Leiden University
Medical Center

Code and data management with Git

Combining changes by merging

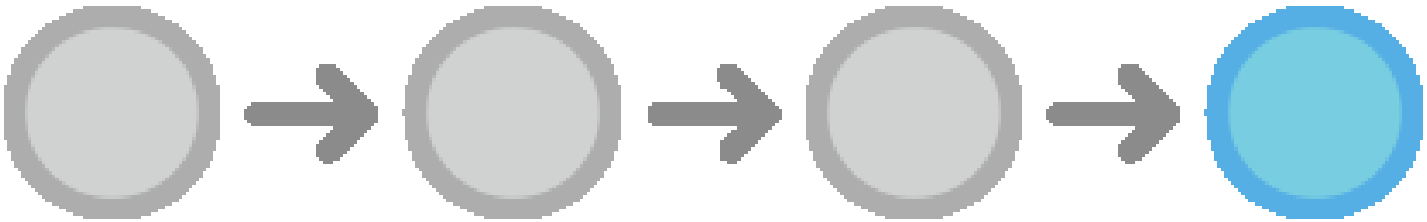
Martijn Vermaat

Department of Human Genetics
Center for Human and Clinical Genetics

Table of contents

The Git commit graph

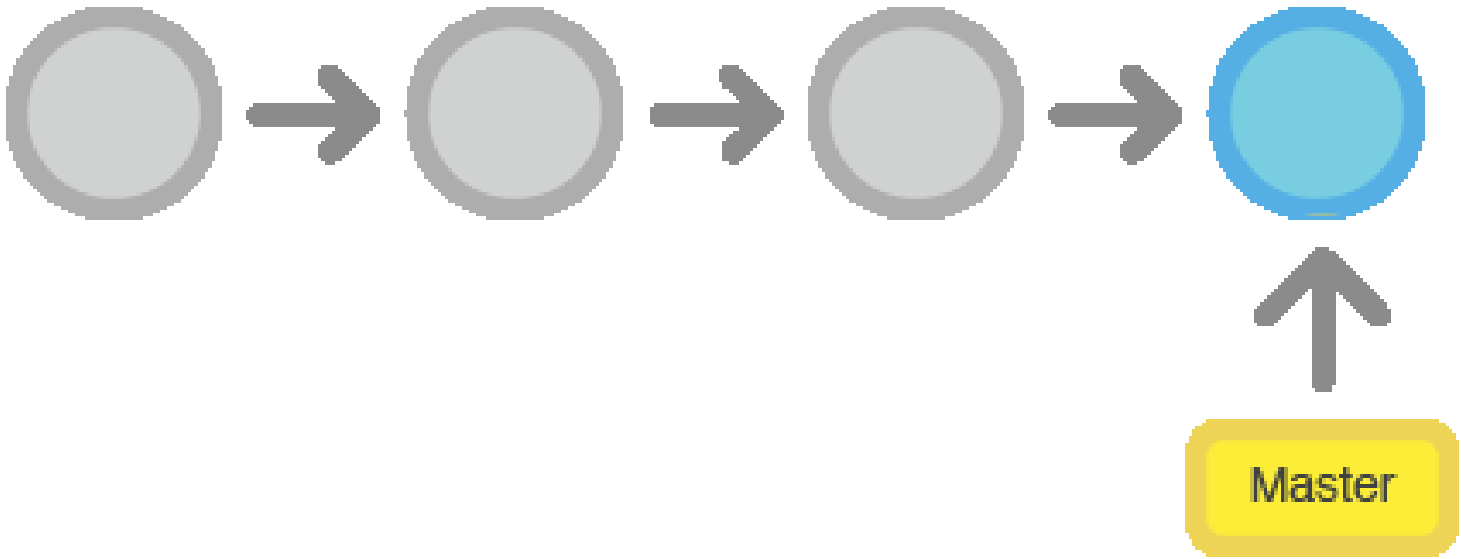
A linear history



- Every commit has a parent.
- *Committing* creates a new commit on top of the current one.

The Git commit graph

The default `master` branch

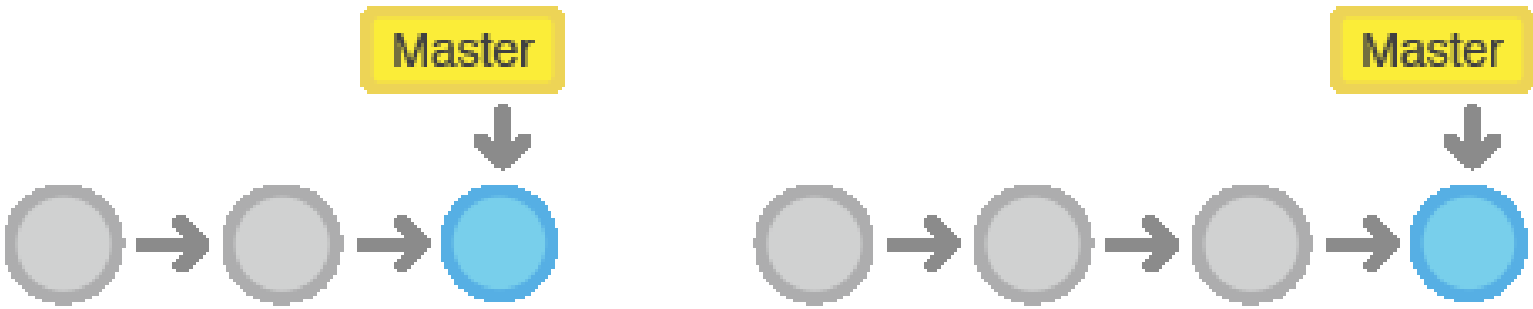


- A *branch* is a pointer to a commit.
- By default there is one branch: `master`.
- `HEAD` is a special pointer, it points to the current branch.

The Git commit graph

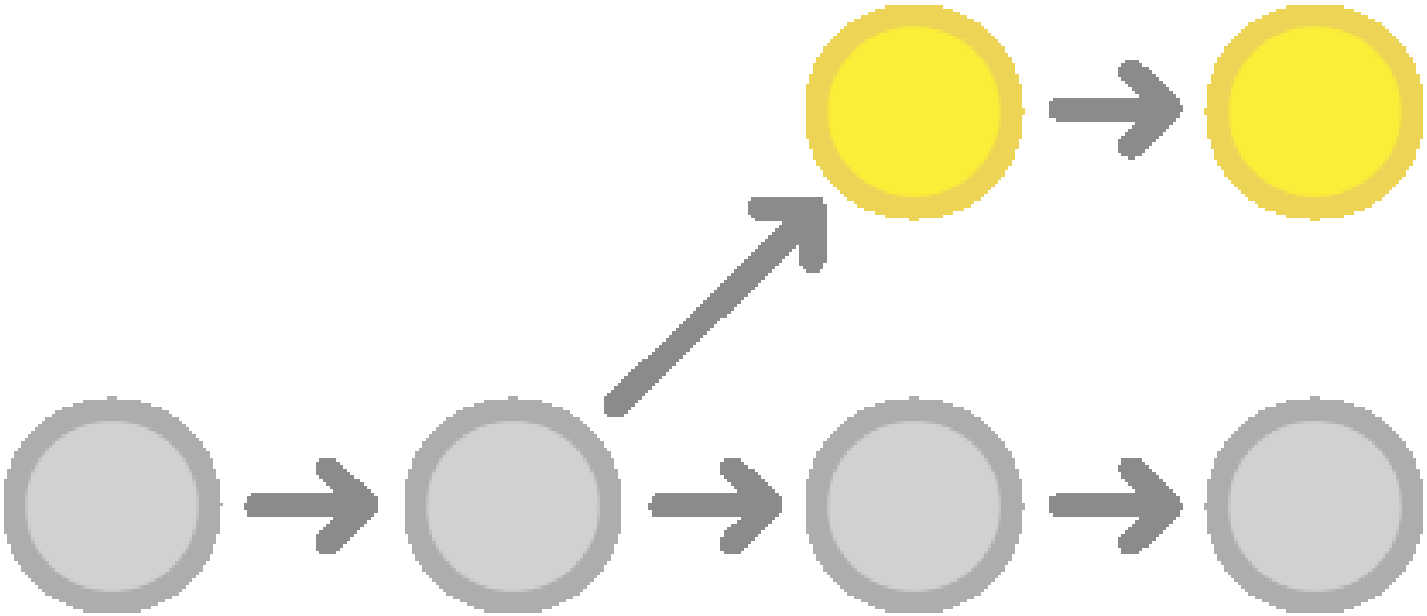
Committing moves the current branch pointer

- Committing moves the current branch to the new commit.
- (Of course, `HEAD` moves with it.)



The Git commit graph

A non-linear history

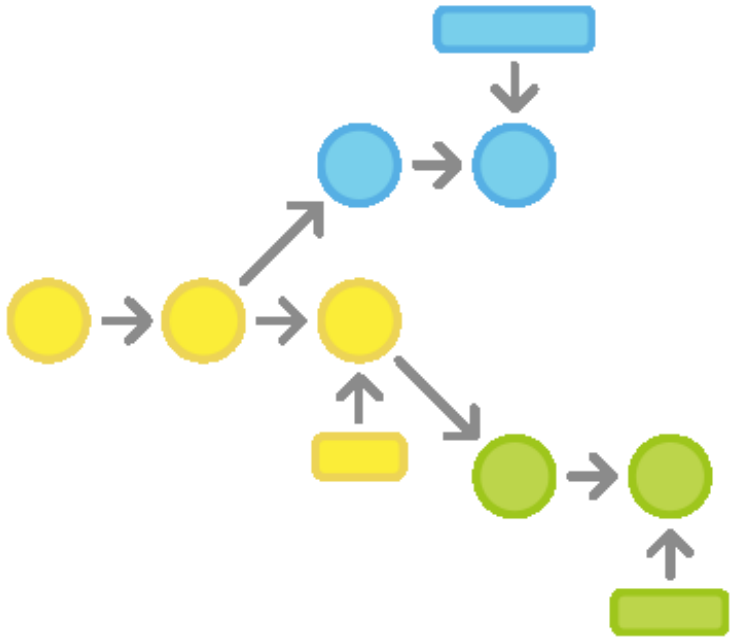


- The commit graph can become non-linear.
- Usually by committing from the same commit twice.

The Git commit graph

Use branches to keep track of different code paths

- Branch names are easier to remember than commit hashes.
- Branch names make it clear what commits are about.



The Git commit graph

Different code paths may later join

- Commits from different branches can be brought together.
- We call this *merging*.

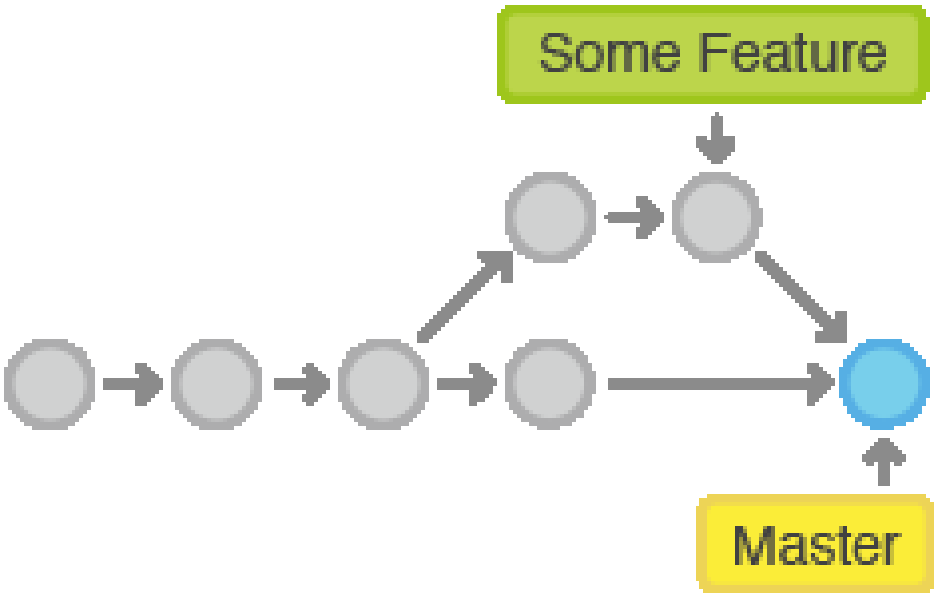
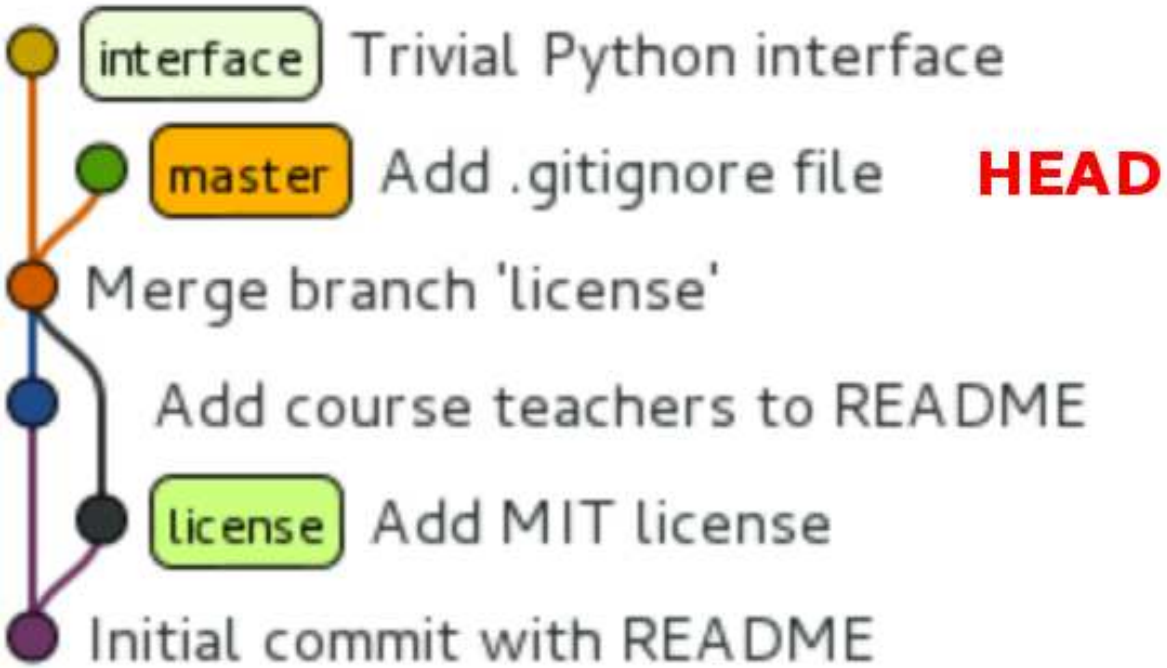


Table of contents

Inspecting the commit graph

An example repository



- Branches `license` and `interface` diverged from `master`.
- Only `license` has been merged back into `master`.
- Current branch is `master`.

Inspecting the commit graph

Showing the current branch: `git status`

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Remember: you cannot type `git status` enough!

Inspecting the commit graph

The commit log: `git log`

```
$ git log --oneline --decorate
c7f3bd9 (HEAD, master) Add .gitignore file
4a44c4e Merge branch 'license'
64af1ee Add course teachers to README
0fbe3e3 (license) Add MIT license
d1c7fd7 Initial commit with README
```

`--oneline`: Shows commit summary on one line.

`--decorate`: Adds branch information.

Inspecting the commit graph

The commit log as a graph: `git log`

```
$ git log --oneline --decorate --graph --all
* 8fc25c1 (interface) Trivial Python interface
| * c7f3bd9 (HEAD, master) Add .gitignore file
|/
*   4a44c4e Merge branch 'license'
|\
| * 0fbe3e3 (license) Add MIT license
* | 64af1ee Add course teachers to README
|/
* d1c7fd7 Initial commit with README
```

`--graph`: Shows the commit graph.
`--all`: Includes all branches instead of just the current.

Annotated log as an alias

For convenience, we can create an alias for the `git log` command with all the arguments we just used:

```
$ git config --global alias.l \
    'log --oneline --decorate --graph --all'
$ git l
* 8fc25c1 (interface) Trivial Python interface
| * c7f3bd9 (HEAD, master) Add .gitignore file
|/
*   4a44c4e Merge branch 'license'
...
```

This alias is stored in the `~/.gitconfig` file. You can also edit that file manually.

Remote branches

In this course, we don't really discuss branches, except:

- Commits from a remote show up as a remote branch.
- Even with only a `master` branch, remotes might still diverge.
- Merging remote branches is just like merging local branches.

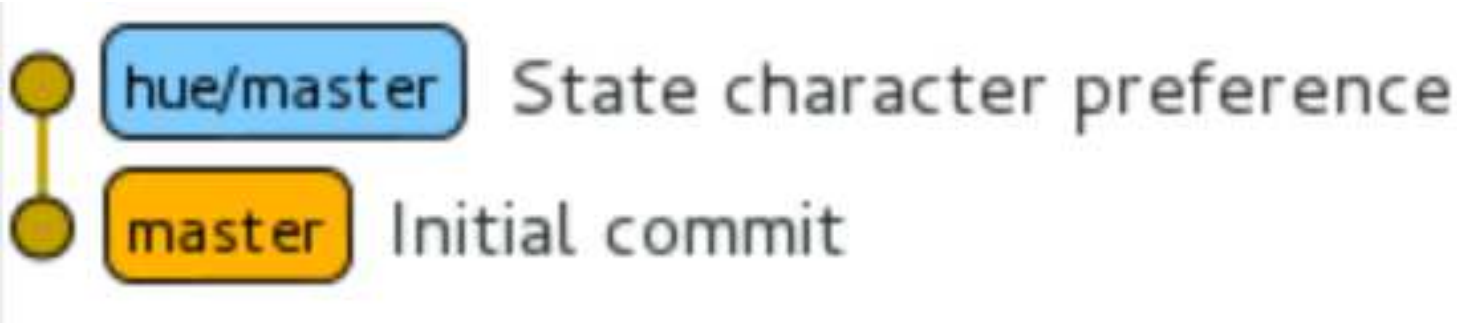
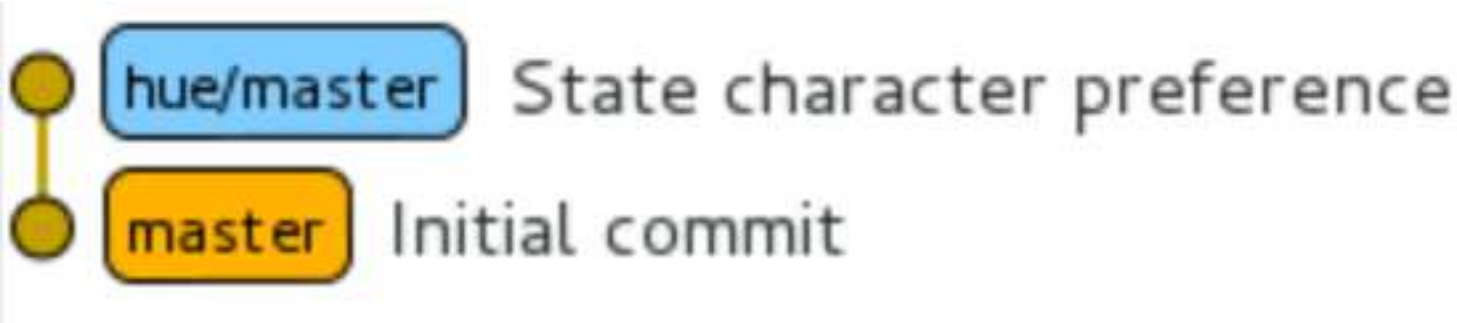


Table of contents

Merging from remotes

An example repository



Branch `master` on `hue` is one commit ahead of our `master` and we have already seen how to merge `hue/master` into `master`:

- `git merge hue/master`
- This was easy: Git just moves `master` (and `HEAD`) to point to `hue/master`.
- It is called a *fast forward merge*.

Merging from remotes

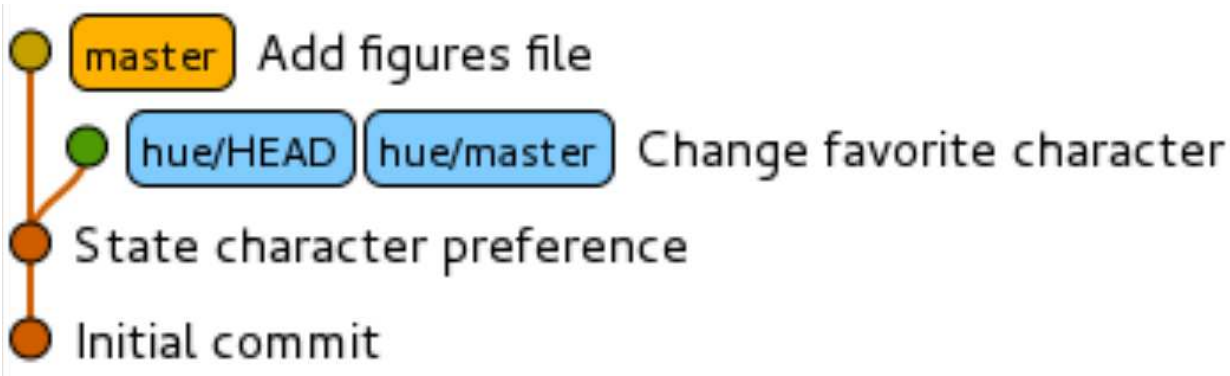
Fast forward merging: `git merge`

If a remote branch is ahead of us, it can be merged easily:

```
$ git merge hue/master
Updating c7f3bd9..251a51b
Fast-forward
testlib.py |    2 +
1 file changed, 2 insertions(+)
```

Merging from remotes

An example repository



Branch `master` on `hue` and our `master` have diverged and merging now seems less straightforward.

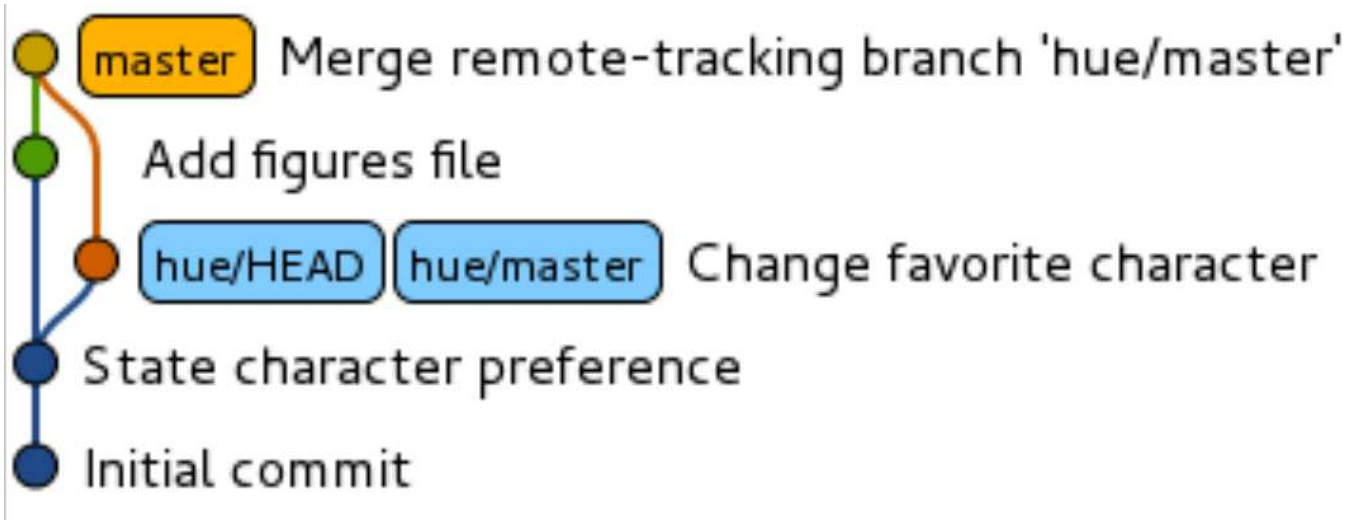
- `git merge hue/master`
- Git will create a new commit in which it combines both changes.
- It is called a *three-way merge*.
- The *merge commit* has two parents.

Merging from remotes

Three-way merging: `git merge`

Let's merge the remote work on `master` back into our local `master`:

```
$ git merge hue/master
Merge made by the 'recursive' strategy.
 FACTS.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```



Merging from remotes

The other way around

`git push` also does an implicit `git merge` on the remote.

- We send our local commits to the remote.
- They are merged into the remote branch.
- But only if this is a fast-forward merge.
- Diverging commits *cannot be pushed directly*.

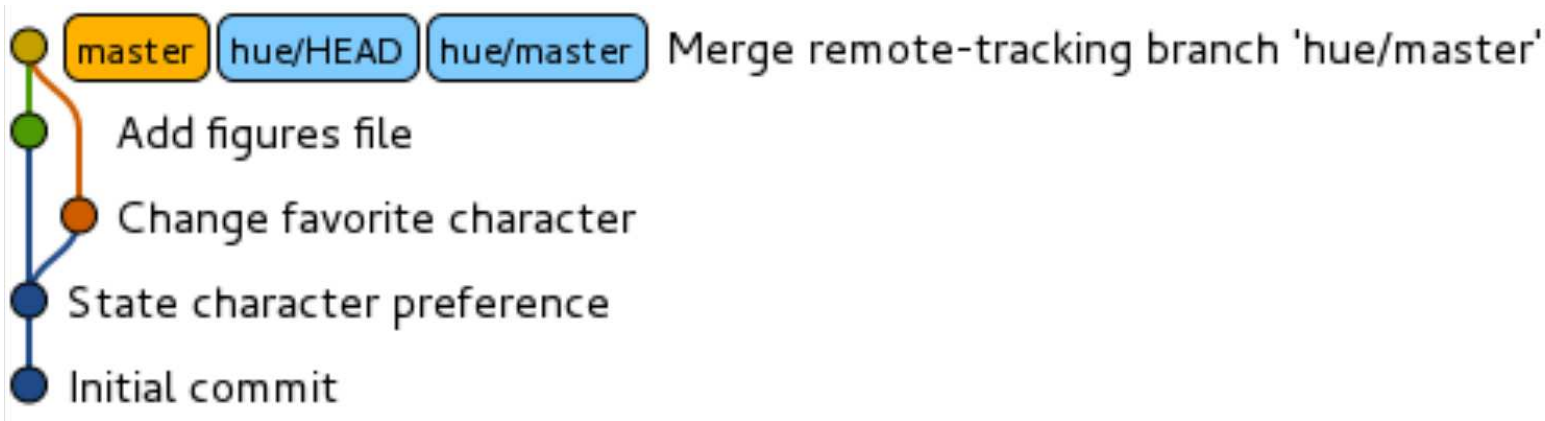


Table of contents

Merge conflicts

Git is pretty good at merging:

- The changes might have been in different files.
- Or in different parts of the same file.
- Git tries to figure out a sensible result.

Merge conflicts

Git is pretty good at merging:

- The changes might have been in different files.
- Or in different parts of the same file.
- Git tries to figure out a sensible result.

Sometimes this is not possible:

- The changes might be incompatible.
- When we try `git merge`, Git gives up.
- This situation is called a *merge conflict*.

Setting the stage (1/3)

```
* 5edaf08 (hue/master, hue/HEAD) State character
| * f1ef19c (HEAD, master) State character prefer
|/
* 1f6d2ab Initial commit
```

We'd like to merge branch `hue/master` into `master`.

Setting the stage (2/3)

The last commit on `master`:

```
f1ef19c State character preference
diff --git a/FACTS.md b/FACTS.md
index de15194..ef40359 100644
--- a/FACTS.md
+++ b/FACTS.md
@@ -1,2 +1,4 @@
Facts about television series
==============================
+
+My favorite character is Eric Cartman.
```

Setting the stage (3/3)

The last commit on `hue/master`:

```
$ git show --oneline hue/master
5edaf08 State character preference
diff --git a/FACTS.md b/FACTS.md
index de15194..5e69508 100644
--- a/FACTS.md
+++ b/FACTS.md
@@ -1,2 +1,4 @@
Facts about television series
==============================
+
+My favorite character is Milhouse.
```

Creating a merge conflict

```
$ git merge hue/master
Auto-merging FACTS.md
CONFLICT (content): Merge conflict in FACTS.md
Automatic merge failed; fix conflicts and then
commit the result.

$ git status
# On branch master
# Unmerged paths:
#   (use "git add/rm <file>.." as appropriate to
#    mark resolution)
#
#               both modified:      FACTS.md
#
no changes added to commit (use "git add" and/or
"git commit -a")
```

Resolving a merge conflict (1/2)

```
$ cat FACTS.md
Facts about television series
==============================

<<<<<<< HEAD
My favorite character is Eric Cartman.
=======
My favorite character is Milhouse.
>>>>>>> hue/master
```

What we had is under `HEAD`, what `hue/master` had is above `hue/master`.

Resolving a merge conflict (1/2)

```
$ cat FACTS.md
Facts about television series
==============================

<<<<<<< HEAD
My favorite character is Eric Cartman.
=======
My favorite character is Milhouse.
>>>>>>> hue/master
```

What we had is under `HEAD`, what `hue/master` had is above `hue/master`.

```
$ nano FACTS.md
```

Resolving a merge conflict (2/2)

We resolve the conflict by hand.

```
$ cat FACTS.md
Facts about television series
==============================

My favorite characters are Eric Cartman
and Milhouse.
```

Resolving a merge conflict (2/2)

We resolve the conflict by hand.

```
$ cat FACTS.md
Facts about television series
==============================

My favorite characters are Eric Cartman
and Milhouse.
```

And can now finish the merge commit.

```
$ git add FACTS.md
$ git commit
[master 1e496cb] Merge remote-tracking branch
'hue/master'
```

Aborting a merge

If you don't feel like resolving the merge conflict, you can go back with `git merge --abort`.

```
$ git merge hue/master
Auto-merging FACTS.md
CONFLICT (content): Merge conflict in FACTS.md
Automatic merge failed; fix conflicts and then
commit the result.
```

```
$ git merge --abort
```

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

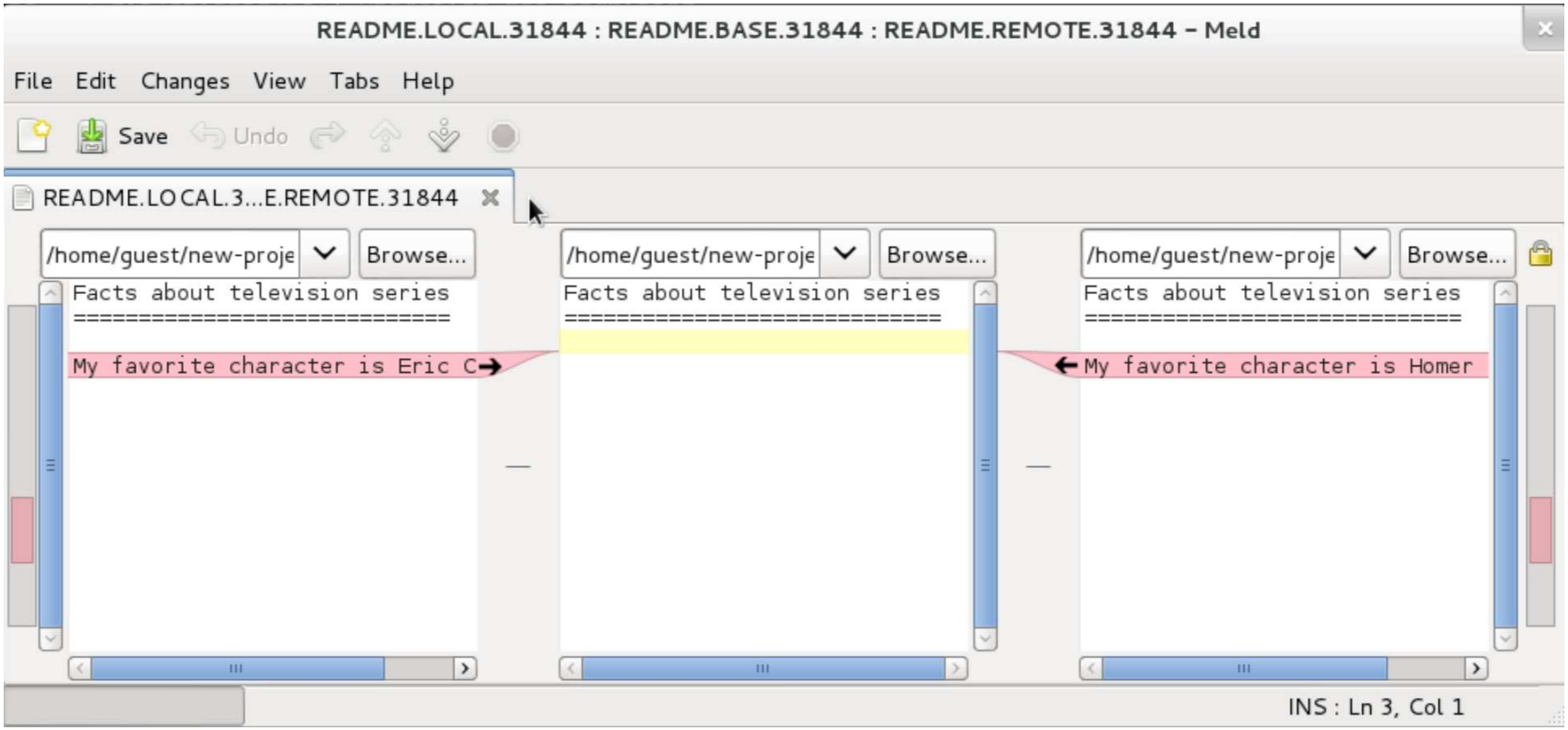
Resolving conflicts with `git mergetool`

We can also use graphical merge tools such as *Meld*.

```
$ git mergetool
merge tool candidates: meld opendiff kdiff3 ...
Merging:
FACTS.md

Normal merge conflict for 'FACTS.md':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (meld):
```

Meld example



Tools like Meld provide an editable three-way diff.

Leiden University
Medical Center

Acknowledgements

Martijn Vermaat
Wibowo Arindrarto
Szymon Kiełbasa
Jeroen Laros

http://git-scm.com/book
https://www.atlassian.com/git