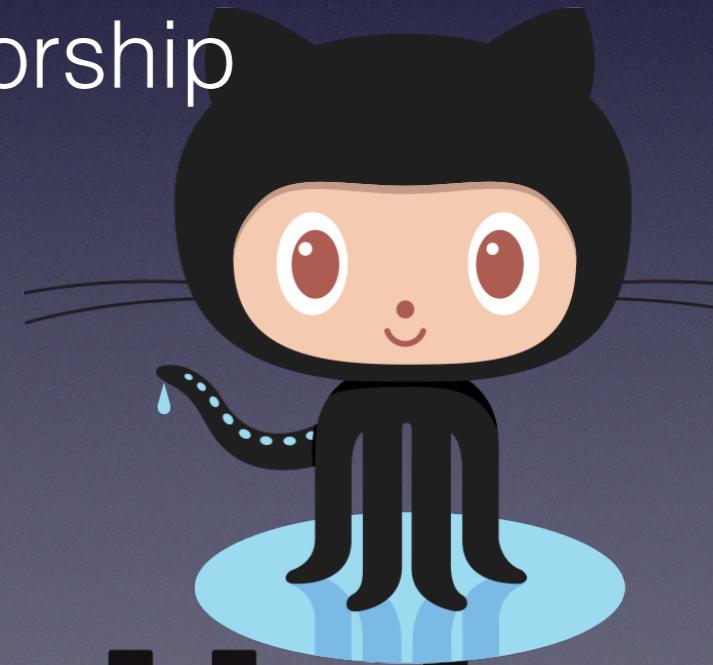
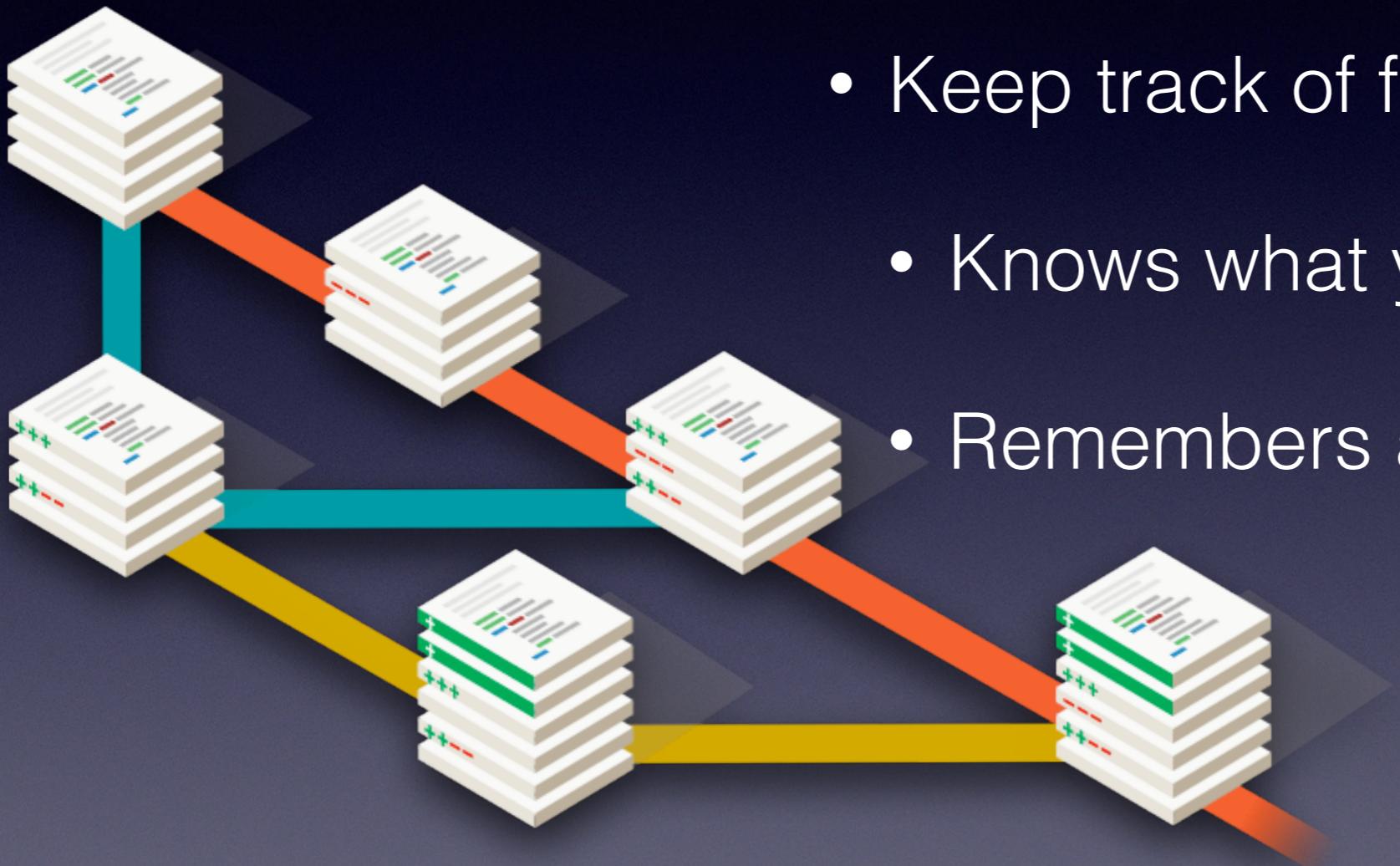


# A brief introduction to git and Github

Kiko Fernandez-Reyes

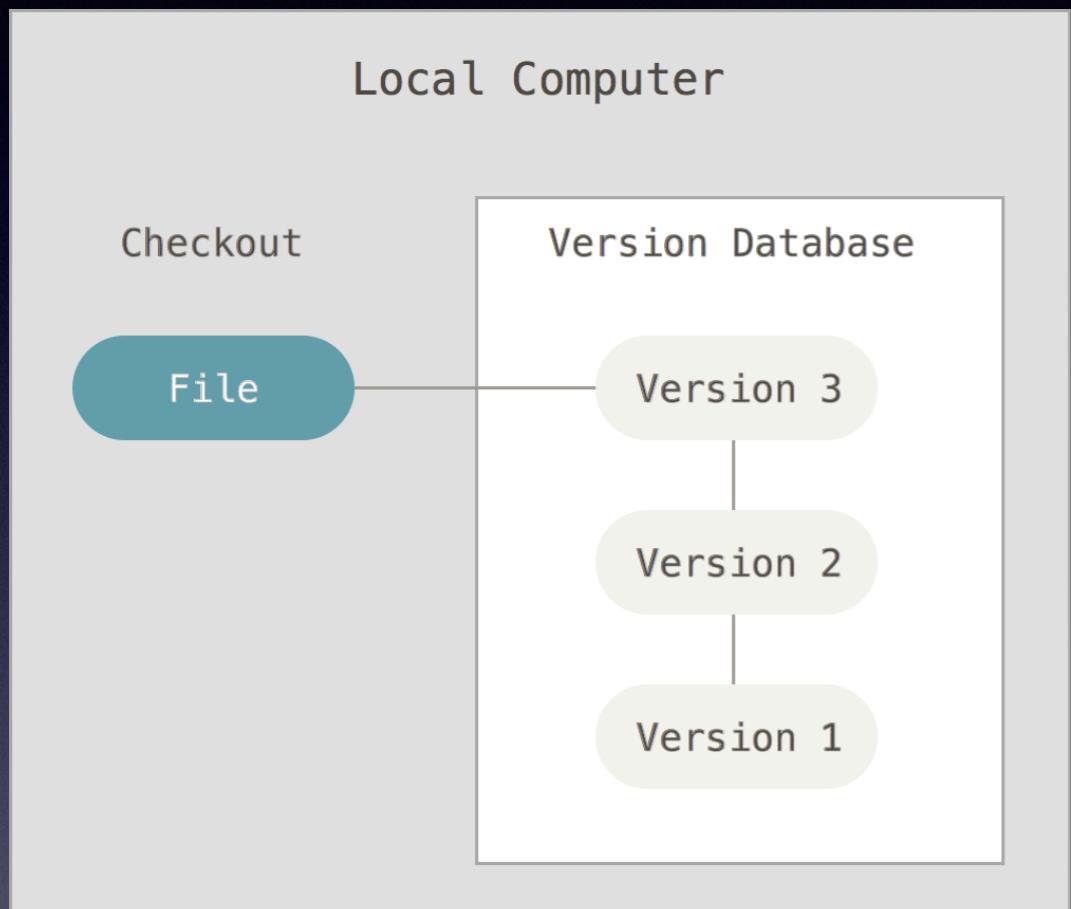
# Git: Distributed Version Control System

- Remote and local copies
- Keep track of file changes
- Knows what you did last summer
- Remembers authorship



# GitHub

# Git: Distributed Version Control System

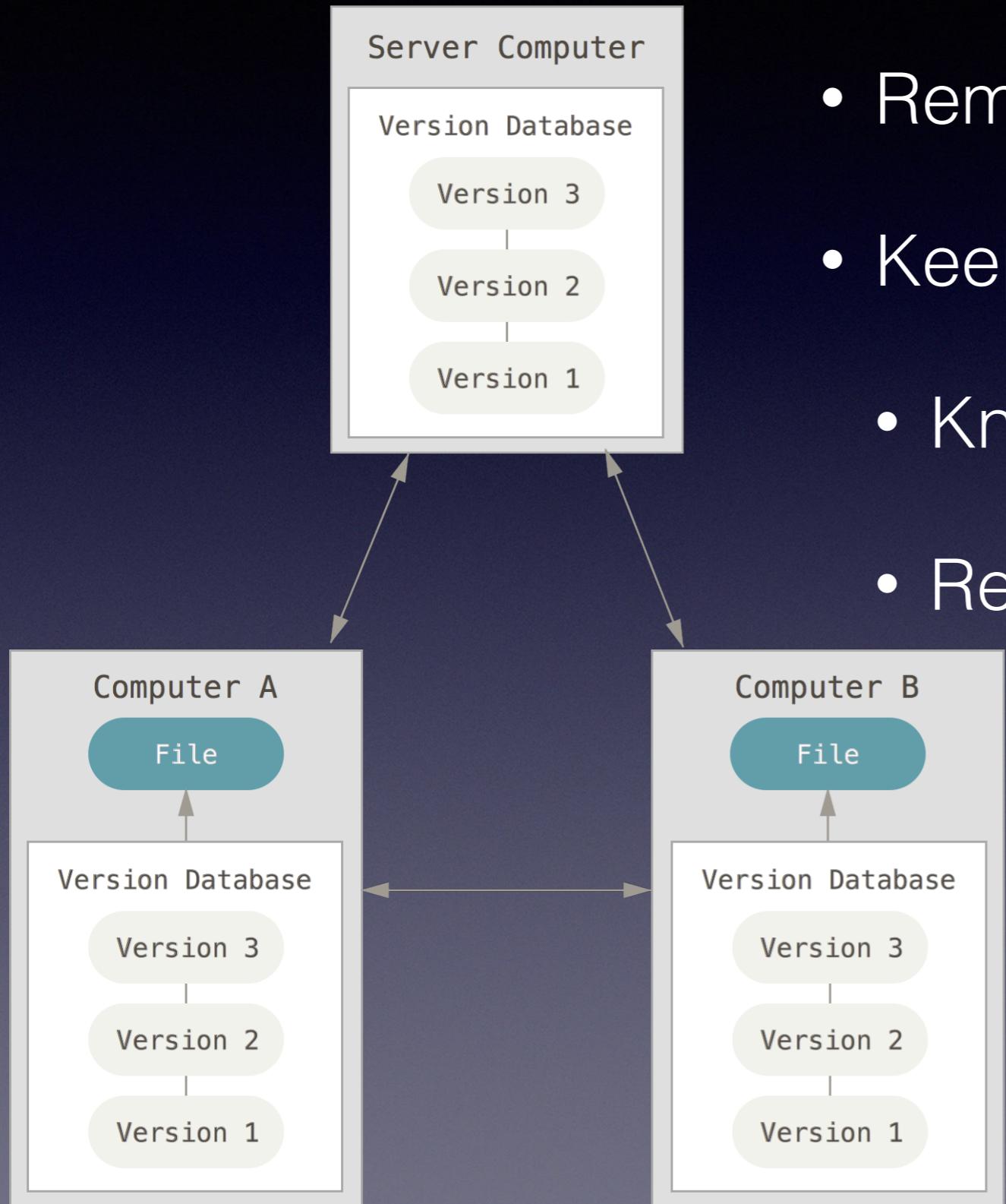


- Remote and local copies
- Keep track of file changes
  - Knows what you did last summer
  - Remembers authorship



# GitHub

# Git: Distributed Version Control System

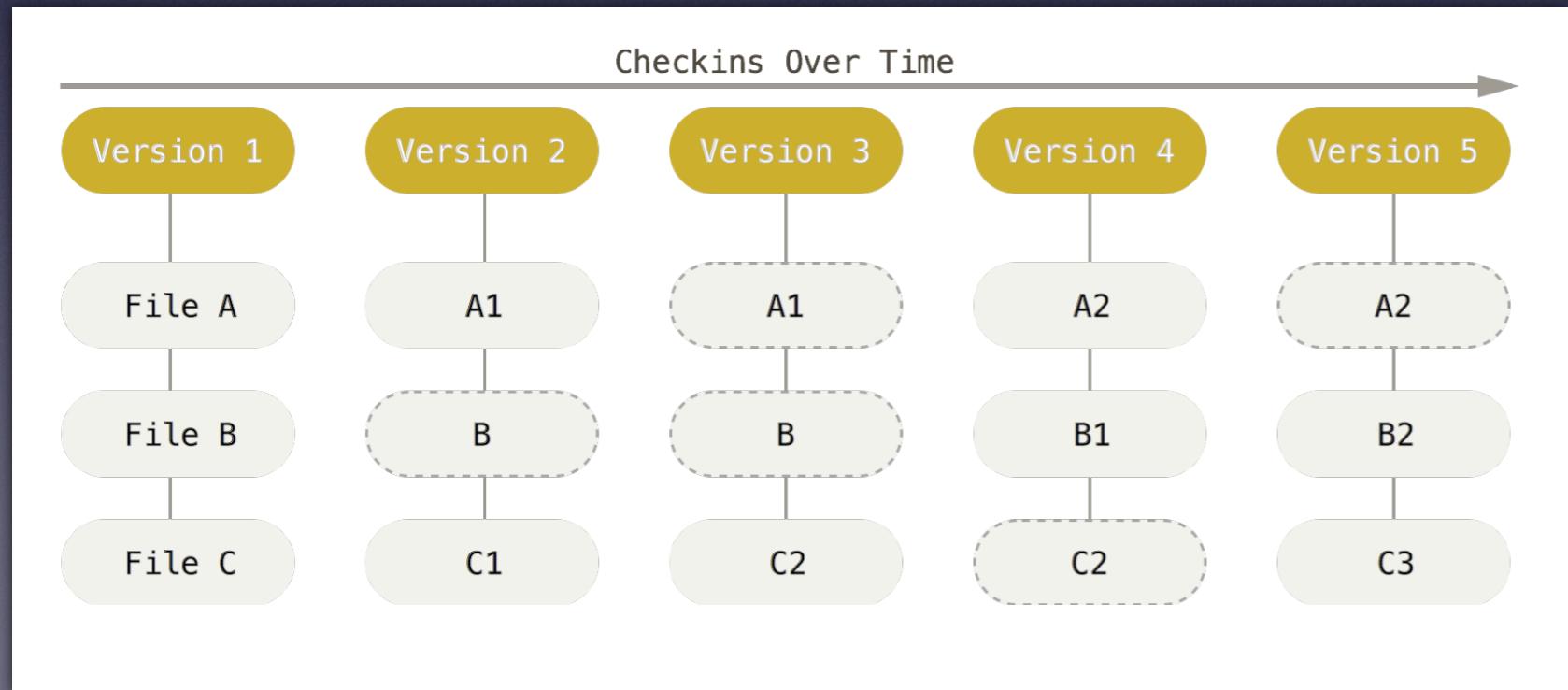


- Remote and local copies
- Keep track of file changes
- Knows what you did last summer
- Remembers authorship



# Git: Distributed Version Control System

- Remote and local copies
- Keep track of file changes
- Knows what you did last summer
- Remembers authorship



# Github & Git

- Set up an account
- Create a repository (repo)
- Add things to your repo
- Branch workflow
- Share changes with your team



# Git Basics



- Create Github account
- Set up Git ([link Github](#))

Linux & OS X

- 1 Download and install the latest version of Git.
- 2 On your computer, open the **Terminal** application.
- 3 Tell Git your *name* so your commits will be properly labeled. Type everything after the `$` here:

```
$ git config --global user.name "YOUR NAME"
```

- 4 Tell Git the *email address* that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](#). To keep your email address hidden, see "[Keeping your email address private](#)".

```
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

```
$ git config --global push.default simple  
$ git config --global core.editor "subl -n -w"
```

[link config editors](#)

- Create Github account
- Set up Git ([link Github](#)) Windows

**1** Download and install the latest version of [GitHub Desktop](#). This will automatically install Git and keep it up-to-date for you.

**2** On your computer, open the **Git Shell** application.

**3** Tell Git your *name* so your commits will be properly labeled. Type everything after the `$` here:

```
$ git config --global user.name "YOUR NAME"
```

**4** Tell Git the *email address* that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](#). To keep your email address hidden, see "[Keeping your email address private](#)".

```
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

```
$ git config --global push.default simple  
$ git config --global core.editor "subl -n -w"
```

[link config editors](#)

# Initial Steps:

Create a repository (repo)

REMOTE

Add collaborators to repo

REMOTE

Clone repo on local machine

LOCAL

```
$ git clone <address-repo>
```

REMOTE

= the remote server, Github

LOCAL

= your local machine

# Initial Steps:

Create a repository (repo)

REMOTE

Add collaborators to repo

REMOTE

Clone repo on local machine

LOCAL

```
$ git clone https://github.com/kikofernandez/git-intro.git
```

REMOTE

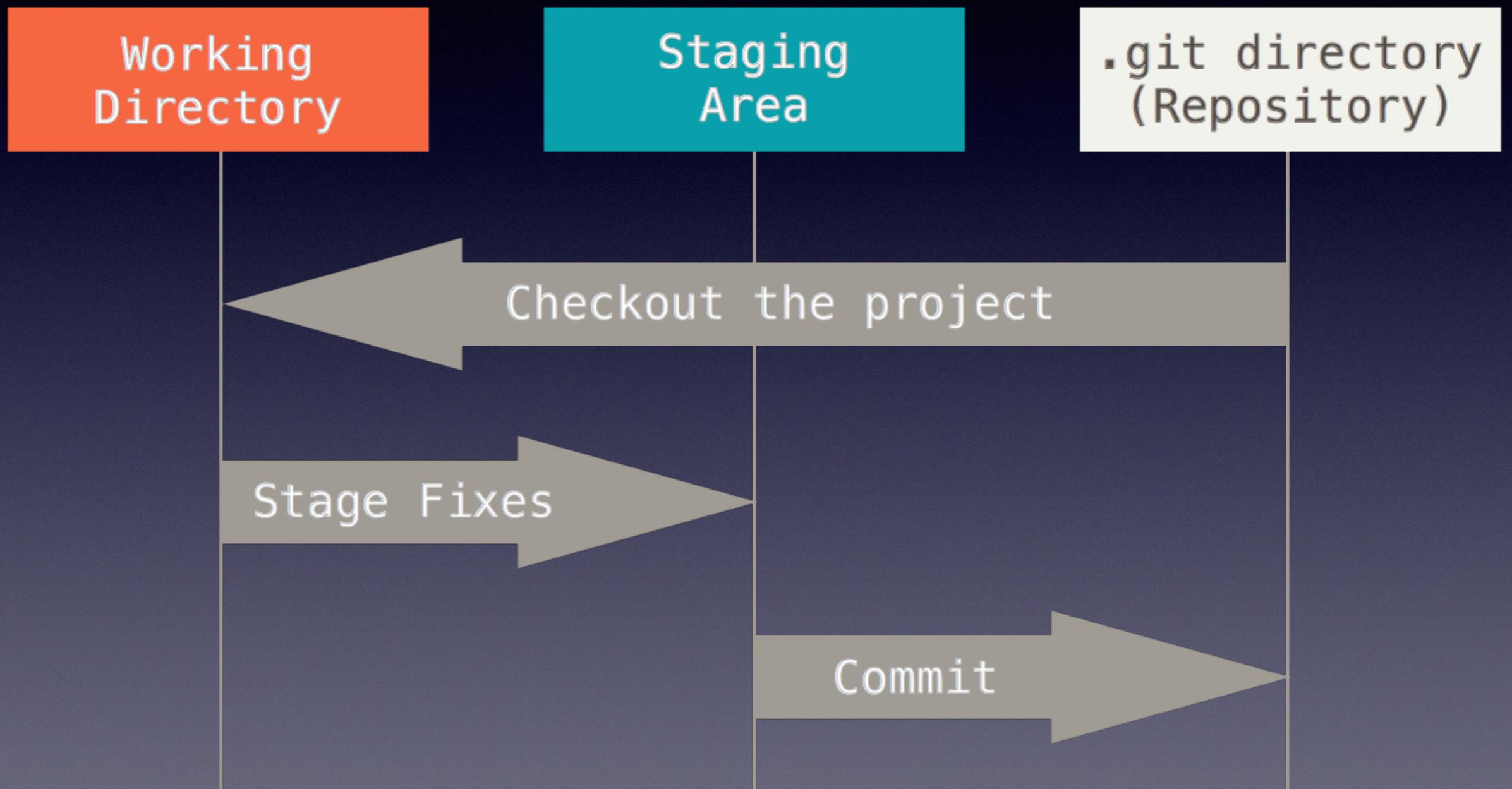
= the remote server, Github

LOCAL

= your local machine

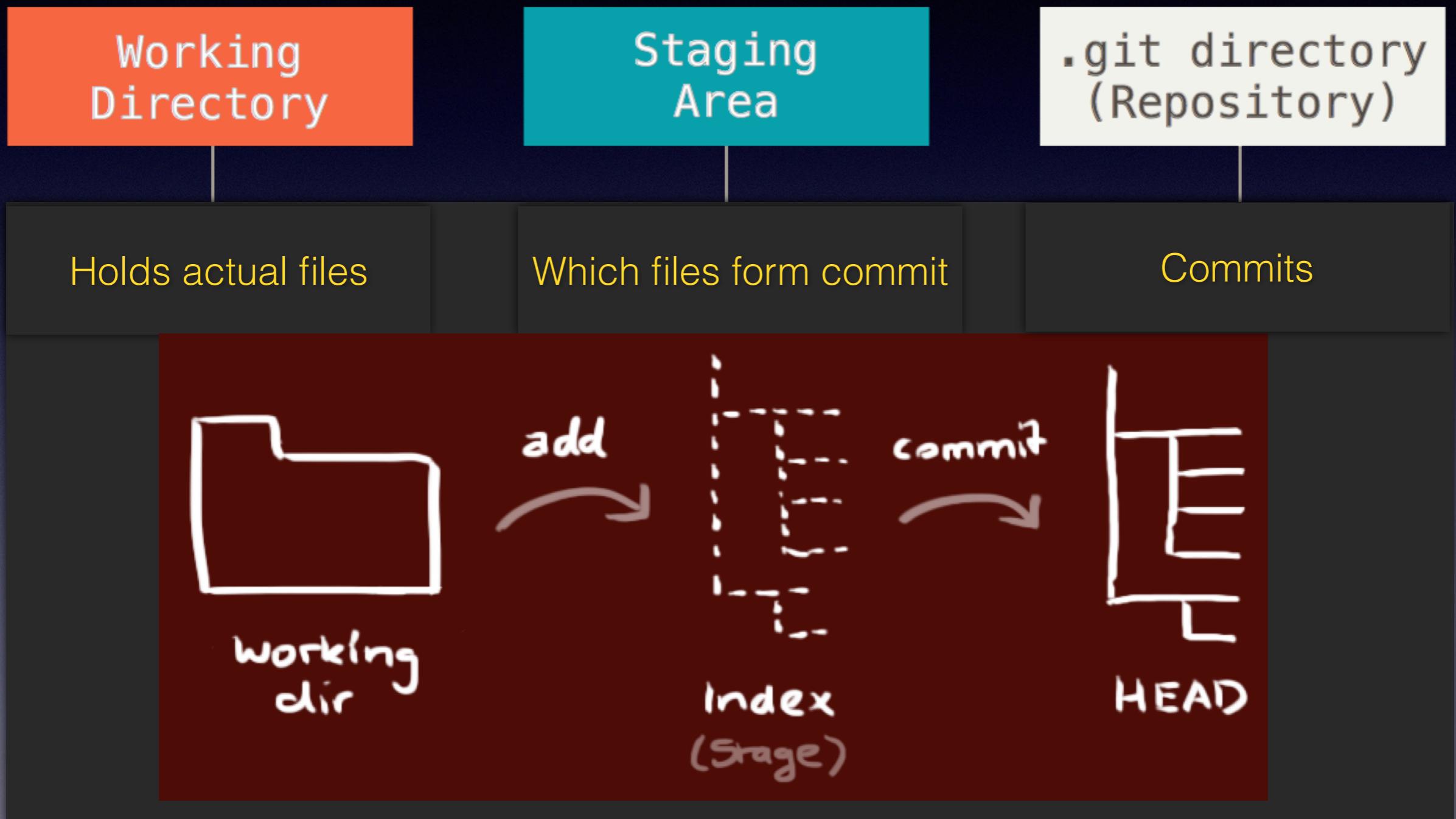
# Git: Basics

LOCAL



# Git: Basics

LOCAL



# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add <file>
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "message"
```

# Git: Basics

LOCAL

Working  
Directory

intro.md

this is an  
introduction  
to the  
course ...

Staging  
Area

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add <file>
```

Changes that make up a commit

Commands:

```
$ git commit -m "message"
```

# Git: Basics

LOCAL

Working  
Directory

intro.md

this is an  
introduction  
to the  
course ...

Staging  
Area

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Changes that make up a commit

Commands:

```
$ git commit -m "message"
```

# Git: Basics

LOCAL

Working  
Directory

Staging  
Area

this is an  
introduction  
to the  
course ...

intro.md

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Changes that make up a commit

Commands:

```
$ git commit -m "message"
```

# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

this is an  
introduction  
to the  
course ...

intro.md

# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

# Git: Basics

LOCAL

Working  
Directory

this is an  
introduction  
to the  
course ...

intro.md

Staging  
Area

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

this is an  
introduction  
to the  
course ...

intro.md

# Git: Basics

LOCAL

Working  
Directory

this is an  
introduction  
to the  
course ...

intro.md

Staging  
Area

this is an  
introduction  
to the  
course ...

intro.md

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

this is an  
introduction  
to the  
course ...

intro.md

# Git: Basics

LOCAL

Working  
Directory

Folder of your repository, e.g. ~/git-intro

Commands:

```
$ git add intro.md
```

Staging  
Area

Changes that make up a commit

Commands:

```
$ git commit -m "add intro.md file"
```

# Git: Logs

How do I see the changes?

REMOTE

The screenshot shows a GitHub repository page for 'parapluu / encore'. The repository has 1,494 commits, 7 branches, 0 releases, and 12 contributors. A red box highlights the '1,494 commits' link. Below the summary, it says 'This branch is 11 commits ahead of master.' and lists recent commits by 'supercooldave' and others.

Commit	Message	Author	Date
#349	Merge pull request #340 from kikofernandez/fix/ghc-locale	supercooldave	Latest commit ecf1b19
.github	add github templates for PR and Issues		
applications/SATSolver	SAT solver with 10 example problems.		
benchmarks	cannot run benchmark unless it's linux		

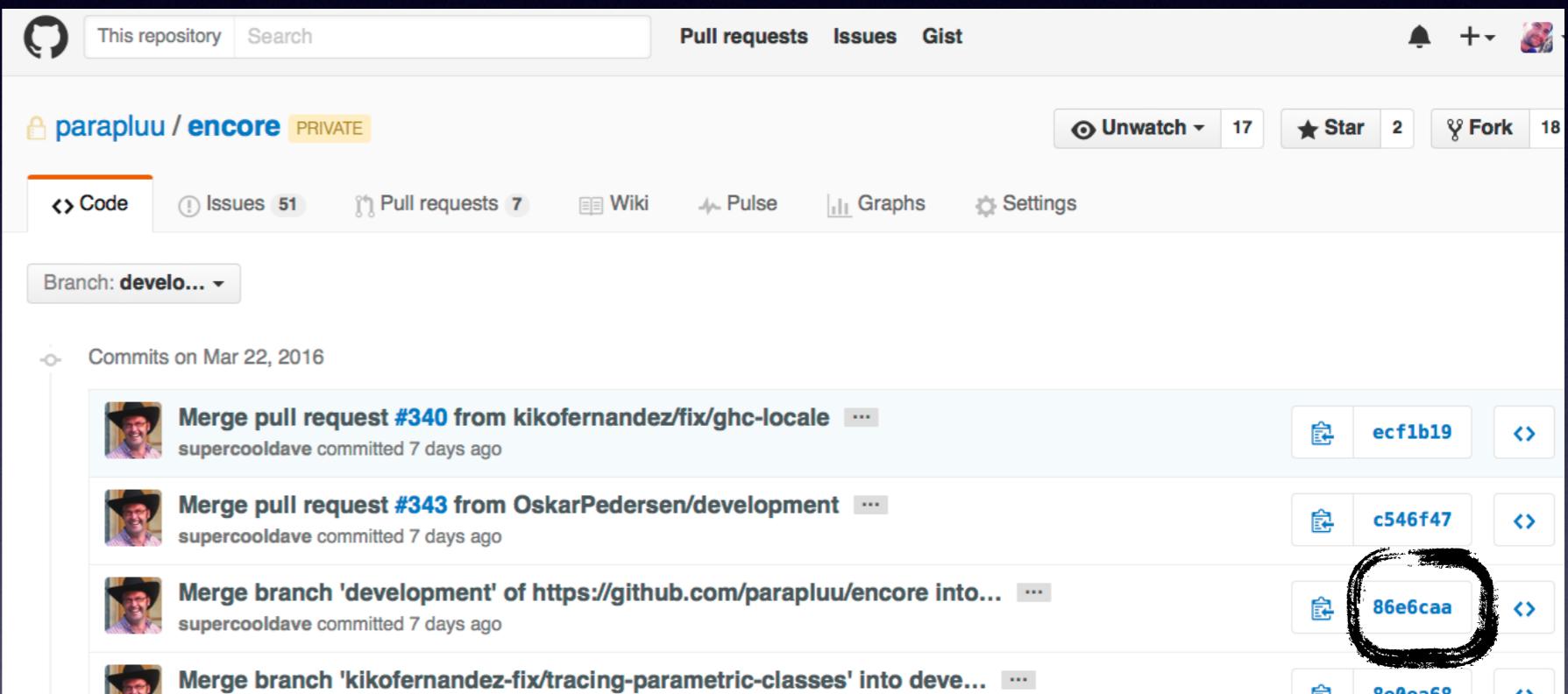
LOCAL

```
$ git log -p
```

# Git: Logs

How do I see the changes?

REMOTE



A screenshot of a GitHub repository page for 'parapluu / encore'. The page shows a list of recent commits on the 'development' branch. The commits are:

- Merge pull request #340 from kikofernandez/fix/ghc-locale ... (commit ecf1b19)
- Merge pull request #343 from OskarPedersen/development ... (commit c546f47)
- Merge branch 'development' of https://github.com/parapluu/encore into... (commit 86e6caa) 86e6caa
- Merge branch 'kikofernandez-fix/tracing-parametric-classes' into deve... (commit 8e0ea68)

LOCAL

```
$ git log -p
```

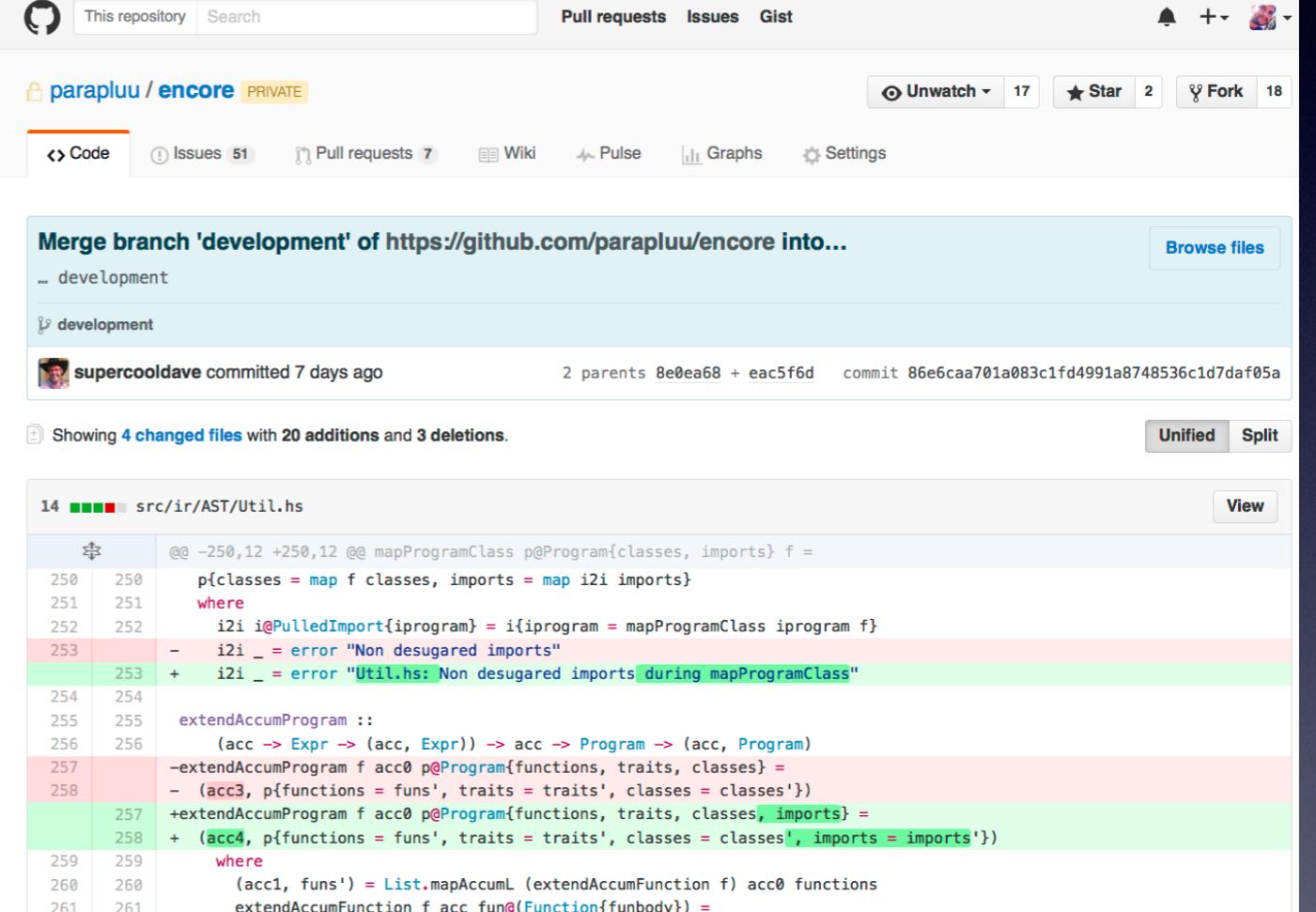
# Git: Logs

How do I see the changes?

REMOTE

LOCAL

\$ git log -p



The screenshot shows a GitHub repository page for 'parapluu / encore' (PRIVATE). The 'Code' tab is selected, displaying a merge commit from the 'development' branch. The commit message is 'Merge branch 'development' of https://github.com/parapluu/encore into...'. It was committed by 'supercooldave' 7 days ago. The commit has 2 parents: 8e0ea68 + eac5f6d, with a commit hash of 86e6caa701a083c1fd4991a8748536c1d7daf05a. The diff view shows 4 changed files with 20 additions and 3 deletions. A specific file, 'src/ir/AST/Util.hs', is shown with a unified diff. The diff highlights changes in lines 253 and 258. Line 253 shows a removal of 'i2i \_ = error "Non desugared imports"' and an addition of 'i2i \_ = error "Util.hs: Non desugared imports during mapProgramClass"'. Line 258 shows a removal of 'acc0 p@Program{functions, traits, classes}' and an addition of '(acc4, p{functions = funs', traits = traits', classes = classes', imports = imports'})'. The code block also includes lines 250-261.

```
14 src/ir/AST/Util.hs
@@ -250,12 +250,12 @@ mapProgramClass p@Program{classes, imports} f =
 250   p{classes = map f classes, imports = map i2i imports}
 251   where
 252     i2i i@PulledImport{iprogram} = i{iprogram = mapProgramClass iprogram f}
 253     - i2i _ = error "Non desugared imports"
 254     + i2i _ = error "Util.hs: Non desugared imports during mapProgramClass"
 255
 256 extendAccumProgram :: (acc -> Expr -> (acc, Expr)) -> acc -> Program -> (acc, Program)
 257 -extendAccumProgram f acc0 p@Program{functions, traits, classes} =
 258   - (acc3, p{functions = funs', traits = traits', classes = classes'})
 259   +extendAccumProgram f acc0 p@Program{functions, traits, classes, imports} =
 260   + (acc4, p{functions = funs', traits = traits', classes = classes', imports = imports'})
 261   where
 262     (acc1, funs') = List.mapAccumL (extendAccumFunction f) acc0 functions
 263     extendAccumFunction f acc fun@(Function{funbody}) =
```

# Git: Logs

How do I see the changes?

REMOTE

LOCAL

\$ git log -p

```
commit 810c19b5d46f94d4d2eac1f0dd361daf9235c7e8
Author: Kiko Fernandez Reyes <kiko.fernandez@it.uu.se>
Date:   Thu Mar 24 08:12:37 2016 +0100

    working on passing parametric function type to class

diff --git a/src/back/CodeGen/Context.hs b/src/back/CodeGen/Context.hs
index 3459177..4de8542 100644
--- a/src/back/CodeGen/Context.hs
+++ b/src/back/CodeGen/Context.hs
@@ -28,6 +28,8 @@ import qualified CodeGen.ClassTable as Tbl

     import qualified CCode.Main as C

+data LexicalContext = ActiveObject | PassiveObject | SharedObject | GlobalFunction
+
     type NextSym = Int

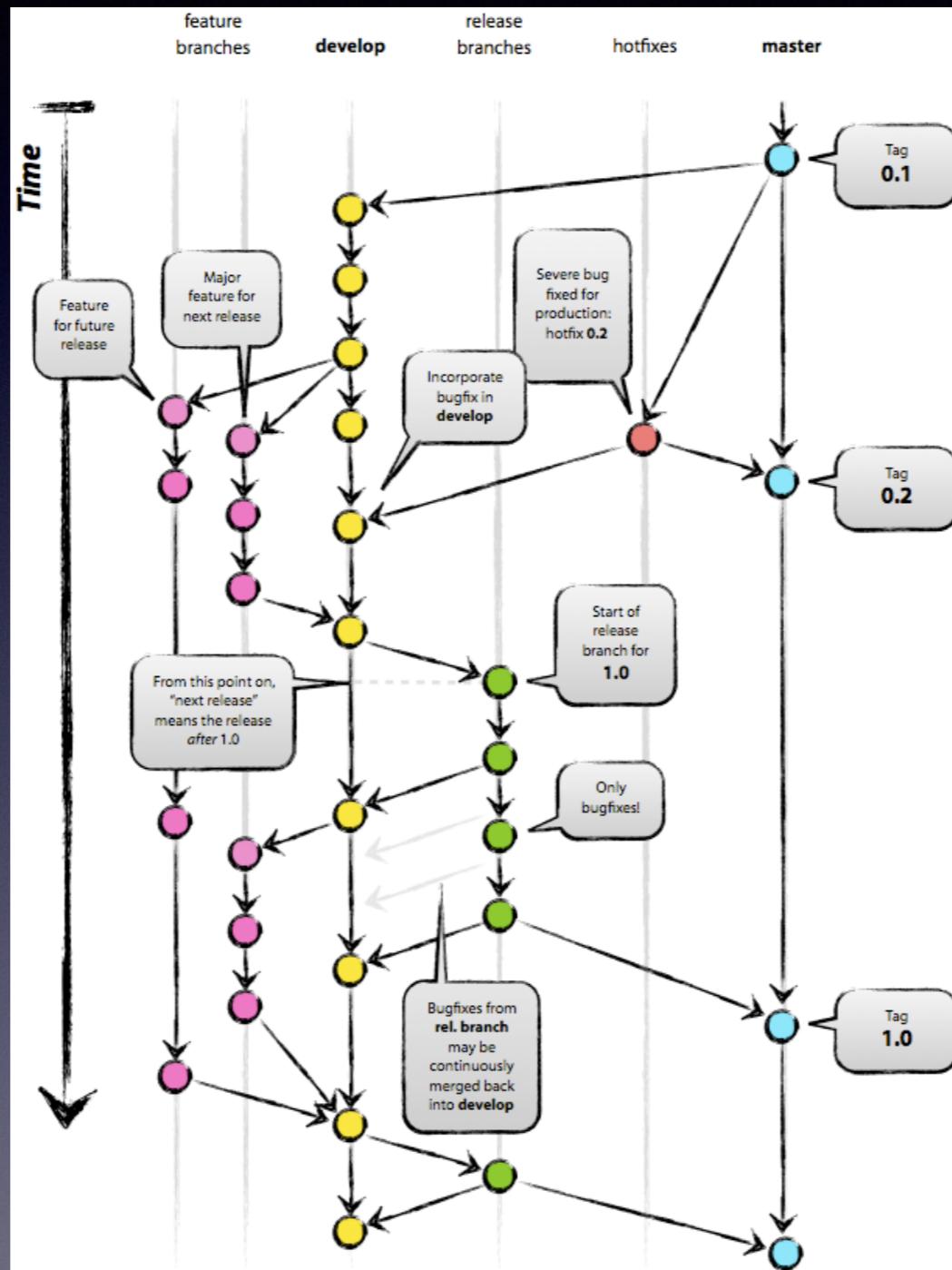
     type VarSubTable = [(Name, C.CCode C.LVal)] -- variable substitutions (for supporting, for instance, nested var declarations)
diff --git a/src/back/CodeGen/Expr.hs b/src/back/CodeGen/Expr.hs
index dd13f44..8194417 100644
--- a/src/back/CodeGen/Expr.hs
+++ b/src/back/CodeGen/Expr.hs
@@ -1040,8 +1040,8 @@ globalFunctionCall fcall@A.FunctionCall{A.name, A.args} = do
     returnType = A.hType formalArgs

     arguments = map translateArg (zip3 (map AsExpr args') formalTypes actualArgs)
-    bindings = zip (A.hParams formalArgs) (A.hParams fcall)
     typeVariables = map translateParametricArguments (A.hParams fcall)
+
     rhs = Call (globalFunctionName name) (typeVariables ++ arguments)
     (callVar, call) <- namedTmpVar "global_f" typ $
         translateReturnType rhs returnType (translate typ)
diff --git a/src/tests/encore/basic/parametricFunctions.enc b/src/tests/encore/basic/parametricFunctions.enc
index 0303053..db00c1a 100644
--- a/src/tests/encore/basic/parametricFunctions.enc
+++ b/src/tests/encore/basic/parametricFunctions.enc
@@ -5,7 +5,8 @@ def <a> testCallToParametricFunction(x: int, y: a): void {
     testCallToParametricFunction<a>(x, y); -- correct trace
     testCreationOfObject<int>(23); -- correct trace
     testCreationOfObject<String>"23"; -- correct trace
-
+ new Test<String>(x, "sd");
+ new Test<a>(x, y);
1
```

Best tip you will ever receive:

Commit early, commit often  
*(A tip for version controlling - not for relationships)*

# Git branching (optional)



# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git add intro.md; git commit -m "update"
```



# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git add intro.md; git commit -m "update"
```



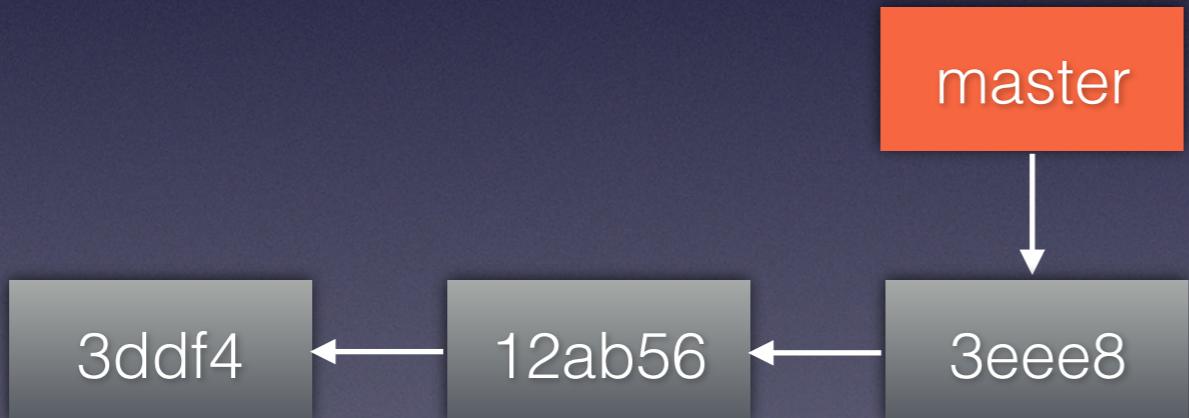
# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git add intro.md; git commit -m "update"
```



# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master



# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git branch frontend
```



# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master



**\$ git branch frontend**

HEAD: pointer to the current branch

# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master



**\$ git checkout frontend**

HEAD: pointer to the current branch

# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git checkout frontend
```



HEAD: pointer to the current branch

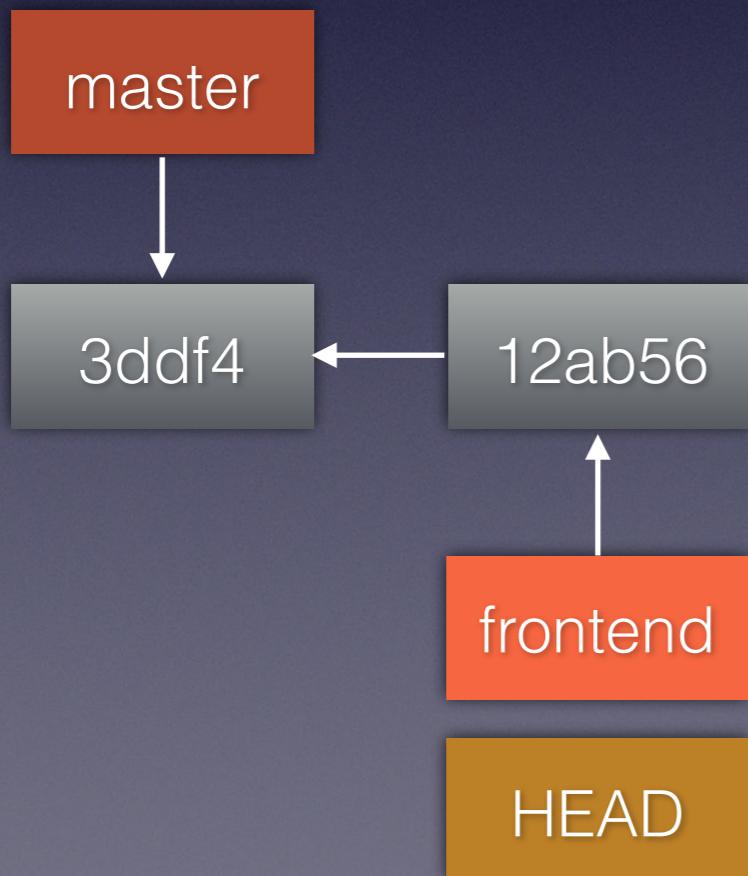
# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git commit -a -m "commit everything"
```



HEAD: pointer to the current branch

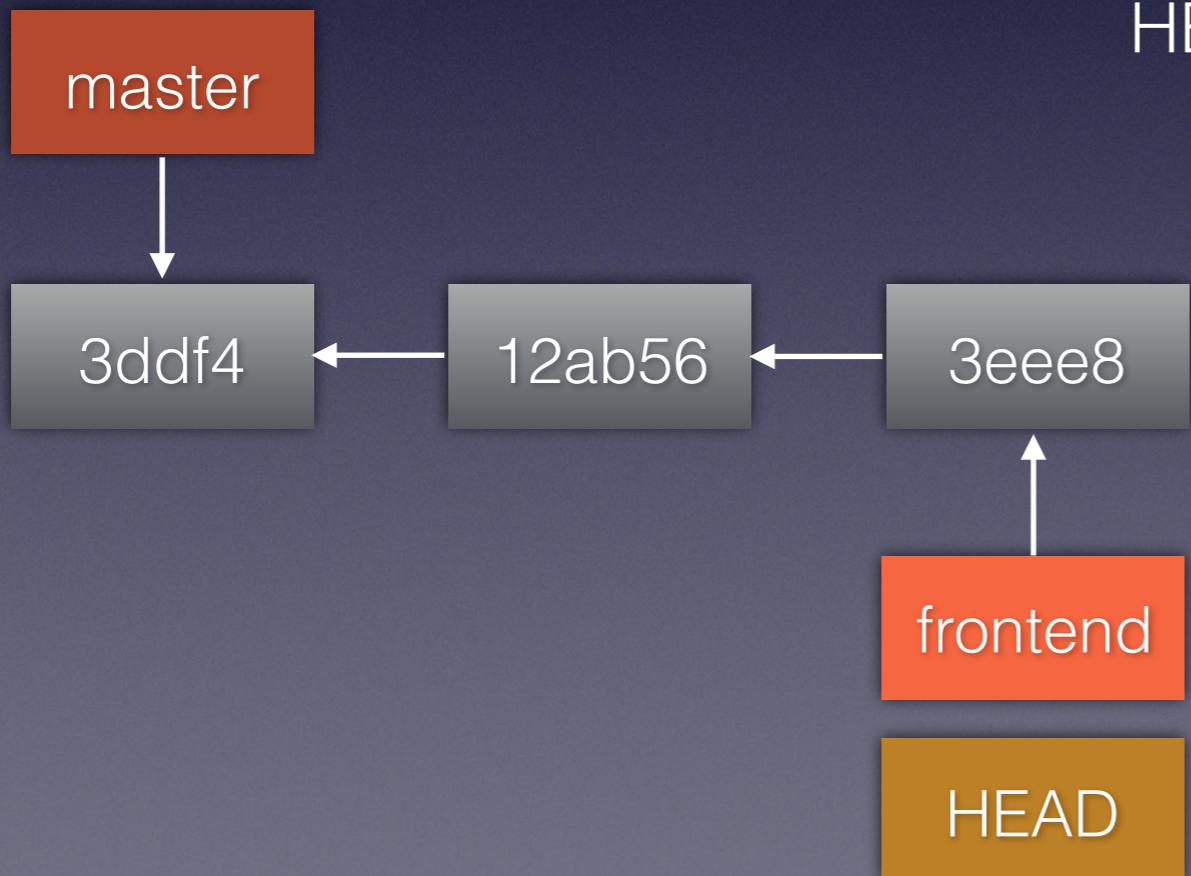
# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git commit -a -m "commit everything"
```

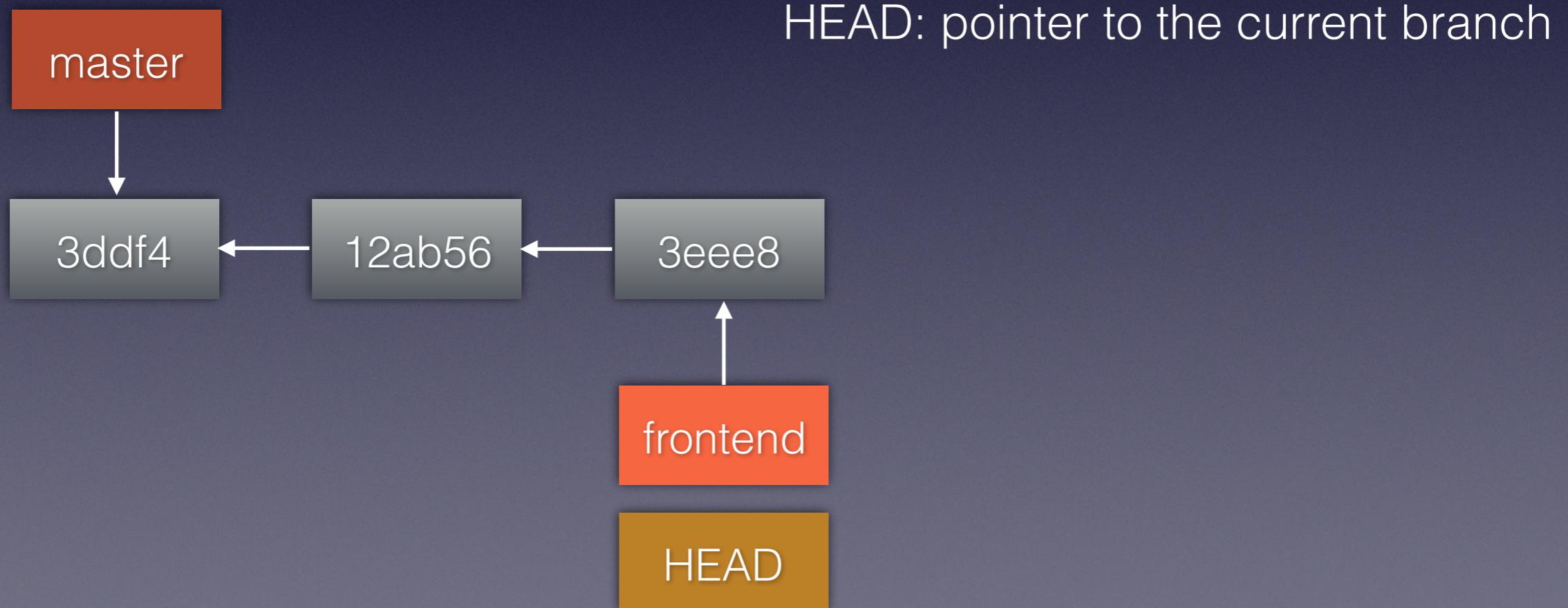


# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master



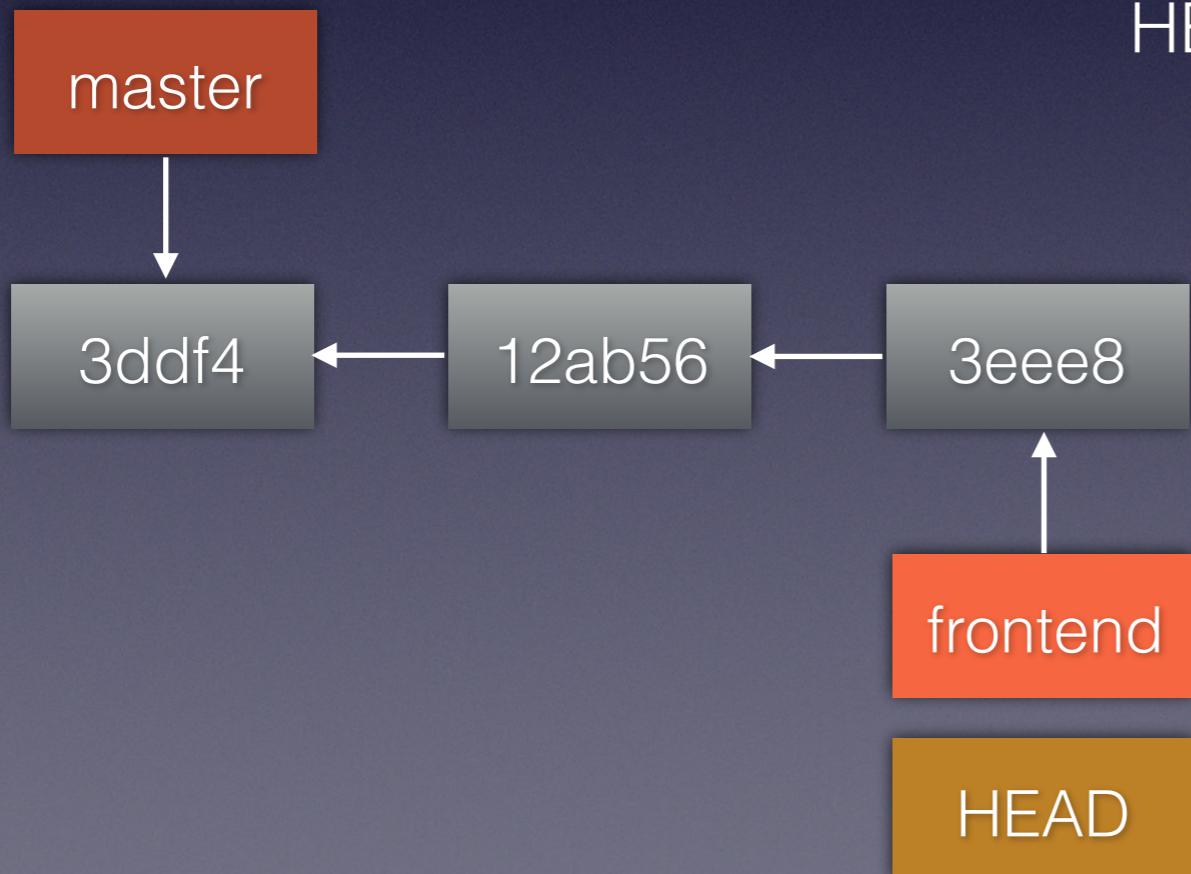
# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git checkout master
```



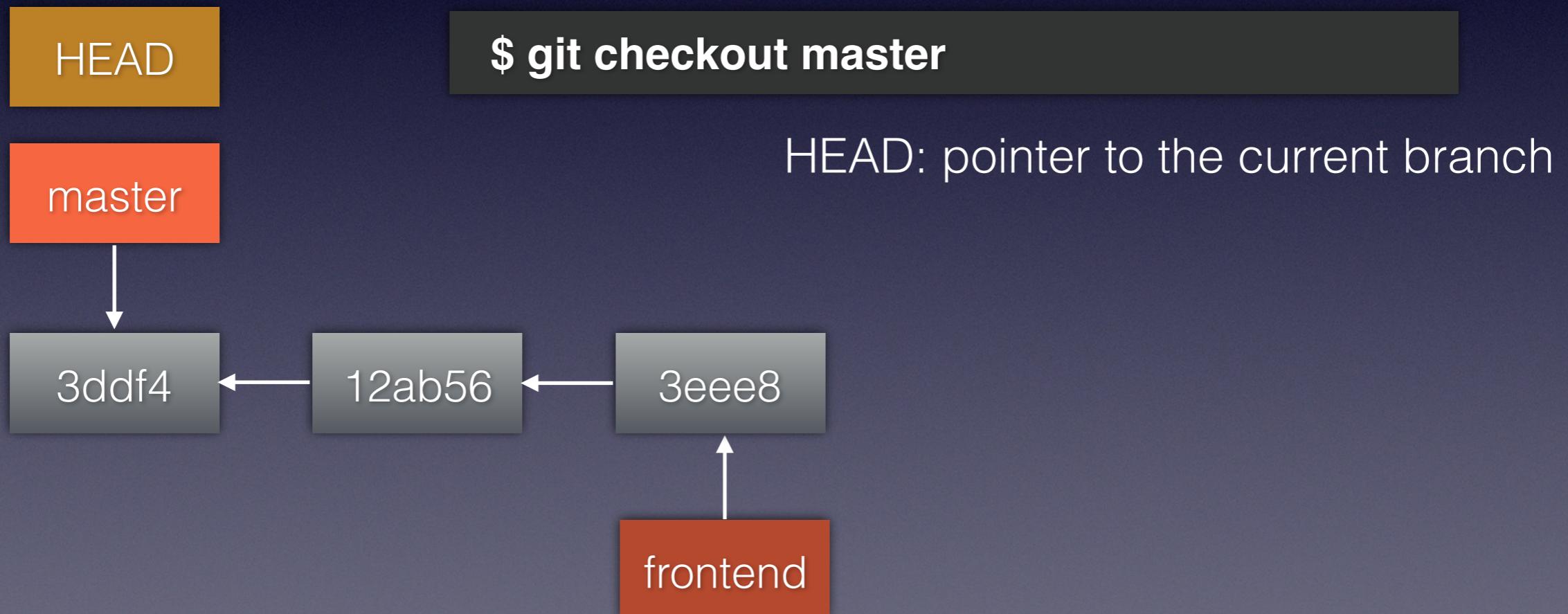
HEAD: pointer to the current branch

# Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master

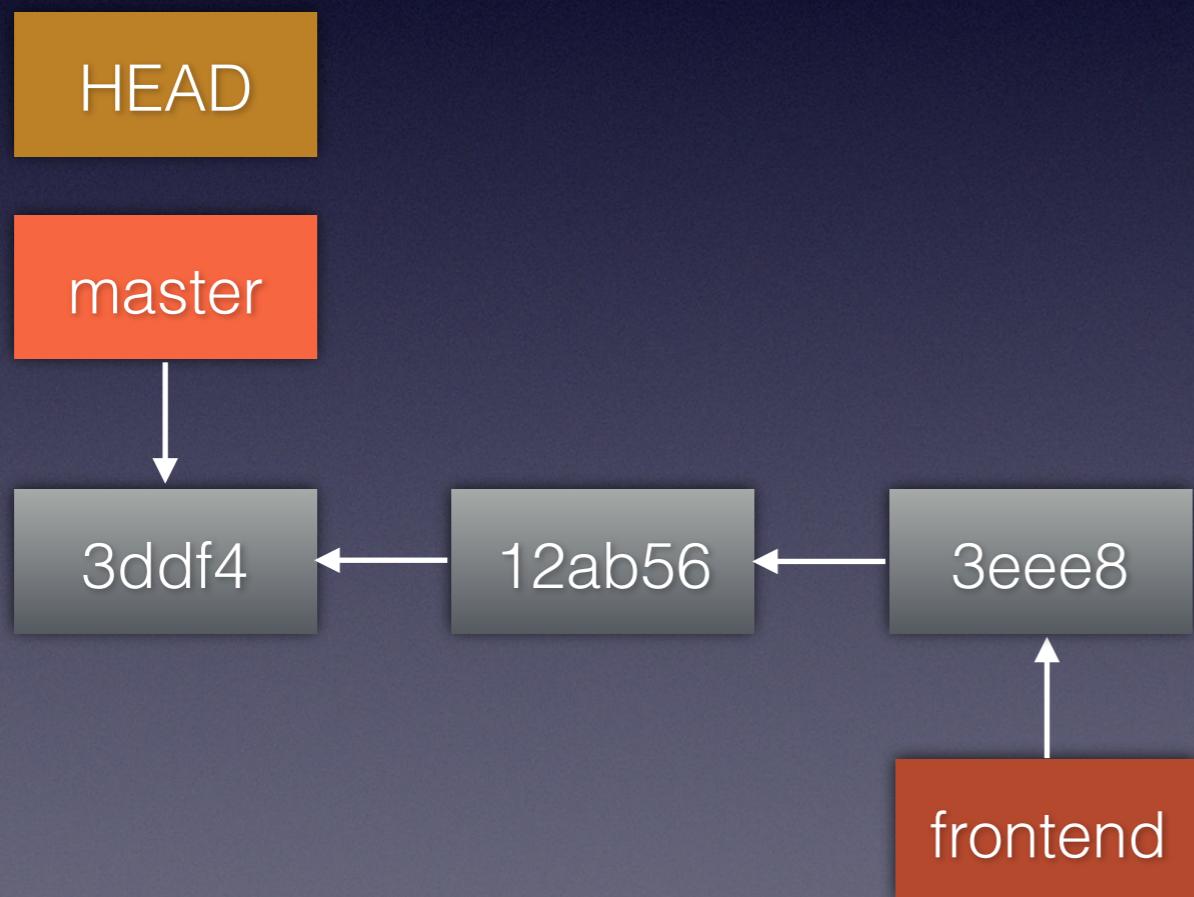


# Git: Branching

LOCAL

How to merge master with the changes in frontend?

```
$ git merge frontend
```



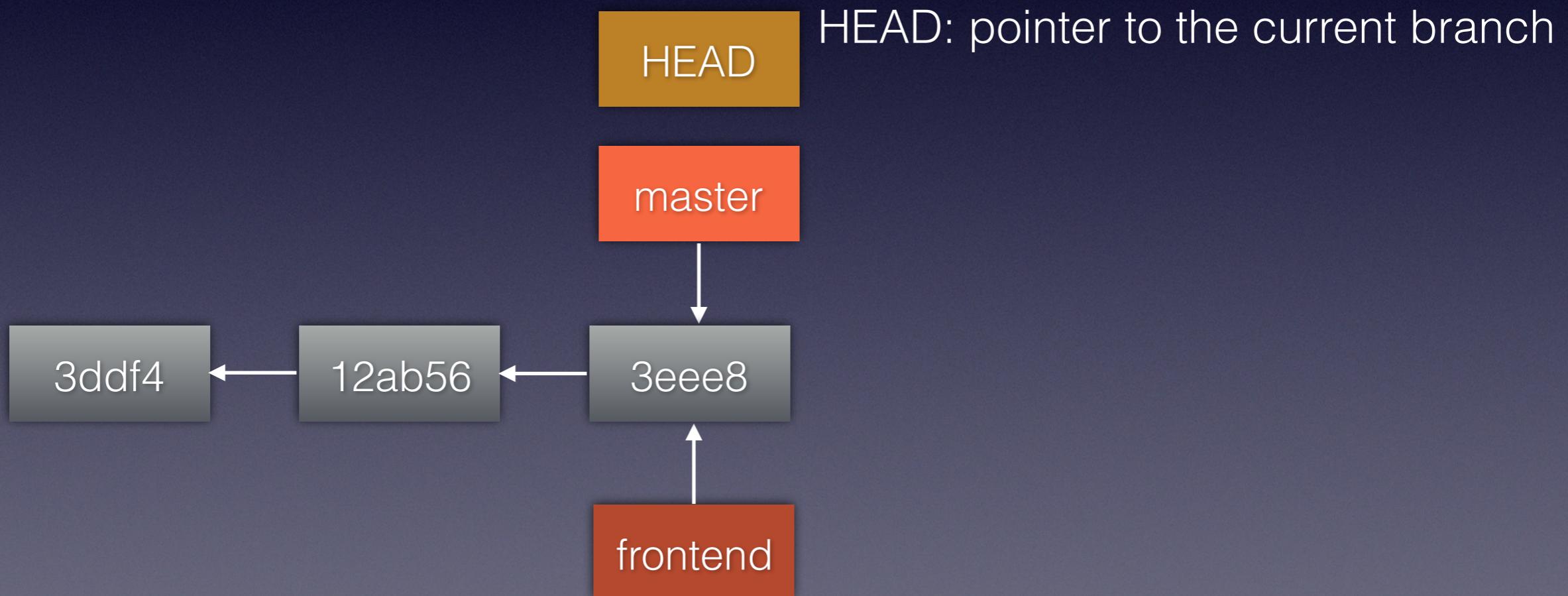
HEAD: pointer to the current branch

# Git: Branching

LOCAL

How to merge master with the changes in frontend?

```
$ git merge frontend
```



# Git Conflicts

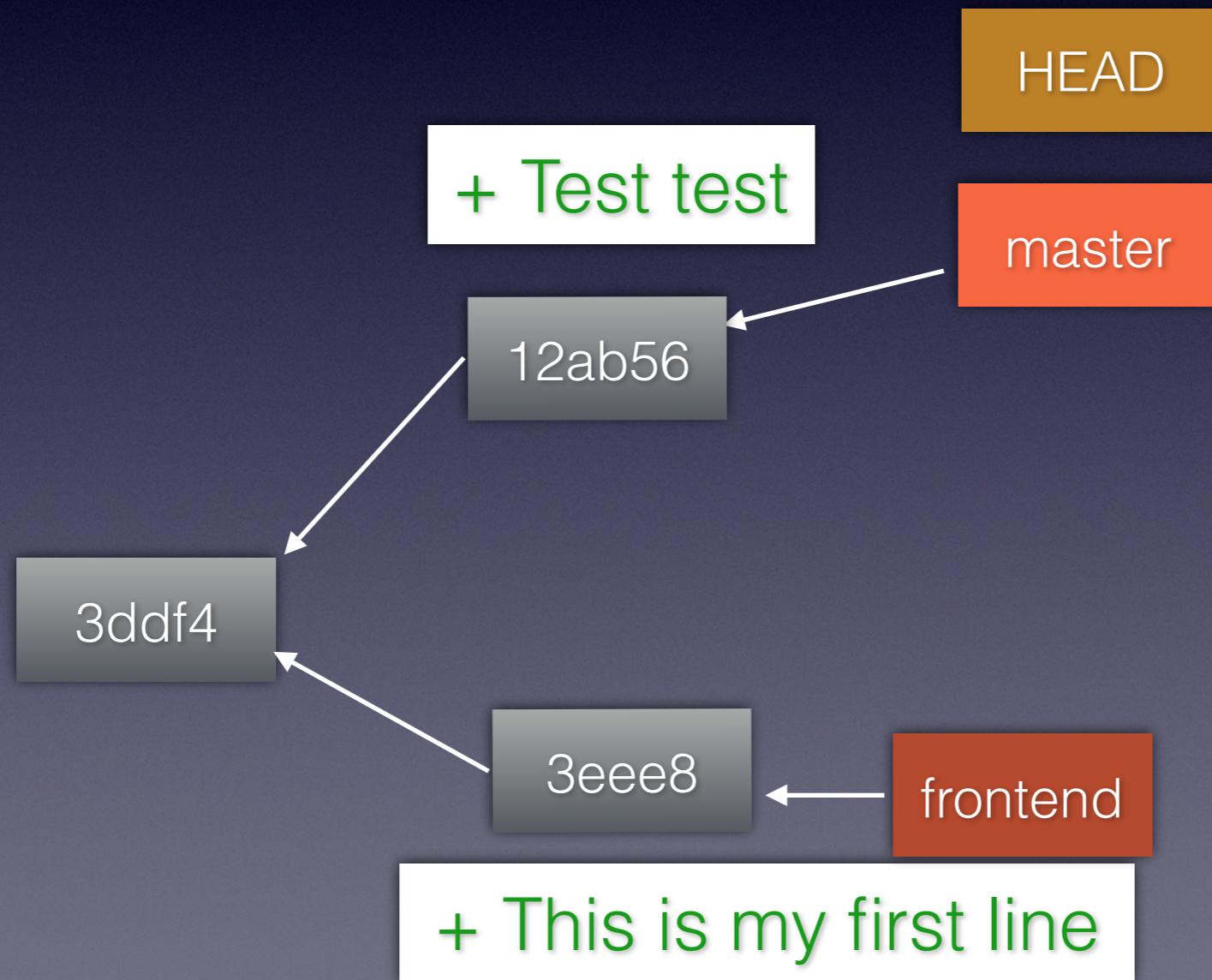


# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

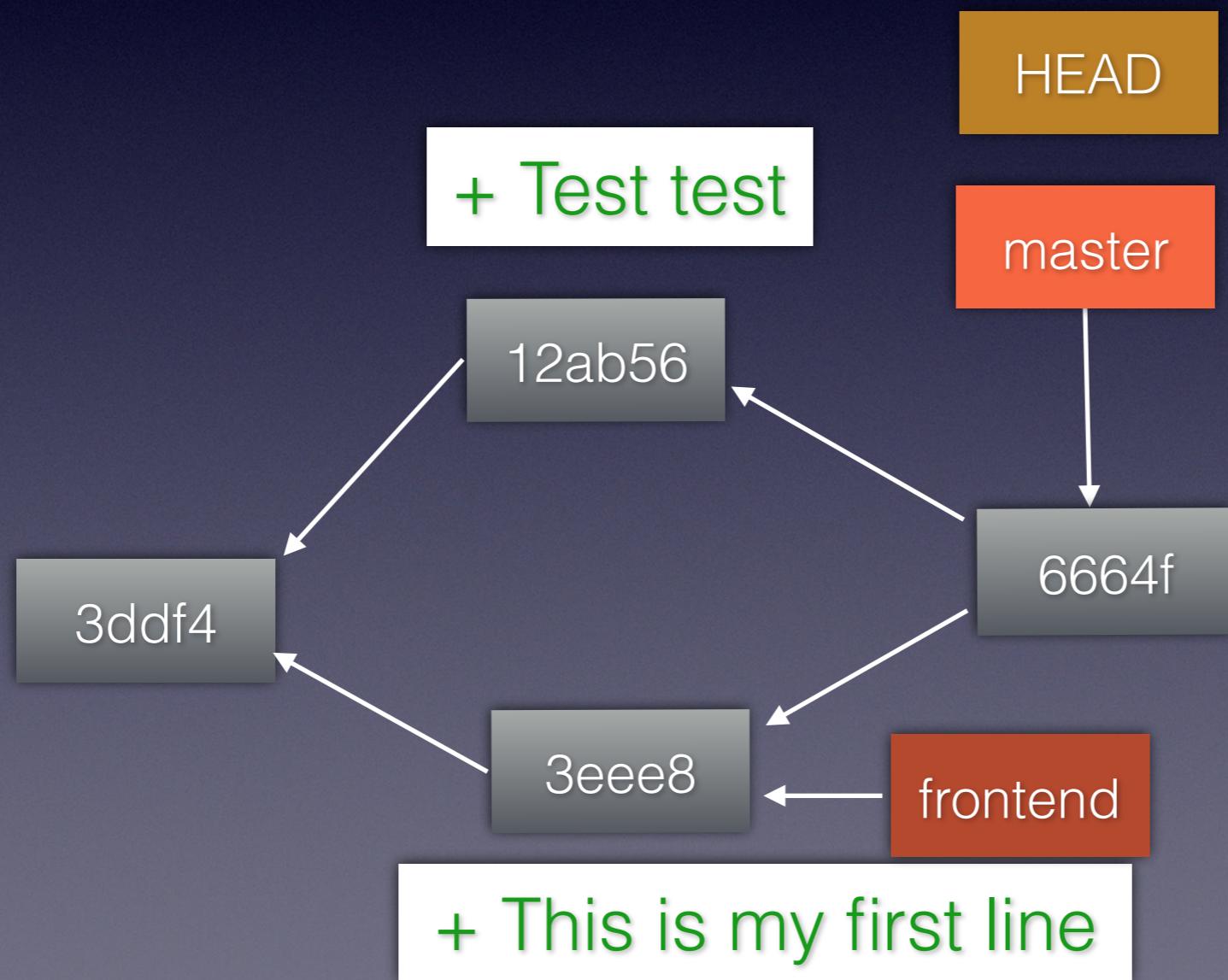


# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```



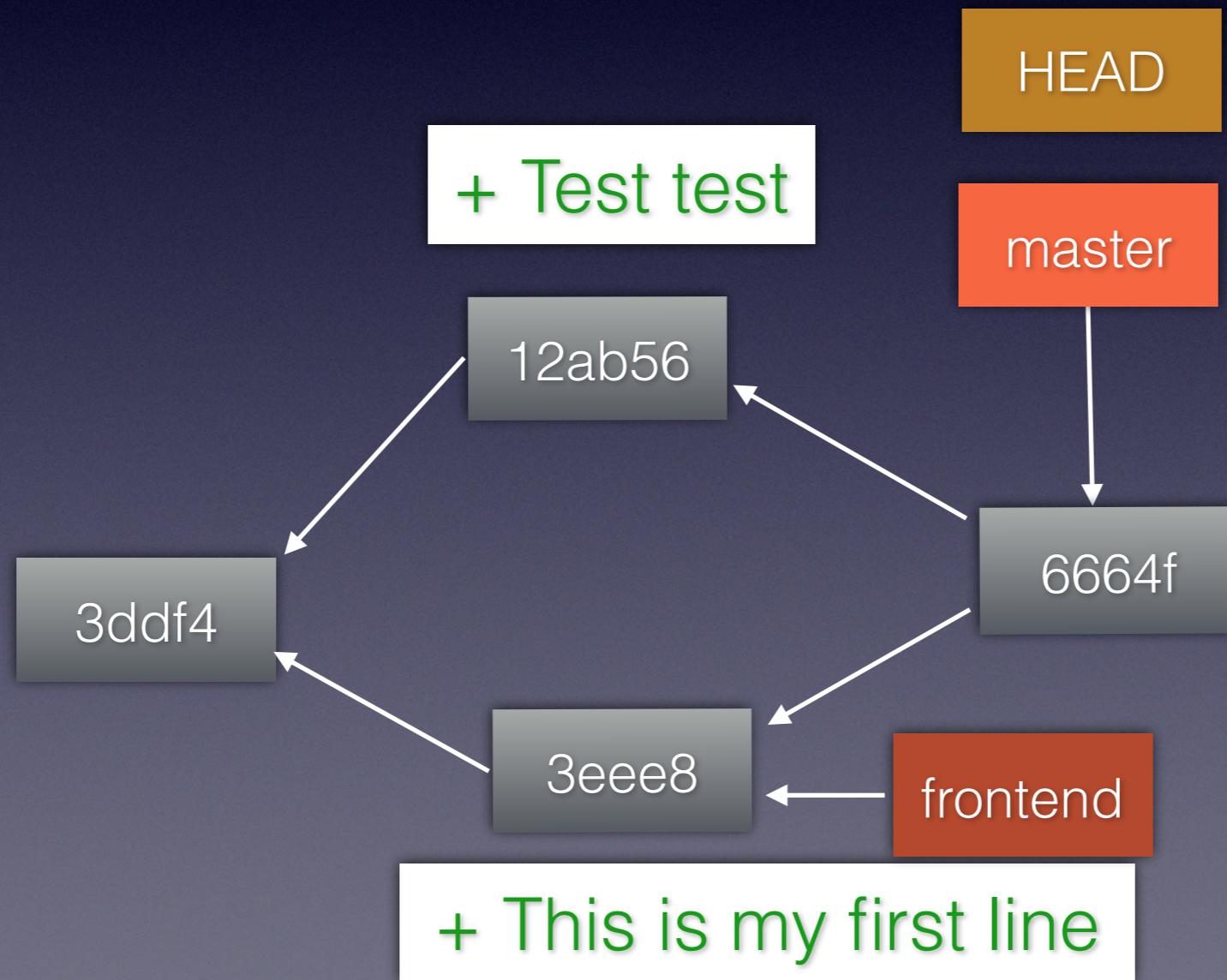
# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

Conflicts: 12ab56 and 3eee8 update same line



# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

Conflicts: 12ab56 and 3eee8 update same line

```
→ intro-git git:(master) ✘ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
You have unmerged paths.
  (fix conflicts and run "git commit")
Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:   example.md

no changes added to commit (use "git add" and/or "git commit -a")
→ intro-git git:(master) ✘
+ THIS IS my first line
```

# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

Conflicts: 12ab56 and 3eee8 update same line

HEAD

```
→ intro-git git:(master) ✘ cat example.md
<<<<< HEAD
Test test
=====
This is my first change
>>>>> first-line
→ intro-git git:(master) ✘
```

+ This is my first line

# Git: Conflicts

# LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

## Conflicts: 12ab56 and 3eee8 update same line

LIFAD

```
→ intro-git git:(master) ✘ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
You have unmerged paths.
  (fix conflicts and run "git commit")
Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified: example.md
no changes added to commit (use "git add" and/or "git commit -a")
→ intro-git git:(master) ✘
```

# Git: Conflicts

LOCAL

Commits: 12ab56 and 3eee8 add a line in same file

```
$ git merge frontend
```

Conflicts: 12ab56 and 3eee8 update same line

```
→ intro-git git:(master) ✘ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)
```

Changes to be committed:

```
modified:   example.md
```

```
→ intro-git git:(master) ✘ █
```

# Git conflicts: Recipe



When you get conflicts:

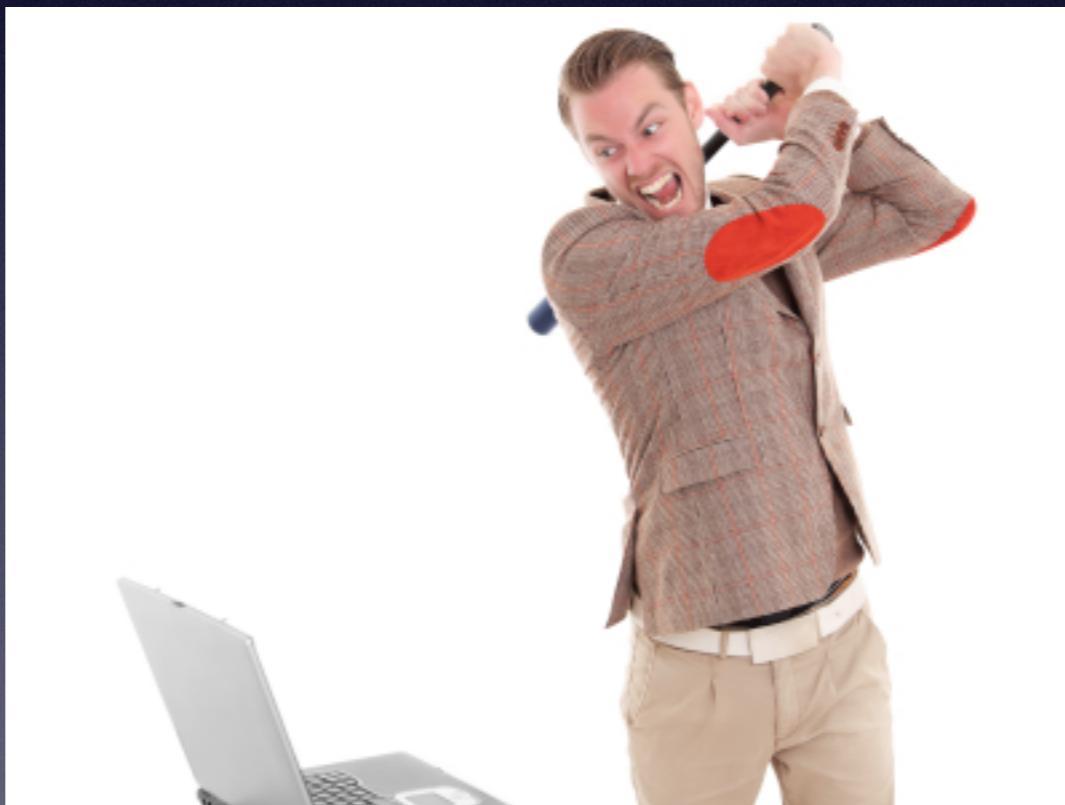
1. fix the conflicts
2. **\$ git add <name-of-files-with-conflicts>**
3. **\$ git commit**

# Team work



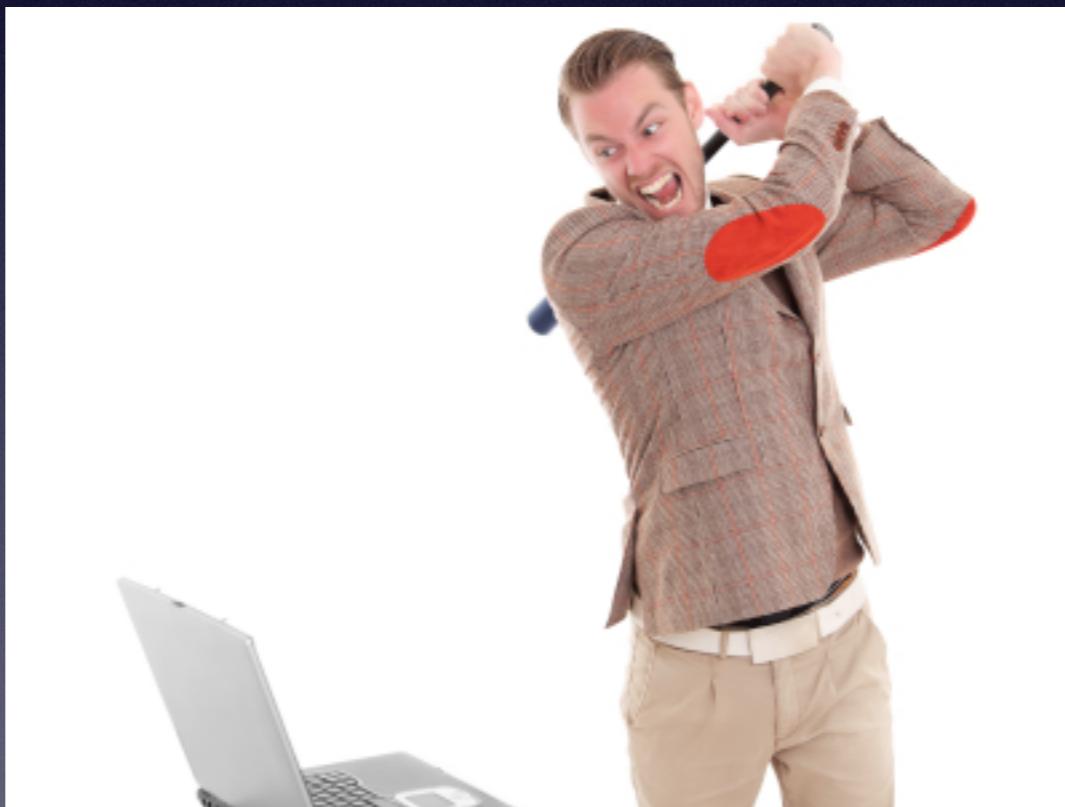
# Github ❤ Git

- So far, your changes are in local computer
- **Problem:** What if you drop the computer? Work in Teams?



# Github ❤ Git

- So far, your changes are in local computer
- **Problem:** What if you drop the computer? Work in Teams?



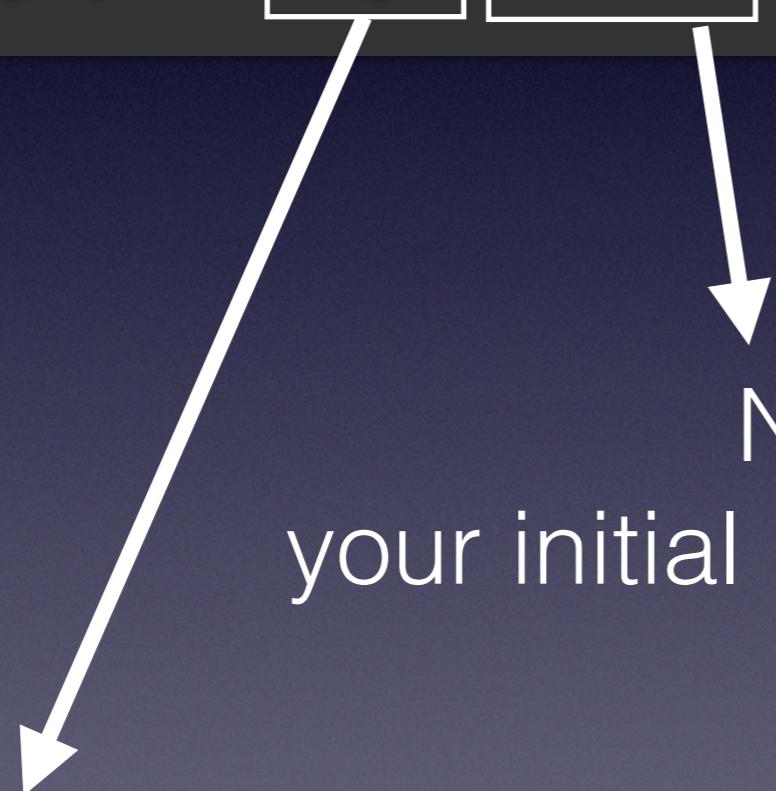
- **Solution:** **Github** saves your work in the “cloud”

# Github ❤ Git

REMOTE

- Push your **local** changes to your **remote** repository

```
$ git push origin master
```



Name of your branch:  
your initial branch is always called master

Name of the remote url:  
by default, it's always called origin

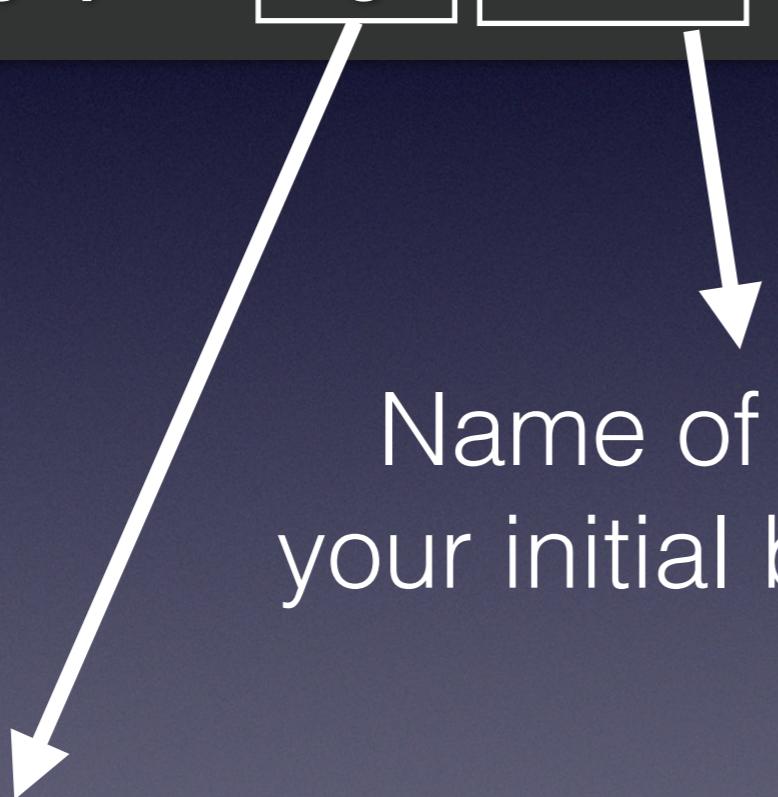
```
$ git remote -v
```

# Github ❤ Git

REMOTE

- Pull **remote** changes to your **local** repository

```
$ git pull origin master
```



Name of the **local** branch to pull into:  
your initial branch is always called master

Name of the remote url:  
by default, it's always called origin

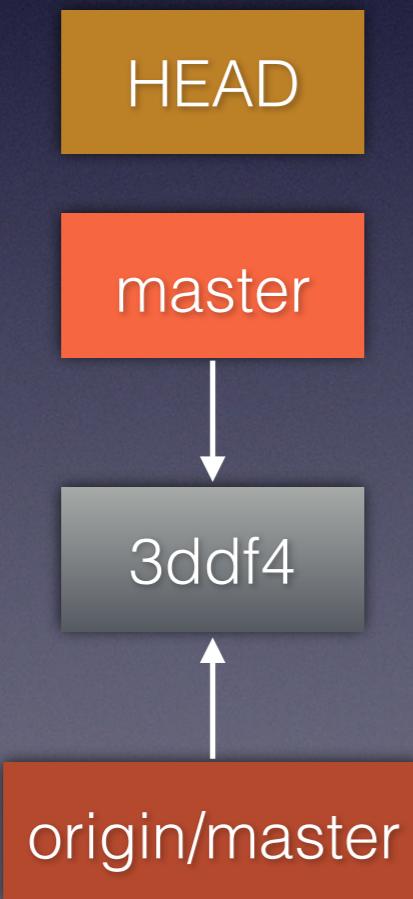
```
$ git remote -v
```

# Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git commit -am “commit everything”
```

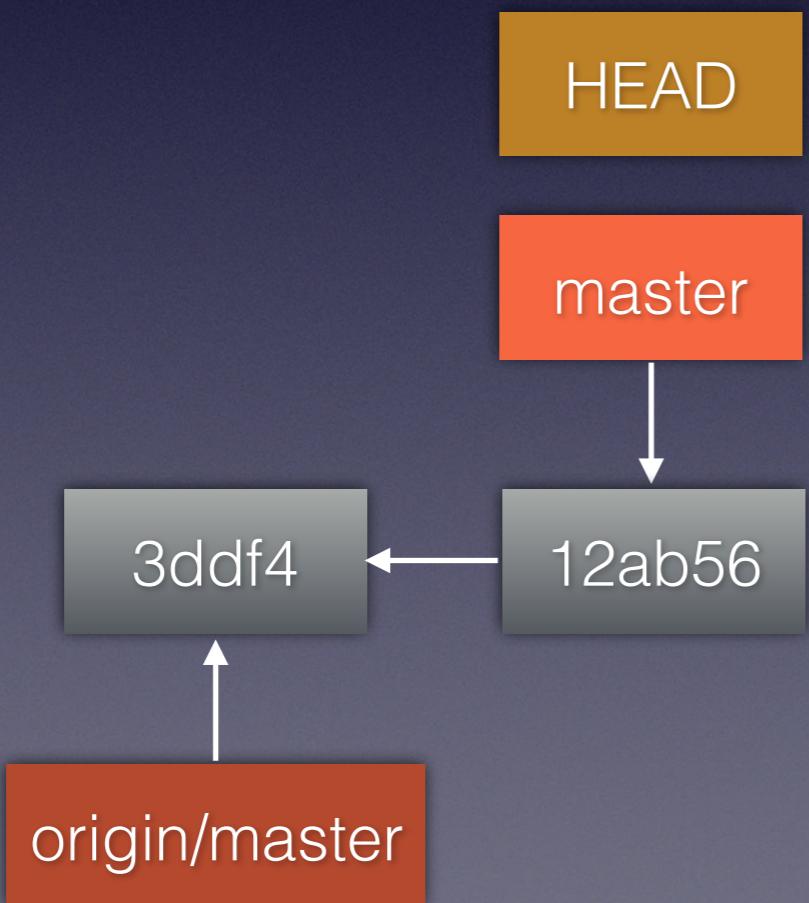


# Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git commit -am “commit everything”
```



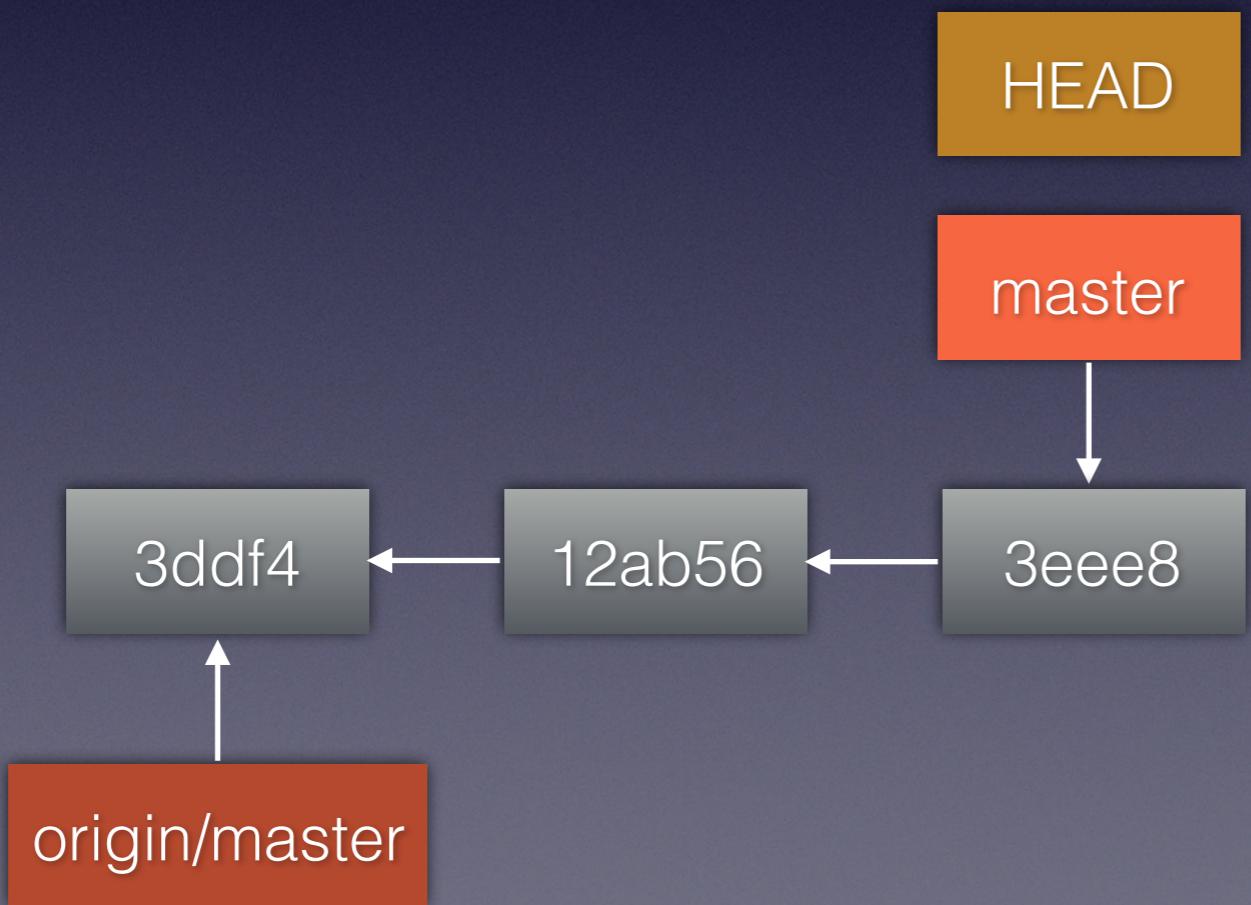
HEAD: pointer to the current branch

# Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git commit -am “commit everything”
```

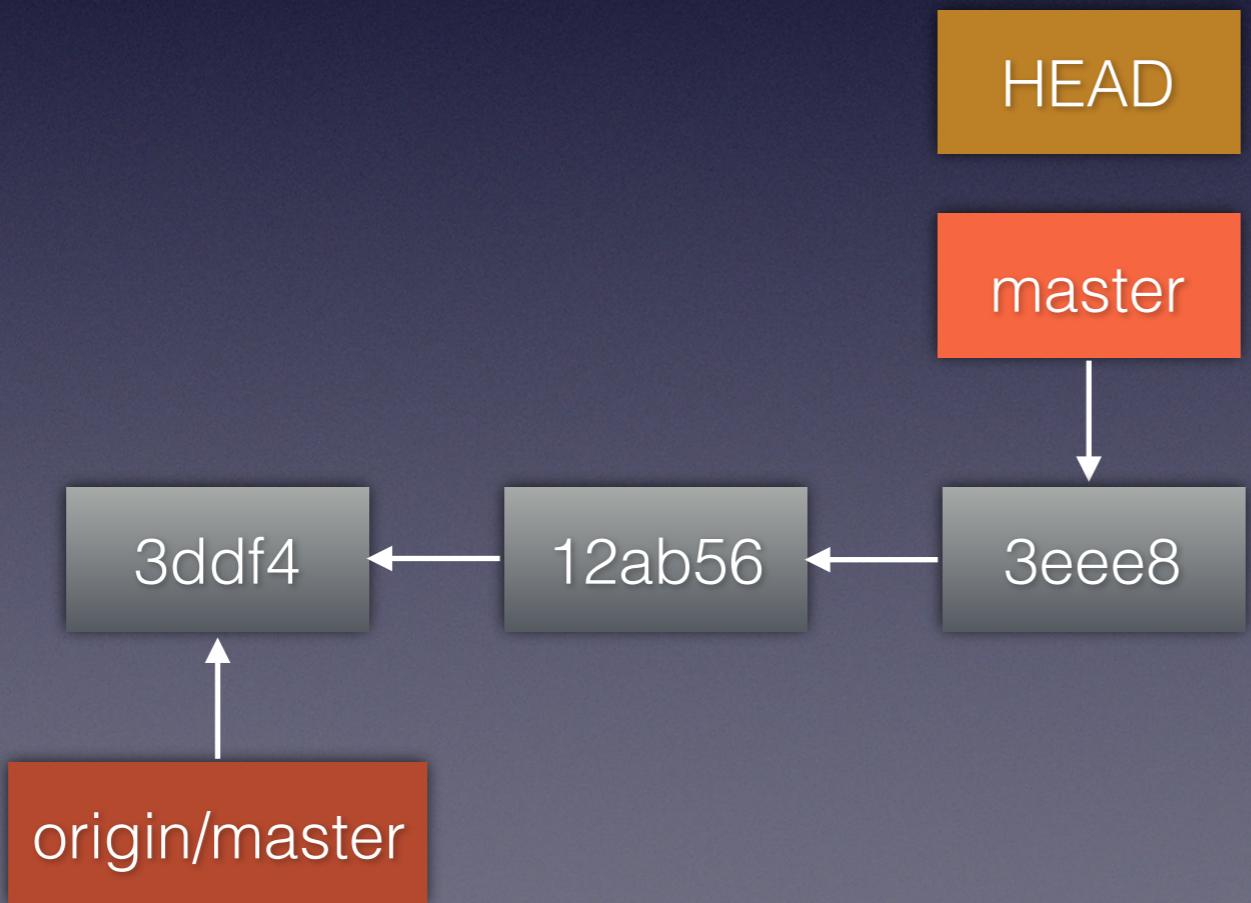


# Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git push origin master
```

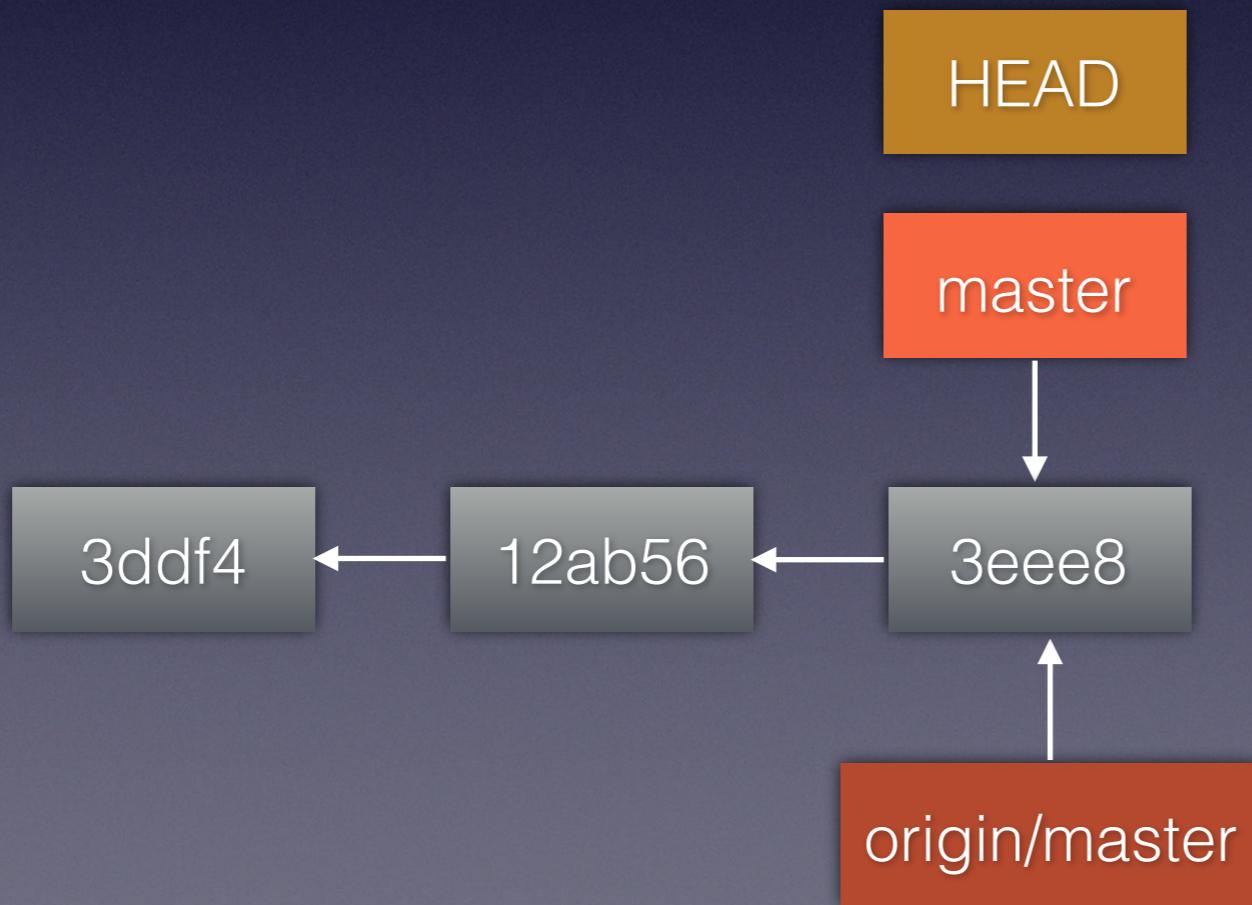


# Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git push origin master
```



# Distributed Recipes



- Centralised Workflow
- Integration-Manager Workflow
- Dictator-Lieutenants Workflow

\*recipes = workflow

# Distributed Recipes



- **Centralised Workflow**
- Integration-Manager Workflow
- Dictator-Lieutenants Workflow

\*recipes = workflow

# Centralised recipe

Easy start: everyone works on the same branch

```
$ git pull origin master
```

Do some work

```
$ git add <files>
```

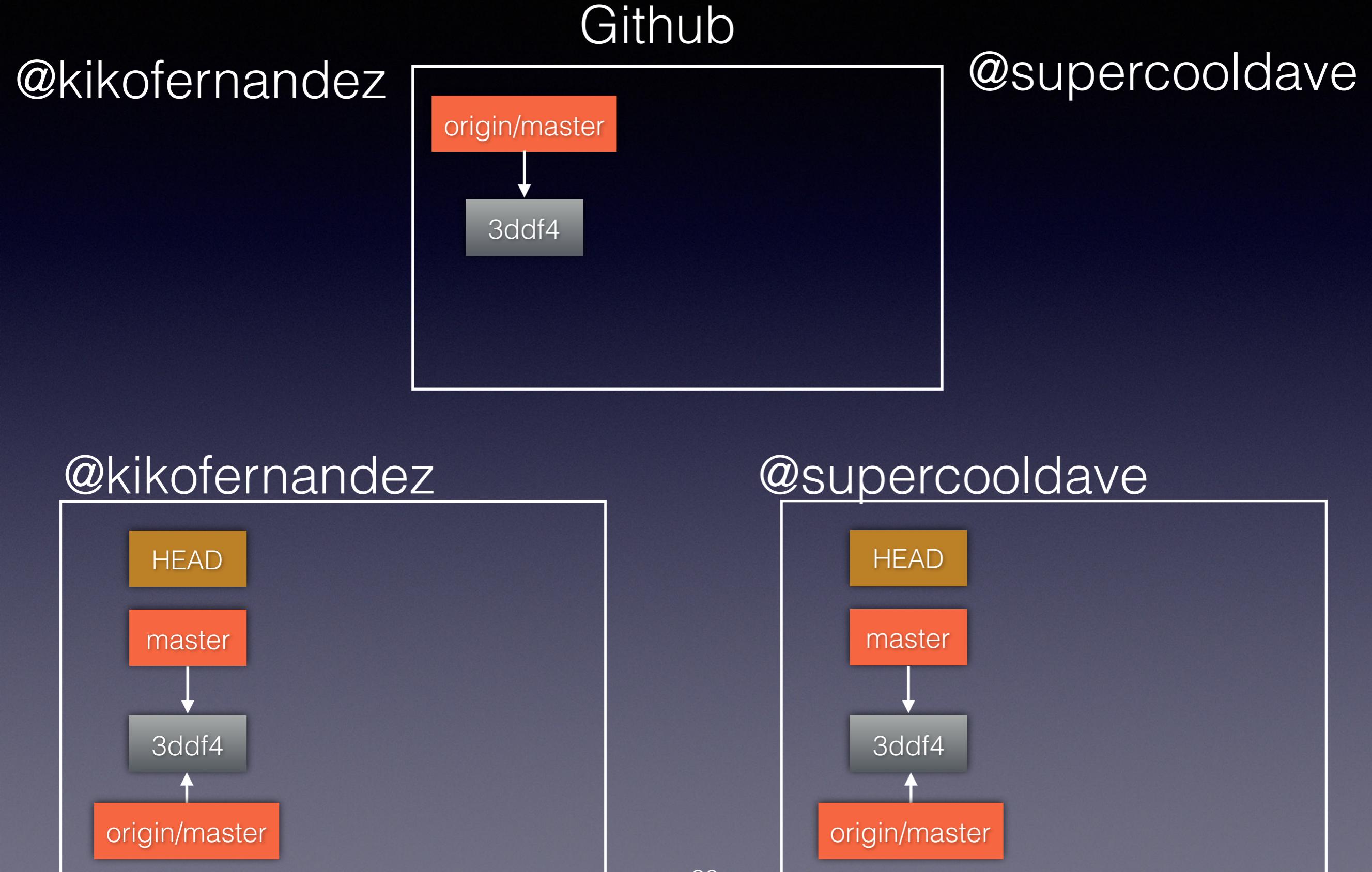
```
$ git commit -m "<message>"
```

```
$ git push origin master
```

rinse and repeat

Works until you get conflicts

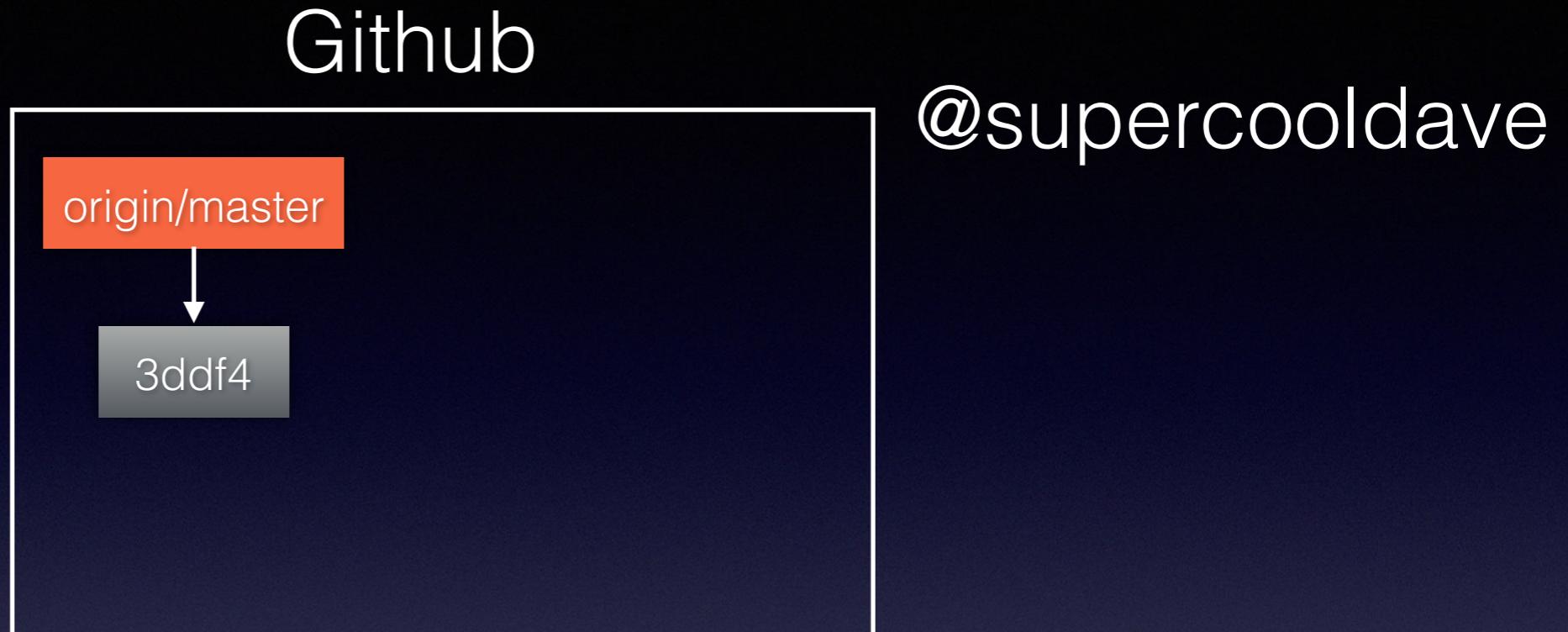
# Conflicts in Centralised recipe



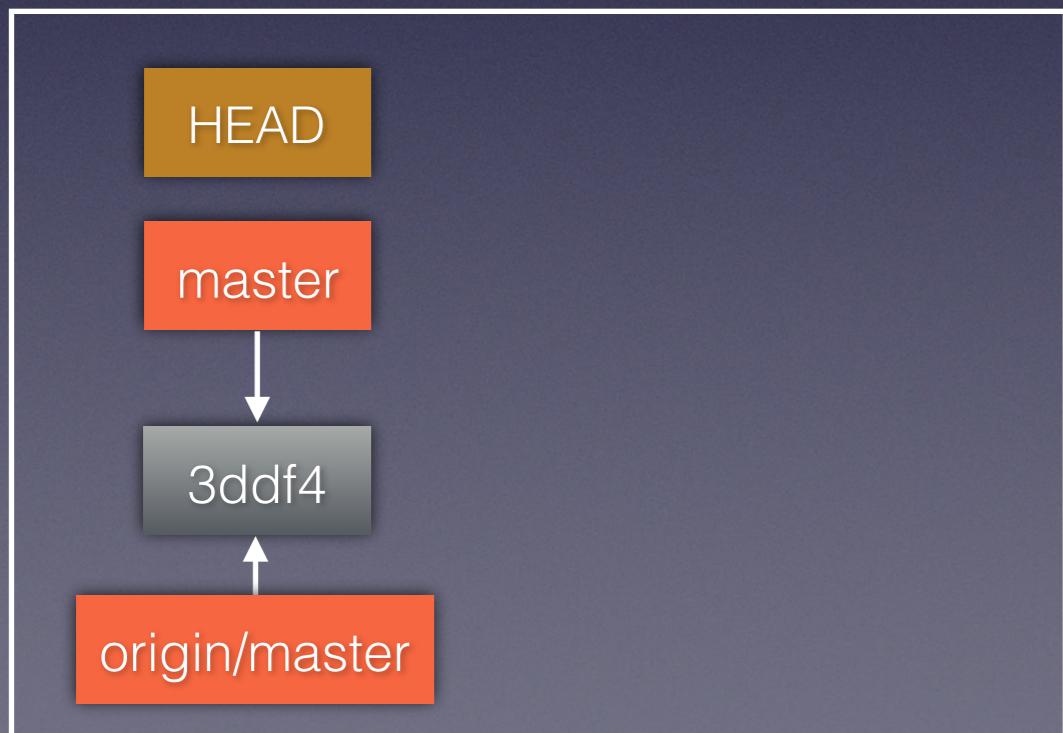
# Conflicts in Centralised recipe

@kikofernandez

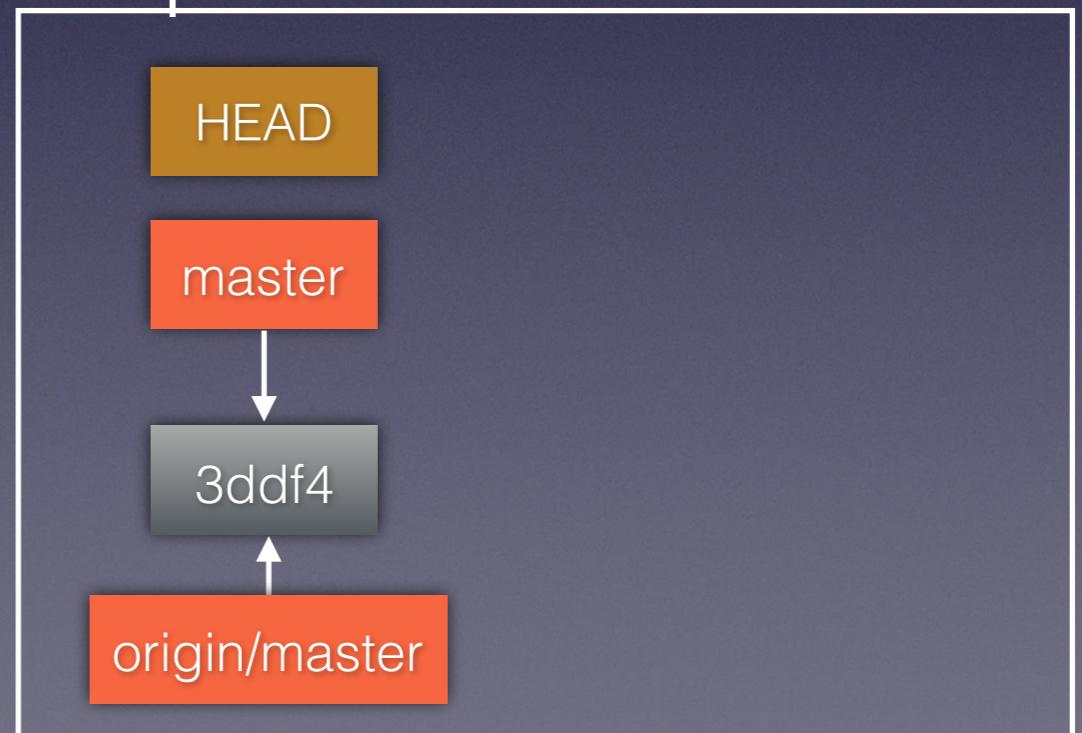
```
$ git commit -am "stuff"
```



@kikofernandez



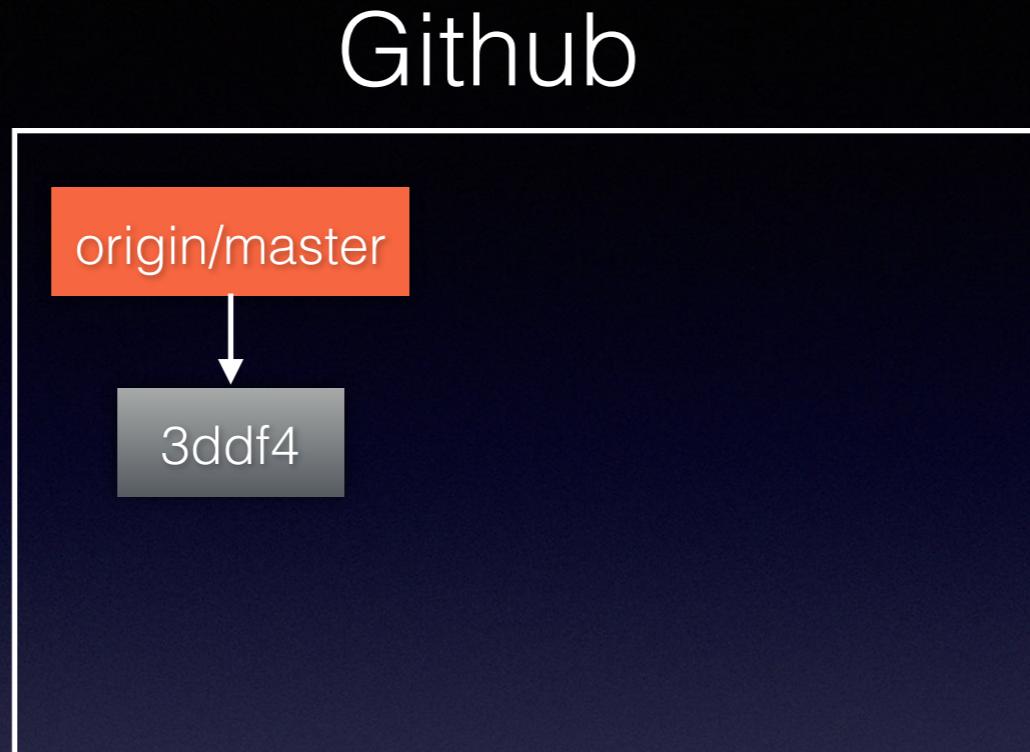
@supercooldave



# Conflicts in Centralised recipe

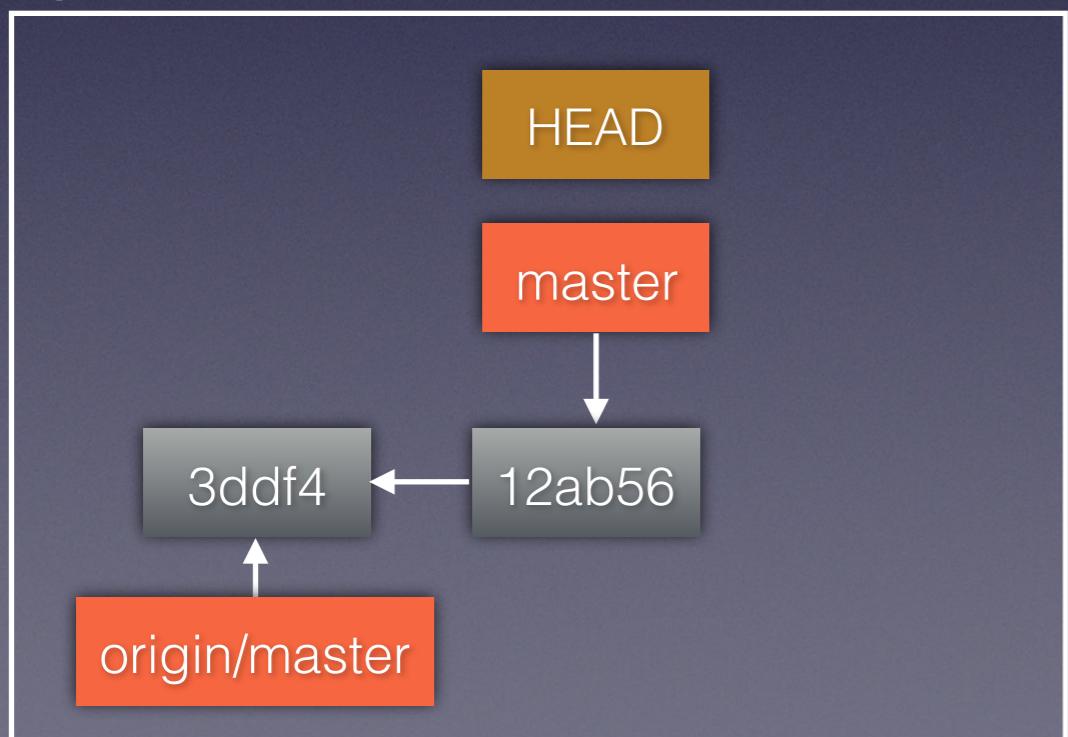
@kikofernandez

```
$ git commit -am "stuff"
```

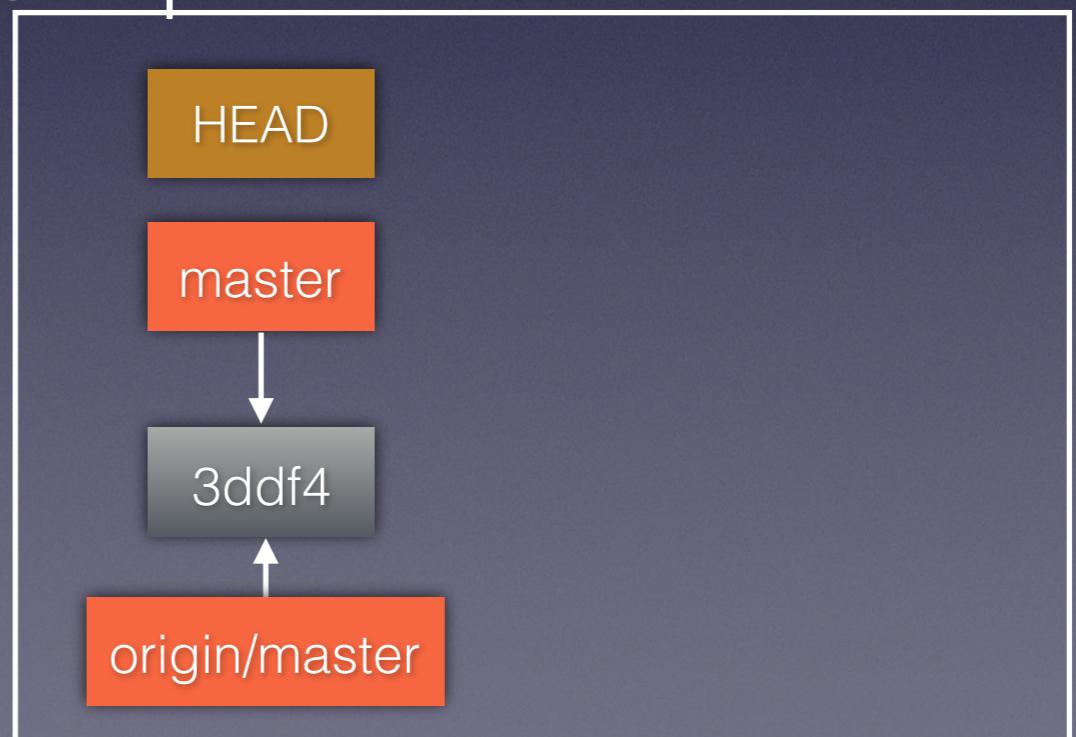


@supercooldave

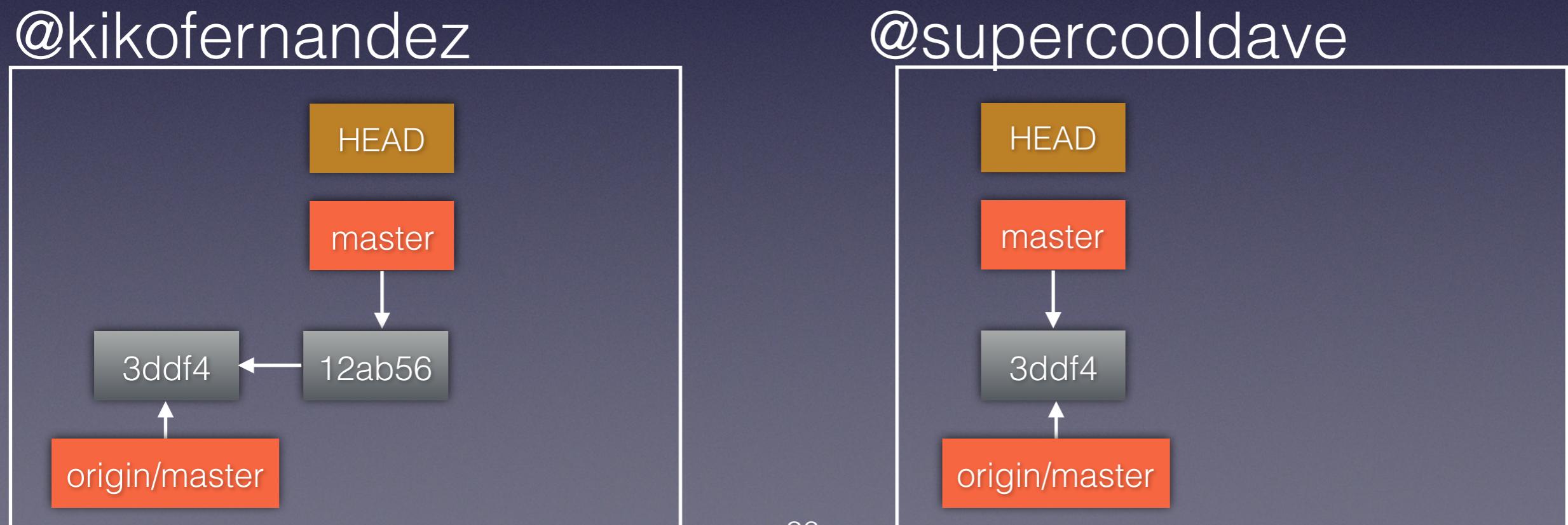
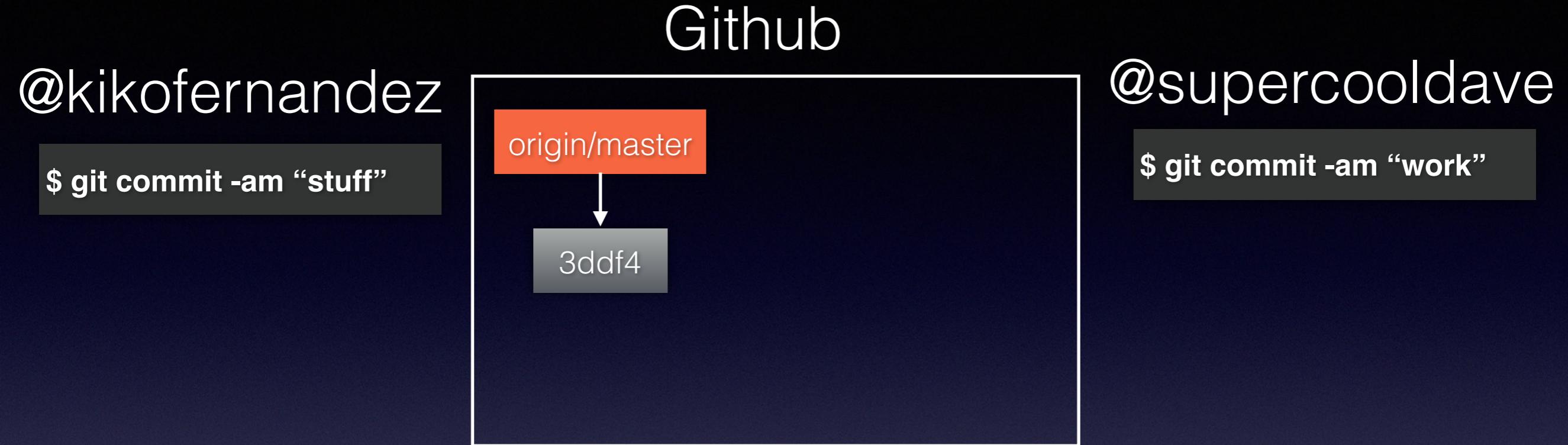
@kikofernandez



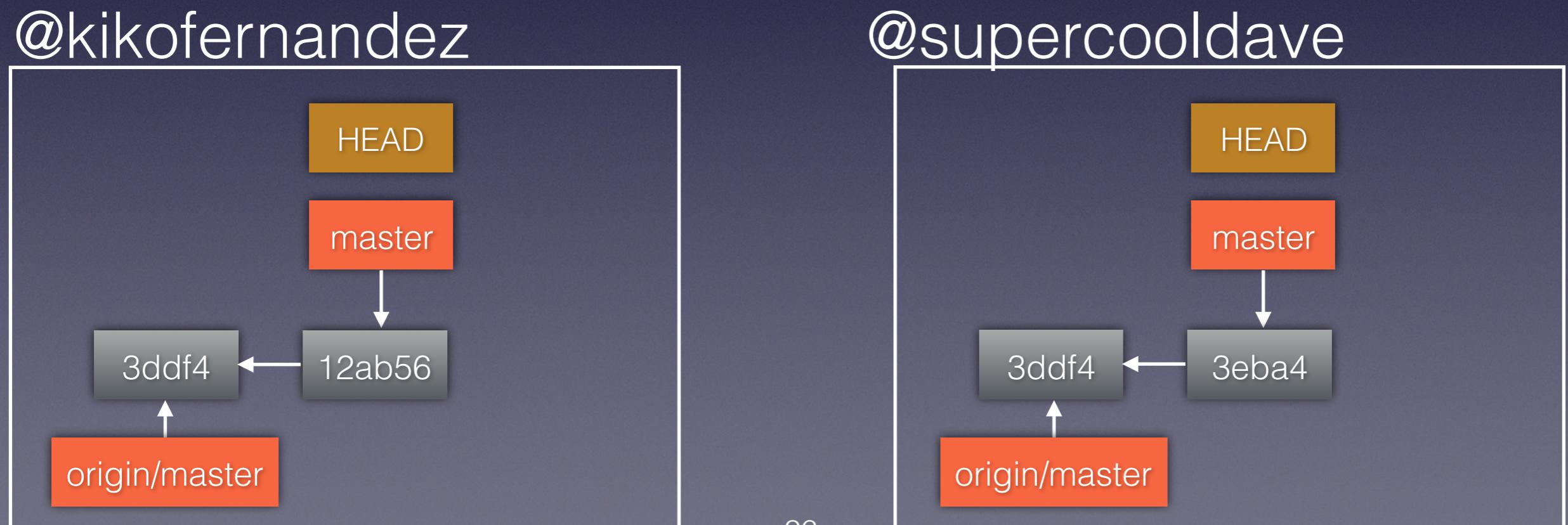
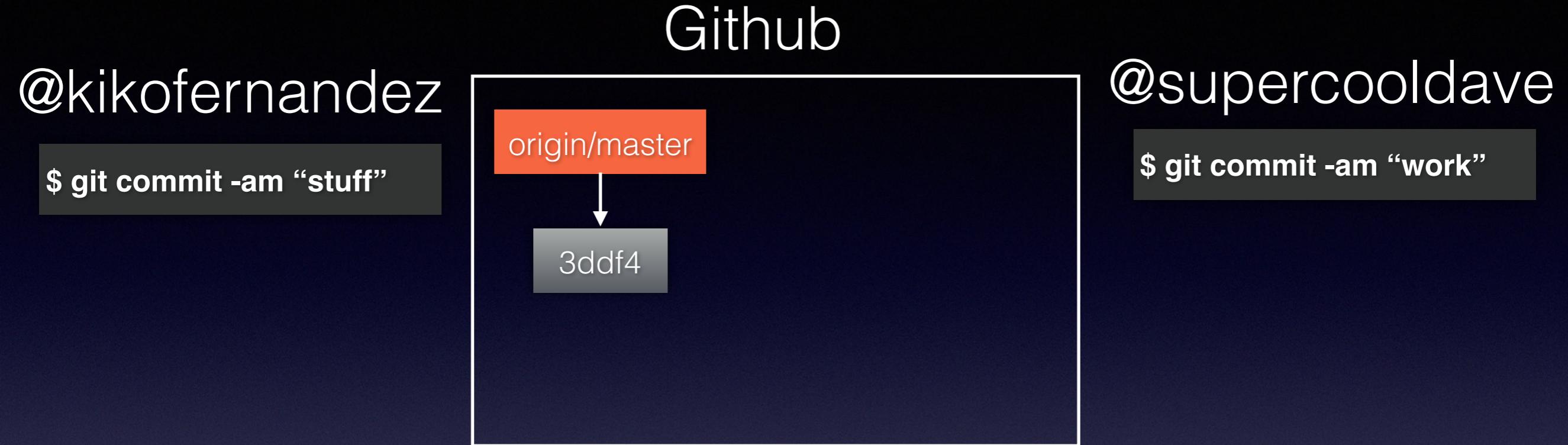
@supercooldave



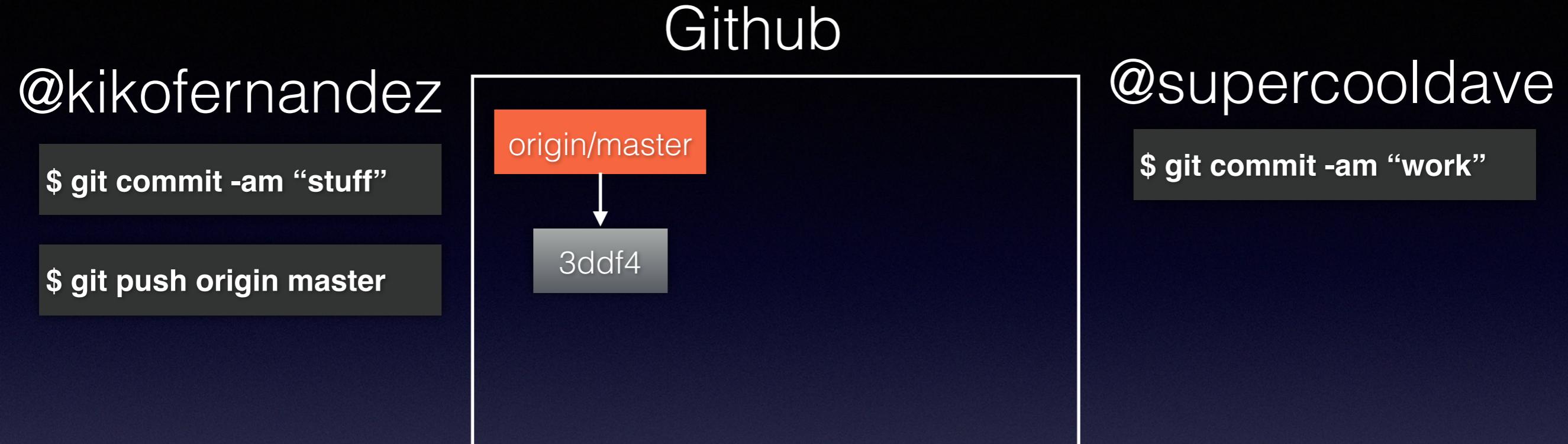
# Conflicts in Centralised recipe



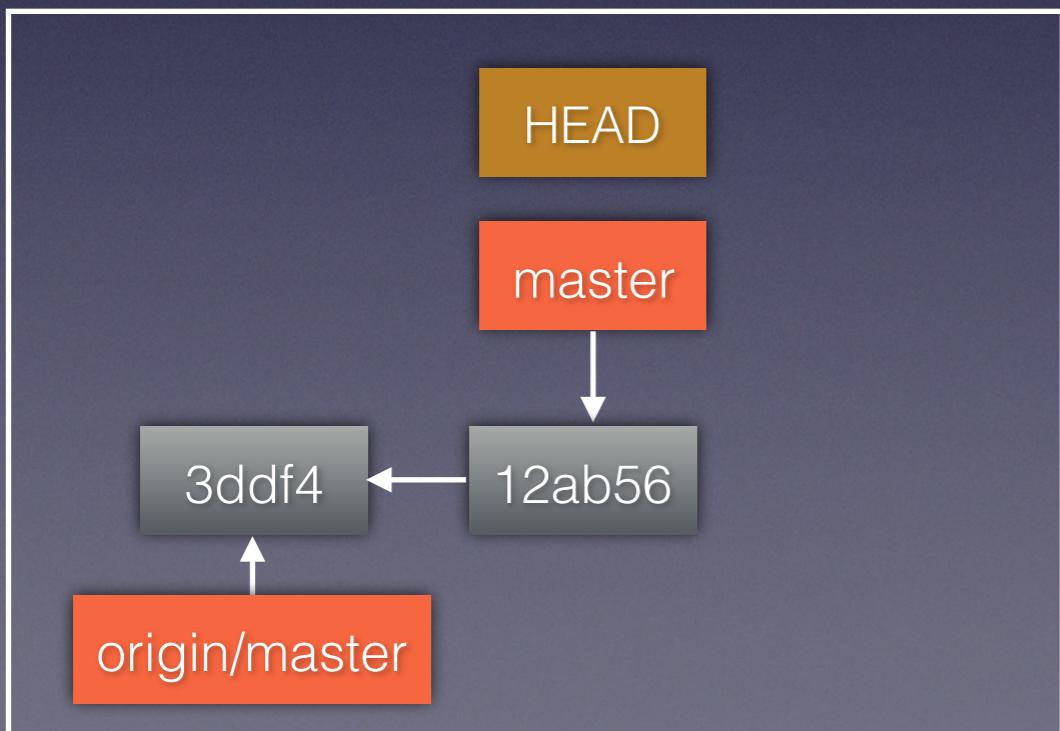
# Conflicts in Centralised recipe



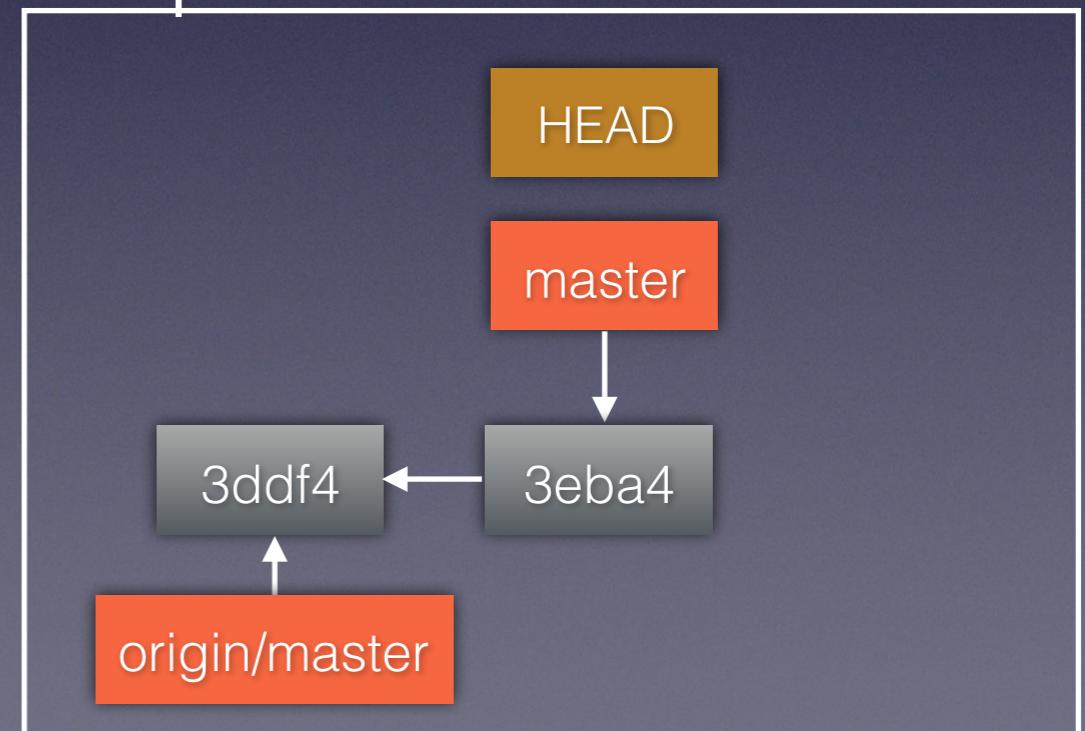
# Conflicts in Centralised recipe



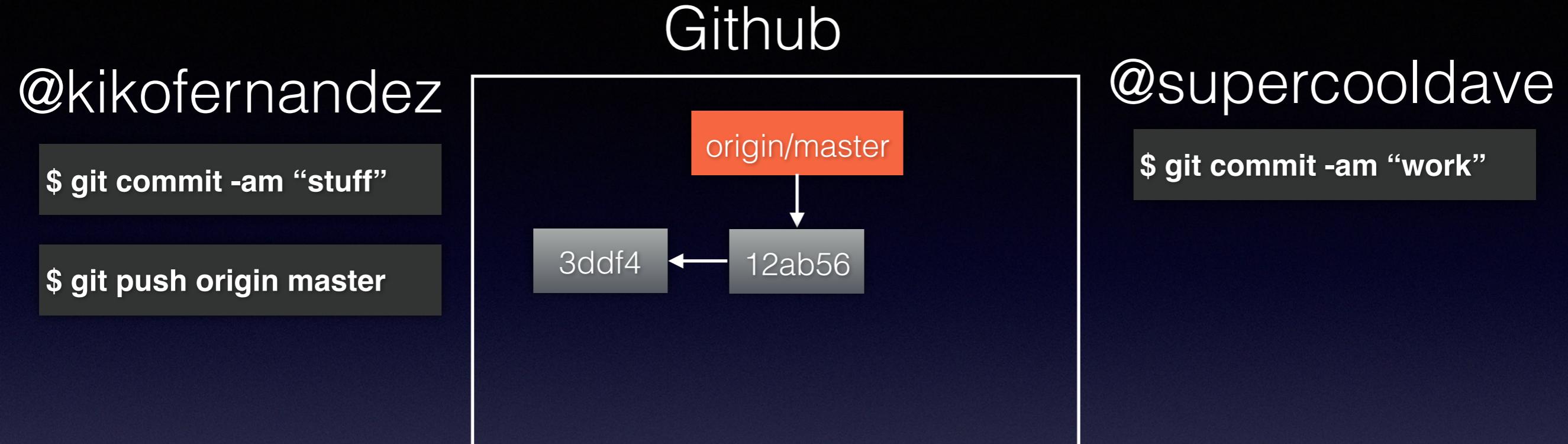
@kikofernandez



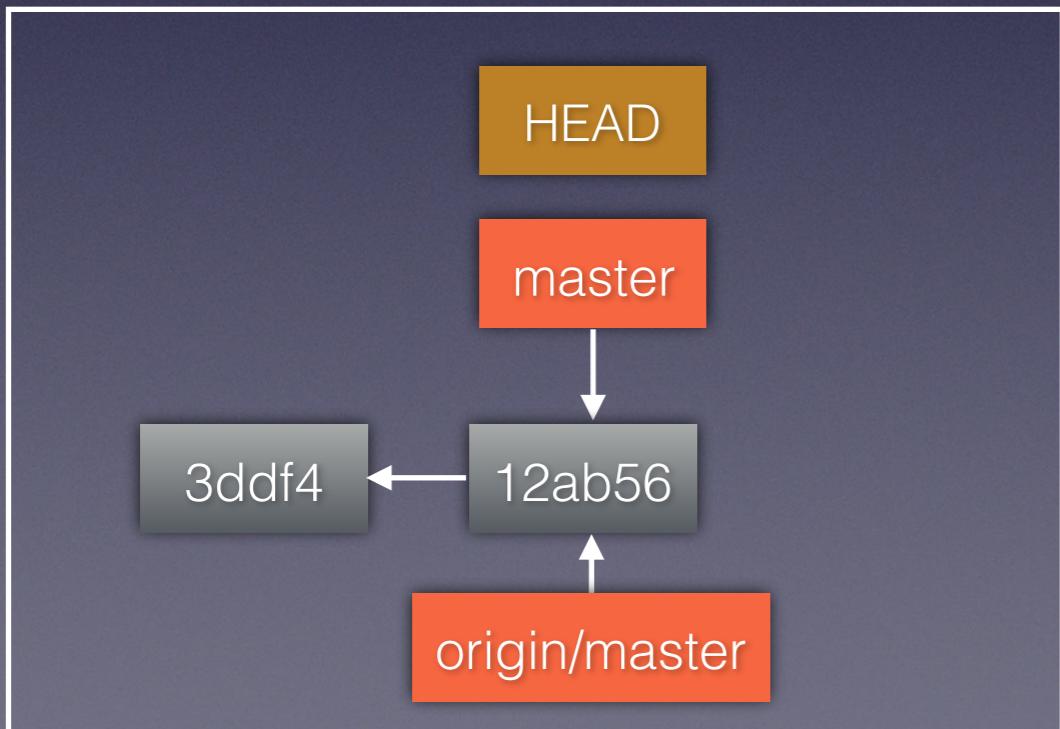
@supercooldave



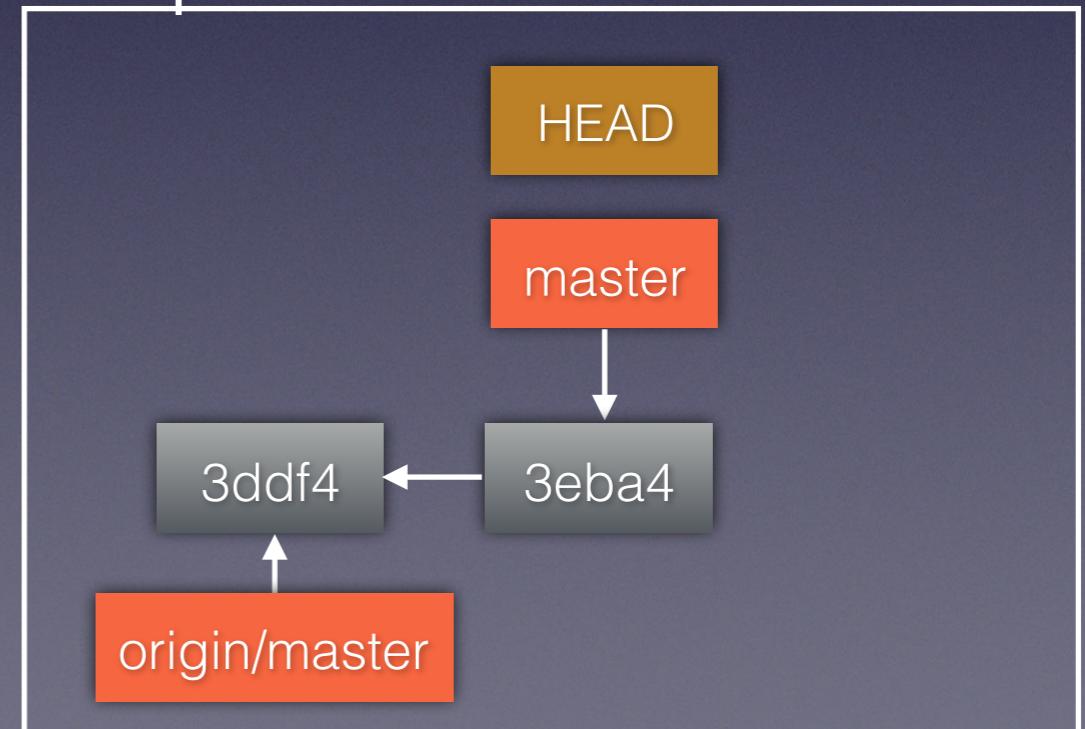
# Conflicts in Centralised recipe



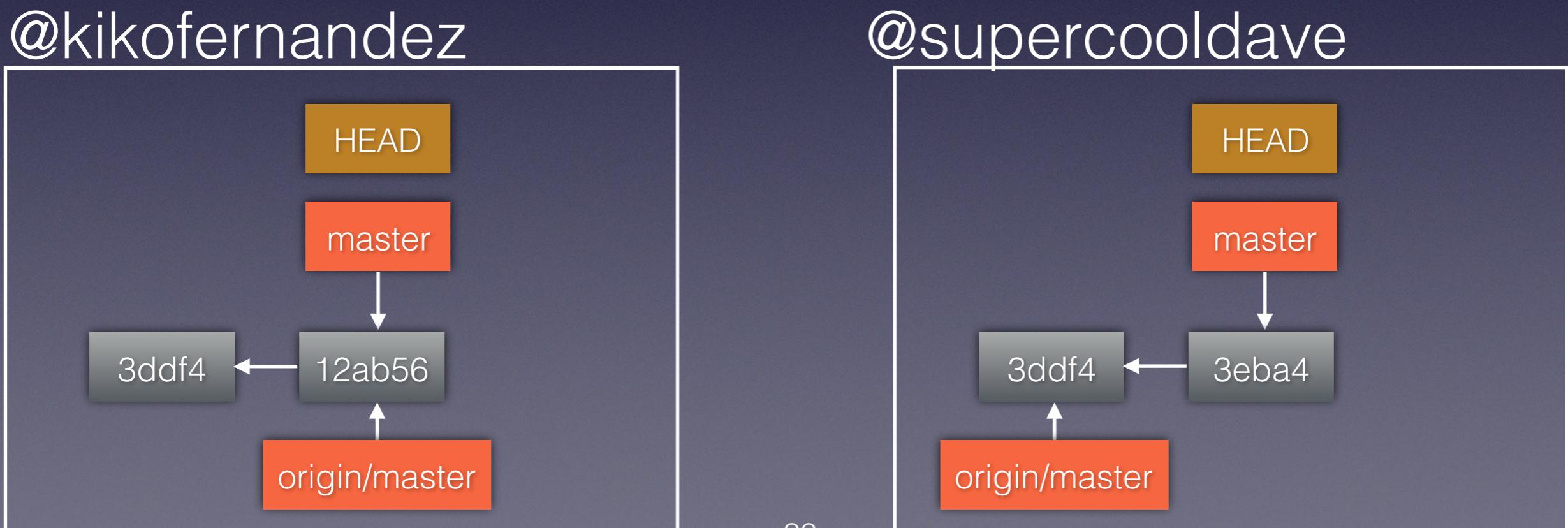
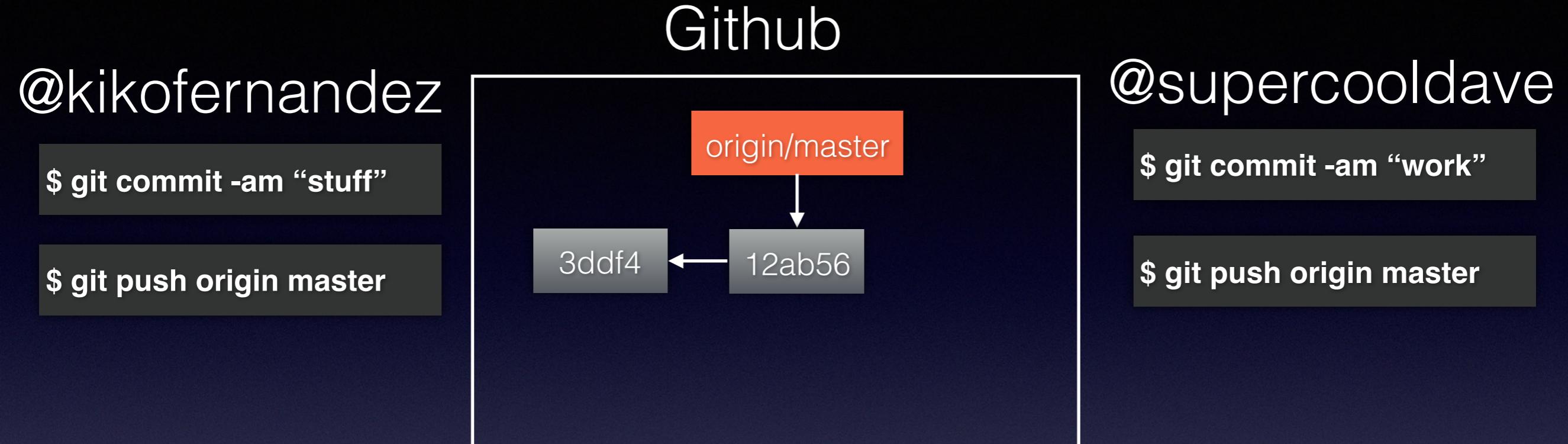
@kikofernandez



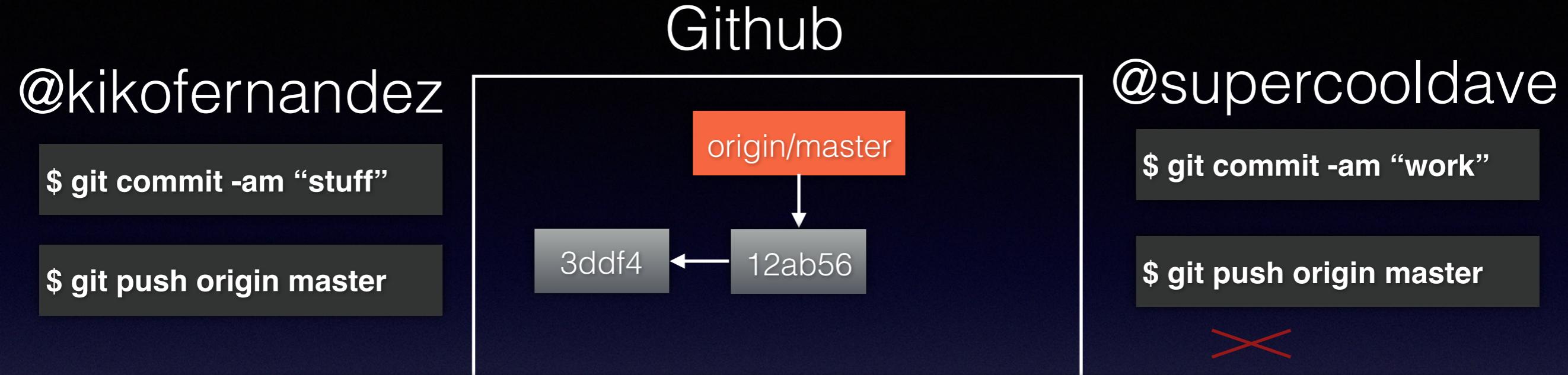
@supercooldave



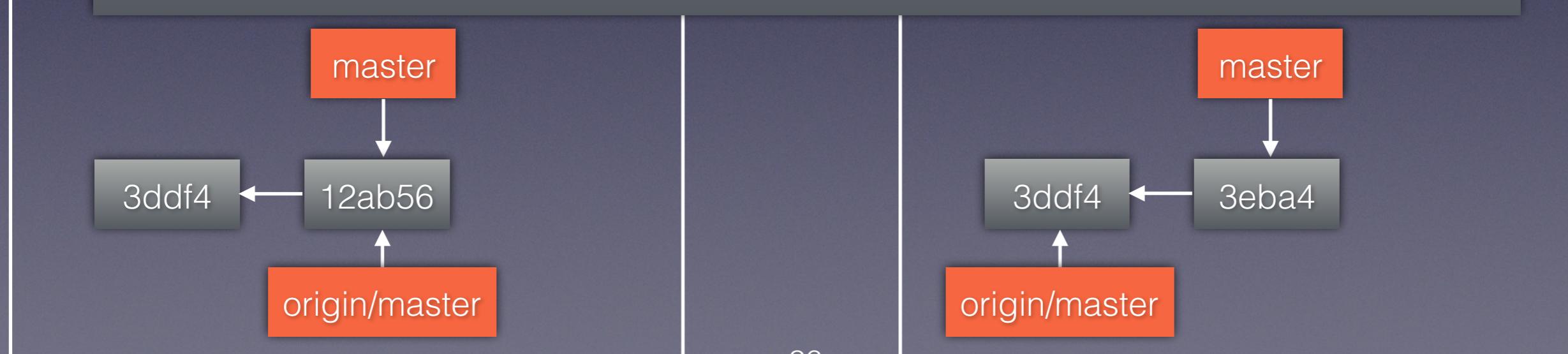
# Conflicts in Centralised recipe



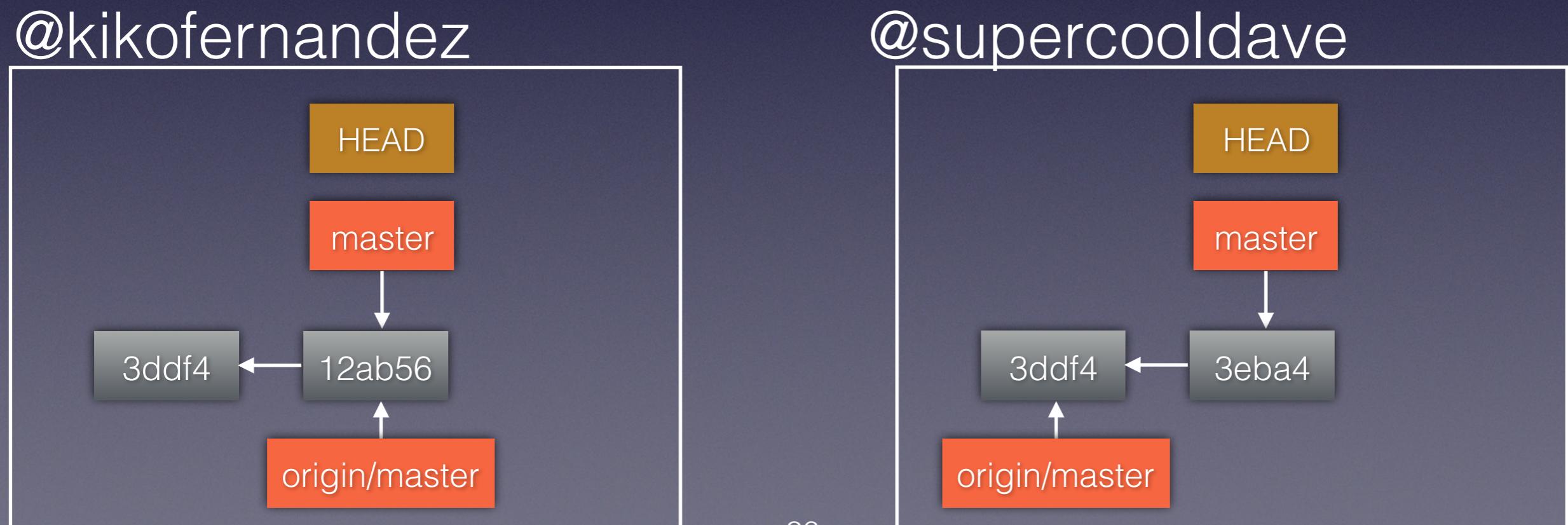
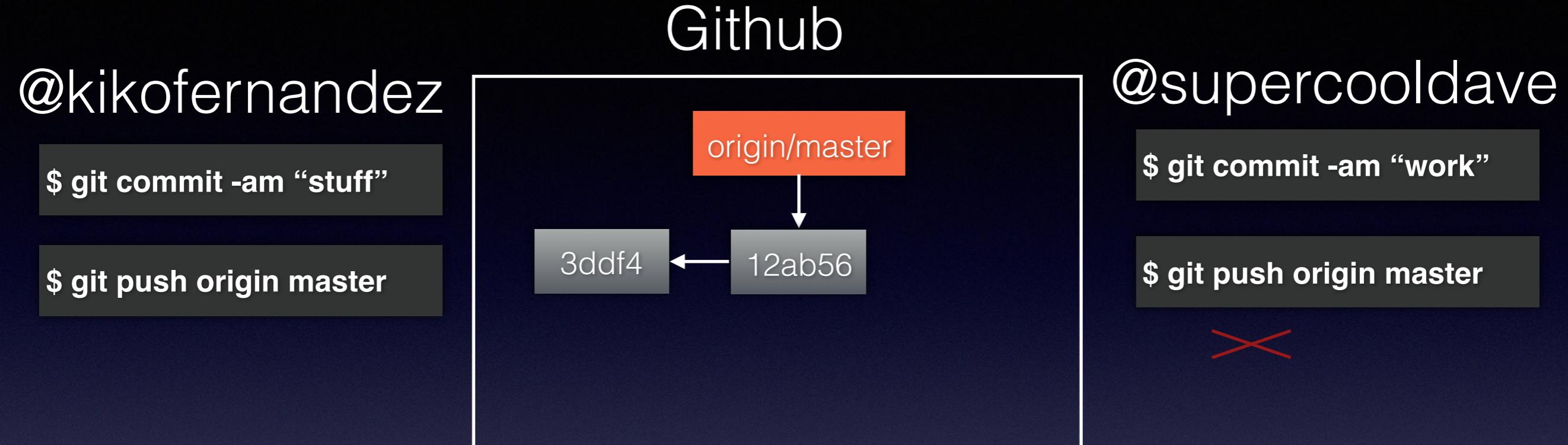
# Conflicts in Centralised recipe



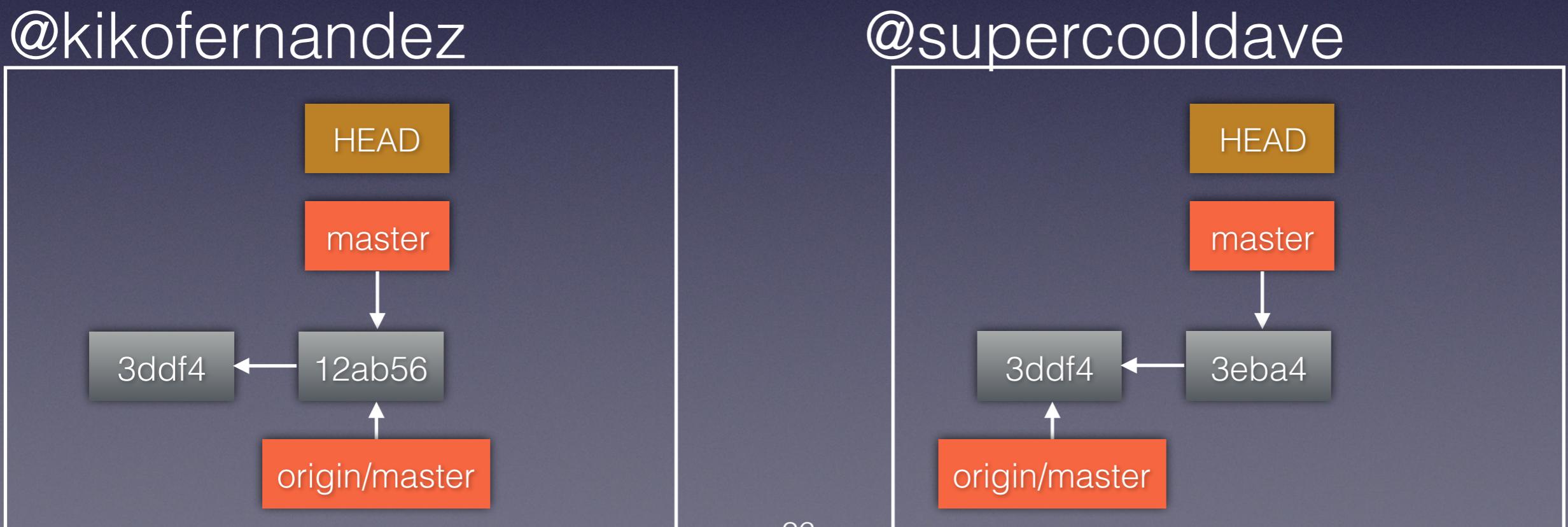
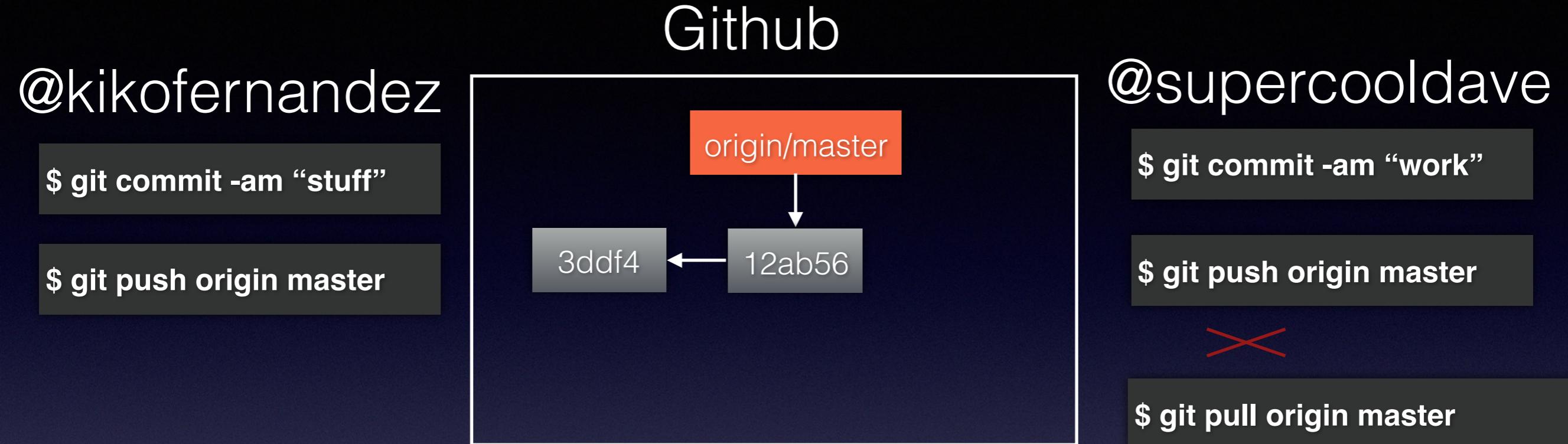
@k  
error: failed to push some refs to '/path/to/repo.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')  
hint: before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.



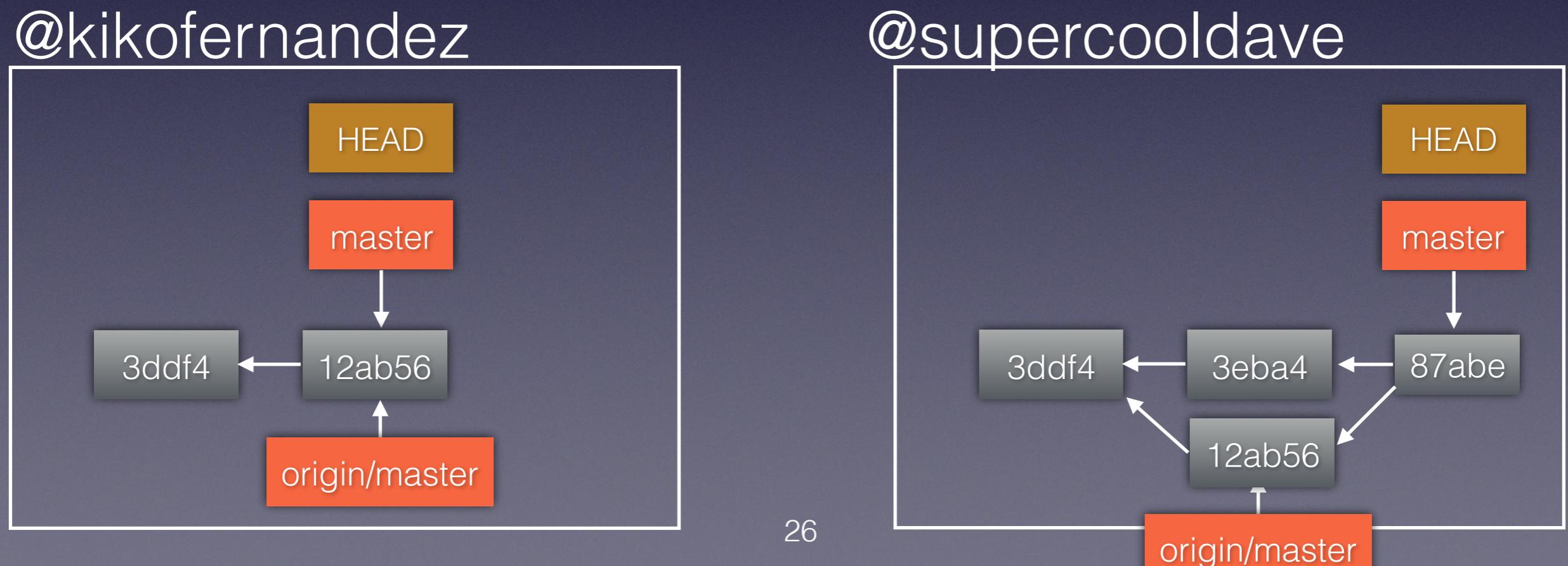
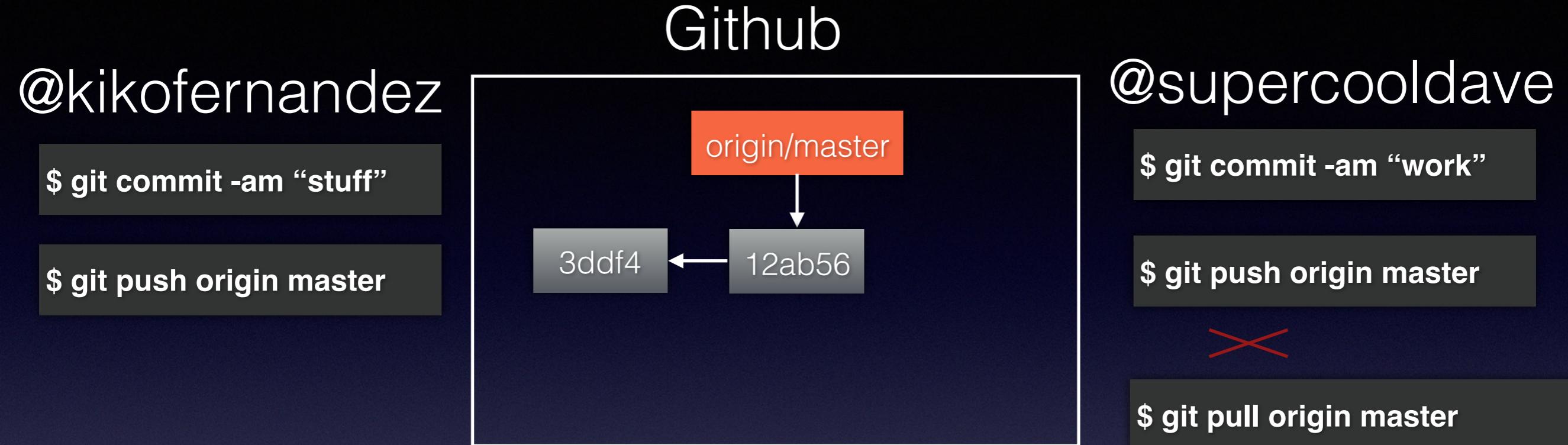
# Conflicts in Centralised recipe



# Conflicts in Centralised recipe



# Conflicts in Centralised recipe



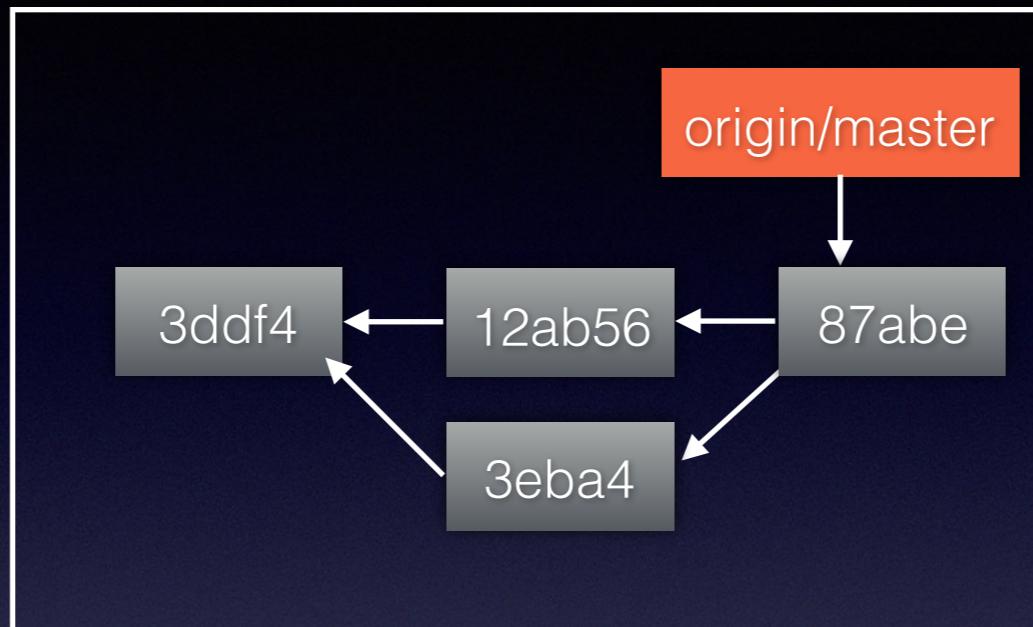
# Conflicts in Centralised recipe

Github

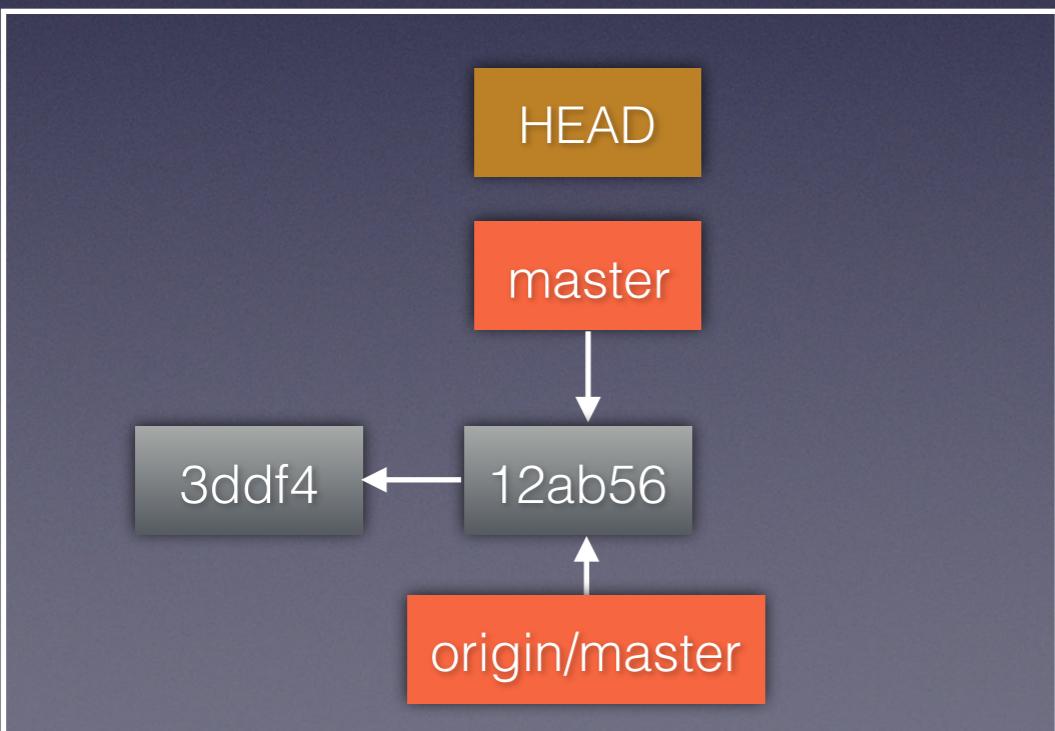
@kikofernandez

```
$ git commit -am "stuff"
```

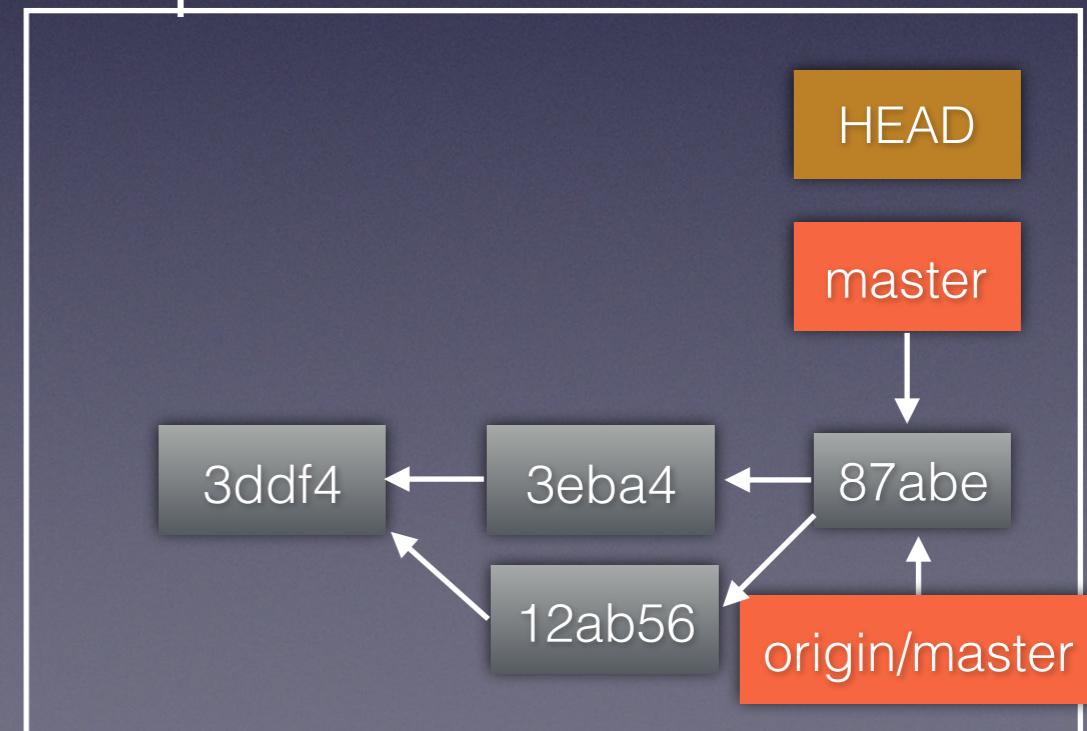
```
$ git push origin master
```



@kikofernandez



@supercooldave



# Useful links

- Conflict example (for you to practice)
- Git - the simple guide
- Github: Bootcamp
- Git: Cheatsheet
- Learn enough git to be dangerous
- CodeSchool: Try git

# Graphical Interfaces



SourceTree



GithubDesktop

# Summary

Introduction to **git** and **Github**  
&  
Distributed workflows

so...

# Summary

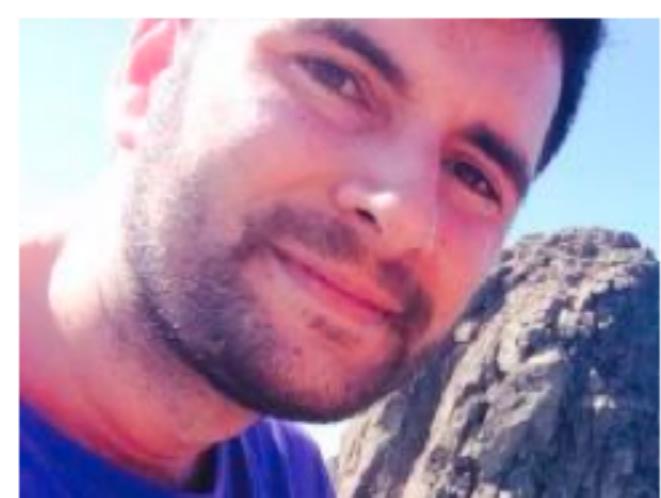
Introduction to **git** and **Github**  
&  
Distributed workflows

so...

let's **git** some fun

# Thanks

Feedback is welcome (email) or at the end of the course



**Address:** Computing Science Division  
Department of Information Technology  
Uppsala University  
Box 337  
SE-751 05 Uppsala  
Sweden

**Visit:** ITC building 1, floor 3, room 1306

**Phone:** [+46 18 - 471 4361](tel:+46184714361)

**Fax:** [+46 18 511925](tel:+4618511925)

**Email:** [kiko.fernandez@it.uu.se](mailto:kiko.fernandez@it.uu.se)