planets.git
git push -u origin master
```

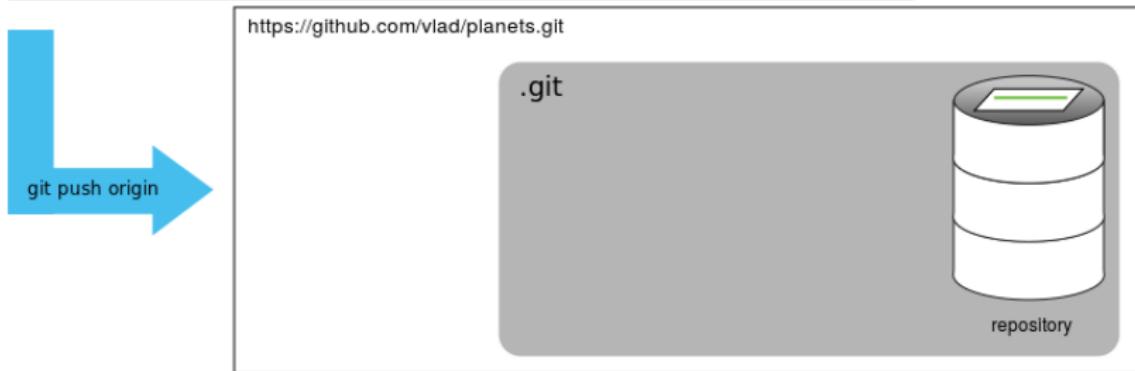
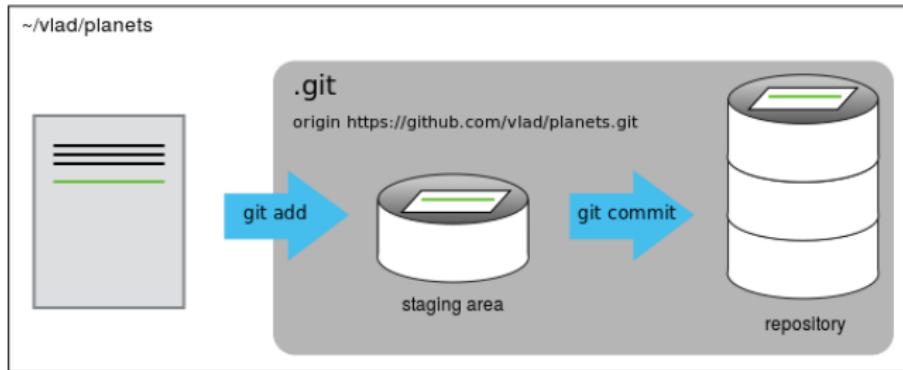
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

How to make your GitHub repository a remote

```
[train106@login01 gittest]$ git remote add origin git@github.com:mahidhar/gittest.git
[train106@login01 gittest]$ git push -u origin master
SSH: Server;Ltype: Version;Remote: 192.30.255.112-22;Protocol: 2.0;Client: babeld-d815c248
The authenticity of host 'github.com (192.30.255.112)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeI0ttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,192.30.255.112' (ECDSA) to the list of known hosts.
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 64 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (22/22), 2.06 KiB | 1.03 MiB/s, done.
Total 22 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To github.com:mahidhar/gittest.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[train106@login01 gittest]$ git branch -a
  br1
* master
  remotes/origin/master
```

Your GitHub repository after the first git push



Your GitHub repository after the first git push

The screenshot shows a GitHub repository page for the user 'mahidhar' named 'gittest'. The repository is public. The master branch has one commit. The commit details show the addition of 'earth.txt' and 'mars.txt' files. A callout box at the bottom encourages adding a README file.

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

gitest Public

master 1 branch 0 tags

Go to file Add file ▾ < Code ▾

mahidhar earth.txt	8b83895 21 minutes ago	8 commits
earth.txt	committed new earth file to branch br1	26 minutes ago
mars.txt	added to mars.txt	32 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

Cloning a GitHub repository with git clone

```
mkandes@comet-ln3:~/planets
*****
[mkandes@comet-ln3 ~]$ git clone https://github.com/mkandes/planets.git
Cloning into 'planets'...
fatal: unable to access 'https://github.com/mkandes/planets.git/': SSL connect error
[mkandes@comet-ln3 ~]$ git config --global http.sslVersion "tlsv1"
[mkandes@comet-ln3 ~]$ git clone https://github.com/mkandes/planets.git
Cloning into 'planets'...
remote: Counting objects: 24, done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 24 (delta 2), reused 24 (delta 2), pack-reused 0
Unpacking objects: 100% (24/24), done.
[mkandes@comet-ln3 ~]$ cd planets/
[mkandes@comet-ln3 planets]$ git config --global user.name "Marty Kandes"
[mkandes@comet-ln3 planets]$ git config --global user.email "mkandes@sdsc.edu"
[mkandes@comet-ln3 planets]$ git log -1
commit 38e66b9c50d3e729afab9f7352167dc70d3b7937 (HEAD -> master, origin/master,
origin/HEAD)
Merge: 48bbd18 6d3b7a4
Author: Marty Kandes <mkandes@sdsc.edu>
Date:   Mon Aug 6 00:31:16 2018 -0700

    Merged earth.txt from earth branch into master branch
[mkandes@comet-ln3 planets]$ █
```

Modify cloned repository and push changes back to GitHub

```
mkanedes@comet-ln3:~/planets
[mkandes@comet-ln3 planets]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
[mkandes@comet-ln3 planets]$ echo "It's where Cohaagen's big secret is buried."
>> mars.txt
[mkandes@comet-ln3 planets]$ git diff
diff --git a/mars.txt b/mars.txt
index ef4c79a..e1223fe 100644
--- a/mars.txt
+++ b/mars.txt
@@ -1,2 +1,3 @@
 Cold and dry, but everything else is my favorite color.
 Well, not the sunsets -- they are blue.
+It's where Cohaagen's big secret is buried.
[mkandes@comet-ln3 planets]$ git add mars.txt
[mkandes@comet-ln3 planets]$ git commit -m 'Come on, Cohaagen! You got what you
want. Give those people air!'
[master 386a046] Come on, Cohaagen! You got what you want. Give those people air!
1 file changed, 1 insertion(+)
[mkandes@comet-ln3 planets]$ git push
Username for 'https://github.com': mkandes
Password for 'https://mkandes@github.com':
Counting objects: 3, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 385 bytes | 385.00 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/mkandes/planets.git
    38e66b9..386a046  master -> master
[mkandes@comet-ln3 planets]$
```

GitHub repository after recent push

mkanedes/planets: Example GitHub Repository for SDSC Summer Institute 2018 - Mozilla Firefox

mkandes/planets: Exam X +

GitHub, Inc. (US) https://github.com/mkandes/planets ... git commit message ↗

Most Visited

Search or jump to... Pull requests Issues Marketplace Explore

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Example GitHub Repository for SDSC Summer Institute 2018 Edit

Add topics

9 commits 1 branch 0 releases 1 contributor

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

 mkanedes Come on, Cohaagen! You got what you want. Give those people air! Latest commit 386a046 a minute ago

 earth.txt Merged earth.txt from earth branch into master branch an hour ago

 mars.txt Come on, Cohaagen! You got what you want. Give those people air! a minute ago

Help people interested in this repository understand your project by adding a README. Add a README



Summary and Conclusion: Benefits of Version Control

- ▶ **Archiving:** You *must* regularly save the changes you make.
- ▶ **Reproducibility:** Creating a history of saved changes allows you to revert selected files or your entire project back to any previous state in its recorded history.
- ▶ **Collaboration:** You and a team of contributors can work on a project independently, but share your changes amongst one another as the project takes shape.
- ▶ **Accountability:** Compare changes, see who last modified something, and who introduced a problem and when.
- ▶ **Recoverability:** Each contributor has their own local, recent copy of the project and its complete history, making it highly unlikely you'll ever lose a significant portion of the project.



References

- ▶ **Version Control with Git** by D.Huang and I. Gonzalez
[https://swcarpentry.github.io/
git-novice/](https://swcarpentry.github.io/git-novice/)
- ▶ **Pragmatic Version Control Using Git**
by T. Swicegood
- ▶ **Pro Git** by S. Chacon and B. Straub
<https://git-scm.com/book/en/v2>

