# Git internals

April 27, 2011

## Part I
## What's inside

Git repository consists of the following objects:

- Object database
  - Blobs
  - Trees
  - Commits
  - Annotated tags
- Refs
  - Your branches and tags
  - Remote branches
  - Special (HEAD etc.)
- Config/misc
  - .git/config
  - hooks
  - reflogs

.

Blob is a piece of content without metadata

Tree is a list of file/directory names pointing at blobs or other trees /* not considering subprojects */.

Commit is an object that 1. has pointer to other commit[s], 2. has pointer to tree (project root), 3. contains metadata (commit message, dates, authors, etc.)

Each ref points to (usually last) commit in some branch (either your or remote)

## Part II

# A short story

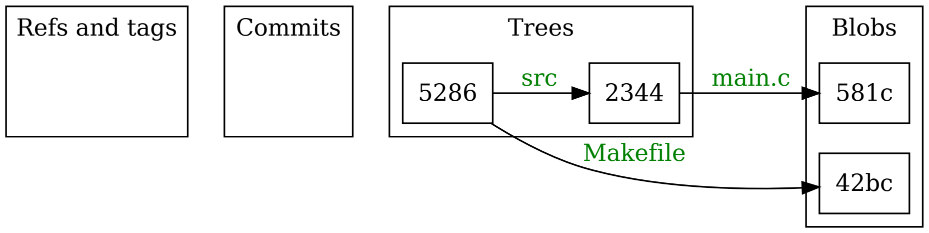
## 1 Adding files

Let's create a simple project:

```
mkdir qqq && cd qqq
mkdir src
echo 'int main(){printf("Helo wrold\r");}' > src/main.c
cat > Makefile << \EOF
all:
        gcc src/*.c -o hello
EOF

git init
git add .
```

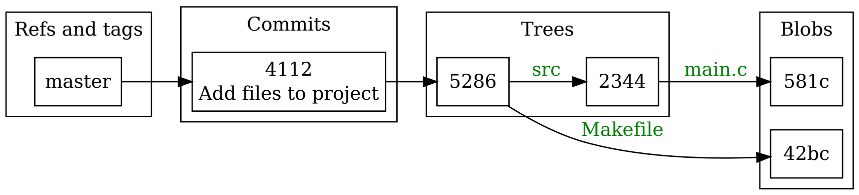
"git add ." addded src/main.c and Makefile. It created two blobs (main.c and Makefile) and two trees (root tree and "src" tree).



## 2 Committing

```
git commit -m "Add files to project"
```

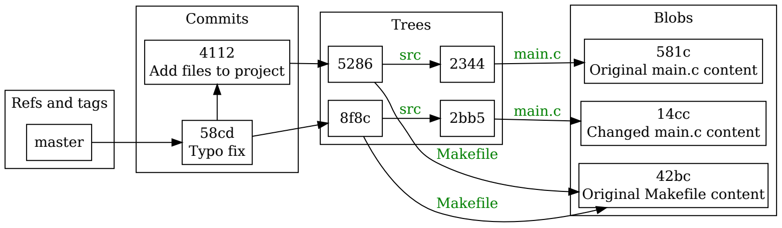
Now we have just created a commit. Commit is pointed by ref (name of branch), tree is pointed by commit, blob is pointed by tree.



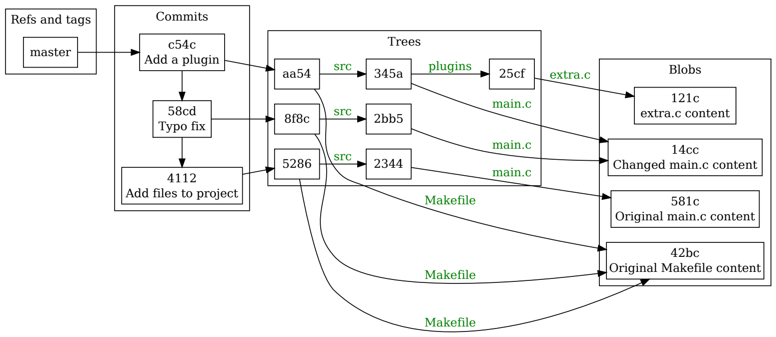
## 3 More commits

```
echo 'int main(){ printf("%s", "Hello world\n"); return 0;}' > src/main.c
git commit -am "Typo fix"
```

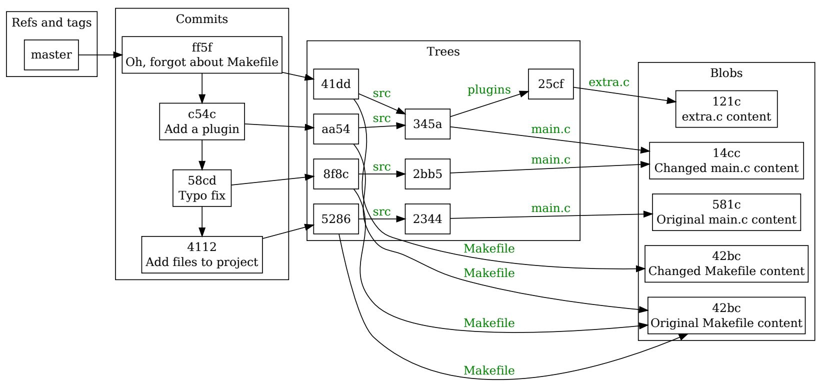
We changed main.c. It gets new blob. So we need to have new "src" tree (with updated blob pointer). So we need to have the new root tree (with updated "src" tree pointer). New commit points to the new tree.



```
mkdir src/plugins
echo 'void func(){}' > src/plugins/extra.c
git add src/plugins
git commit -m "Add a plugin"
```



```
cat > Makefile << \EOF
all:
          gcc src/*.c src/plugins/*.c -o hello
EOF
git add Makefile
git commit -m "Oh, forgot about Makefile"
```



Note that now "src" tree is reused from previous commit (as nothing have changed there)

## Part III

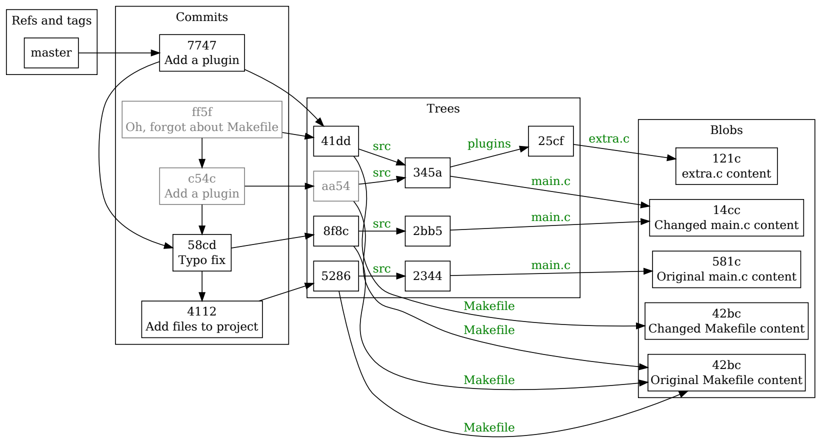
# Rewriting history

Commit "c54c" is not useful: the very next commit fixes obvious thing. We want to merge that two commits into one, eliminating the "wrong" intermediate c54c.

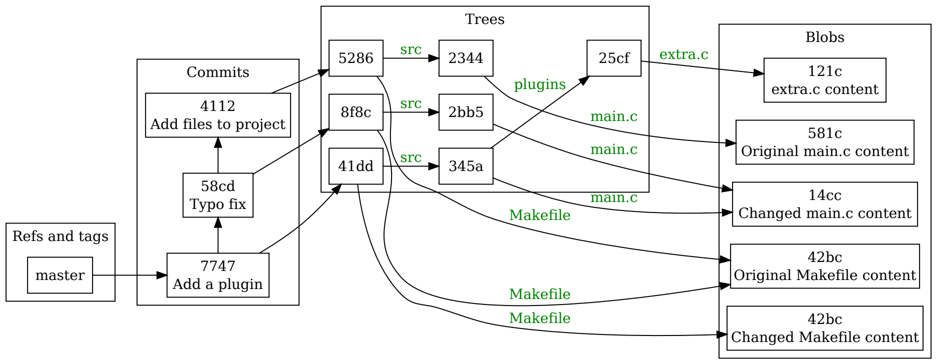
```
git rebase -i HEAD~2  # ~2 means we need to process 2 last commits
# in editor:
pick c54c
squash ff5f
```

This is convert commits c54c and ff5f to patches, rewing to commit 58cd then apply that two patches, but producing only one commit (you can also split and edit them). /* Note: this case can be done simpler: `git reset --soft HEAD~2 && git commit -m "Add a plugin"`*/

Stale commits "ff5f" and "c54c" still remain in repository and can be used for recovery (unless finally cleaned by "git gc")


Refs and tags
master
Commits
7747 Add a plugin
ff5f Oh, forgot about Makefile
c54c Add a plugin
58cd Typo fix
4112 Add files to project
Trees
41dd
aa54
8f8c
5286
345a
2bb5
2344
25cf
src
plugins
main.c
extra.c
Makefile
Blobs
121c extra.c content
14cc Changed main.c content
581c Original main.c content
42bc Changed Makefile content
42bc Original Makefile content

If/when stale commits get cleaned (they will be there for a certain time), it will be like this:



“ff5f” and “c54c” are gone, tree “aa54” is gone too (as it is not referenced by anything anymore).

## Part IV
# Interaction with other repositories

We can fetch or push commits to other repositores. Obtaining (or publishing) a commit implies transferring it’s trees and blobs as well. For example,

```
git remote add origin git://github.com/hello/exaple.git
git fetch origin +refs/heads/master:refs/remotes/origin/master
```

Registers URL of remote repository under the name “origin” and then fetches origin’s ref named “refs/heads/master” (or just “master”) to our local ref “refs/remotes/origin/master”, replacing old “refs/remotes/origin/master” even if it is not direct child (this will occur if, for example, somebody have fetched our commit ff5f or c54c and now we rewritten it to new one 7747).

Note: This was a full form of fetch command, usually “git fetch origin” or “git fetch” does the required thing.

Pushing command example is “git push origin 4112:refs/heads/master”. It also can have forcing “+”, source ref can be “master” or “HEAD” (or even nothing - means “delete it”) instead of direct commit number “4112” or be configured for just “git push” to do what you expect.