Git: The stupid content tracker

Joseph Hallett

January 10, 2023

What's this all about?

Writing source code is hard.
We need a mechanism to systematically track changes.

Luckilly

Software Engineers solved this problem back in the 70s.

- (and perfected it in the 2000s)

For example

Suppose *Alice* goes and writes the following program:

```
package uk.ac.bristol.cs.SoftwareTools;
public class Hello {
    public static void main(String[] args) {
        if (args.length == 0) {
            args = new String[1];
            args[0] = "World";
        }
        for (final var name : args)
            System.out.print("Hello "+name+"!\n");
    }
}
```

```
Hello World!
```

Later updates

Later she makes a new version of her program... Whats changed?

```
package uk.ac.bristol.cs.SoftwareTools;
public class Hello2 {
    public static void main(String[] args) {
        if (args.length == 0) {
            args = new String[1];
            args[0] = "World";
        }
        for (final var name : args)
            System.out.println("Hello "+name+"!");
    }
}
```

```
Hello World!
```

A bad solution

We *could* go and track changes manually…

- Each version of the file has a different name with a number on the end
- Write a suite of tools for spotting what the differences in files are…

  `diff uk/ac/bristol/cs/SoftwareTools/Hello*.java`

  2c2 < public class Hello { — > public class Hello2 { 9c9 < System.out.print(”Hello ”~~name~~”!”); — > System.out.println(”Hello ”~~name~~”!”);

```
2c2
< public class Hello {
---
> public class Hello2 {
9c9
<               System.out.print("Hello "+name+"!\n");
---
>               System.out.println("Hello "+name+"!");
```

Suppose Bob forks the code?

Suppose *Bob* takes *Alice*'s original program and makes his own changes?

```
package uk.ac.bristol.cs.SoftwareTools;
import java.util.*;
public class Hello2 {
    public static void main(String[] args) {
        final var people = new LinkedList<String>();
        people.addAll(Arrays.asList(args));
        if (people.size() == 0)
            people.add("World");
        for (final var name : people)
            System.out.print("Greetings "+name+".\n");
    }
}
```

```
Greetings World.
```

- How are we going to *merge* Alice's changes with Bob's?
- How do we deal with *divergence*?

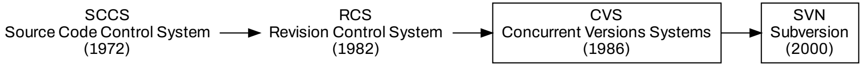
Don’t work hard! Work Lazy!

Clearly managing source code like this is going to be a lot of manual work.
But we’re *computer scientists*… we can automate *anything*.

So lets do that!

- Write software to do all the management of software for you
- Let it keep track of who has changed what and when
- Let the programmer step in and fix things as a last resort

Version Control Systems



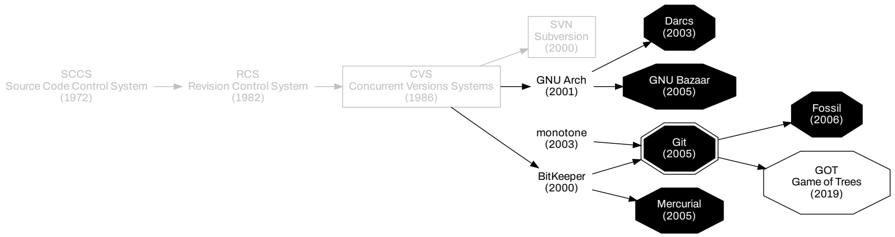
Initially all the version control systems are *centralised*...

- that is they each have an *official* central repository that stores the latest versions.

Decentralised Version Control Systems

But around 2000 we start to see a shift away from *centralised* models to decentralised ones

- Every user has a *master* version of the source control
- Changes are accepted from other people through *merges*



Git

Linus Torvalds develops Git to help with the development of the Linux Kernel.

- The kernel is developed by taking *diffs* of source code with the changes you want to make
- Email whoevers in charge the bit of the kernel you want to change with the changes and an explanation
- If they take the changes they email the changes to *Linus* to merge into his tree

Git is designed to be a tool to help Linus do his job

- Not designed to be user friendly
- Worse is better
- Fast for working with plaintext files (source code)
- Works well with *huge* numbers of files
- Source code isn't that complex

This is *still* how the kernel gets developed!

Modern Git

Whilst the *email-based* workflow is still used… there are now alternatives

- Git *forges* offer an alternative to email-based workflows
  - Of which the most popular is Microsoft's *GitHub*
- Terminal commands have been made more usable
- GUIs for those who like them
  - (But learn the command line too…)
- Editor plugins for those who like them
  - (If you use Emacs try *Magit*…)

If in doubt: `man 1 git`

```
GIT(1)                         Git Manual                         GIT(1)

NAME
     git - the stupid content tracker

SYNOPSIS
     git [-v | --version] [-h | --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>]



DESCRIPTION
     Git is a fast, scalable, distributed revision control system with an
     unusually rich command set that provides both high-level operations and
     full access to internals.

     See gittutorial(7) to get started, then see giteveryday(7) for a useful
```

If in further doubt...

`https://git-scm.com/book/en/v2` The *official* Git book. Free and well written. Great for understanding internals

`https://ohshitgit.com` A guide for how to get out of silly situations in Git. Somewhat sweary.

Okay lets get started!

To create a *Git repo* we can use the `git init` command:

```
mkdir tutorial
cd tutorial
git init
```

```
Initialized empty Git repository in
private/tmp/tutorial.git/
```

```
ls -a
```

```
. .. .git
```

```
git status
```

```
On branch main
No commits yet
nothing to commit (create/copy files and use
"git add" to track)
```

Lets add some code

```
cat >hello.c <<EOF
#include <stdio.h>

int main(void) {
    printf("Hello, World\n");
    return 0;
}
EOF
git add hello.c
git status
```

```
On branch main
No commits yet
Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: hello.c
```

Staging

At this point, the file `hello.c` is *staged* but it hasn't been *commited* yet.
When you *stage* a file:

- You're saying this will be part of a new commit
- You're adding the changes into Git's versioning
- But you're not saving anything
- Things can still change!

When you *commit*:

- Everything you've staged so far gets written into the history as a single change.
- With a note explaining it
- And your name associated with it
- Things *shouldn't* change
  - (techincally they still can… but it gets harder)

Lets commit!

```
git commit -m 'Initial commit of the greeting program.

Greets the user and then exits.'
```

```
[main (root-commit) b377fa3] Initial commit of the greeting program. 1 file changed, 6
insertions(+) create mode 100644 hello.c
```

Note

Sometimes when your on a new system you'll get a prompt to set your name and email… just follow the instructions provided. All Git commits need a name and an email address attributed to them.

```
git config --global user.name 'Joseph Hallett'
git config --global user.email 'joseph.hallett@bristol.ac.uk'
```

Lets make some edits

```
ed hello.c <<EOS
3c
int main(int argc, char *argv[]) {
.
4c
    for (int i=0; i<argc; i++)
        printf("Hello, %s\n", argv[i]);
.
wq
EOS
```

```
83 142
```

```
git add hello.c
git commit -m "Greets all the people passed."
```

```
[main 8f3fafd] Greets all the people passed.
1 file changed, 3 insertions(+), 2
deletions(-)
```

```
make hello
./hello Alice Bob
```

```
cc hello.c -o hello Hello, ./hello Hello,
Alice Hello, Bob
```

One more edit...

```
ed hello.c <<EOS
4s/0/1/
wq
EOS
git add hello.c
git commit -m "Stops greeting the program itself."
```

```
142 142 [main 24c96de] Stops greeting the
program itself. 1 file changed, 1
insertion(+), 1 deletion(-)
```

```
make hello
./hello Alice Bob
```

```
make: `hello' is up to date. Hello, ./hello
Hello, Alice Hello, Bob
```

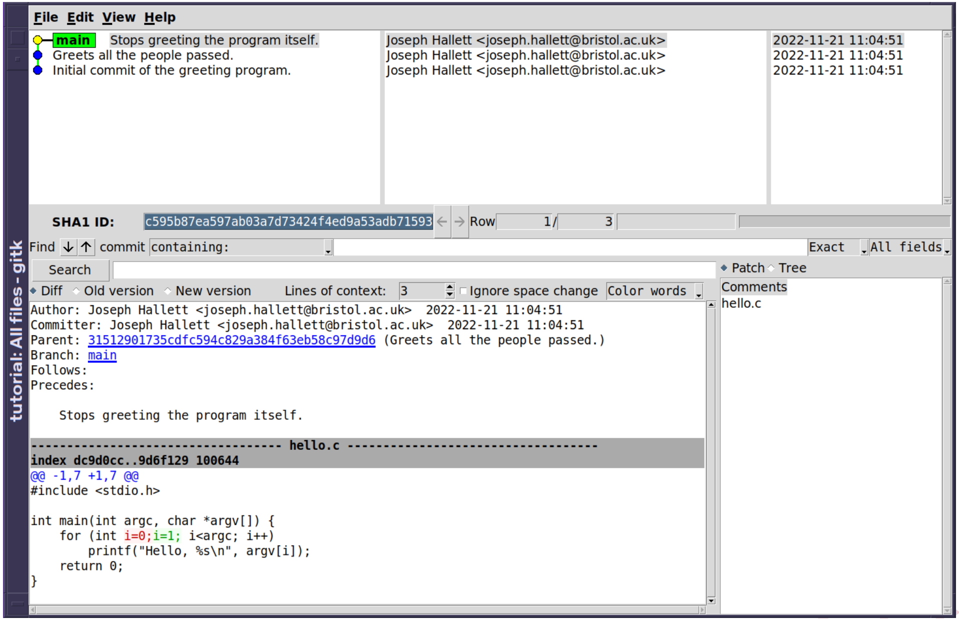
So what have we done?

So far we've made three changes to our code: lets see what these look like in Git!

```
git log --oneline | cat
```

```
24c96de Stops greeting the program itself. 8f3fafd Greets all the people passed. b377fa3
Initial commit of the greeting program.
```

Or just use gitk



```
main  Stops greeting the program itself.   Joseph Hallett <joseph.hallett@bristol.ac.uk>   2022-11-21 11:04:51
      Greets all the people passed.         Joseph Hallett <joseph.hallett@bristol.ac.uk>   2022-11-21 11:04:51
      Initial commit of the greeting program.  Joseph Hallett <joseph.hallett@bristol.ac.uk>   2022-11-21 11:04:51

SHA1 ID: c595b87ea597ab03a7d73424f4ed9a53adb71593   Row 1/ 3

Lines of context: 3

Author: Joseph Hallett <joseph.hallett@bristol.ac.uk>  2022-11-21 11:04:51
Committer: Joseph Hallett <joseph.hallett@bristol.ac.uk>  2022-11-21 11:04:51
Parent: 31512901735cdfc594c829a384f63eb58c97d9d6 (Greets all the people passed.)
Branch: main
Follows:
Precedes:

    Stops greeting the program itself.

----------------------------------- hello.c ----------------------------------
index dc9d0cc..9d6f129 100644
@@ -1,7 +1,7 @@
#include <stdio.h>

int main(int argc, char *argv[]) {
    for (int i=0;i=1; i<argc; i++)
        printf("Hello, %s\n", argv[i]);
    return 0;
}
```

Tags, branches and `HEAD`...

Commits are all *identified* by their hash...

- but you can name specific commits by using the `git tag` command
- (this is useful for marking releases or submitted versions of your code)

All commits are made to a *branch* which is a *tag*

- When the commit is made the *branch* tag is *updated* to point to the new commit at the top of *branch*.
- The *default* branch is usually called `main` (or `master`)
- (This is wrong in all important respects; but it's an okay simplification)

There is also a *special* tag called `HEAD`

- Always points to wherever your code is currently at
- Minus any unstaged work

Working with commits

Say you've made a bunch of changes to a file, but not commited them. You'd like to threw away the changes you made:

```
git checkout HEAD -- hello.c
```

Or if you've changed a lot of stuff and want to go back to clean:

```
git reset --hard HEAD # Remove all changes
git clean -dfx        # Delete all untracked files
```

```
HEAD is now at 24c96de Stops greeting the program itself. Removing hello
```

Say you'd like to go back to how the code was *before* the last commit:

```
git checkout HEAD~1
```

Say you're done looking at the code in an old state, and want to go back to working on the `main` branch:

```
git checkout main
```

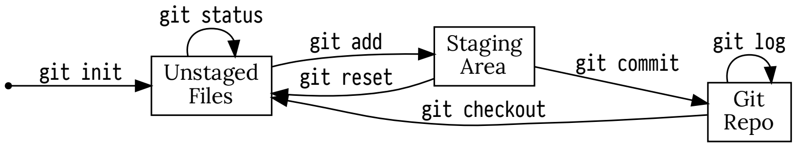
```
Say a commit was a horrible mistake and you'd like to apply it in reverse and undo all the changes of it:
```

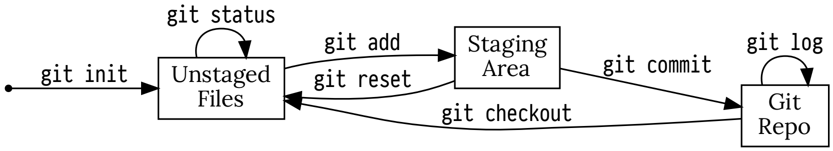
```
git revert HEAD
```

```
[main 3d0eaae] Revert "Stops greeting the program itself." Date: Tue Jan 10 11:33:04
2023 ~~0000 1 file changed, 1 insertion(~~), 1 deletion(-)
```

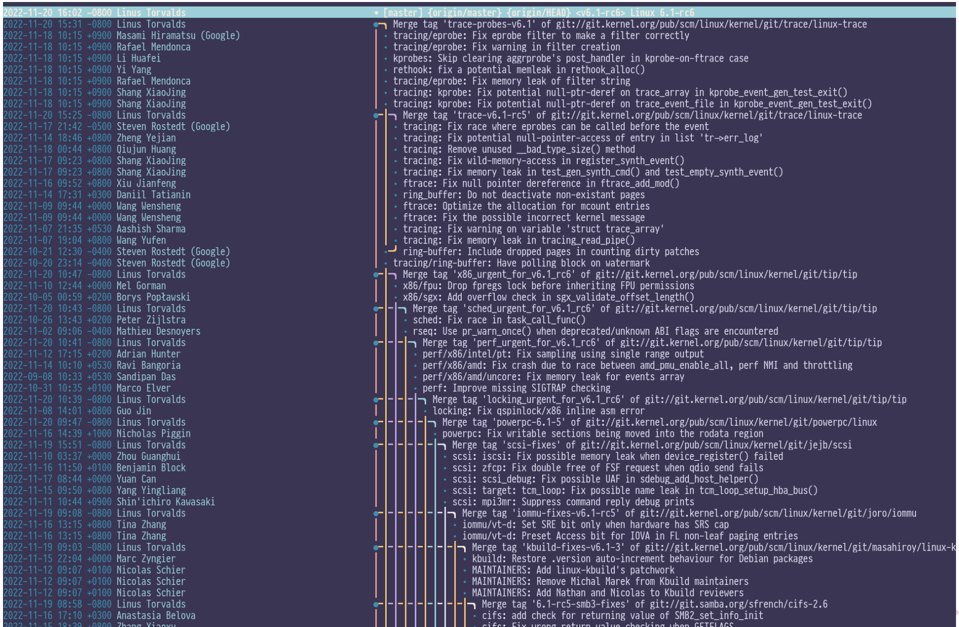
Top Tips

1. Write good descriptive commit messages (`updates` is not a good message)
2. Never commit broken code (if it doesn't *at least* comple don't commit yet)
3. Read the `man git` pages (git is fiddly!)





One more thing…



```
2022-11-20 16:02 -0800 Linus Torvalds             • [master] {origin/master} {origin/HEAD} <v6.1-rc6> Linux 6.1-rc6
2022-11-20 15:31 -0800 Linus Torvalds             Merge tag 'trace-probes-v6.1' of git://git.kernel.org/pub/scm/linux/kernel/git/trace/linux-trace
2022-11-18 10:15 +0900 Masami Hiramatsu (Google)    tracing/eprobe: Fix eprobe filter to make a filter correctly
2022-11-18 10:15 +0900 Rafael Mendonca              tracing/eprobe: Fix warning in filter creation
2022-11-18 10:15 +0900 Li Huafei                    kprobes: Skip clearing aggrprobe's post_handler in kprobe-on-ftrace case
2022-11-18 10:15 +0900 Yi Yang                      rethook: fix a potential memleak in rethook_alloc()
2022-11-18 10:15 +0900 Rafael Mendonca              tracing/eprobe: Fix memory leak of filter string
2022-11-18 10:15 +0900 Shang XiaoJing               tracing: kprobe: Fix potential null-ptr-deref on trace_array in kprobe_event_gen_test_exit()
2022-11-18 10:15 +0900 Shang XiaoJing               tracing: kprobe: Fix potential null-ptr-deref on trace_event_file in kprobe_event_gen_test_exit()
2022-11-20 15:25 -0800 Linus Torvalds             Merge tag 'trace-v6.1-rc5' of git://git.kernel.org/pub/scm/linux/kernel/git/trace/linux-trace
2022-11-17 21:42 -0500 Steven Rostedt (Google)      tracing: Fix race where eprobes can be called before the event
2022-11-14 18:46 +0800 Zheng Yejian                 tracing: Fix potential null-pointer-access of entry in list 'tr->err_log'
2022-11-18 00:44 +0800 Qiujun Huang                 tracing: Remove unused __bad_type_size() method
2022-11-17 09:23 +0800 Shang XiaoJing               tracing: Fix wild-memory-access in register_synth_event()
2022-11-17 09:23 +0800 Shang XiaoJing               tracing: Fix memory leak in test_gen_synth_cmd() and test_empty_synth_event()
2022-11-16 09:52 +0800 Xiu Jianfeng                 ftrace: Fix null pointer dereference in ftrace_add_mod()
2022-11-14 17:31 +0300 Daniil Tatianin              ring_buffer: Do not deactivate non-existant pages
2022-11-09 09:44 +0000 Wang Wensheng                ftrace: Optimize the allocation for mcount entries
2022-11-09 09:44 +0000 Wang Wensheng                ftrace: Fix the possible incorrect kernel message
2022-11-07 21:35 +0530 Aashish Sharma               tracing: Fix warning on variable 'struct trace_array'
2022-11-07 19:04 +0800 Wang Yufen                   tracing: Fix memory leak in tracing_read_pipe()
2022-10-21 12:30 -0400 Steven Rostedt (Google)      ring-buffer: Include dropped pages in counting dirty patches
2022-10-20 23:14 -0400 Steven Rostedt (Google)    tracing/ring-buffer: Have polling block on watermark
2022-11-20 10:47 -0800 Linus Torvalds             Merge tag 'x86_urgent_for_v6.1_rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
2022-11-10 12:44 +0000 Mel Gorman                   x86/fpu: Drop fpregs lock before inheriting FPU permissions
2022-10-05 00:59 +0200 Borys Popławski              x86/sgx: Add overflow check in sgx_validate_offset_length()
2022-11-20 10:43 -0800 Linus Torvalds             Merge tag 'sched_urgent_for_v6.1_rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
2022-10-26 13:43 +0200 Peter Zijlstra               sched: Fix race in task_call_func()
2022-11-02 09:06 -0400 Mathieu Desnoyers            rseq: Use pr_warn_once() when deprecated/unknown ABI flags are encountered
2022-11-20 10:41 -0800 Linus Torvalds             Merge tag 'perf_urgent_for_v6.1_rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
2022-11-12 17:15 +0200 Adrian Hunter                perf/x86/intel/pt: Fix sampling using single range output
2022-11-14 10:10 +0530 Ravi Bangoria                perf/x86/amd: Fix crash due to race between amd_pmu_enable_all, perf NMI and throttling
2022-09-08 10:33 +0530 Sandipan Das                 perf/x86/amd/uncore: Fix memory leak for events array
2022-10-31 10:35 +0100 Marco Elver                  perf: Improve missing SIGTRAP checking
2022-11-20 10:39 -0800 Linus Torvalds             Merge tag 'locking_urgent_for_v6.1_rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
2022-11-08 14:01 +0800 Guo Jin                      locking: Fix qspinlock/x86 inline asm error
2022-11-20 09:47 -0800 Linus Torvalds             Merge tag 'powerpc-6.1-5' of git://git.kernel.org/pub/scm/linux/kernel/git/powerpc/linux
2022-11-16 14:39 +1000 Nicholas Piggin              powerpc: Fix writable sections being moved into the rodata region
2022-11-19 15:51 -0800 Linus Torvalds             Merge tag 'scsi-fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/jejb/scsi
2022-11-10 03:37 +0000 Zhou Guanghui                scsi: iscsi: Fix possible memory leak when device_register() failed
2022-11-16 11:50 +0100 Benjamin Block               scsi: zfcp: Fix double free of FSF request when qdio send fails
2022-11-17 08:44 +0000 Yuan Can                     scsi: scsi_debug: Fix possible UAF in sdebug_add_host_helper()
2022-11-15 09:50 +0800 Yang Yingliang               scsi: target: tcm_loop: Fix possible name leak in tcm_loop_setup_hba_bus()
2022-11-11 10:44 +0900 Shin'ichiro Kawasaki         scsi: mpi3mr: Suppress command reply debug prints
2022-11-19 09:08 -0800 Linus Torvalds             Merge tag 'iommu-fixes-v6.1-rc5' of git://git.kernel.org/pub/scm/linux/kernel/git/joro/iommu
2022-11-16 13:15 +0800 Tina Zhang                   iommu/vt-d: Set SRE bit only when hardware has SRS cap
2022-11-16 13:15 +0800 Tina Zhang                   iommu/vt-d: Preset Access bit for IOVA in FL non-leaf paging entries
2022-11-19 09:03 -0800 Linus Torvalds             Merge tag 'kbuild-fixes-v6.1-3' of git://git.kernel.org/pub/scm/linux/kernel/git/masahiroy/linux-k
2022-11-15 22:04 +0000 Marc Zyngier                 kbuild: Restore .version auto-increment behaviour for Debian packages
2022-11-12 09:07 +0100 Nicolas Schier               MAINTAINERS: Add linux-kbuild's patchwork
2022-11-12 09:07 +0100 Nicolas Schier               MAINTAINERS: Remove Michal Marek from Kbuild maintainers
2022-11-12 09:07 +0100 Nicolas Schier               MAINTAINERS: Add Nathan and Nicolas to Kbuild reviewers
2022-11-19 08:58 -0800 Linus Torvalds             Merge tag '6.1-rc5-smb3-fixes' of git://git.samba.org/sfrench/cifs-2.6
2022-11-16 17:10 +0300 Anastasia Belova             cifs: add check for returning value of SMB2_set_info_init
2022-11-15 18:39 +0800 Zhang Xiaoxu                 cifs: Fix wrong return value checking when GETFLAGS
```