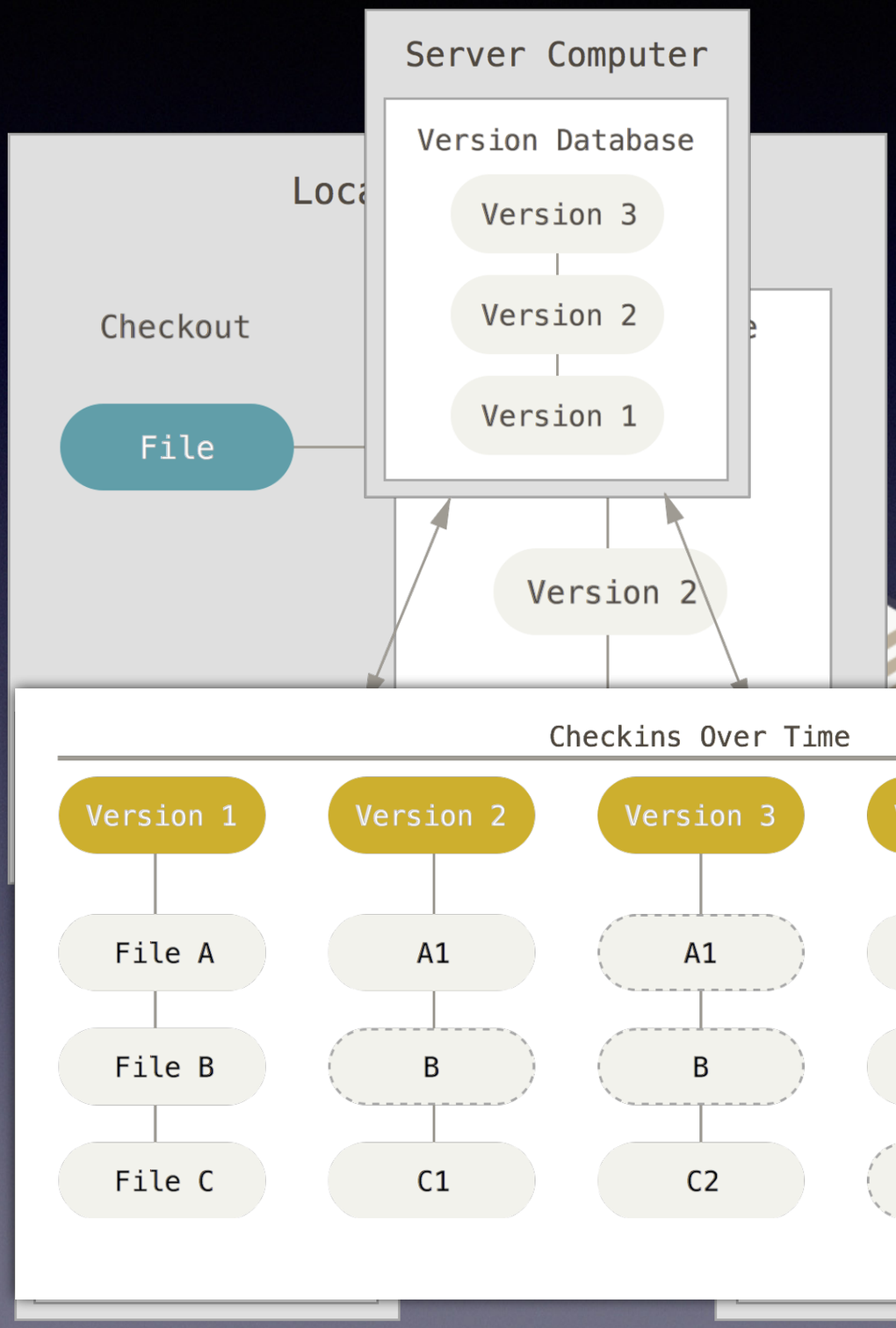
A brief introduction to git and Github

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

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)

Windows
Linux & OS X

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>
```

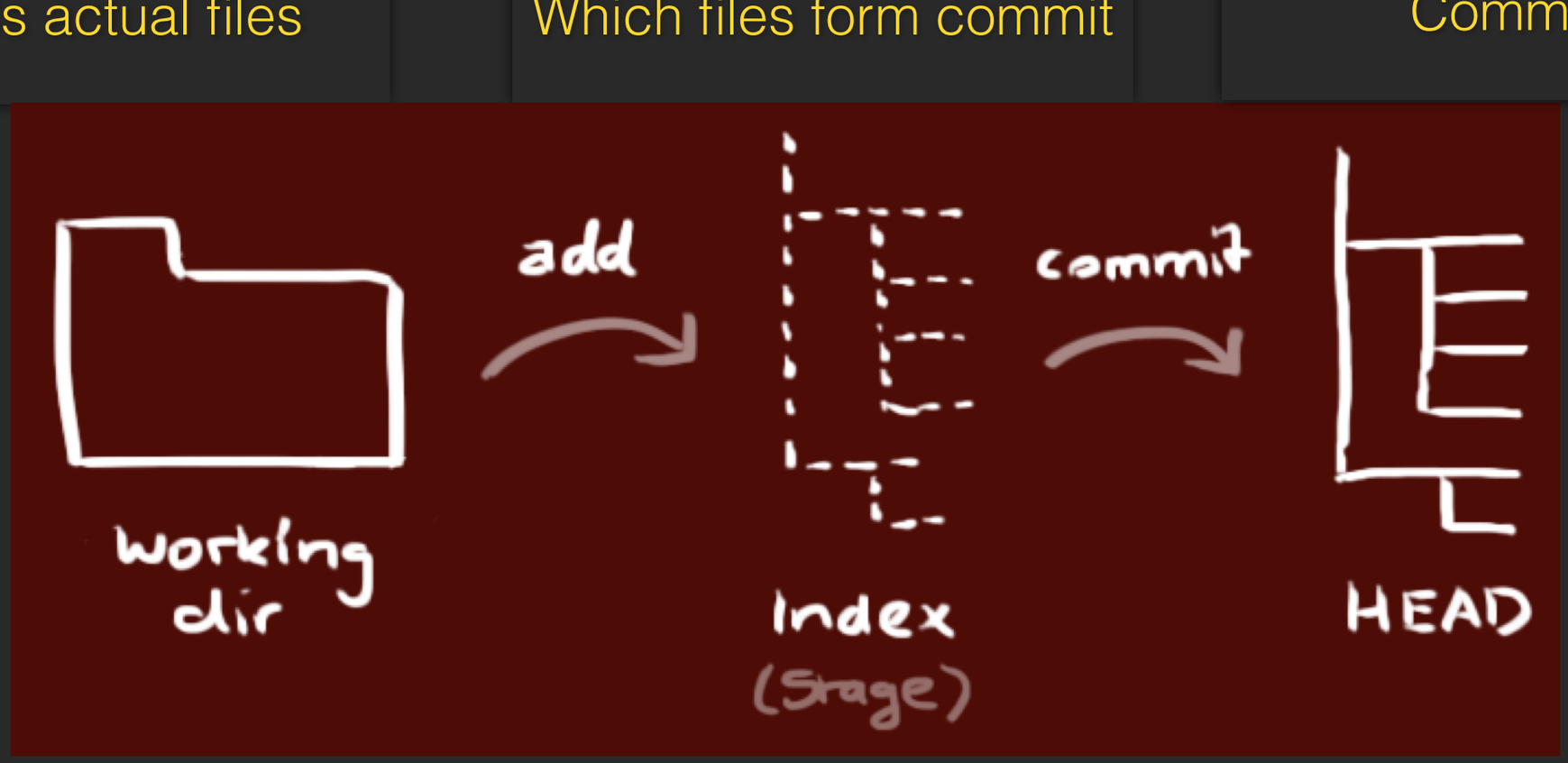
REMOTE = the remote server, Github

LOCAL = your local machine

Git: Basics

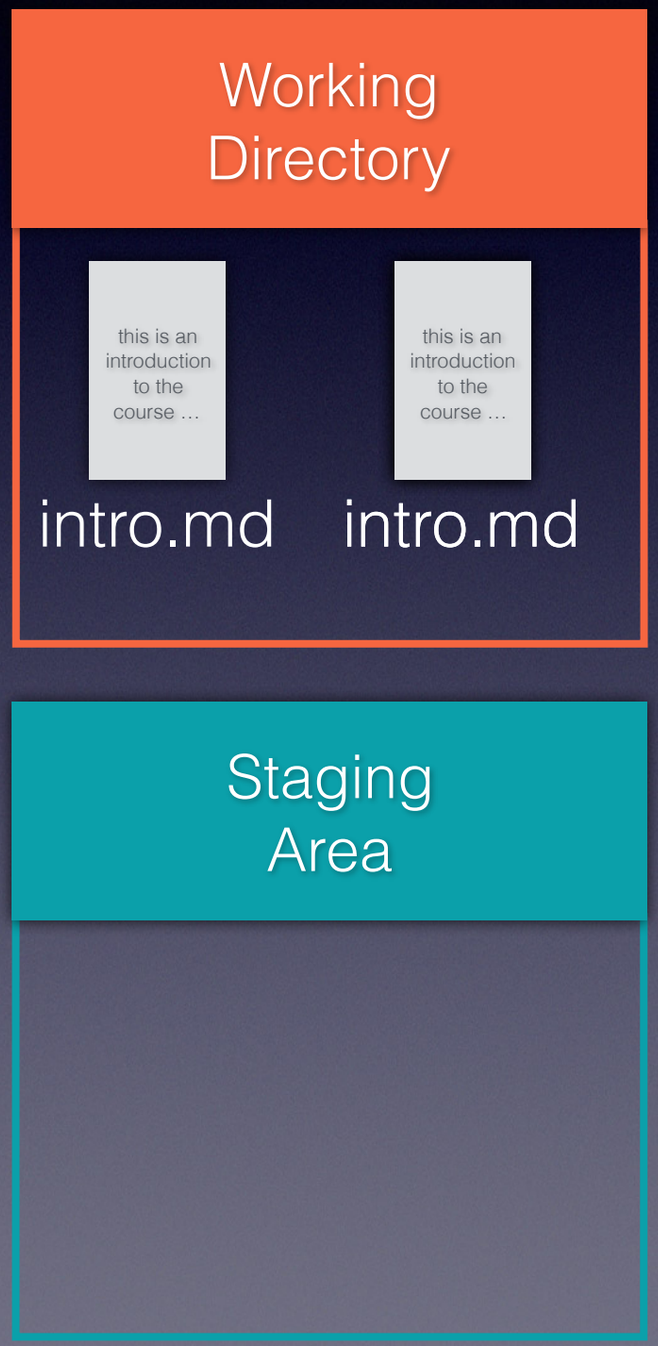
LOCAL

Working Directory	Staging Area	.git directory (Repository)
Holds actual files	Which files form commit	Commits



Git: Basics

LOCAL



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”
```

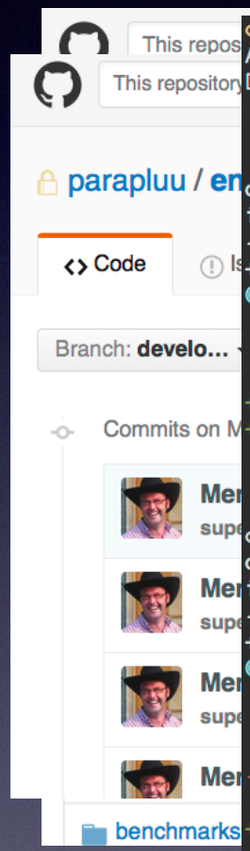
Git: Logs

How do I see the changes?

REMOTE

LOCAL

$ git log -p



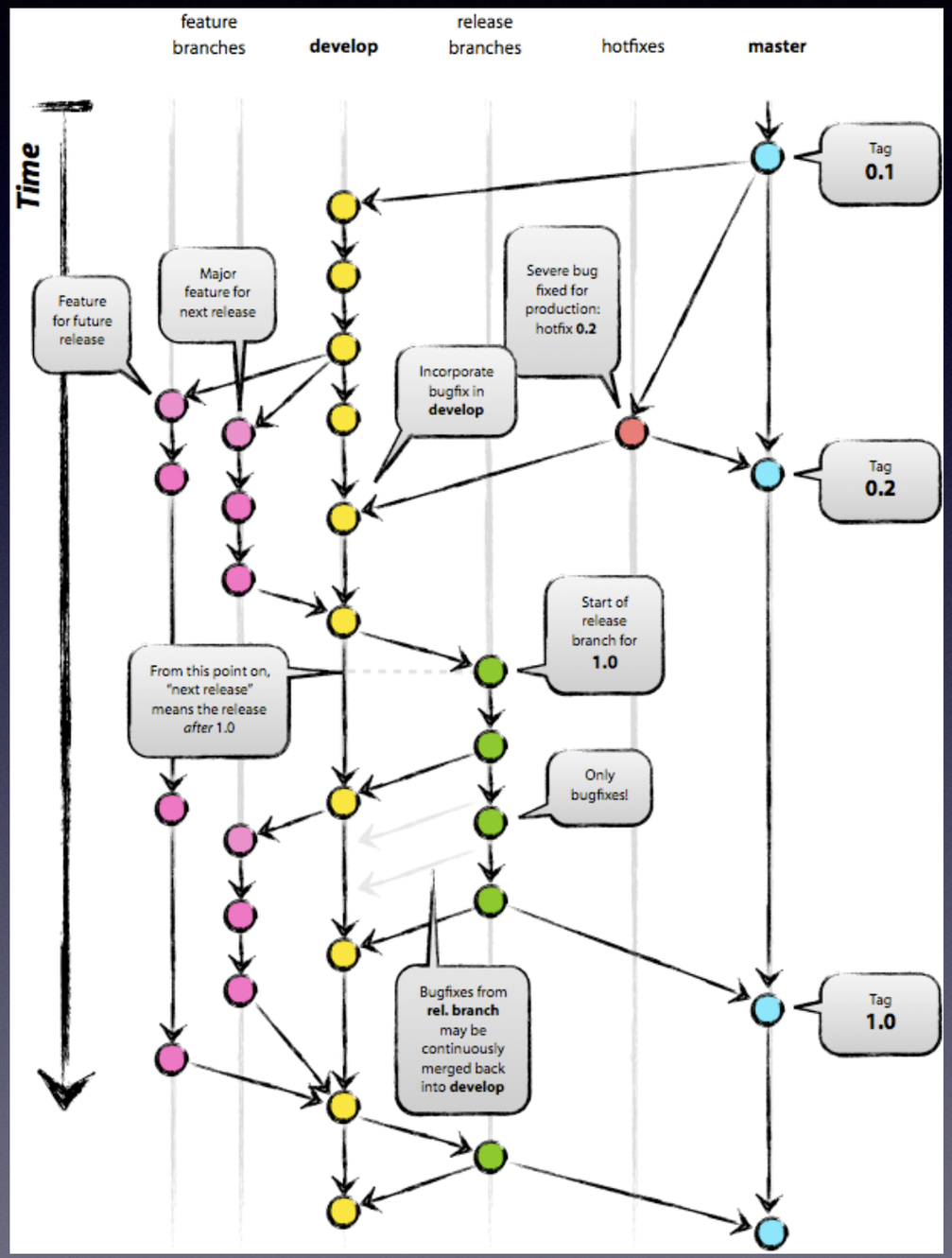

```
commit 810c19b5d46f94d4d2eac1f0dd361daf9235c7e8
Author: Kiko Fernandez Reyes <kiko.fernandez@it.uu.se>
Date:   Thu Mar 24 08:12:37 2016 +0100

    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 de
cls)
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.hpparams formalArgs) (A.pparams fcall)
       typeVariables = map translateParametricArguments (A.pparams 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);
 }
```

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

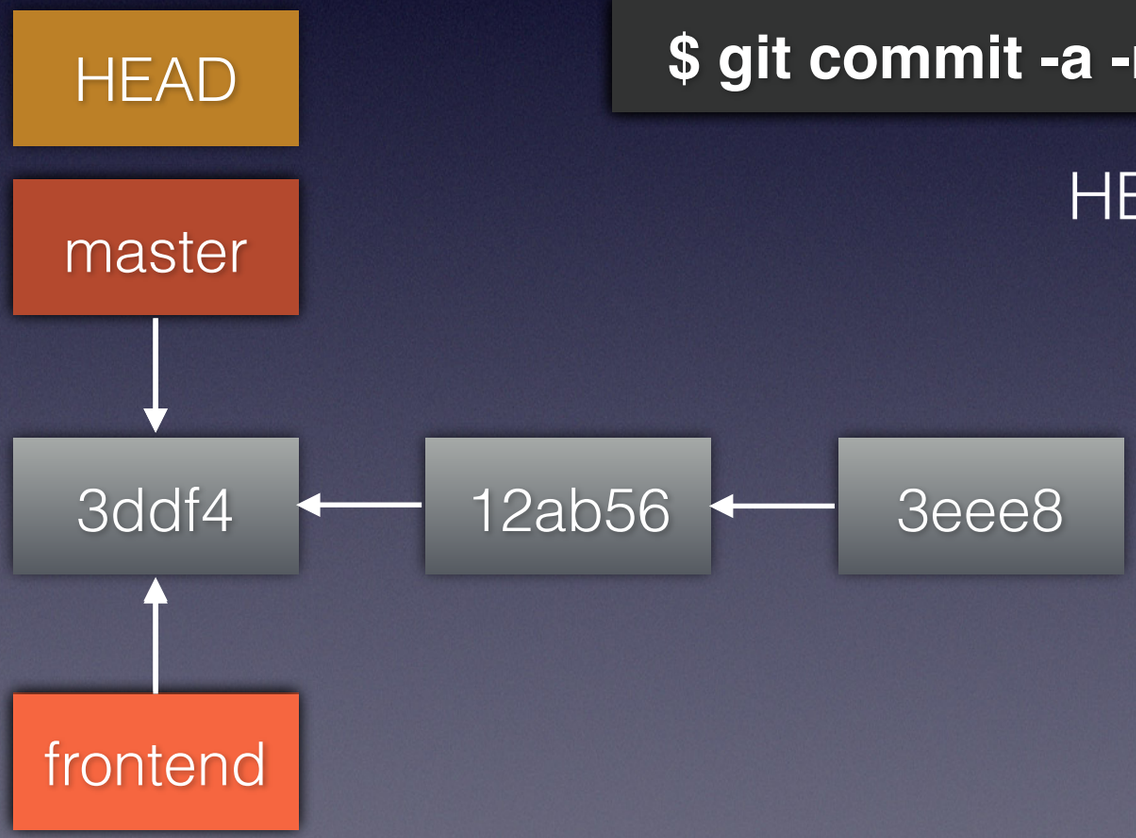
```
$ git add intro.md; git commit -m “update”
```

Git: Branching

LOCAL

Branches: they are pointers to commits / versions

Default branch: master




```
$ git commit -a -m “commit everything”
```

HEAD: pointer to the current branch

Git: Branching

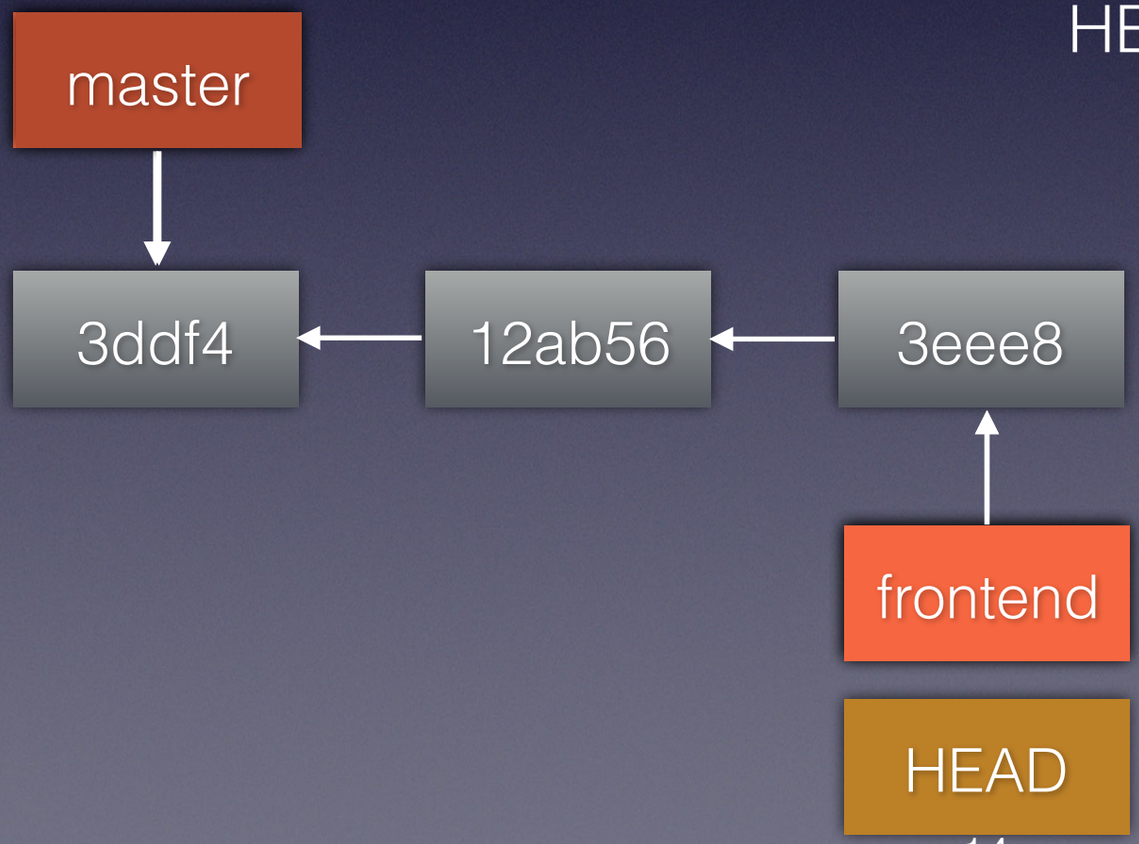
LOCAL

Branches: they are pointers to commits / versions

Default branch: master

```
$ git checkout master
```

HEAD: pointer to the current branch



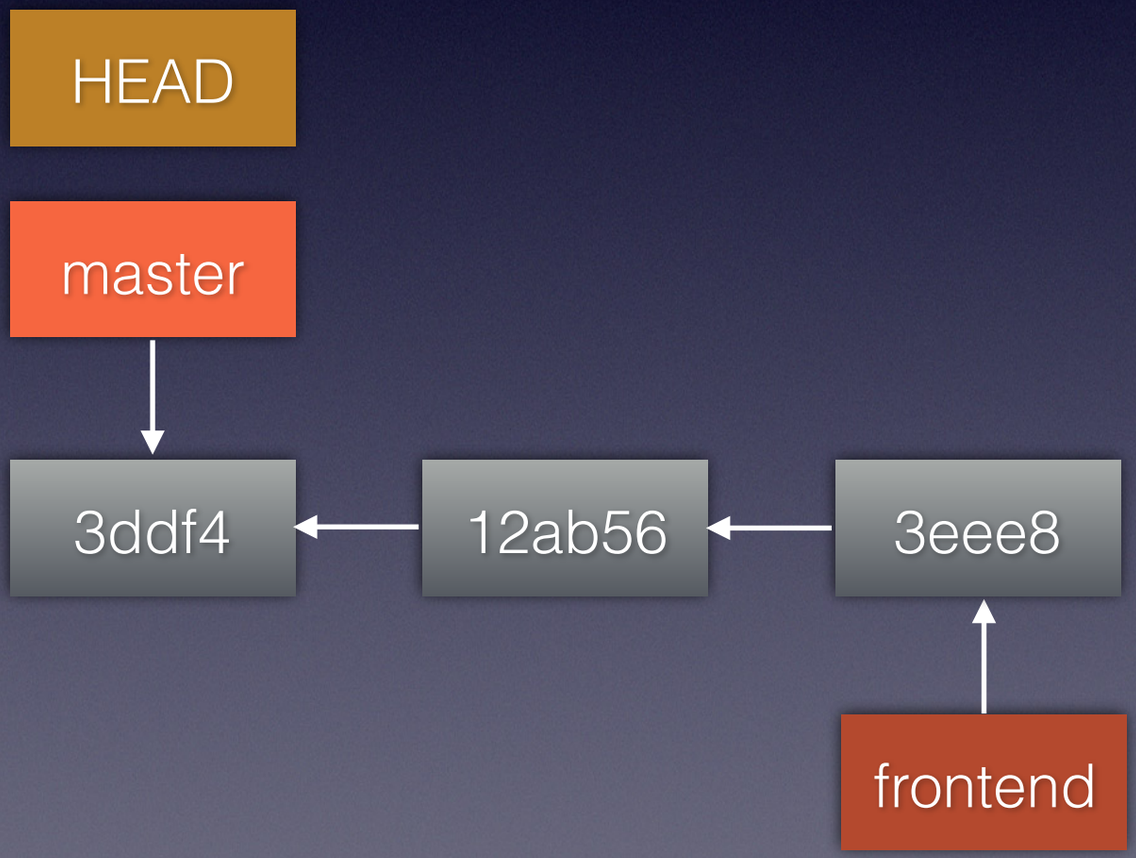
Git: Branching

LOCAL

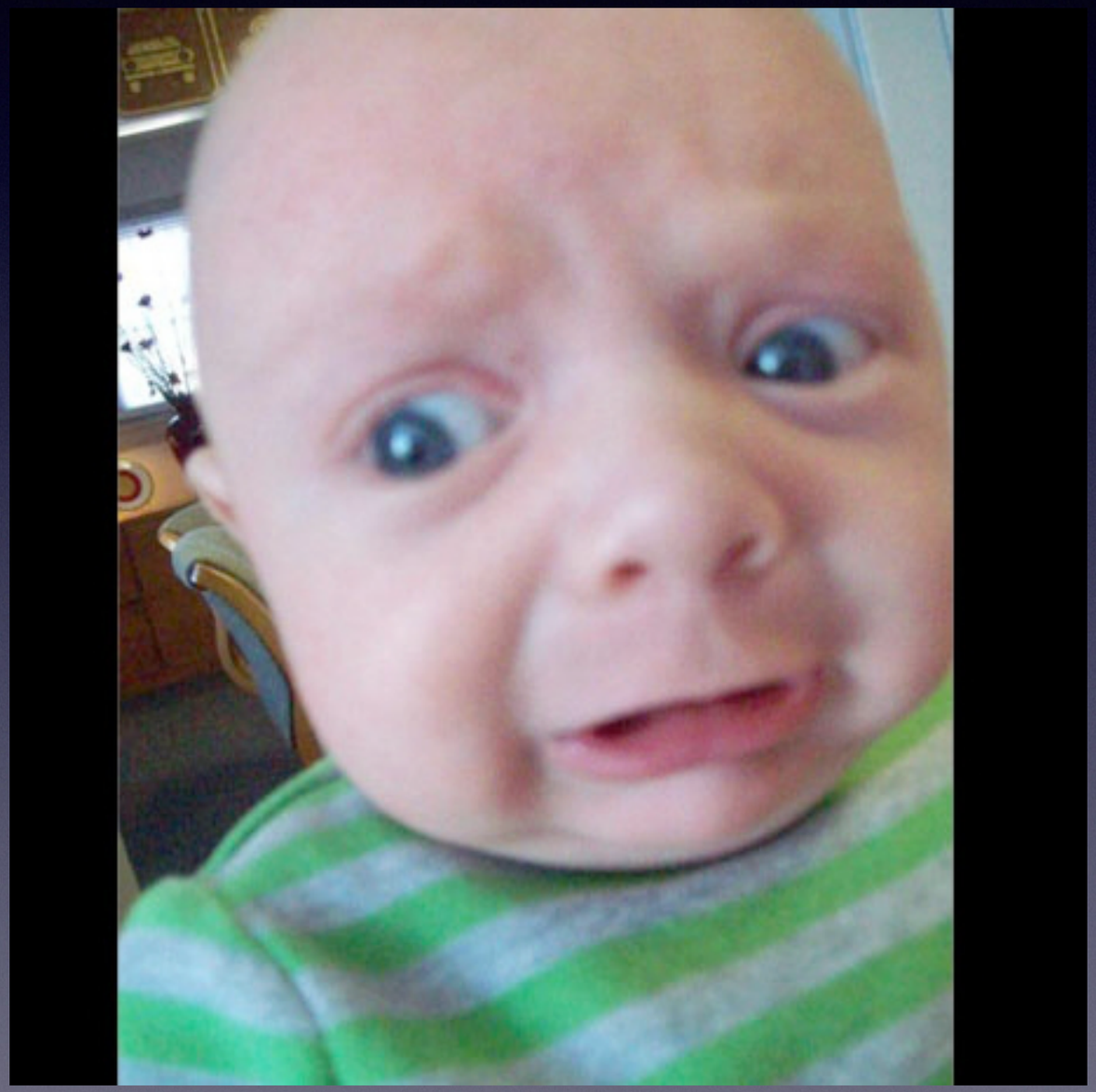
How to merge master with the changes in frontend?

```
$ git merge frontend
```

HEAD: pointer to the current branch



Git Conflicts



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?

- **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

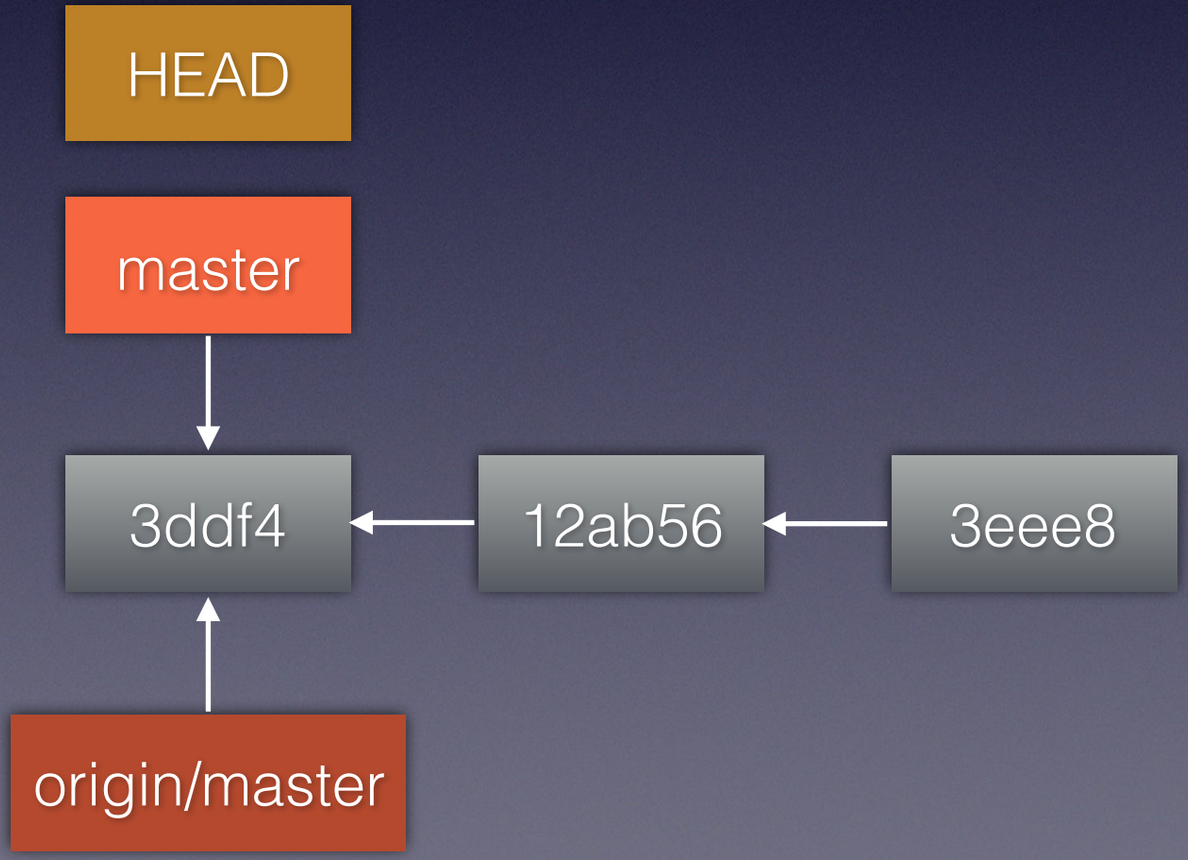
```
$ git remote -v
```

Github ❤ Git

REMOTE

The remote end is similar to a branch

```
$ git push origin master
```



HEAD: pointer to the current branch

Distributed Recipes

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

*recipes = workflow

Centralised recipe

Easy start: everyone works on the same branch

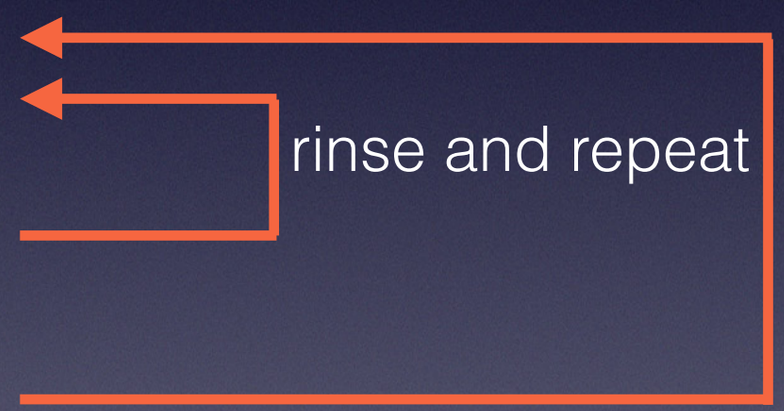
```
$ git pull origin master
```

Do some work

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

```
$ git commit -m “<message>”
```

```
$ git push origin master
```



Works until you get conflicts

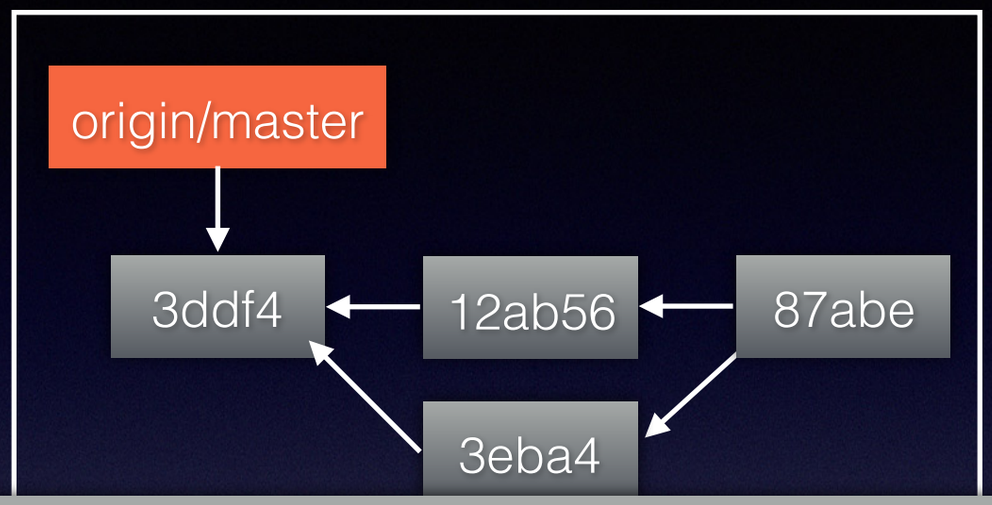
Conflicts in Centralised recipe

Github

@kikofernandez

$ git commit -am “stuff”

$ git push origin master

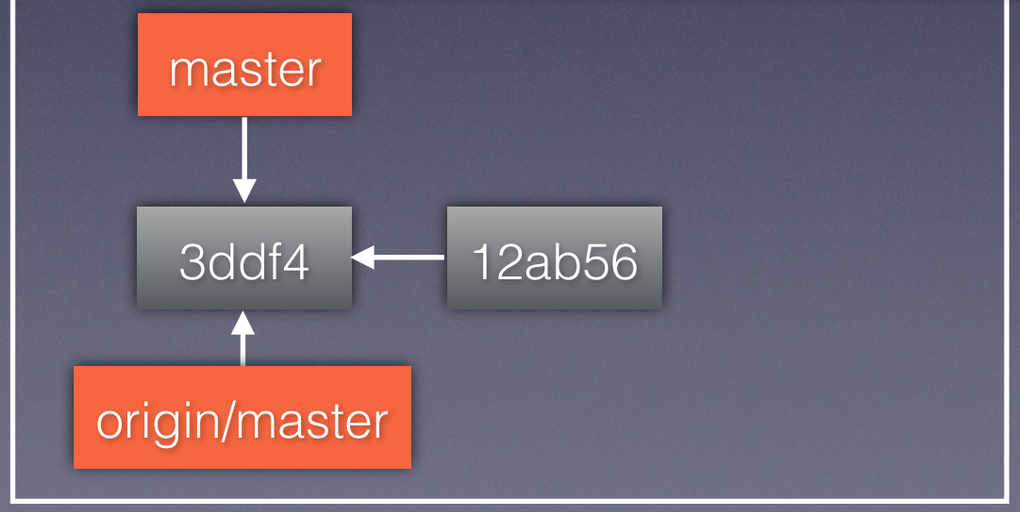


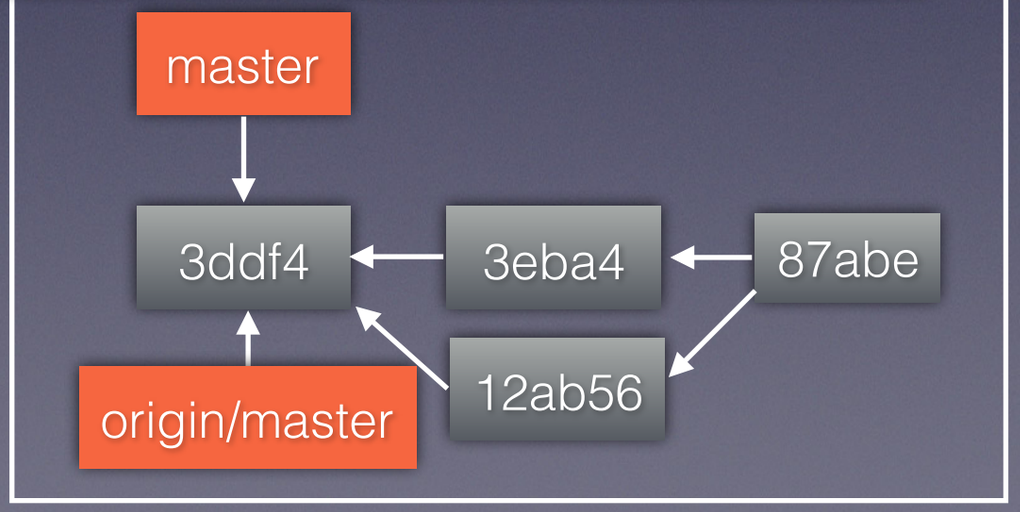
@supercooldave

$ git commit -am “work”

$ git push origin master

```
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.
```





Summary

Introduction to **git** and **Github**

&

Distributed workflows

so…

let’s **git** some fun

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

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

Fax: +46 18 511925

Email: kiko.fernandez@it.uu.se