

Git and GitHub



Gwyn Jones, Lead Front End Developer

Version control

What is it?

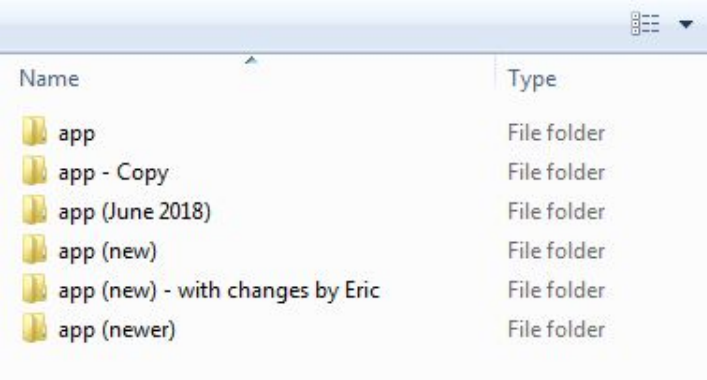
A *system* to **record changes** to files **over time** so that you can **recall specific versions later**.

Informal 'systems' people use for version control

It's not long after being a computer user that people start creating their own 'systems' for local version control. Common approaches include:

- **Ctrl+Z**
- Creating copies to act as a 'backup'

This are very common because they are so simple, but are **crude**, **limited** and **incredibly error prone**.



Name	Type
app	File folder
app - Copy	File folder
app (June 2018)	File folder
app (new)	File folder
app (new) - with changes by Eric	File folder
app (newer)	File folder

**Before looking at
formal version control,
let's introduce the
'commit' (aka 'patch
set', 'change set' etc.)**

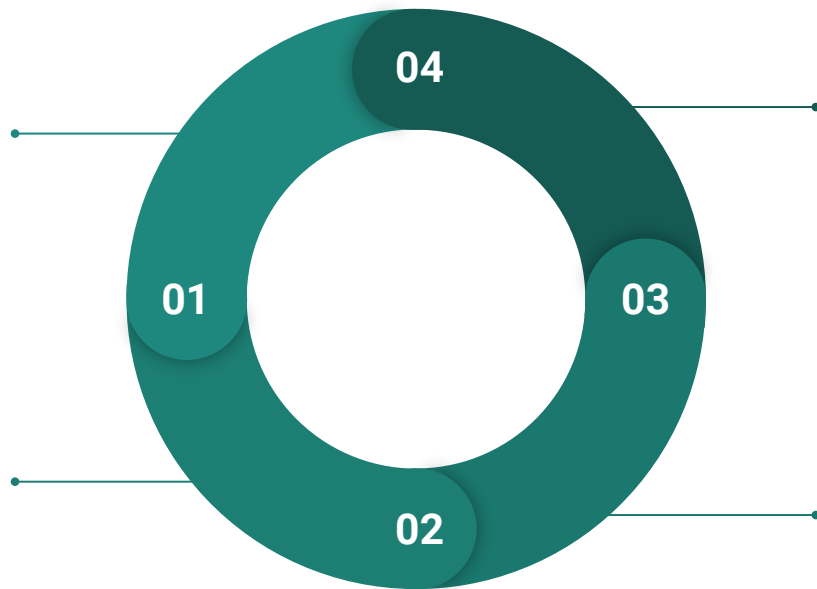
A representation of the commit cycle

Create or change code files to fulfil a specific task

In order to fulfil a task, the developer creates or amends files

Select files to be included in the commit

This will often be all the files that have changed, but it may be that the developer feels there would be benefit if splitting the commit over two or more commits



Commit the changes

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

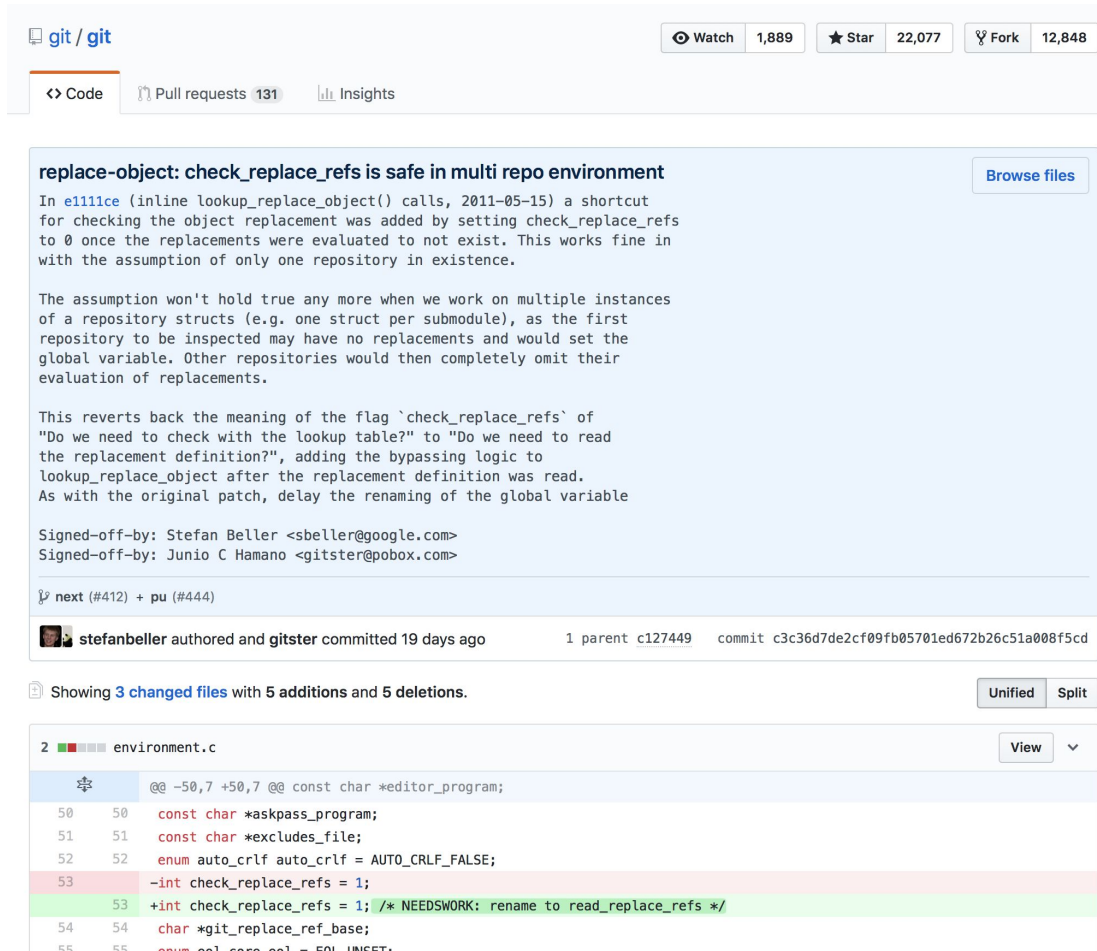
Describe the changes

Provide a message that communicates the **context** of a change to future developers

Commit messages communicate the context of a change

*“The contributors to these repositories know that a well-crafted Git commit message is the best way to communicate context about a change to fellow developers (and indeed to their future selves). A diff will tell you **what** changed, but only the commit message can properly tell you why.”*

Chris Beams. “How to Write a Git Commit Message”



git / git

Watch 1,889 Star 22,077 Fork 12,848

<> Code Pull requests 131 Insights

replace-object: check_replace_refs is safe in multi repo environment

[Browse files](#)


In [e1111ce](#) (inline `lookup_replace_object()` calls, 2011-05-15) a shortcut for checking the object replacement was added by setting `check_replace_refs` to 0 once the replacements were evaluated to not exist. This works fine in with the assumption of only one repository in existence.

The assumption won't hold true any more when we work on multiple instances of a repository structs (e.g. one struct per submodule), as the first repository to be inspected may have no replacements and would set the global variable. Other repositories would then completely omit their evaluation of replacements.



This reverts back the meaning of the flag `'check_replace_refs'` of "Do we need to check with the lookup table?" to "Do we need to read the replacement definition?", adding the bypassing logic to `lookup_replace_object` after the replacement definition was read. As with the original patch, delay the renaming of the global variable

Signed-off-by: Stefan Beller <sbeller@google.com>
Signed-off-by: Junio C Hamano <gitster@pobox.com>

🔗 next (#412) + 📁 pu (#444)

 stefanbeller authored and gitster committed 19 days ago 1 parent [c127449](#) commit [c3c36d7de2cf09fb05701ed672b26c51a008f5cd](#)

Showing 3 changed files with 5 additions and 5 deletions. Unified Split

```
2   environment.c View
```

```
@@ -50,7 +50,7 @@ const char *editor_program;

50 50  const char *askpass_program;
51 51  const char *excludes_file;
52 52  enum auto_crlf auto_crlf = AUTO_CRLF_FALSE;
53 53  -int check_replace_refs = 1;
53 53  +int check_replace_refs = 1; /* NEEDSWORK: rename to read_replace_refs */
54 54  char *git_replace_ref_base;
55 55  enum enl core_enl = ENL_UNSET;
```

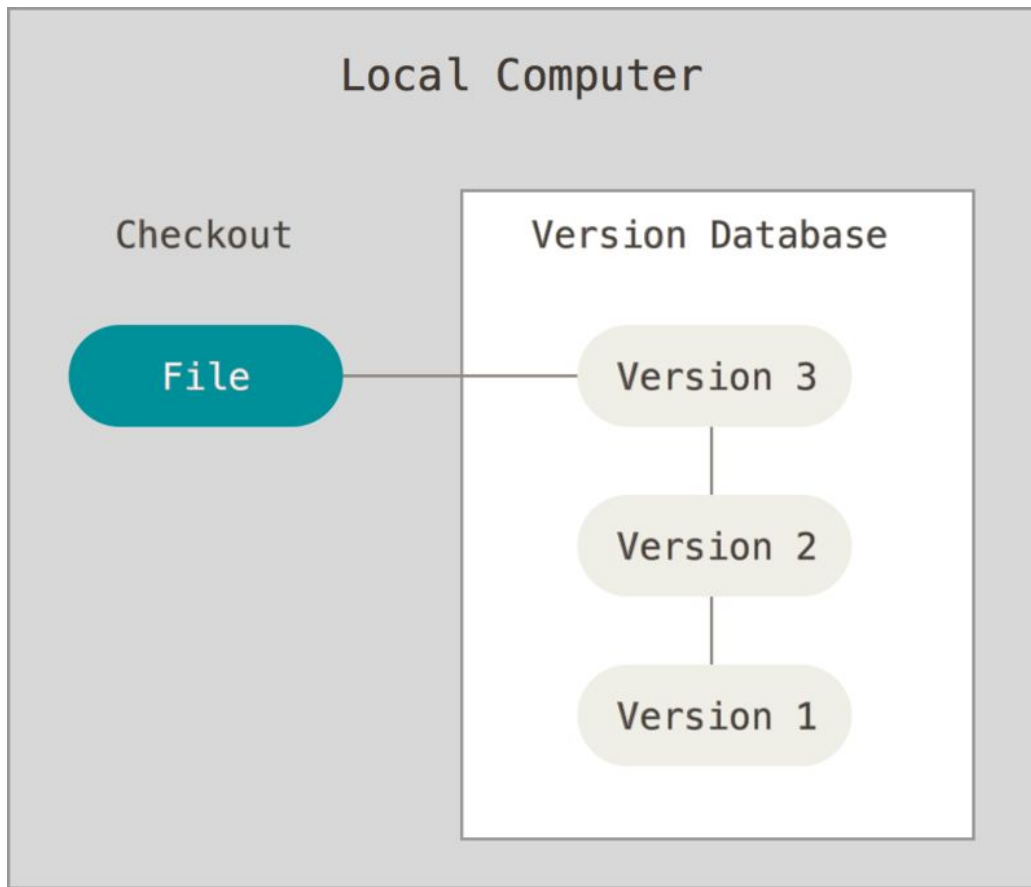
Evolution of formal Version Control Systems (VCSs)

Local Version Control

A long time ago programmers introduced local version control systems which stored 'patch sets' (i.e. the difference between files at different points in time).

Primary benefit:

- This allowed a programmer to recreate a specific state of a file at any given point by applying or removing specific patches

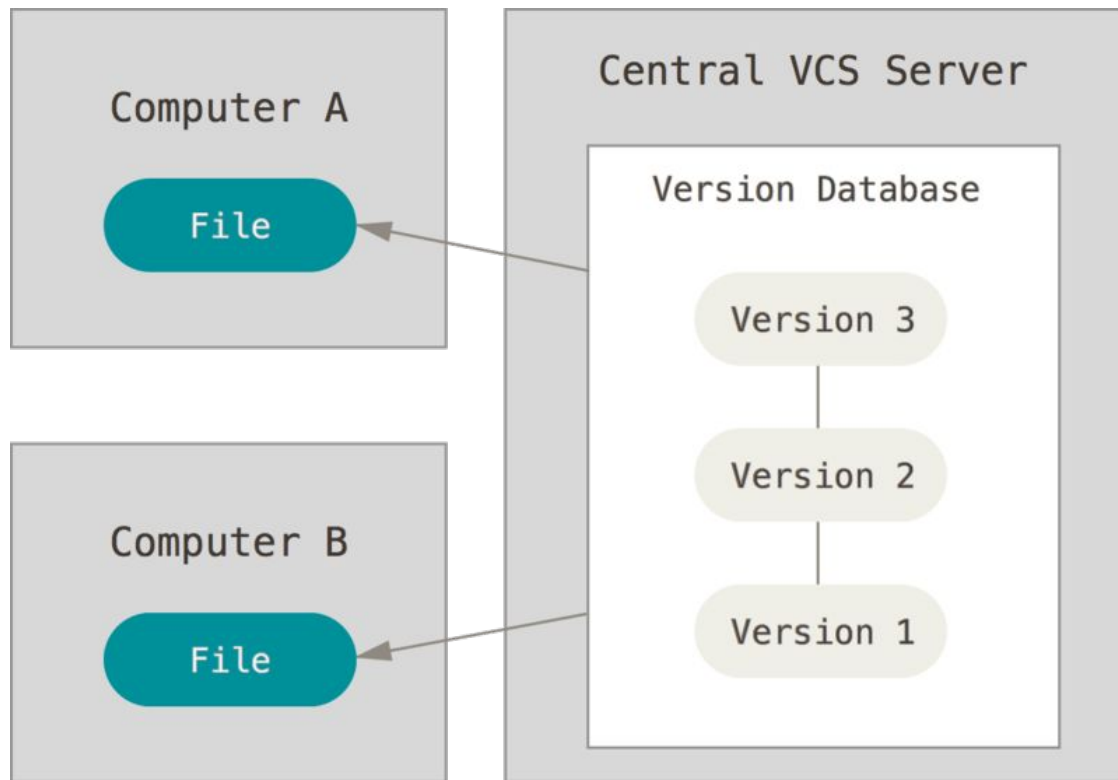


Centralized Version Control Systems (CVCSs)

A single server that contains all the versioned files and programmers 'check out' files from that single place

Primary benefit: Allowed programmers to collaborate with others

Drawbacks: huge SPOF

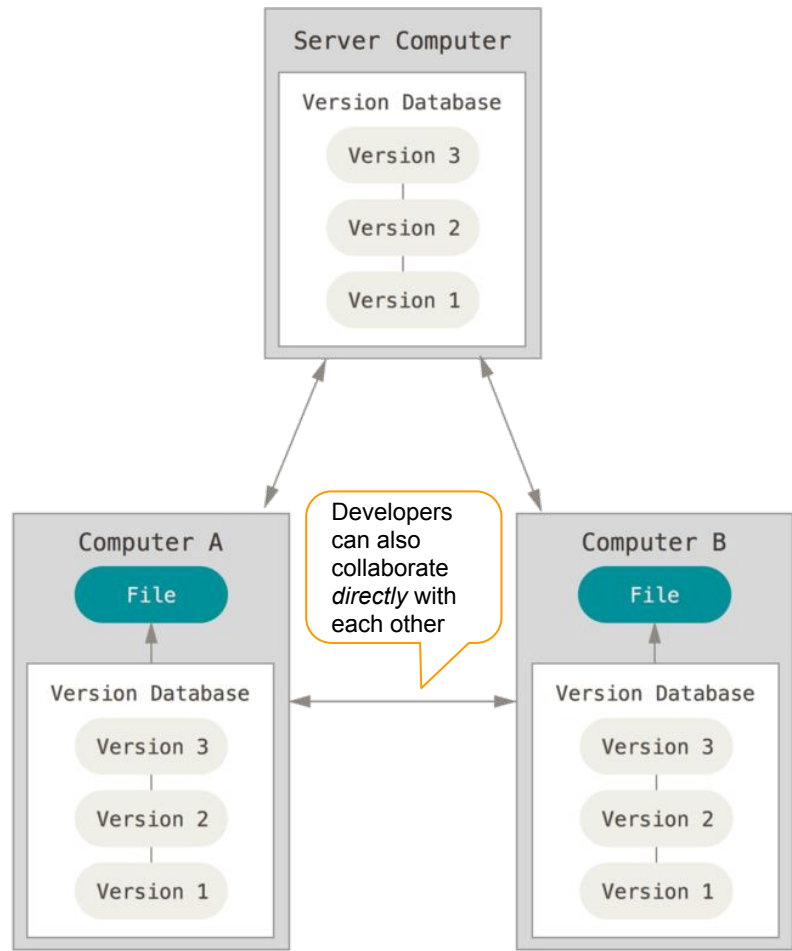


Distributed Version Control Systems (DVCSs)

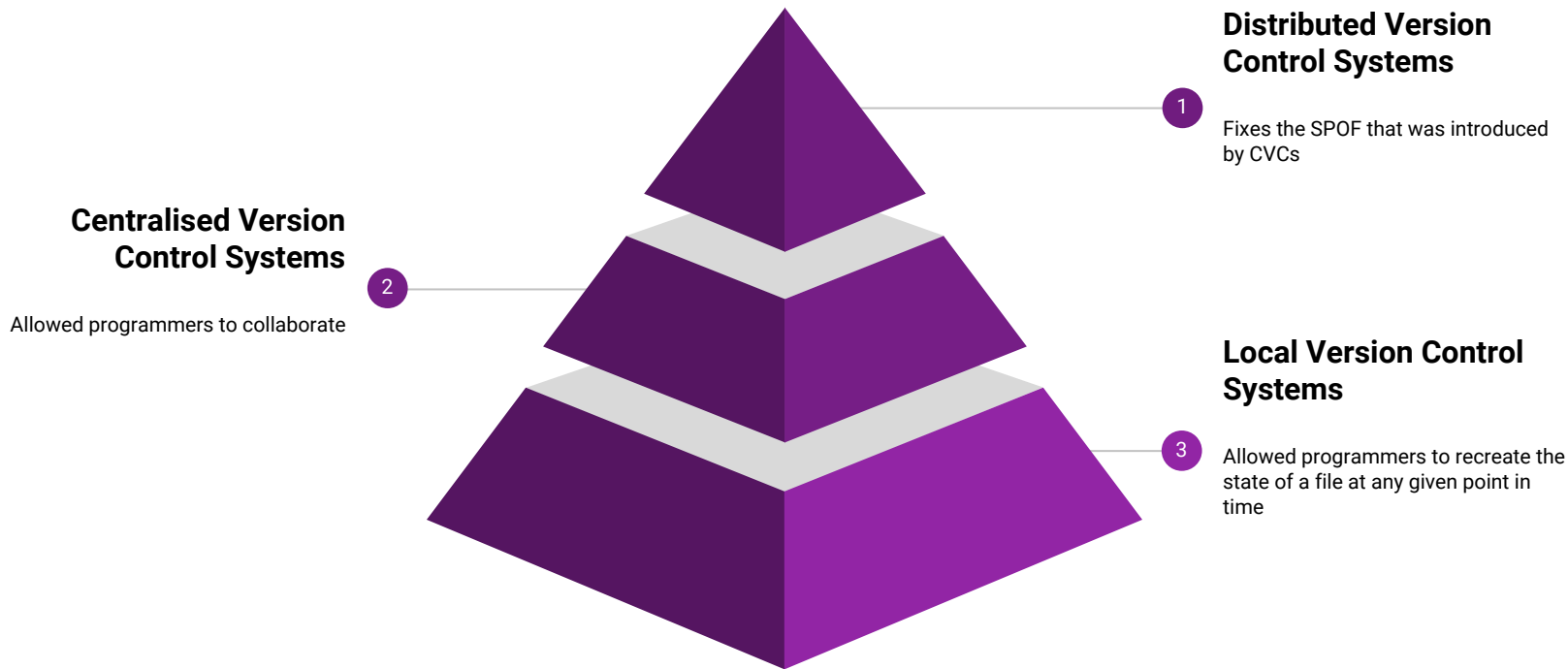
In a DVCS programmers don't just check out the latest snapshot of the files; they fully mirror the repository - including its history.

Primary benefit: every clone is a full backup of the code and its history.

Git is a DVCS, but it is not the only one



Each builds upon the capabilities of its predecessor

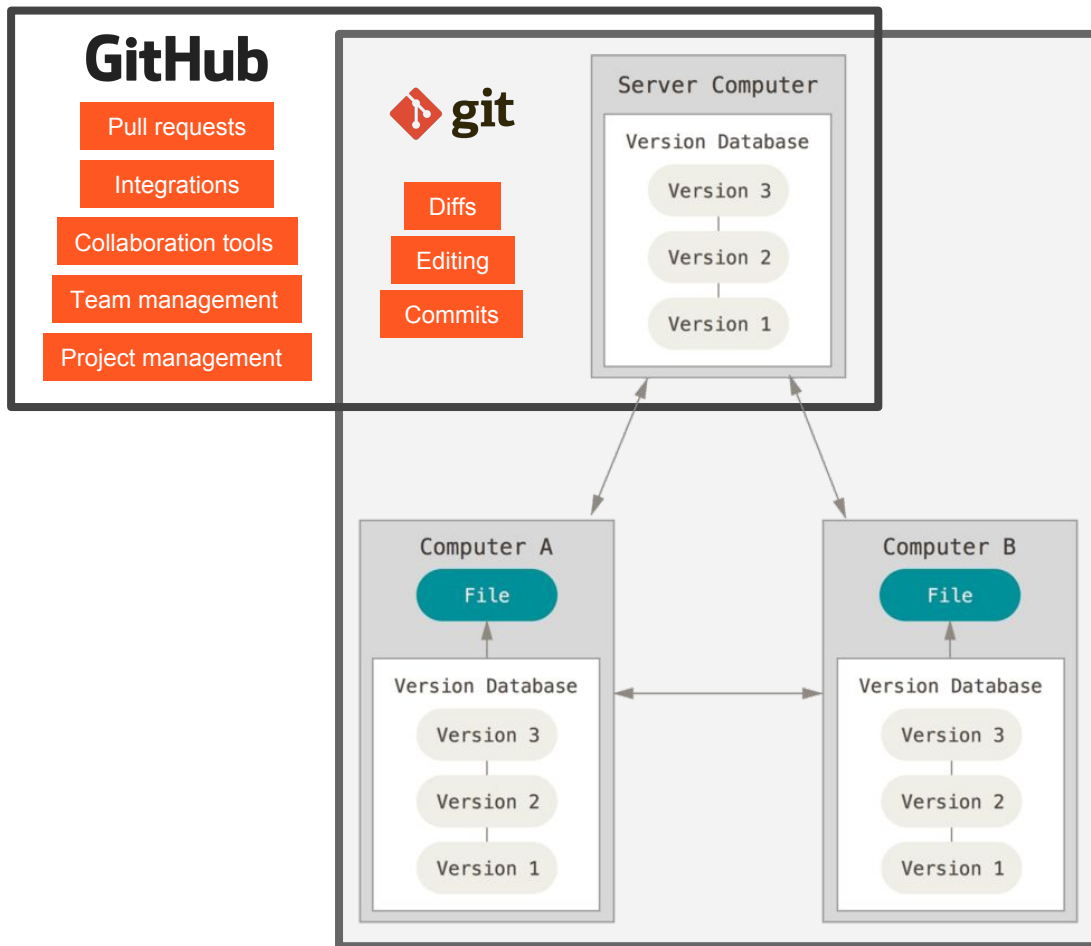


**Think of GitHub
as a place for
collaboration
with Git**

GitHub

A hosting service for version control using Git. In addition to being somewhere that developers can ‘pull’ and ‘push’ code, it provides several other features, including:

- Integrations
- Code review (Pull requests)
- Team management
- Project management



**So, what is a
Pull Request?**

Here's a good explanation



So let's look at a real Pull Request

The screenshot shows a GitHub Pull Request (PR) titled "Feature/banner click #7" in the repository "nationalarchives / tna-google-tag-manager-scripts". The PR is open, showing 5 commits and 9 files changed. The PR is from the "feature/bannerClick" branch to the "develop" branch. The PR is created by "punalsc" and is currently pending review. The PR description is "punalsc wants to merge 5 commits into develop from feature/bannerClick".

The PR details section shows the following information:

- Conversation: 4
- Commits: 5
- Checks: 0
- Files changed: 9
- Reviewers: gtvj (reviewed), punalsc (assigned)
- Assignees: punalsc
- Labels: None yet
- Projects: None yet
- Milestone: No milestone
- Notifications: You're receiving notifications because your review was requested.
- Participants: 2 participants (gtvj, punalsc)

The PR comment history shows a comment from "punalsc" 3 days ago:

Hi @gtvj,
Could you possibly have a look at this PR, when you can.

```
let output;  
if (Reviewer === 'Happy') {  
  output = 'Merge and delete feature branch';  
} else {  
  output = 'The assignee is happy to answer any questions';  
}
```

The PR commit history shows the following commits:

- added clickBanner GTM script
- more click banner tests
- test to check the value of the object id
- object type tests
- changed description of test

The PR was self-assigned by "punalsc" 3 days ago.