Pro Git(https://git-scm.com/book/ko/v2/)를 바탕으로 작성하였습니다.

YoonJoon Lee
SoC KAIST

Fall 2018

What we will take a look in this series

1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

What we will take a look today

1.Getting Started

2. Git Basics

3. Git Branch

4. Git Server - GitLab

What I will talk about in this section

1. Branches in a Nutshell
2. Basic Branching and Merging
3. Branch Management
4. Branching Workflows
5. Remote Branches
6. Rebasing

Branching means we diverge from the main line of development and continue to do work without messing with that main line.

Some people refer to Git's branching model as its "killer feature," and it certainly sets Git apart in the VCS community.

Why is it so special? The way Git branches is incredibly lightweight, making branching operations nearly instantaneous, and switching back and forth between branches generally just as fast.

Branches in a Nutshell

Git doesn't store data as a series of changesets or differences, but instead as a series of *snapshots*.

When we make a commit, Git stores a commit object that contains a pointer to the snapshot of the content we staged.

This object also contains the author's name and email address, the message that we typed, and pointers to the commit or commits that directly came before this commit (its parent or parents):

MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ cat README
This is a demo for Branch


YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ cat test.py
# This is a python library for Branch Demo
import sys


YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ cat LICENSE
The Apache License

Copyright (c) KAIST
```

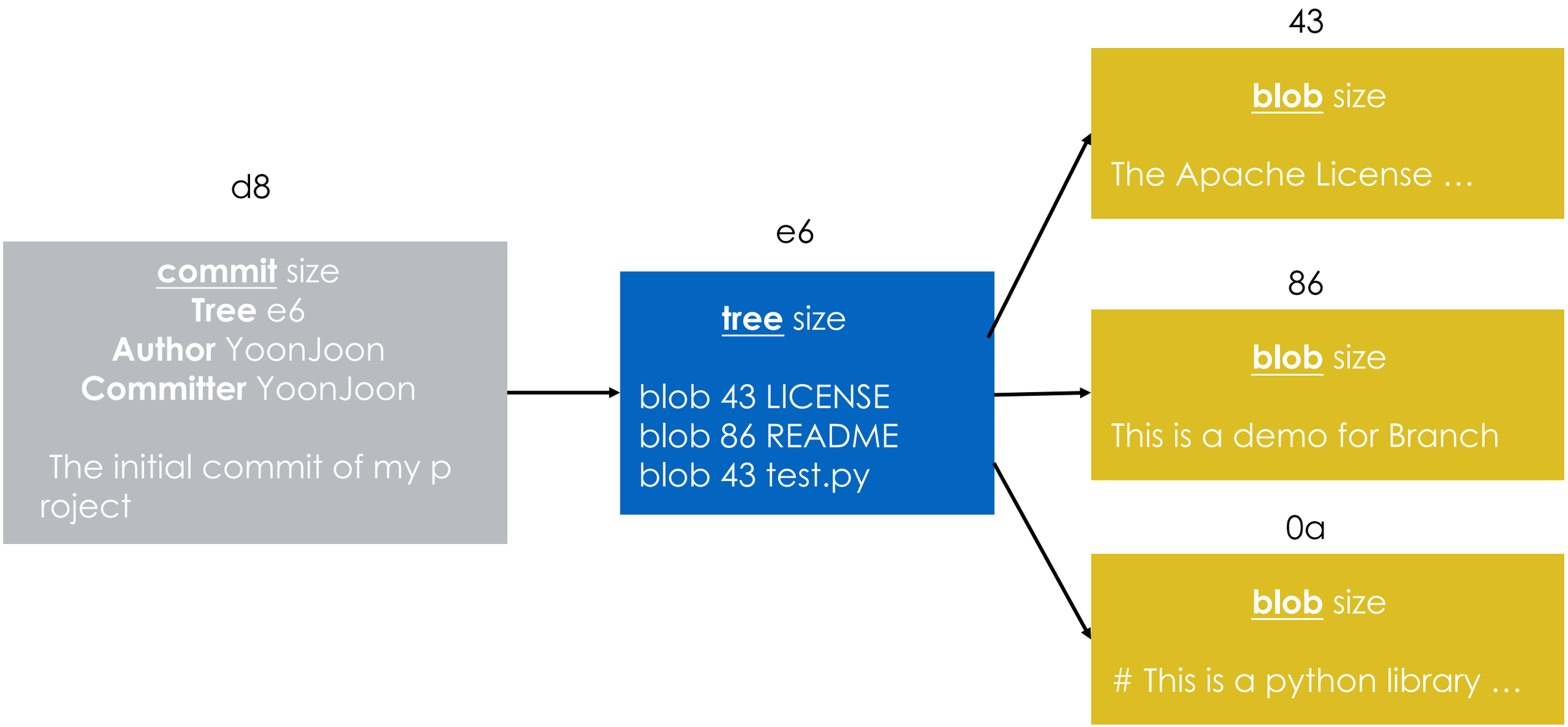
MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

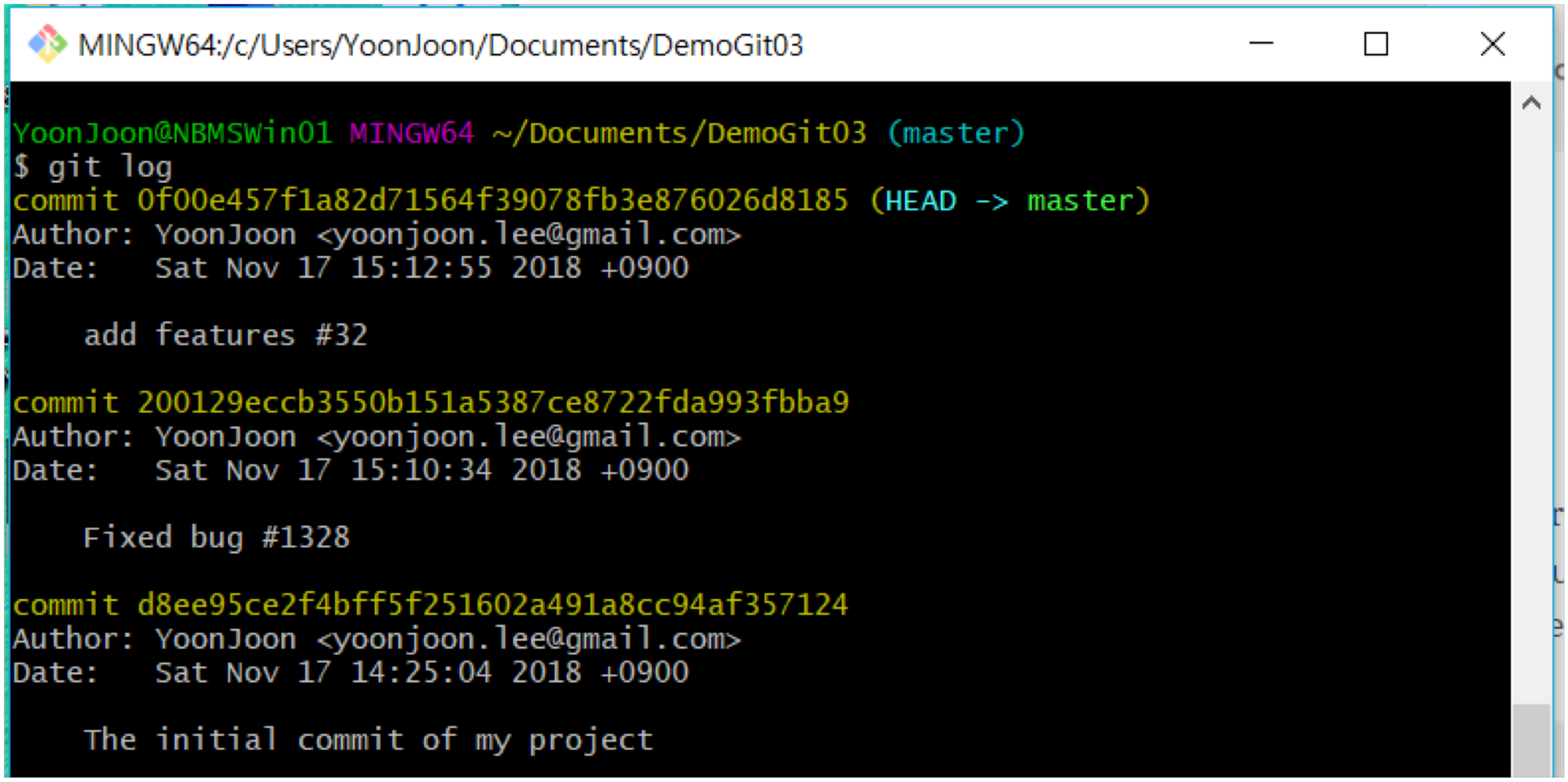
```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ git add README test.py LICENSE
warning: LF will be replaced by CRLF in LICENSE.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in README.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in test.py.
The file will have its original line endings in your working directory

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ git commit -m 'The initial commit of my project'
[master (root-commit) d8ee95c] The initial commit of my project
 3 files changed, 9 insertions(+)
 create mode 100644 LICENSE
 create mode 100644 README
 create mode 100644 test.py
```

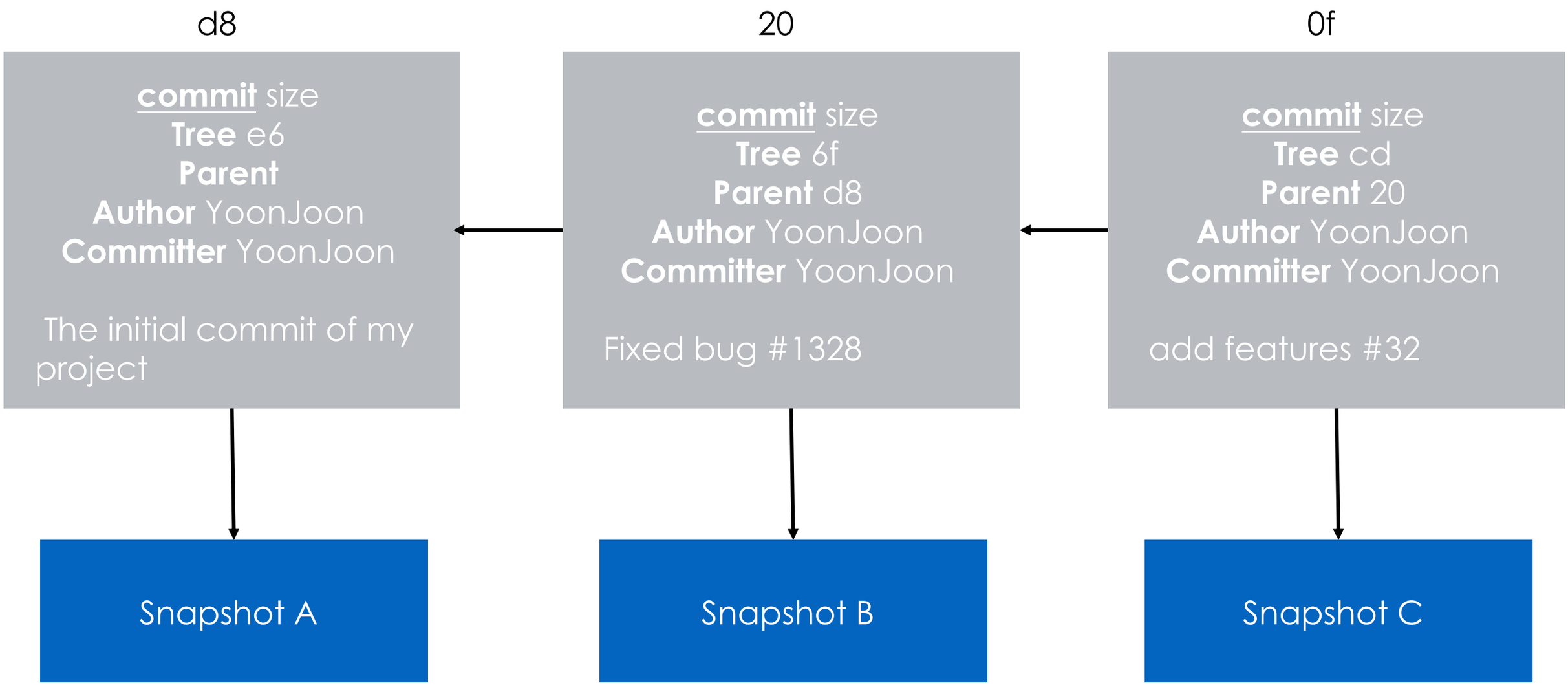
MINGW64:/c/Users/YoonJoon/Documents/DemoGit03/.git/objects

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03/.git/objects (GIT_DIR!)
$ ls
0a/  43/  86/  d8/  e6/  info/  pack/

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03/.git/objects (GIT_DIR!)
$ git show
commit d8ee95ce2f4bff5f251602a491a8cc94af357124 (HEAD -> master)
Author: YoonJoon <yoonjoon.lee@gmail.com>
Date:   Sat Nov 17 14:25:04 2018 +0900

    The initial commit of my project

diff --git a/LICENSE b/LICENSE
new file mode 100644
index 0000000..43981ef
--- /dev/null
+++ b/LICENSE
@@ -0,0 +1,4 @@
+The Apache License
+
+Copyright (c) KAIST
+
diff --git a/README b/README
new file mode 100644
index 0000000..86582cb
--- /dev/null
+++ b/README
@@ -0,0 +1,2 @@
+This is a demo for Branch
+
diff --git a/test.py b/test.py
new file mode 100644
index 0000000..0af139b
--- /dev/null
+++ b/test.py
@@ -0,0 +1,3 @@
+# This is a python library for Branch Demo
+import sys
+
```

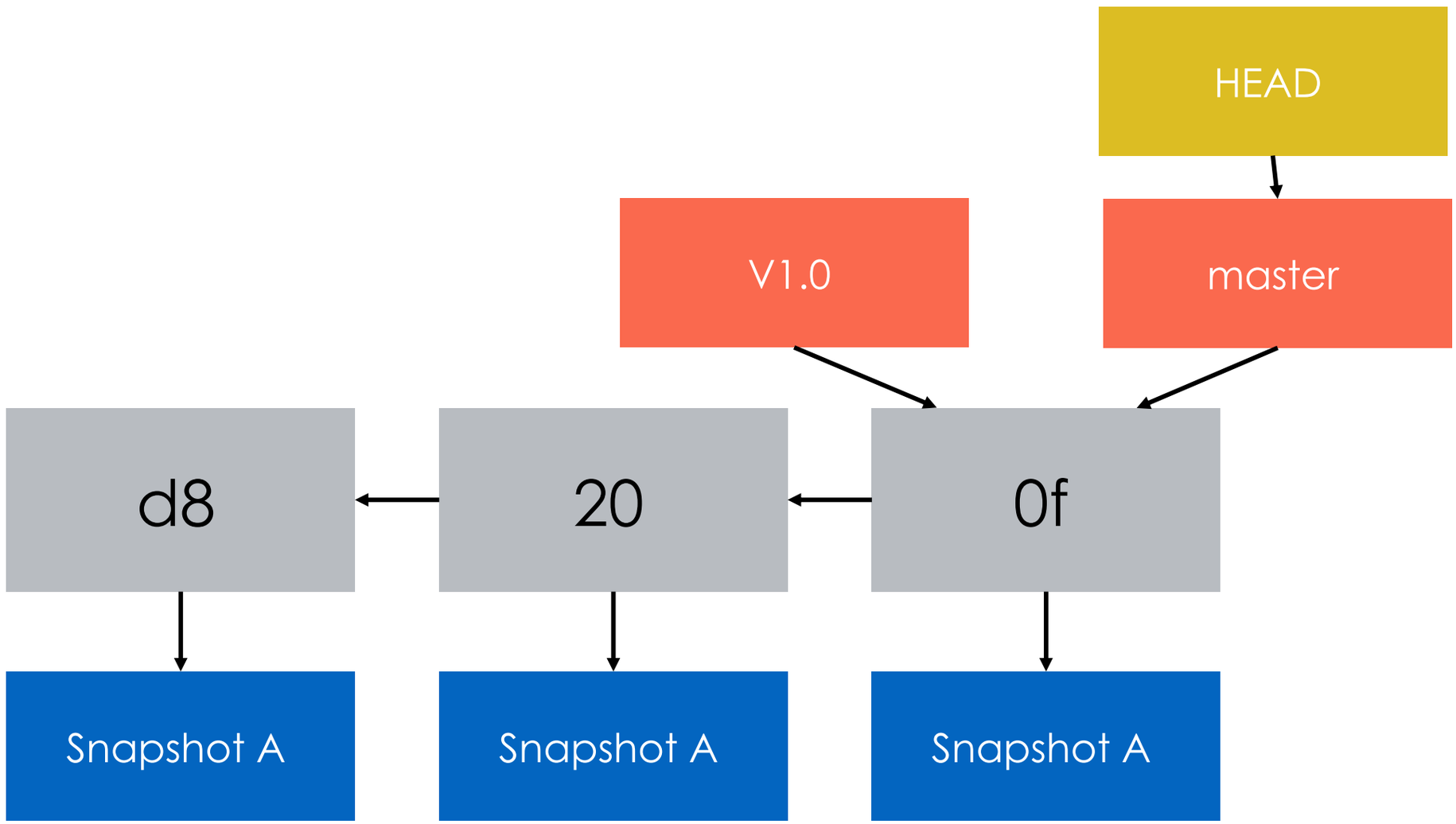
Git repository now contains five objects: three blobs (each representing the contents of one of the three files), one tree that lists the contents of the directory and specifies which file names are stored as which blobs, and one commit with the pointer to that root tree and all the commit metadata.



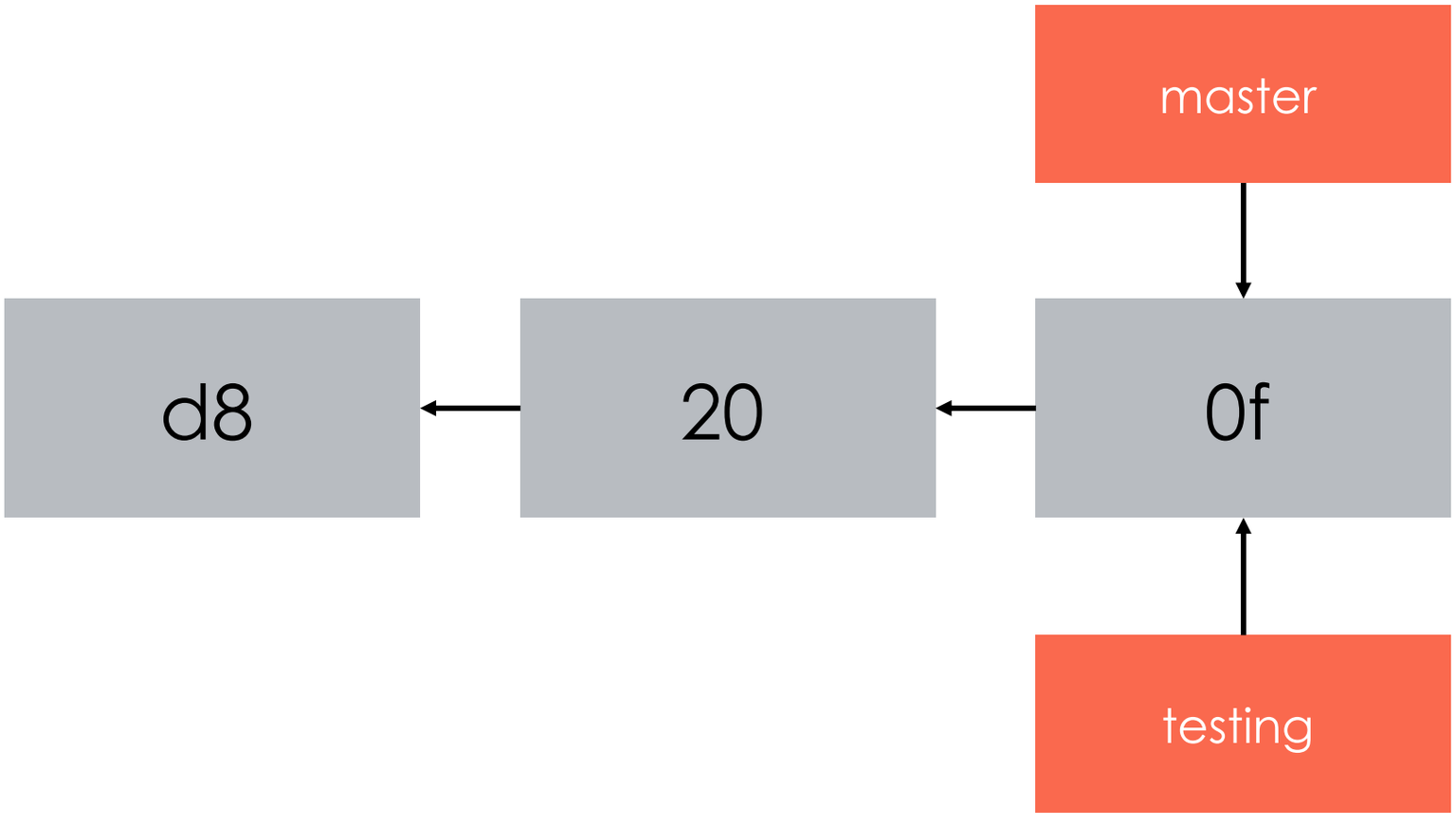

MINGW64:/c/Users/YoonJoon/Documents/DemoGit03
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ git log
commit 0f00e457f1a82d71564f39078fb3e876026d8185 (HEAD -> master)
Author: YoonJoon <yoonjoon.lee@gmail.com>
Date: Sat Nov 17 15:12:55 2018 +0900
add features #32
commit 200129eccb3550b151a5387ce8722fda993fbba9
Author: YoonJoon <yoonjoon.lee@gmail.com>
Date: Sat Nov 17 15:10:34 2018 +0900
Fixed bug #1328
commit d8ee95ce2f4bff5f251602a491a8cc94af357124
Author: YoonJoon <yoonjoon.lee@gmail.com>
Date: Sat Nov 17 14:25:04 2018 +0900
The initial commit of my project


d8
commit size
Tree e6
Parent
Author YoonJoon
Committer YoonJoon
The initial commit of my project
20
commit size
Tree 6f
Parent d8
Author YoonJoon
Committer YoonJoon
Fixed bug #1328
0f
commit size
Tree cd
Parent 20
Author YoonJoon
Committer YoonJoon
add features #32
Snapshot A
Snapshot B
Snapshot C

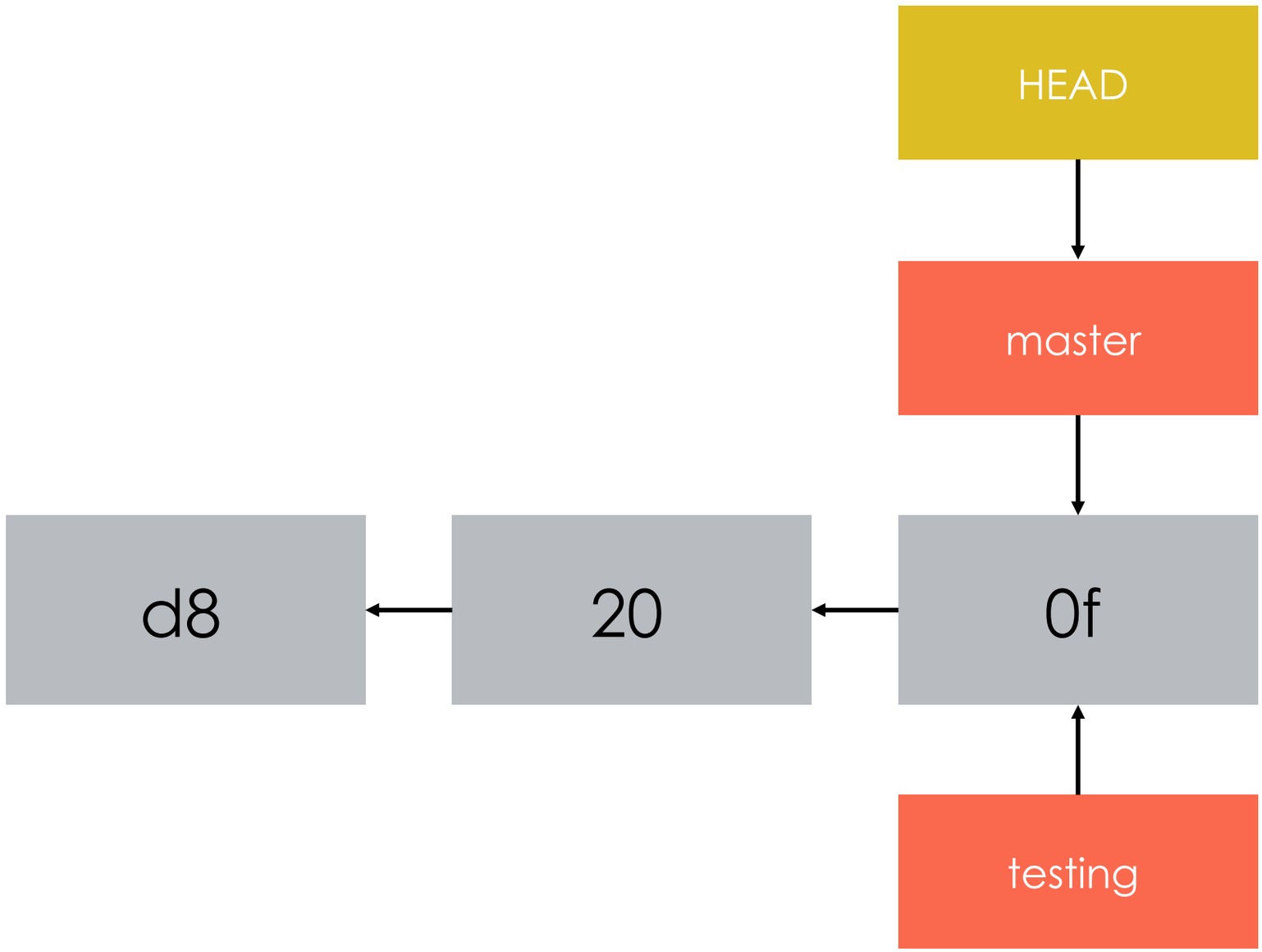
A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is `master`. As we start making commits, we're given a `master` branch that points to the last commit we made. Every time we commit, the `master` branch pointer moves forward automatically.



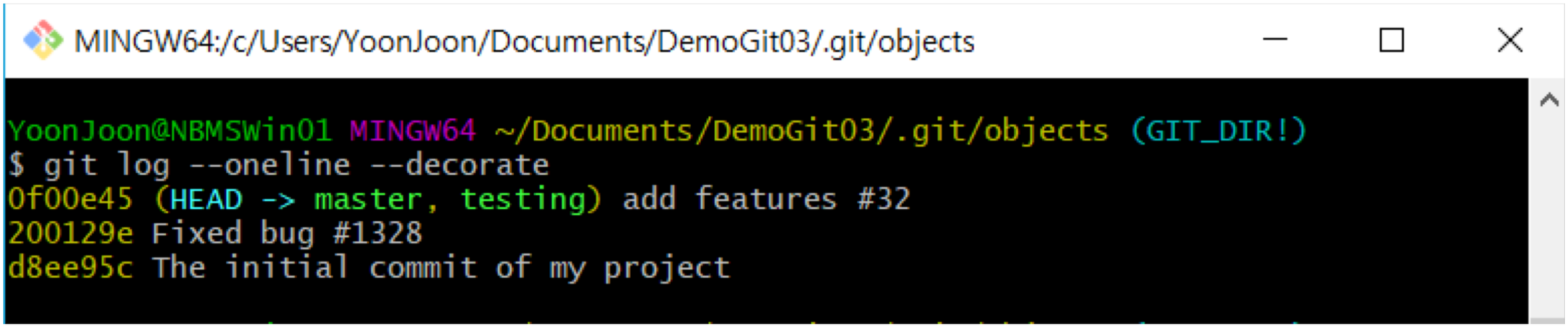
Creating a New Branch



How does Git know what branch we're currently on? It keeps a special pointer called `HEAD`. In Git, this is a pointer to the local branch we're currently on. In this case, we're still on master. The `git branch` command only *created* a new branch — it didn't switch to that branch.

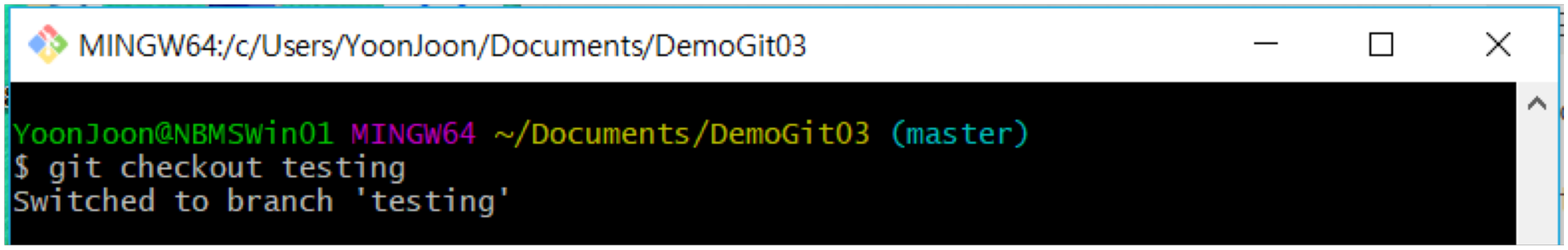


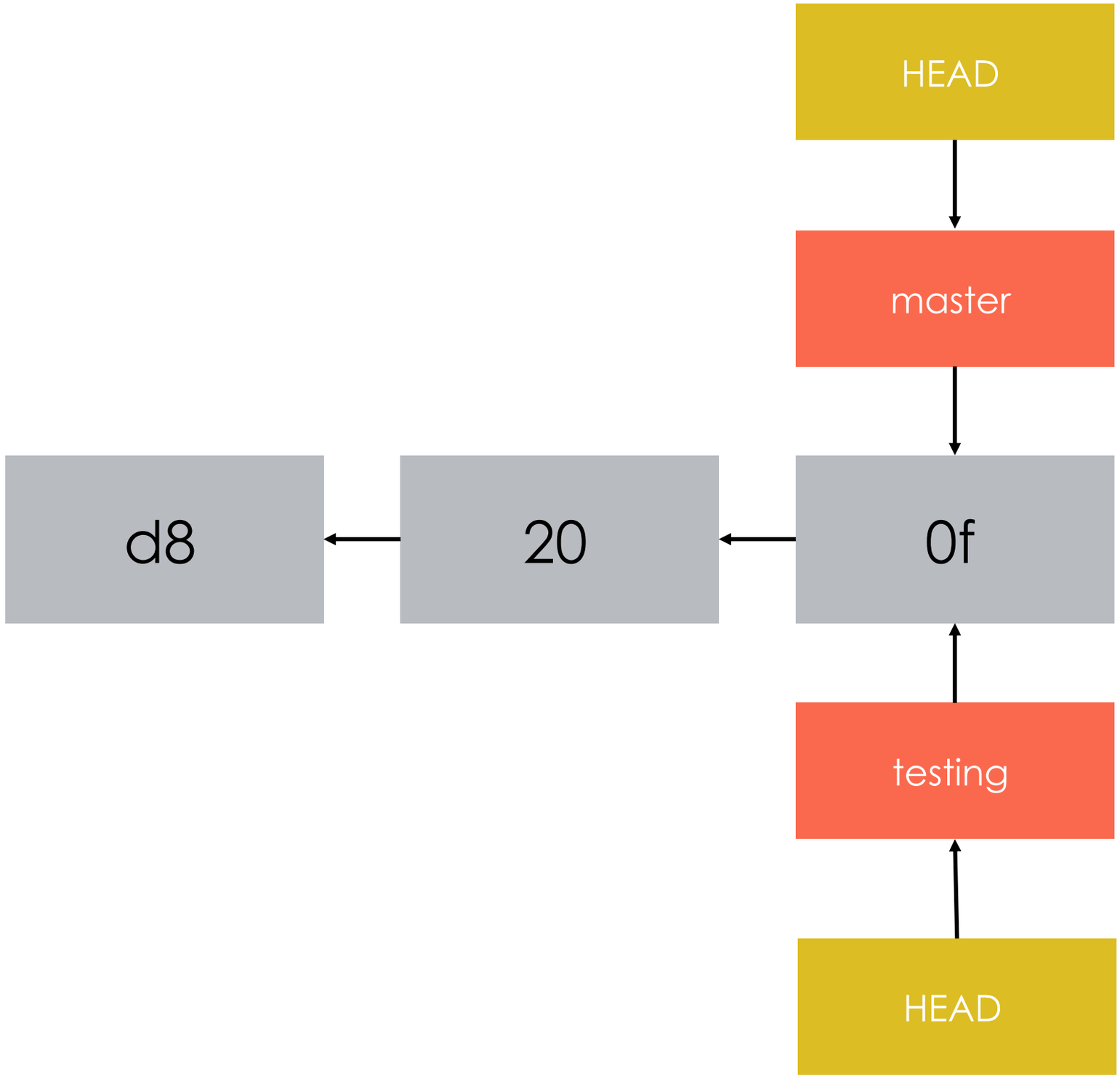
We can easily see this by running a simple `git log` command that shows us where the branch pointers are pointing. This option is called `--decorate`.



Switching Branches

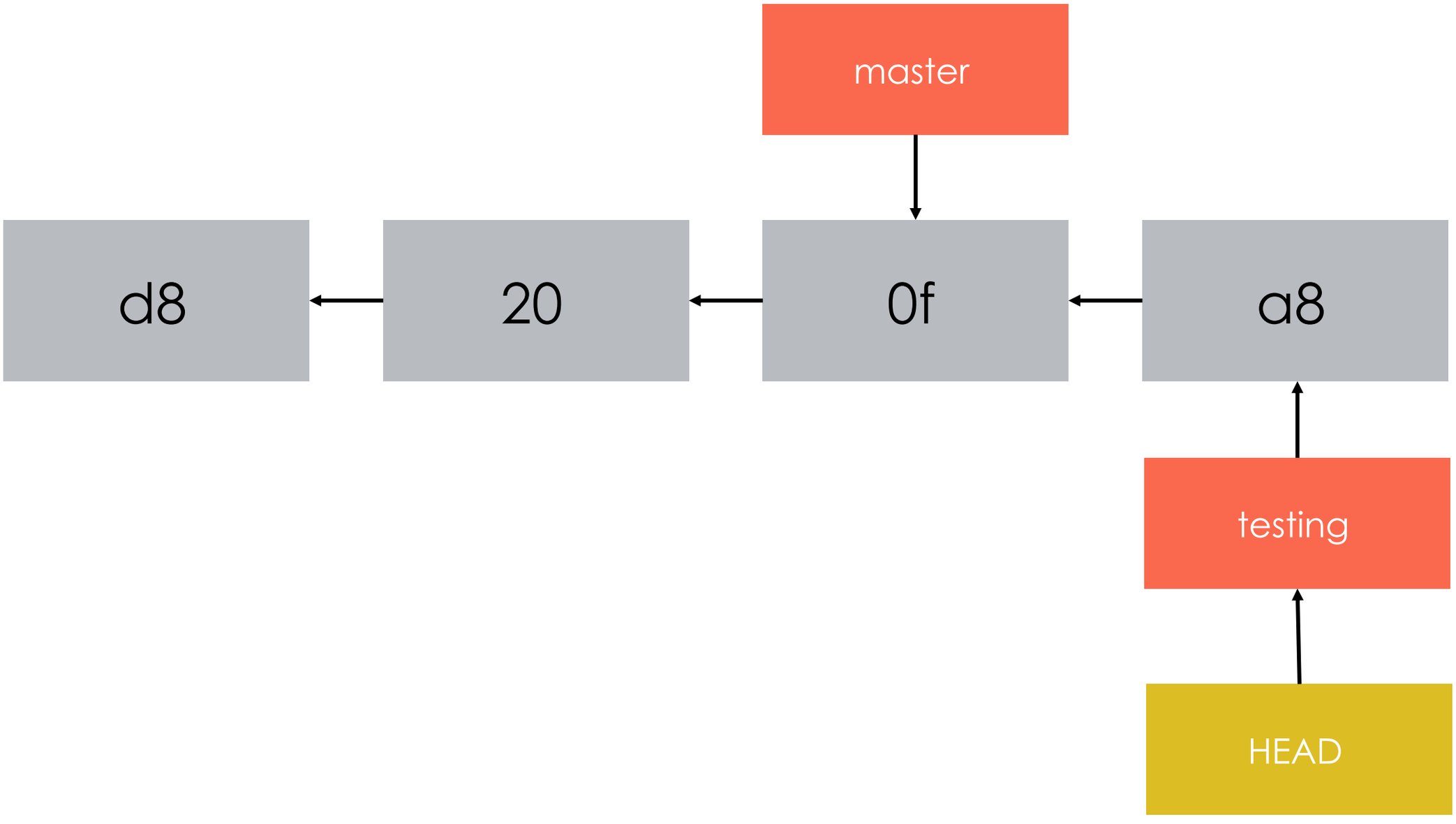
To switch to an existing branch, we run the `git checkout` command. Let's switch to the new testing branch:



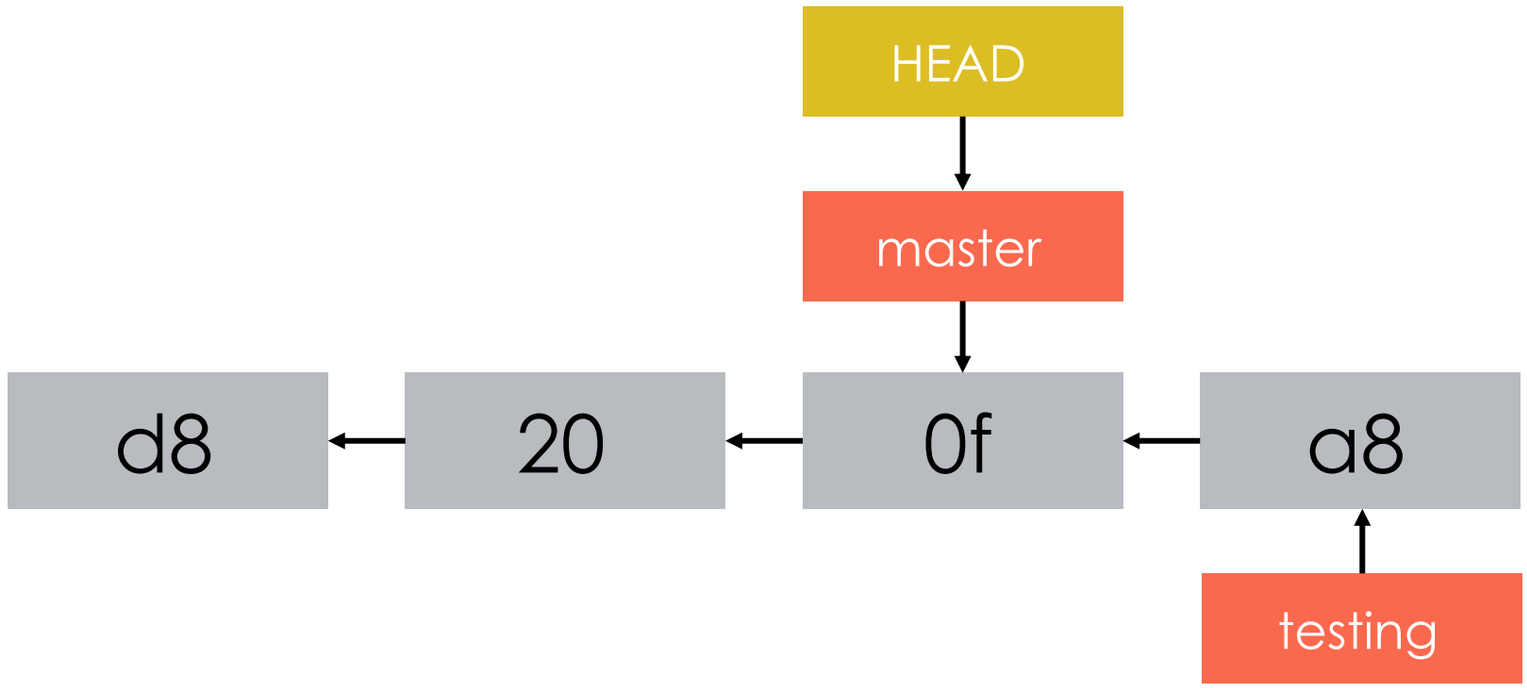

HEAD
master
d8
20
0f
testing
HEAD

MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (testing)
$ vi test.py

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (testing)
$ git commit -a -m 'made a change'
warning: LF will be replaced by CRLF in test.py.
The file will have its original line endings in your working directory
[testing a863b58] made a change
 1 file changed, 2 insertions(+)
```



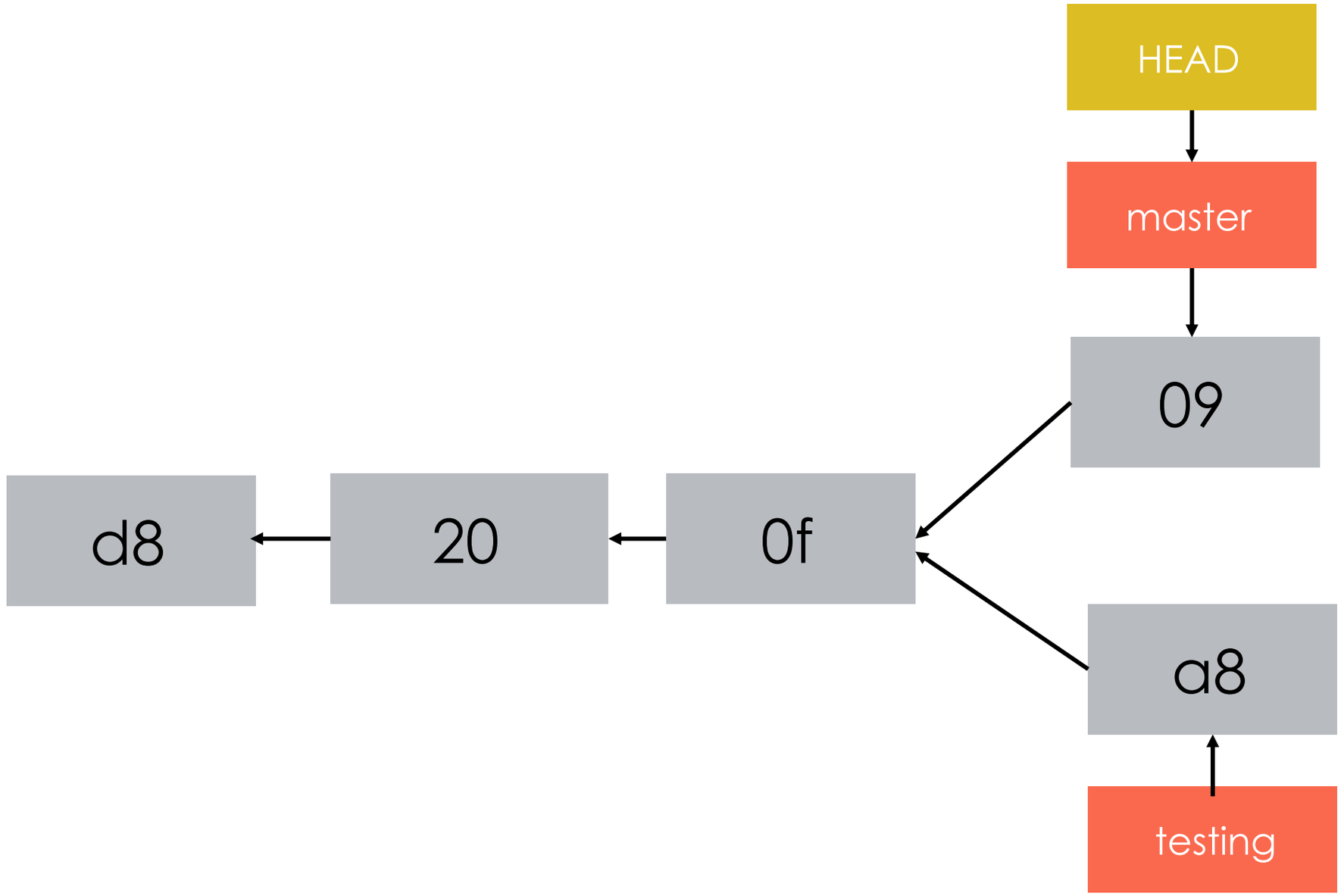
```
MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (testing)
$ git checkout master
Switched to branch 'master'
```

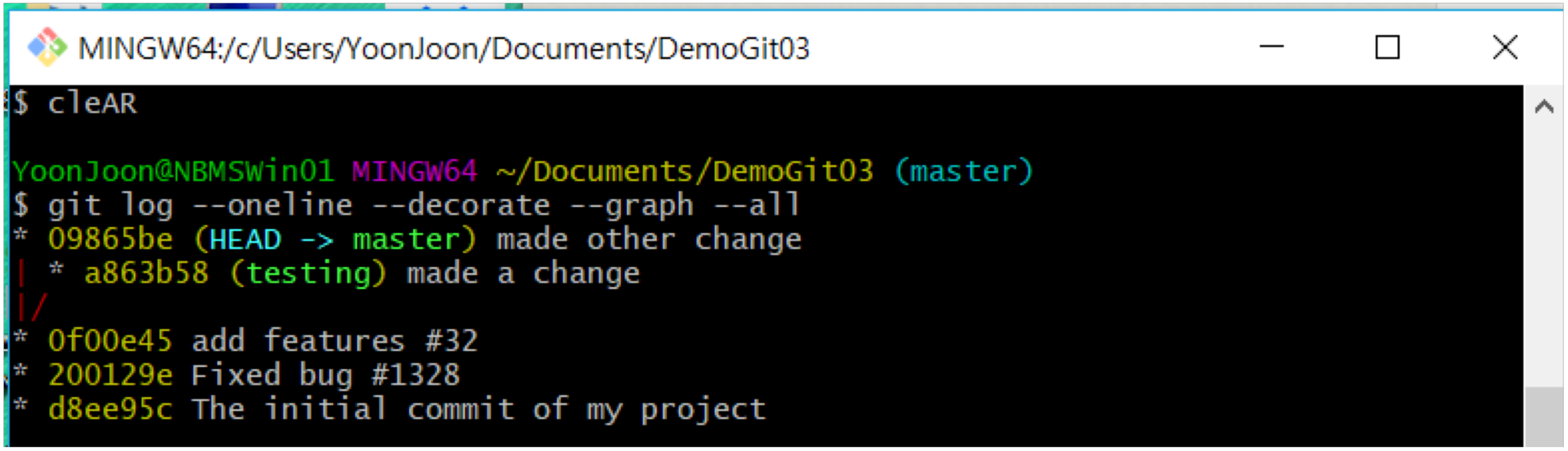



```
MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ cat test.py
# This is a python library for Branch Demo
import sys

# 1st update

# 2nd update
```

MINGW64:/c/Users/YoonJoon/Documents/DemoGit03

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ vi test.py

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ git commit -a -m 'made other change'
[master 09865be] made other change
 1 file changed, 3 insertions(+)
```




MINGW64:/c/Users/YoonJoon/Documents/DemoGit03
$ cleAR
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit03 (master)
$ git log --oneline --decorate --graph --all
* 09865be (HEAD -> master) made other change
| * a863b58 (testing) made a change
|/
* 0f00e45 add features #32
* 200129e Fixed bug #1328
* d8ee95c The initial commit of my project

Basic Branching and Merging

Let's go through a simple example of branching and merging with a workflow that we might use in the real world. We'll follow these steps:

1. Do some work on a website.

2. Create a branch for a new story we're working on.

3. Do some work in that branch.

At this stage, we'll receive a call that another issue is critical and we need a hotfix. We'll do the following:

1. Switch to our production branch.

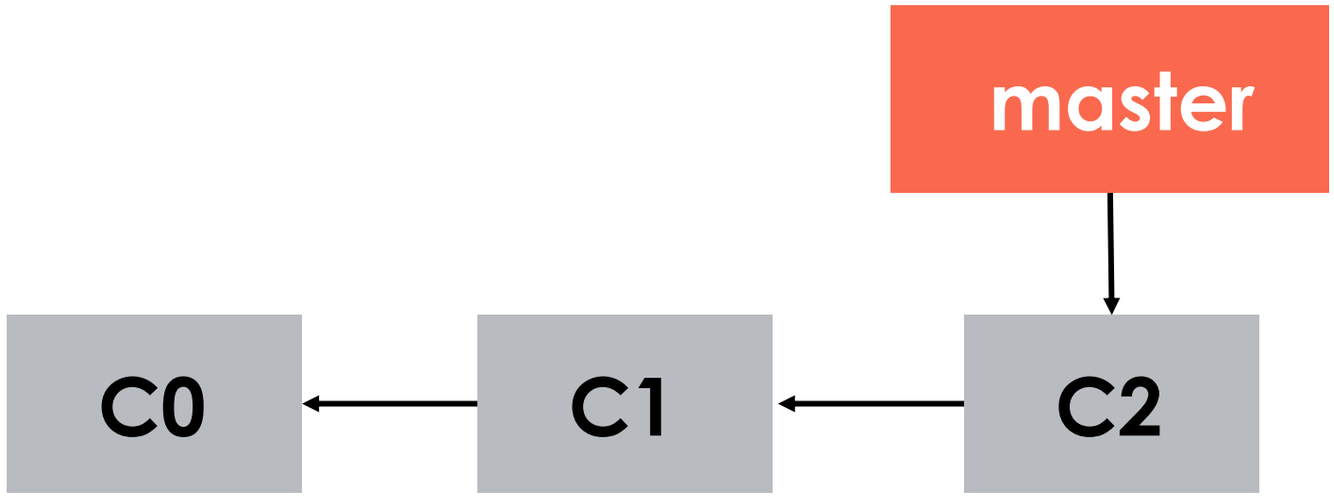
2. Create a branch to add the hotfix.

3. After it's tested, merge the hotfix branch, and push to production.

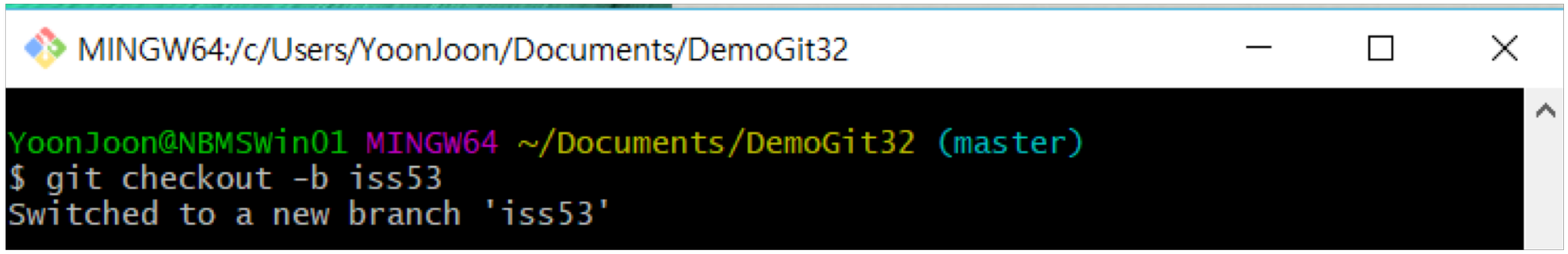
4. Switch back to our original story and continue working.

Basic Branching

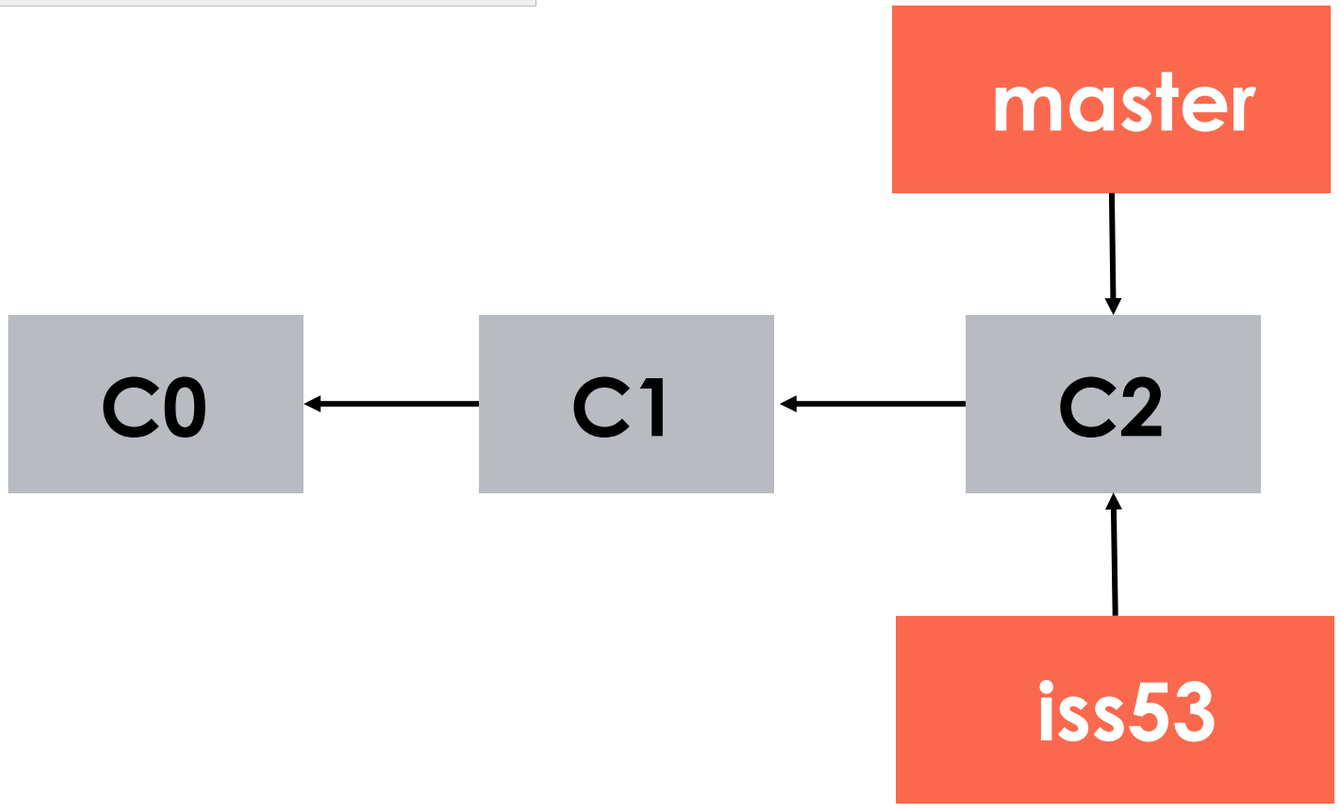
First, let's say we're working on our project and have a couple of commits already on the `master` branch.



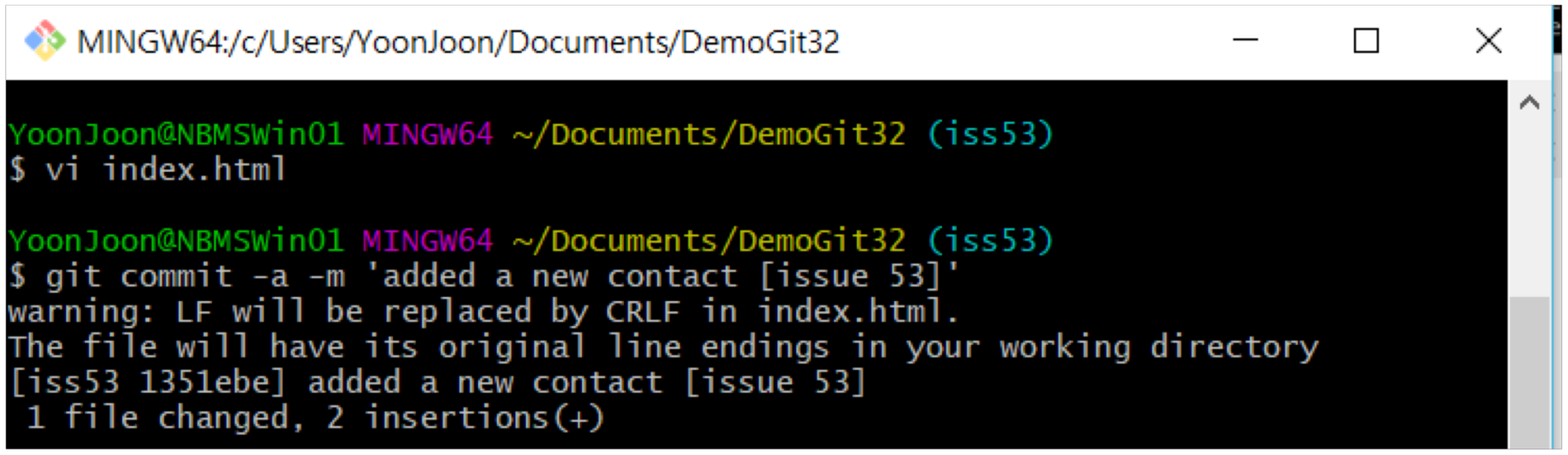
We've decided that we're going to work on issue #53. To create a new branch and switch to it at the same time, we can run the `git checkout` command with the `-b` switch:

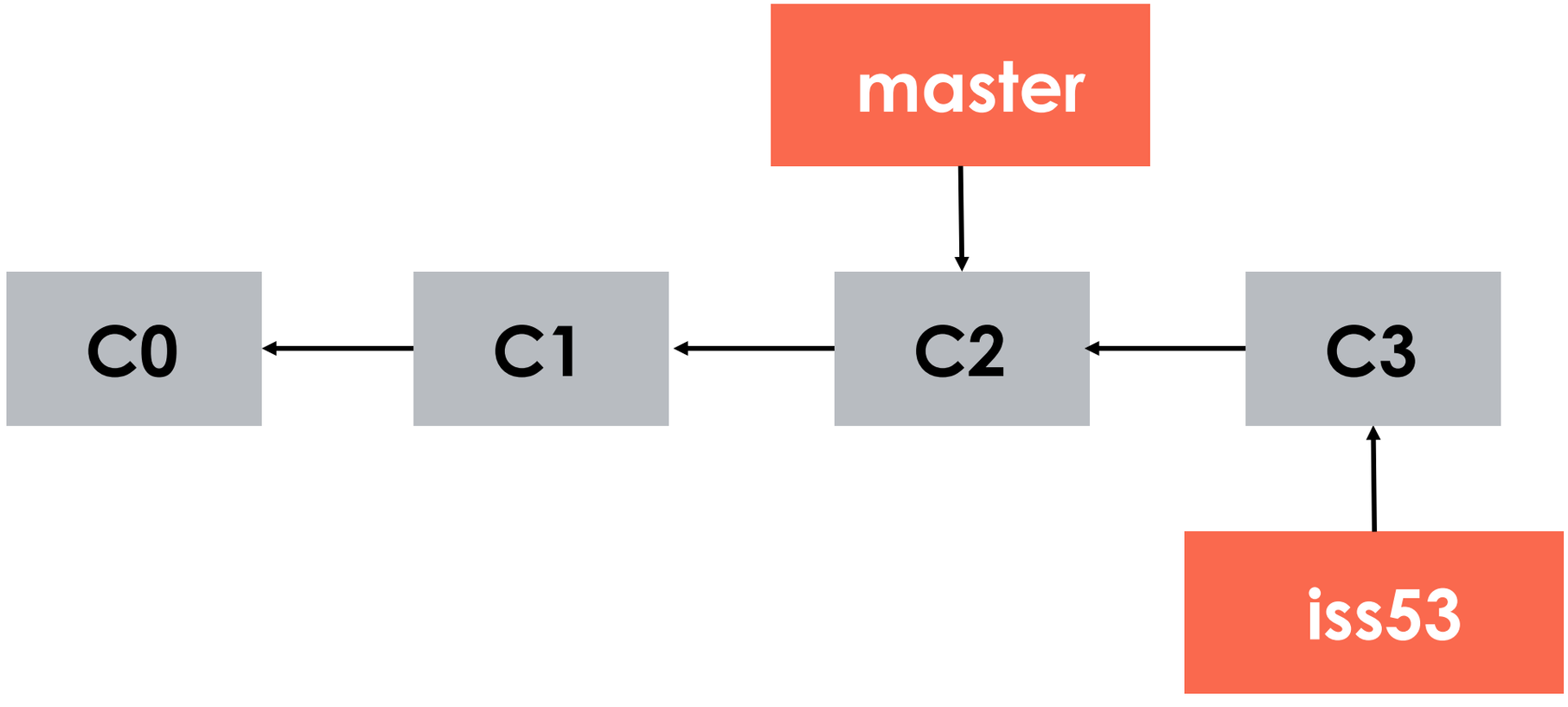



```
$ git branch iss53
$ git checkout iss53
```



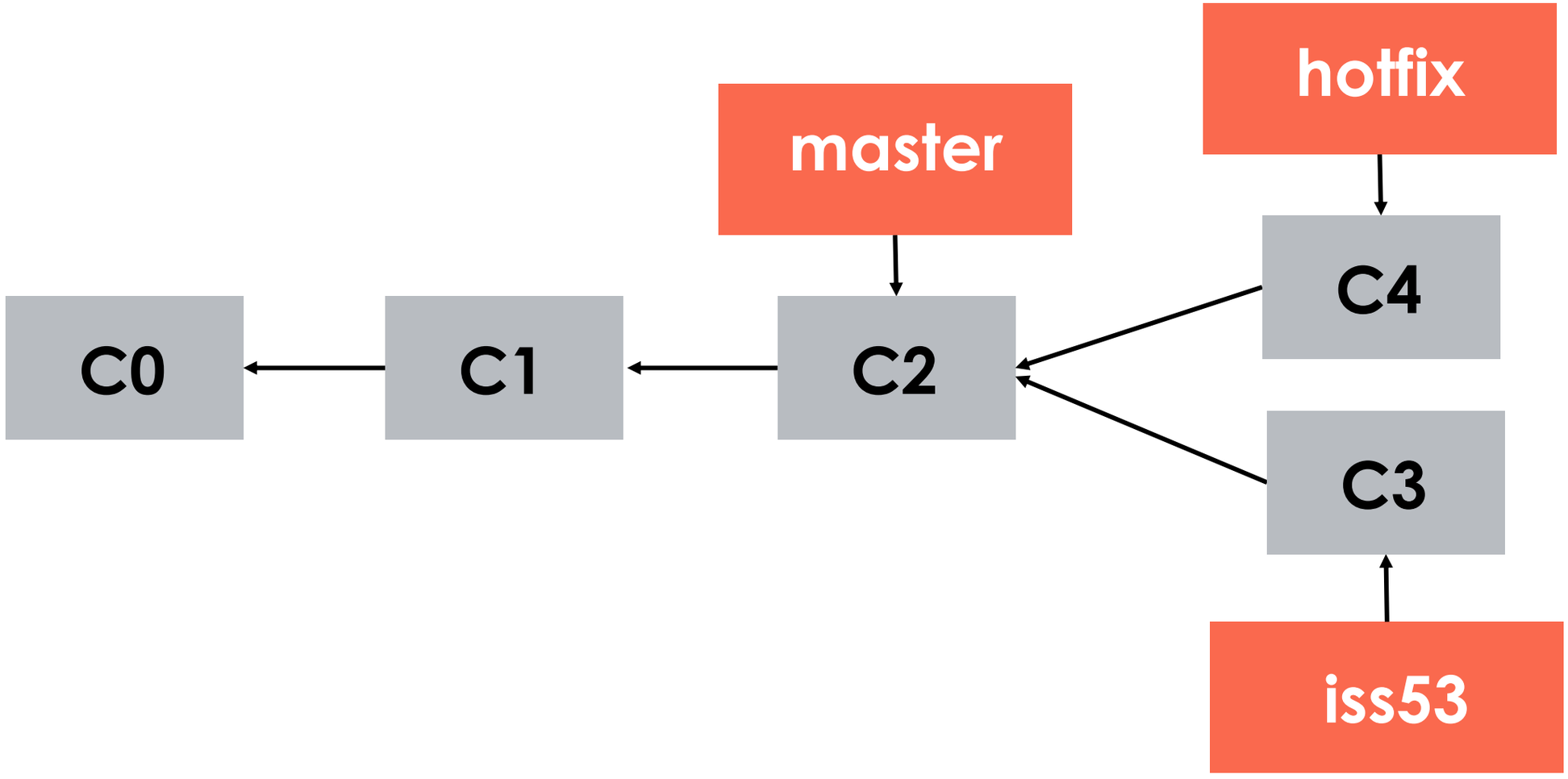
We work on our website and do some commits. Doing so moves the iss53 branch forward, because we have it checked out (that is, our HEAD is pointing to it):



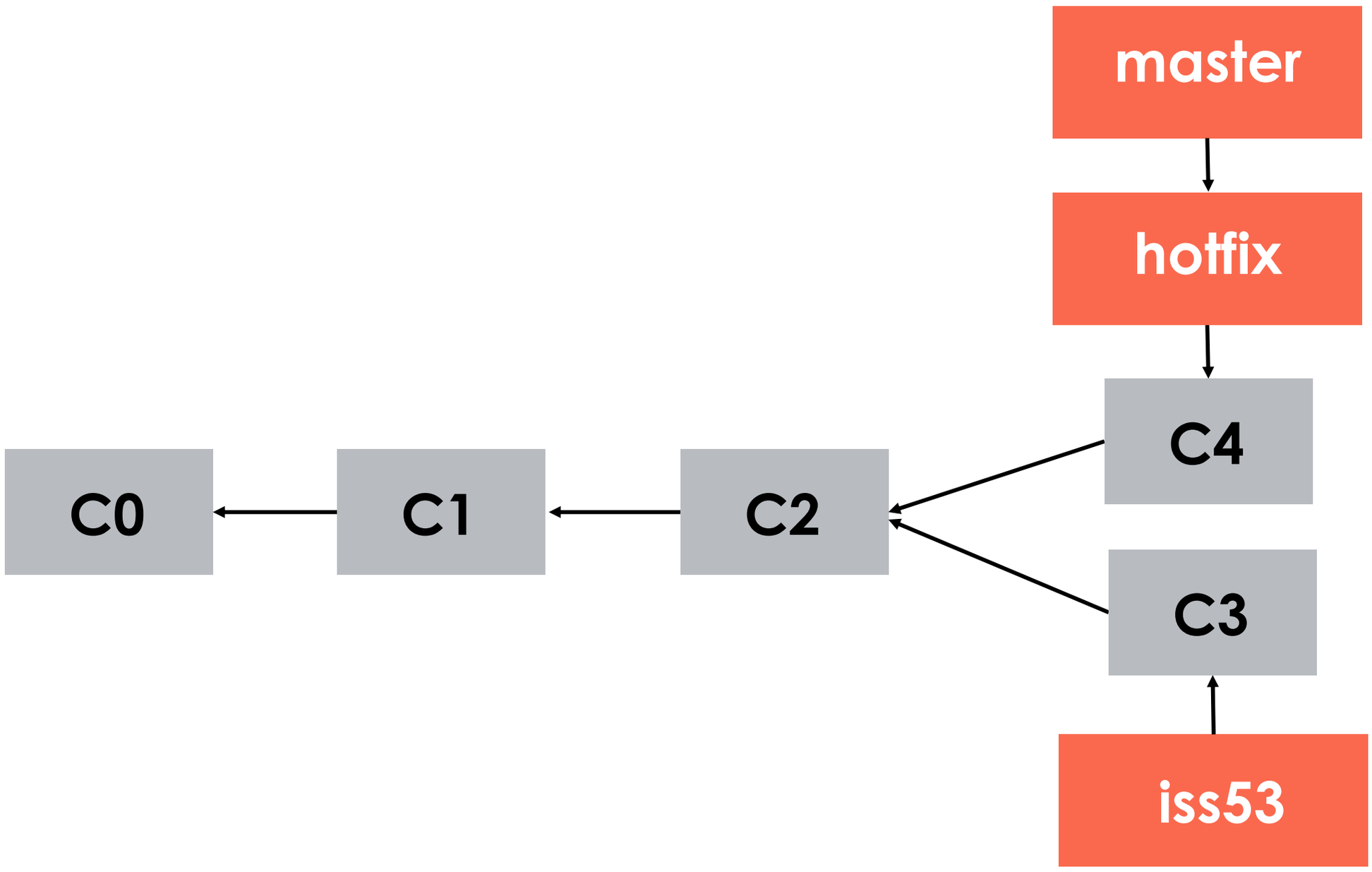


MINGW64:/c/Users/YoonJoon/Documents/DemoGit32

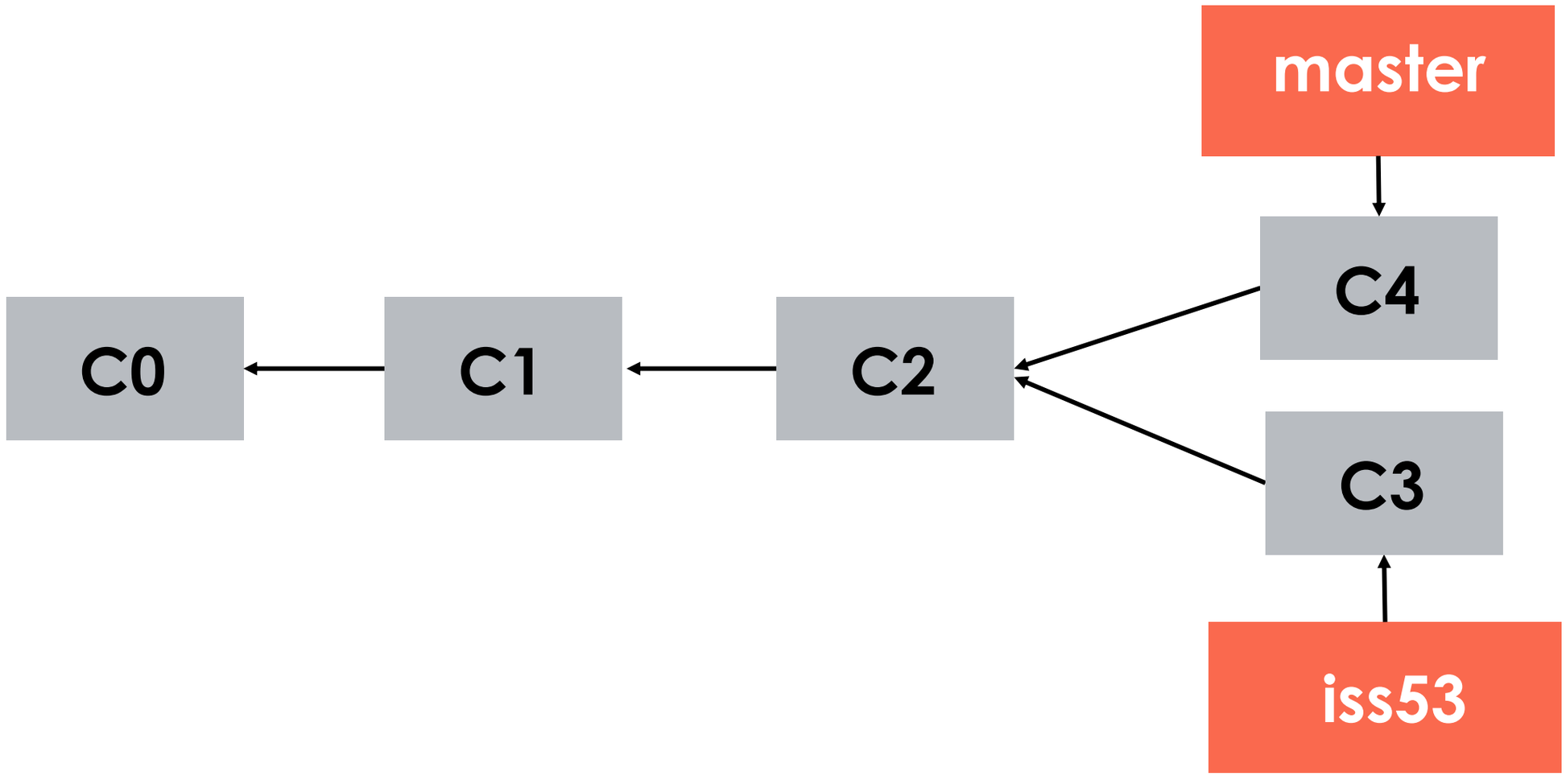
```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (iss53)
$ git checkout master
Switched to branch 'master'

$ git checkout -b hotfix
Switched to a new branch 'hotfix'

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (hotfix)
$ vi index.html

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (hotfix)
$ git commit -a -m 'fixed the broken email address'
[hotfix abf2a5e] fixed the broken email address
 1 file changed, 2 insertions(+)
```



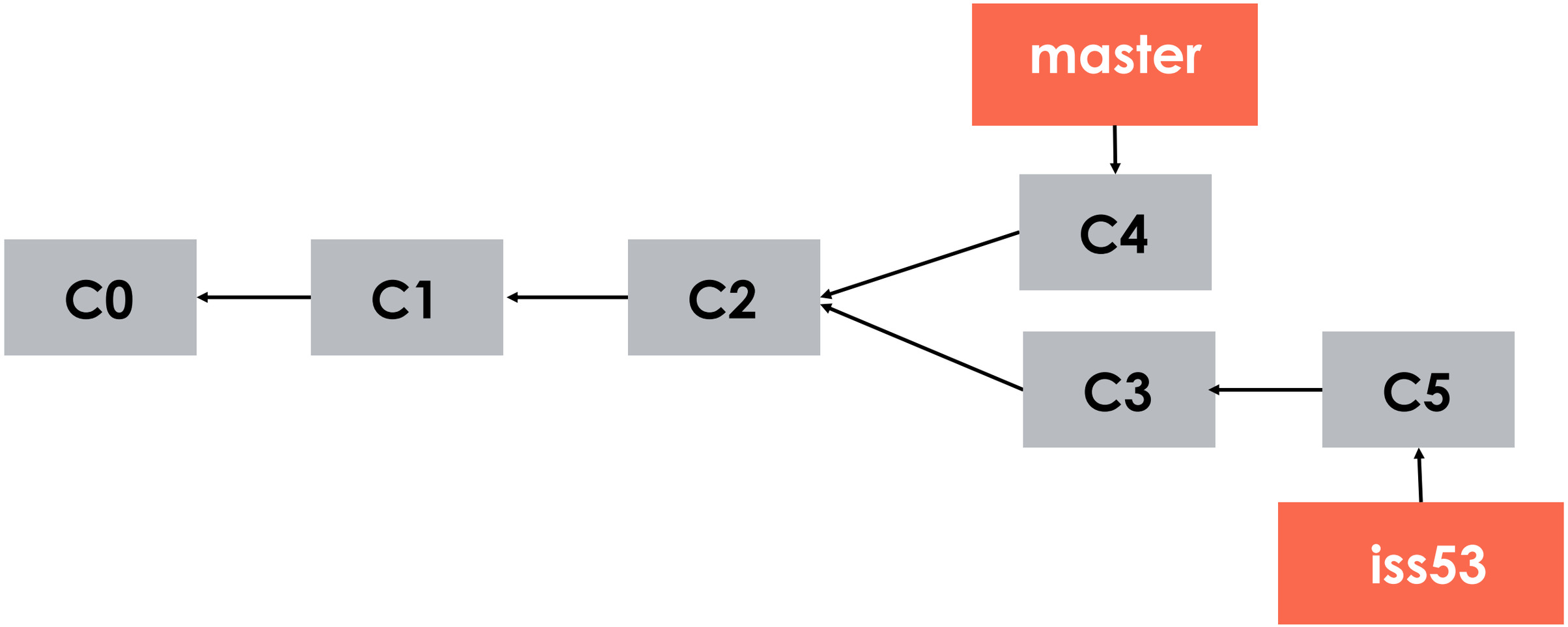
```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (hotfix)
$ git checkout master
Switched to branch 'master'

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git merge hotfix
Updating 5af714f..abf2a5e
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```



```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git branch -d hotfix
Deleted branch hotfix (was abf2a5e).
```



```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git checkout iss53
Switched to branch 'iss53'

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (iss53)
$ vi index.html

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (iss53)
$ git commit -a -m 'finished the new footer [issue 53]'
[iss53 59c94f2] finished the new footer [issue 53]
 1 file changed, 1 insertion(+), 1 deletion(-)
```



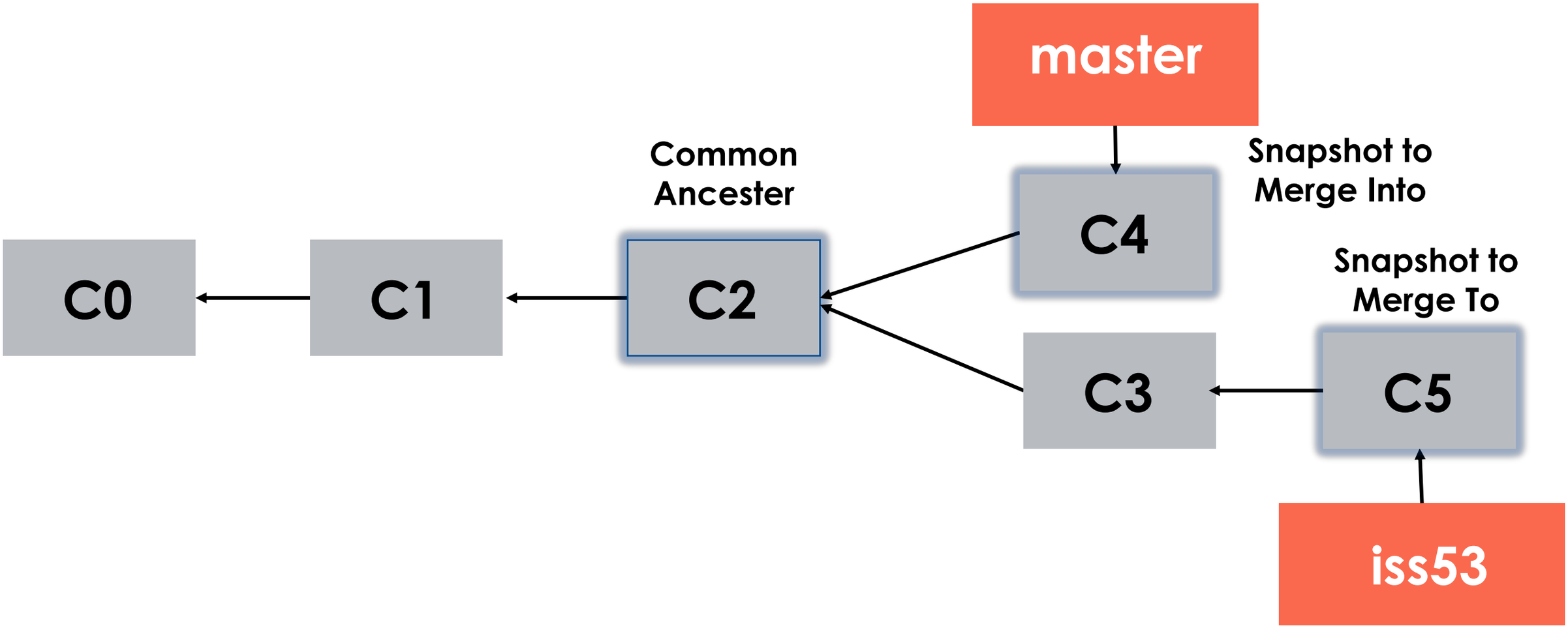
```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (iss53)
$ cat index.html
# This is demo for 3.2

C2

contact at us : email.support@github.com
```

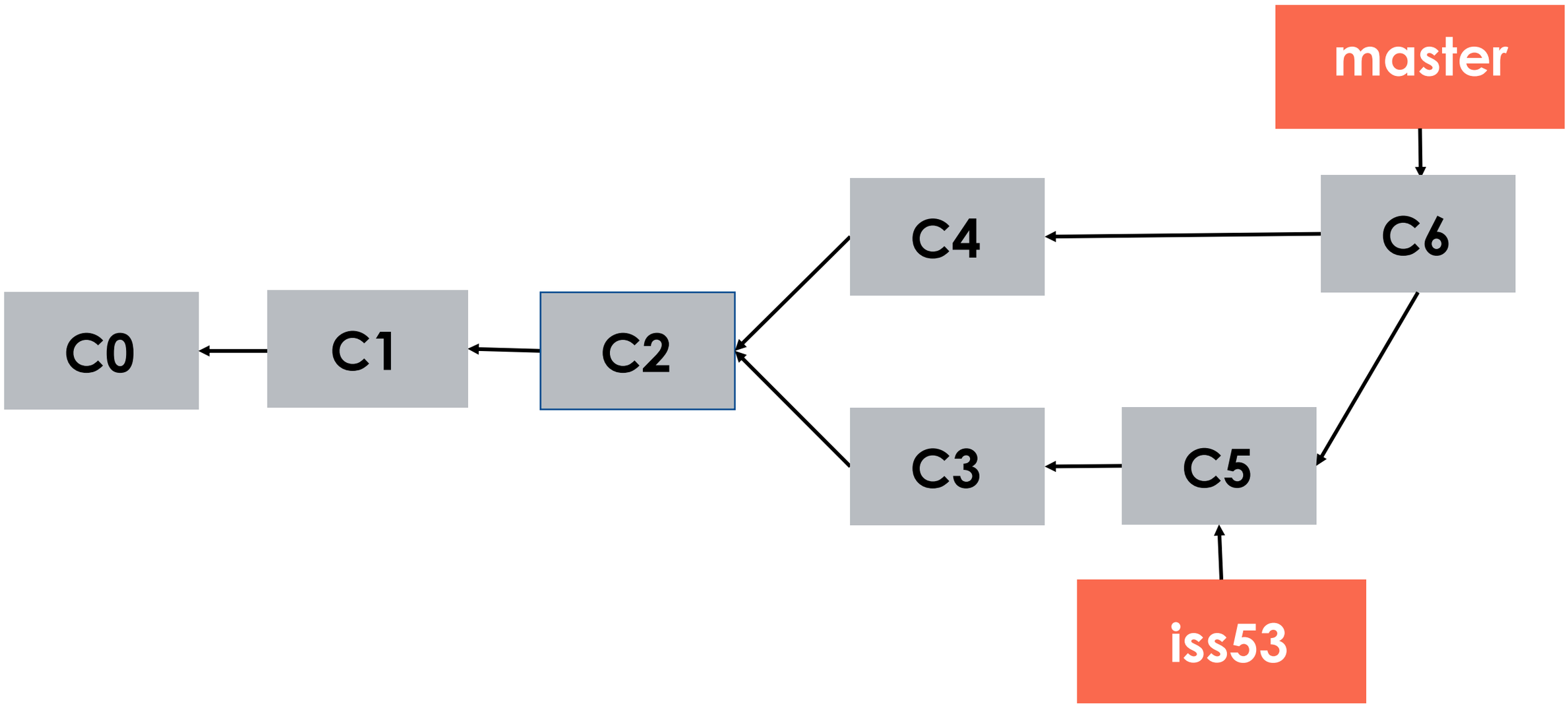
```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ cat index.html
# This is demo for 3.2

C2

contact us at suuport@github.com
```

Basic Merging

Suppose we've decided that our issue #53 work is complete and ready to be merged into our `master` branch. In order to do that, we'll merge our `iss53` branch into `master`, much like we merged our `hotfix` branch earlier. All we have to do is check out the branch we wish to merge into and then run the `git merge` command:

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```


master
Common Ancester
Snapshot to Merge Into
Snapshot to Merge To
C0
C1
C2
C4
C3
C5
iss53


master
C4
C6
C0
C1
C2
C3
C5
iss53

Basic Merge Conflicts

Occasionally, this process doesn't go smoothly. If we changed the same part of the same file differently in the two branches we're merging together, Git won't be able to merge them cleanly. If our fix for issue #53 modified the same part of a file as the `hotfix` branch, we'll get a merge conflict that looks something like this:

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git checkout master
Already on 'master'

YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

If we want to see which files are unmerged at any point after a merge conflict, we can run `git status`:

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

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

        both modified:   index.html

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

Our file contains a section that looks something like this:

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master|MERGING)
$ cat index.html
# This is demo for 3.2

C2

<<<<<<< HEAD
contact us at suuport@github.com
=======
contact at us : email.support@github.com
>>>>>>> iss53
```

In order to resolve the conflict, we have to either choose one side or the other or merge the contents ourselves. For instance, we might resolve this conflict by replacing the entire block with this:

```
please contact us at email.support@github.com
```

If we want to use a graphical tool to resolve these issues, we can run `git mergetool`, which fires up an appropriate visual merge tool and walks we through the conflicts:

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecme
rge p4merge araxis bc codecompare emerge vimdiff
Merging:
index.html

Normal merge conflict for 'index.html':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 파 일 을  고 치 기
```

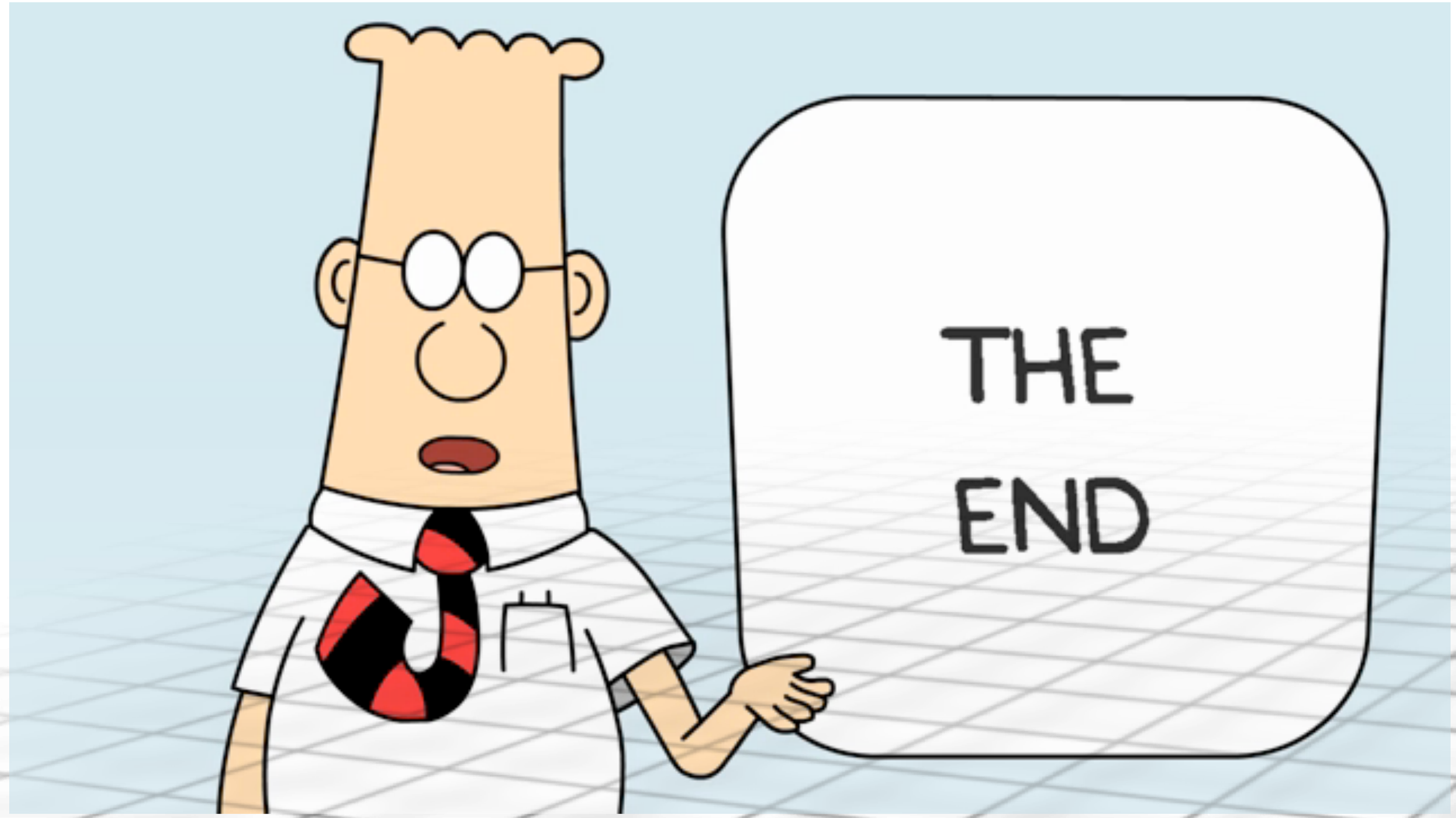
We can run `git status` again to verify that all conflicts have been resolved:

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master|MERGING)
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   index.html

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

        index.html.orig
```

If we're happy with that, and we verify that everything that had conflicts has been staged, we can type `git commit` to finalize the merge commit.

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master|MERGING)
$ git commit -m 'resolve merge conflict'
[master 850becc] resolve merge conflict
```

```
YoonJoon@NBMSWin01 MINGW64 ~/Documents/DemoGit32 (master)
$ git log --oneline --decorate --graph --all
*   850becc (HEAD -> master) resolve merge conflict
|\
| * 59c94f2 (iss53) finished the new footer [issue 53]
| * 1351ebe added a new contact [issue 53]
* | abf2a5e fixed the broken email address
|/
* 5af714f 2nd update
* e8da603 the ist update
* 584715b created index.html
```



출처: metachannels.com

감사합니다