Git: Remotes - Patch v Pull

What is a remote?

- someone else's copy of a Git repo
- these let you collab with others
    - i.e. instead of making our own repository, we will clone someone else's and then made changes to that before sharing those changes back with them

This idea harkens back to the idea of a decentralised version control system

- With *centralised version control* (such as SVN or CVS) you would make a copy of the master source code, make a copy of it locally, make your changes and then send it back tot he cenrtralised version
- **Git** has no centralised copy instead EVERY single copy of a repo can act as the master copy of it
    - therefore you can integrate changes to it and send changes from it as everyone has a master copy

Example

Alice has made a Git repo for her group's coursework in:

```
~alice/coursework/
```

- Bob is in the same group and wants to collaborate with Alice, and wants his own copy

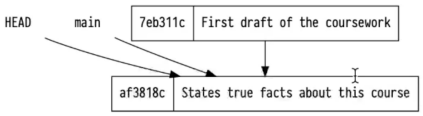
```
$ git clone ~alice/coursework/ ~bob/coursework/
```

- clone takes 2 arguments, the path to the folder you're cloning, then the destination

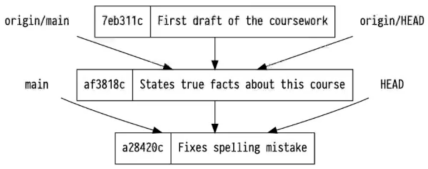
say now bob has made changes and is ready to commit?

this is the state of their repos:

On Alice's repo



On Bob's repo



- bobs repo is now ahead of Alice

How do you get Bob's changes back to Alice?

There are two ways to do this:

1. Patch Based approach
2. Pull based approach

Patch/email Approach

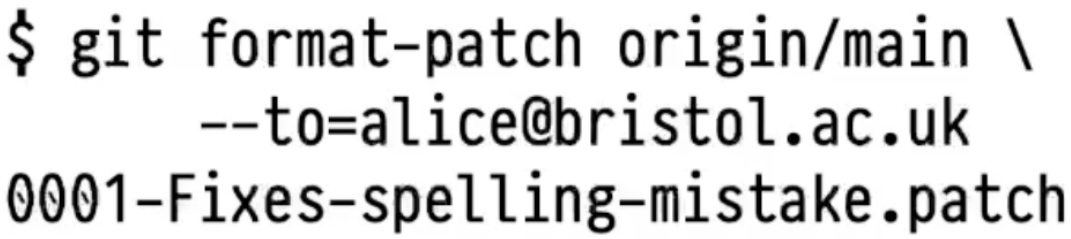
Patch Based Approach (Bob's side)

- this is how the *Linux Kernel* and many other open source projects manage commits

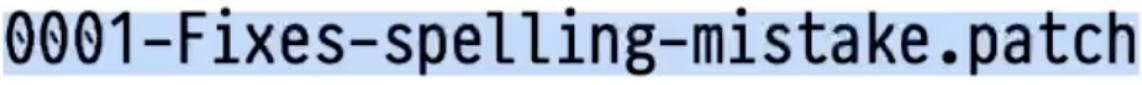
So Bob starts by preparing a patch using the format patch command:

```
git format-patch
```

- these being the changes he wants to commit:



- so the above code is saying: Bob wants to format a patch of origin/main to be sent to alice@bristol.ac.uk
- this will then give him a single patch file for each commit
  - each of these can then be sent to Alice using a **send mail client**
    - e.g. gmail, outlook etc
    - (check out: git sent-email)



```
From a28420cd5c45d06c9a51625d5a03c37bb77e2ca9 Mon Sep 17 00:0
From: bob <bob@bristol.ac.uk>
Date: Tue, 22 Nov 2022 11:53:04 +0000
Subject: [PATCH] Fixes spelling mistake
To: alice@bristol.ac.uk

---
 coursework.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

diff --git a/coursework.c b/coursework.c
index 2e191f8..8927b2f 100644
--- a/coursework.c
+++ b/coursework.c
@@ -1,7 +1,7 @@
 #include <stdio.h>

 int main(void) {
-    printf("Softwaer tools is cool!\n
+    printf("Software tools is cool!\n
     printf("Hello World!\n");
     return 0;
 }
--
2.38.1
```

Patch Based Approach (Alice's side)

- Alice will then review the patch bob sent her and if she likes it she can incorporate it into her tree using a git am command:

```
git am
```

```
$ git am ../bob/0001-Fixes-spelling-mistake.patch
Applying: Fixes spelling mistake
```

- then if Alice checks her log (git log) she can see the Bob's commit has been integrated:

```
$ git log --oneline
575dcde Fixes spelling mistake
af3818c States true facts about this course
7eb311c First draft of the coursework
```

- the reason the hash (575dcde) is different to Bobs is because it is a different commit that has been make to Alice's tree - Alice is the commit author now instead of bob

Git am is a high level git command known as a Git Porcelain Command

- these use a a few low level commands to create a useful bit of functionality
    - e.g. git am takes the commands *git apply* and *git commit* and runs them in a sequence to produce the commit message automatically for you
- similar high-level git commands are commonplace

Benefits of patch approach

- great when youre working on **open-source** or when you want to tightly control how the integration of other people's work goes

The alternative to the patch-based system is the pull-based system...

Pull Based Approach

- this is where you trust the other person you are workling with
- are you trust git

In this approach, Alice instead of getting emailed the patch, will fetch it from Bob's repo...

- firstly, Alice's repo doesnt know anything about Bob's repo
  - she needs to pair Bobs repo as a remote:

```
git remote add bob ~bob/coursework
```

- then running the git fetch command will get the changes that are in bobs verison of the repo:

```
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 255 bytes | 127.00 KiB/s, done.
From ~bob/coursework
 * [new branch]      main       -> bob/main
```

- if Alice is happy, she will want to have Bob's changes saved in her repo

for this she can use the pull command:

(git pull is basically a fetch and a merge rolled into one)

```
git pull
```

```
$ git pull bob main
From ~bob/coursework
 * branch            main       -> FETCH_HEAD
Updating af3818c..a28420c
Fast-forward
 coursework.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git log | cat
commit a28420cd5c45d06c9a51625d5a03c37bb77e2ca9
Author: bob <bob@bristol.ac.uk>
Date:   Tue Nov 22 11:53:04 2022 +0000

    Fixes spelling mistake

commit af3818ce392c983a2d5523ef7b43f5e294bd674e
Author: alice <alice@bristol.ac.uk>
Date:   Tue Nov 22 11:15:29 2022 +0000
```