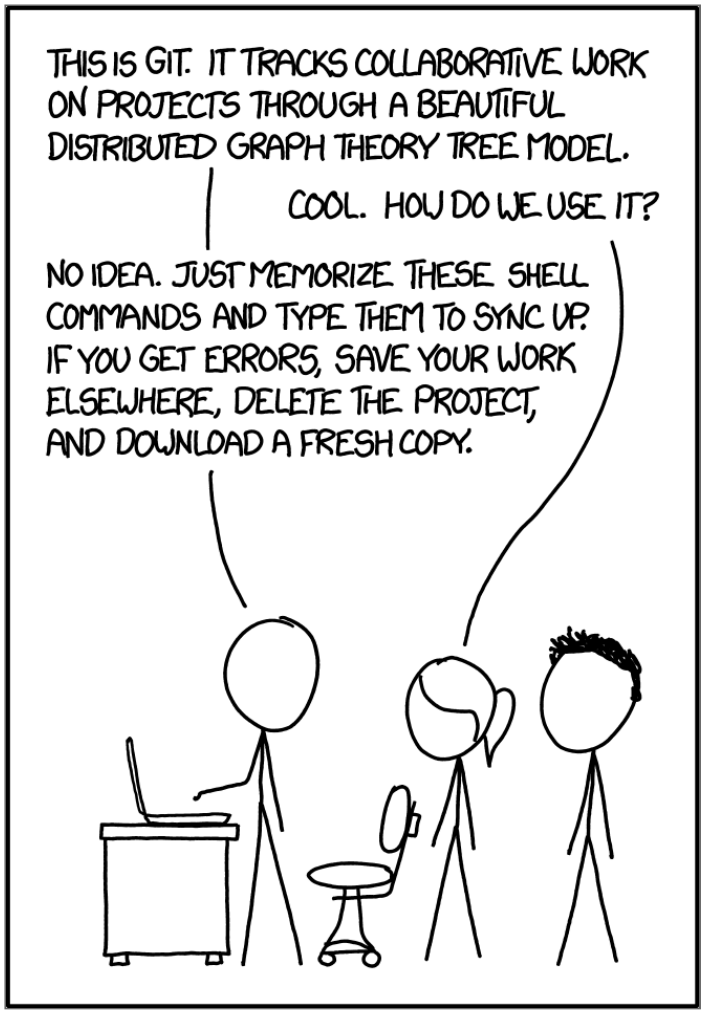
# GIT BASICS

## OR: HOW I LEARNED TO STOP WORRYING AND LOVE THE REBASE


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

# REQUIREMENTS

If you know how to type commands in a terminal and parse its output, this traning is made for you!

# THE TANGLED WORKING COPY PROBLEM

# BOOT YOUR NOTEBOOKS

# SLIDES

**https://escodebar.github.io/trainings/git/basics/**

# SETUP

**Install Git** if it is not already provided by your operating system.

# GIT HELP

## Display help information about Git

# GIT? HELP!

```
$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--b
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
    clone      Clone a repository into a new directory
    init       Create an empty Git repository or reinitialize an existing

work on the current change (see also: git help everyday)
    add        Add file contents to the index
    mv         Move or rename a file, a directory, or a symlink
    reset      Reset current HEAD to the specified state
    rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
    bisect     Use binary search to find the commit that introduced a bug
    grep       Print lines matching a pattern
    log        Show commit logs
    show       Show various types of objects
    status     Show the working tree status

grow, mark and tweak your common history
    branch     List, create, or delete branches
    checkout   Switch branches or restore working tree files
    commit     Record changes to the repository
    diff       Show changes between commits, commit and working tree, etc
```

# GIT CONFIG

## Get and set repository or global options

# CONFIGURE YOUR USERNAME AND EDITOR

## On your machine you may want to run the following

```
$ git config --global user.name 'Pablo Escodebar'
$ git config --global user.email 'escodebar@gmail.com'
```

## Then configure the editor you want to use

```
$ git config --global core.editor 'vim'
$ git config --global core.editor 'emacs'
$ git config --global core.editor 'subl -n -w'
$ git config --global core.editor 'atom --wait'
```

# GIT INIT

Create an empty Git repository or reinitialize an existing one.

# FIRST STEPS

## Create a repository:

```
$ mkdir -p ~/working/directory/ && cd $_ && git init .
Initialized empty Git repository in ~/working/directory/.gi
```

```
$ ls -blah
total 0
drwxr-xr-x 3 escodebar escodebar  60 Aug  2 10:39 .
drwxr-xr-x 3 escodebar escodebar  60 Aug  2 10:38 ..
drwxr-xr-x 7 escodebar escodebar 200 Aug  2 10:39 .git
```

## See that `.git` folder there? That's the repository.

# DIGGING DEEPER

## Don't panic!

```
$ ls -blah .git
total 12K
drwxr-xr-x 7 escodebar escodebar 200 Aug  2 10:39 .
drwxr-xr-x 3 escodebar escodebar  60 Aug  2 10:39 ..
drwxr-xr-x 2 escodebar escodebar  40 Aug  2 10:39 branches
-rw-r--r-- 1 escodebar escodebar  92 Aug  2 10:39 config
-rw-r--r-- 1 escodebar escodebar  73 Aug  2 10:39 descriptio
-rw-r--r-- 1 escodebar escodebar  23 Aug  2 10:39 HEAD
drwxr-xr-x 2 escodebar escodebar 260 Aug  2 10:39 hooks
drwxr-xr-x 2 escodebar escodebar  60 Aug  2 10:39 info
drwxr-xr-x 4 escodebar escodebar  80 Aug  2 10:39 objects
drwxr-xr-x 4 escodebar escodebar  80 Aug  2 10:39 refs
```

## This is deep enough for now!

# GITCEPTION

## Behold, run this in a separate terminal!

```
$ cd ~/working/directory/.git
```

## We are creating a repository inside the repository

```
$ git init . && git add . && git commit -m 'Add the repositor
Initialized empty Git repository in ~/working/directory/.git/
[master (root-commit) 1c2f932] Add the repository
 15 files changed, 653 insertions(+)
 create mode 100644 HEAD
 create mode 100644 config
 create mode 100644 description
 create mode 100755 hooks/applypatch-msg.sample
 [...]
 create mode 100755 hooks/update.sample
 create mode 100644 info/exclude
```

## Don't do this at home!


YO DAWG, I HEARD YOU LIKED GIT
SO I PUT A REPOSITORY IN YOUR REPOSITORY
SO YOU CAN GIT WHILE YOU GIT
imgflip.com

# BREAK

# GIT STATUS

Show the working tree status.

# WHAT'S THE STATUS?

## To get an overview of the repository run:

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to t
```

# LET'S DO SOMETHING!

## Documentation first!

```
$ echo '# My awwwesome training' > README.md
```

## What's the status now?

```
$ git status
On branch master

No commits yet

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

        README.md

nothing added to commit but untracked files present (use "git add" to
```

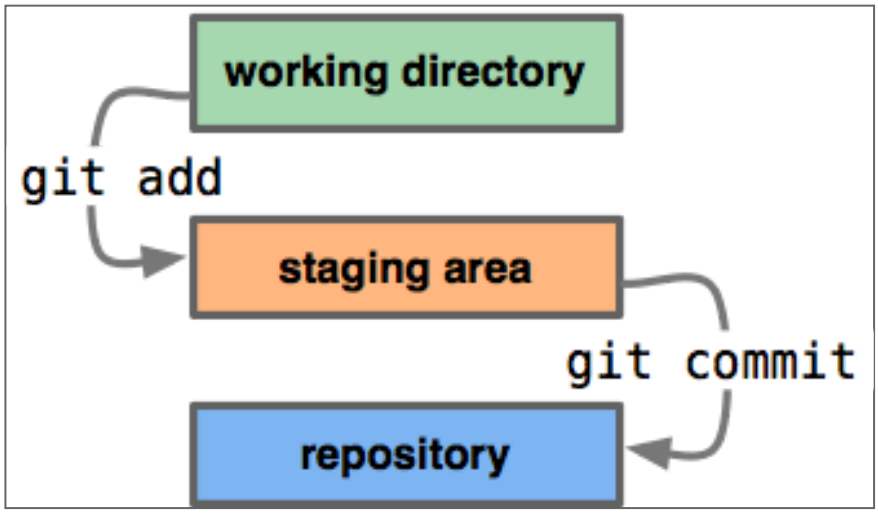
# GIT ADD

Add file contents to the index.

...but what's the index?

# THE INDEX

aka. *the staging area*




*"...is an intermediate area which allows to setup the change before making the commit."*

# LET'S STAGE!

## Put files in the staging area:

```
$ git add README.md && git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md
```

# WHAT HAPPENED IN THE REPOSITORY?

```
$ git add . && git commit -m 'Add files to index'
[master 6147d79] Add files to index
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index
 create mode 100644 objects/b2/7501ade65f39bc91a5e6eb0d707903ba2
```

# GIT CAT-FILE

Provide content or type and size information for repository objects.

# WHAT'S IN THE NEWLY CREATED OBJECT?

## Inspect the created object

```
$ git cat-file -t b27501a
blob
```

```
$ git cat-file -p b27501a
# My awwwesome training
```

# GIT DIFF

Show changes between commits, commit and working tree, etc.

# READY TO COMMIT?

See staged changes to check if they are ready to be committed:

```
$ git diff --cached
diff --git a/README.md b/README.md
new file mode 100644
index 0000000..b27501a
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+# My awwwesome training
```

# GIT COMMIT

Record changes to the repository.

# A COMMIT

aka. *a change*

"*...represents a complete version of your code.*"

# SAVE THE CHANGES

## Commit the changes to the repository

```
$ git commit -m 'Describe the training'
[master (root-commit) 78d7aa6] Describe the training
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

# TIME TO DIG DEEPER

## The repository's content must have changed, commit the changes and go back to the main repository

```
$ git add . && git commit -m 'Commit file'
[master 85240e8] Commit file
 7 files changed, 4 insertions(+)
 create mode 100644 COMMIT_EDITMSG
 create mode 100644 logs/HEAD
 create mode 100644 logs/refs/heads/master
 create mode 100644 objects/78/d7aa680e7ac5f3e851727ac29dd34afeb
 create mode 100644 objects/a4/4f211c376b94d122c6429ef8e87ffa785
 create mode 100644 refs/heads/master
```

# COMMIT OBJECTS!

## What is the object with the commit's hash?

```
$ git cat-file -t 78d7aa6
commit
```

```
$ git cat-file -p 78d7aa6
tree a44f211c376b94d122c6429ef8e87ffa7856419d
author Pablo Escodebar <escodebar@gmail.com> 1533199402 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1533199402 -

Describe the training
```

## ...so this is what a commit looks like!

# TREE OBJECTS!

## What is the object with the tree's hash?

```
$ git cat-file -t a44f211
tree
```

```
$ git cat-file -p a44f211
100644 blob b27501ade65f39bc91a5e6eb0d707903ba225a00    READ
```

## ...it's collection of references to objects!

# GIT SHOW

## Show various types of objects

# WHAT WAS THE LAST COMMIT?

## Take a look at a change using:

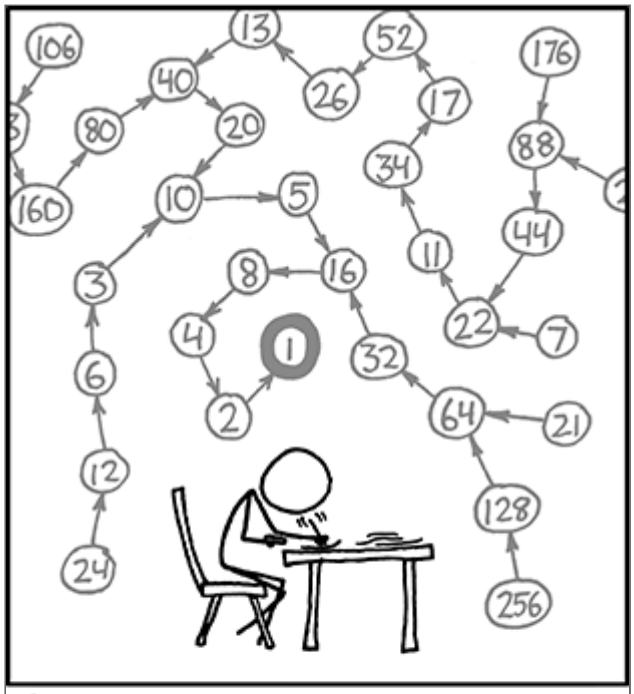
```
$ git show
commit 78d7aa680e7ac5f3e851727ac29dd34afeb766f6 (HEAD -> master)
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Thu Aug 2 10:43:22 2018 +0200

    Describe the training

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..b27501a
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+# My awwwesome training
```

# BREAK

# GRAPH THEORY




THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

# ADD IN PATCH MODE

## ...to select the changes you want to stage

```
$ echo 'This training will make you better!' >> README.md && git ad
diff --git a/README.md b/README.md
index b27501a..22d2d62 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
 # My awwwesome training
+This training will make you better!
Stage this hunk [y,n,q,a,d,/,e,?]?
```

```
$ git commit -m 'Motivate the participant'
[master 113b2fe] Motivate the participant
 1 file changed, 1 insertion(+)
```

## This is a great way to group your code!

# HOW DOES THE NEW COMMIT LOOK LIKE?

This second commit shouldn't be a root commit:

```
$ git cat-file -p 113b2fe
tree 10d06a676fb65acc4b1a2e57454039d904318393
parent 78d7aa680e7ac5f3e851727ac29dd34afeb766f6
author Pablo Escodebar <escodebar@gmail.com> 1534533899 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1534533899 -

Motivate the participant
```

...it has a parent!

# COMMIT IN PATCH MODE

## My favorite way of committing!

```
$ echo 'Buy me a beer if it made you better.' >> README.md
$ git commit -p -m 'Motivate the speaker'
diff --git a/README.md b/README.md
index 22d2d62..3f652ed 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
# My awwwesome training
 This training will make you better!
+Buy me a beer if it made you better.
Stage this hunk [y,n,q,a,d,/,e,?]?
```

## Once all hunks are decided, a commit will be created

```
[master a894a8e] Motivate the speaker
 1 file changed, 1 insertion(+)
```

# GIT LOG

## Show commit logs

# WHAT DID WE DO SO FAR?

Take a look back at your work using:

```
$ git log --oneline --abbrev-commit
a894a8e (HEAD -> master) Motivate the speaker
113b2fe Motivate the participant
78d7aa6 Describe the training
```

...so this is why we want short commit titles?

# COMMIT THE REPOSITORY'S CHANGES

## Add the new objects to the repository's repository:

```
$ git add . && git commit -m 'Add two more commits in patch mode'
[master 9b01029] Add two more commits
 11 files changed, 7 insertions(+), 2 deletions(-)
 create mode 100644 objects/10/d06a676fb65acc4b1a2e57454039d904318393
 create mode 100644 objects/11/3b2feae77a409de1c17d0c400490ac49b6a348
 create mode 100644 objects/22/d2d6223474b8b442b8aae05d4deab6f57a4a2a
 create mode 100644 objects/38/52d81df67551ce4174a25ce844cf690499f55c
 create mode 100644 objects/3f/652ededa8ed2a054ffa2c02bb34f99b53e94dd
 create mode 100644 objects/a8/94a8e197ea8e5a59323522ac9549a5f974f483
```

# BREAK

# GIT BRANCH

List, create, or delete branches

...but what's a branch?

# A BRANCH

aka. *a reference*

"*References are pointers to commits.*"

- simplify complex workflows.
- allow to group the logic of a feature.
- allow to work in parallel on several features.

# CREATE A BRANCH

## Branches are created using

```
$ git branch pe/new_branch
```

# DIGGING AGAIN!

## How are branches stored in the repository?

```
$ git add . && git commit -m 'Add a new branch'
[master ffce5ba] Add a new branch
 3 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 refs/heads/pe/new_branch
```

```
$ cat refs/heads/pe/new_branch
a894a8e197ea8e5a59323522ac9549a5f974f483
```

# REMEMBER GIT SHOW?

## A branch is just a file with the hash of a commit

```
$ git show a894a8e197ea8e5a59323522ac9549a5f974f483
commit a894a8e197ea8e5a59323522ac9549a5f974f483 (HEAD -> pe/new_branch, ma
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Thu Aug 2 10:46:30 2018 +0200

    Motivate the speaker

diff --git a/README.md b/README.md
index 22d2d62..3f652ed 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
 # My awwwesome training
 This training will make you better!
+Buy me a beer if it made you better.
```

## ...that's why creating branches is so fast!

# GIT CHECKOUT

## Switch branches or restore working tree files

# CHECKOUT THE NEWLY CREATED BRANCH

To checkout a branch, run:

```
$ git checkout pe/new_branch
Switched to branch 'pe/new_branch'
```

# CREATE AND CHECKOUT BRANCHES

# ...IN ONE STEP!

Check out a *new* branch using checkout:

```
$ git checkout -b pe/add_list_of_favorite_beers master
Switched to branch 'pe/add_list_of_favorite_beers'
```

...one command is faster than two!

# ADD A COMMIT TO THE NEW BRANCH

```
$ cat << EOBL > beers.md && git add beers.md
* To Øl - 1 ton of Happiness
* Rokki - Muikea
* Felsenau - Bärner Müntschi
* Rokki - Happo
* Egger - Galopper
EOBL
$ echo 'My list of [favorite beers](beers.md).' >> README.md
$ git commit -a -m 'Let people know, what beer to buy'
[pe/add_list_of_favorite_beers 000ce0a] Let people know, what beer
 2 files changed, 6 insertions(+)
 create mode 100644 beers.md
```

# CREATE ANOTHER BRANCH

## ...with another commit

```
$ git checkout -b pe/whiskey_is_also_an_option master
Switched to branch 'pe/whiskey_is_also_an_option'
```

```
$ echo 'Whiskey is also a good reward.' >> README.md
$ cat << EOWL > whiskeys.md && git add whiskeys.md
* Lagavulin - 16
* Ledaig - 10
* Talisker - Storm
* Ledaig - 18
* Laphroaig - Quarter Cask
EOWL
$ echo '[These whiskeys](whiskeys.md) are great!' >> README
$ git commit -a -m 'Accept whiskey as reward'
[pe/whiskey_is_also_an_option 68f2339] Accept whiskey as rev
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

# WHAT A BEAUTIFUL TREE

Take a look at the graph of the repository using:

```
$ git log --oneline --abbrev-commit --all --graph
* 68f2339 (HEAD -> pe/whiskey_is_also_an_option) Accept whiskey as rewa
| * 000ce0a (pe/add_list_of_favorite_beers) Let people know, what beer
|/
* a894a8e (pe/new_branch, master) Motivate the speaker
* 113b2fe Motivate the participant
* 78d7aa6 Describe the training
```

Our tree starts growing branches!

# CLEAN UP!

## We do not want to have uncommitted changes!

```
$ git add . && git commit -m 'Add branches with commits'
[master ccc7056] Add braches with commits
 17 files changed, 20 insertions(+), 12 deletions(-)
 create mode 100644 logs/refs/heads/pe/add_list_of_favorite_beers
 create mode 100644 logs/refs/heads/pe/new_branch
 create mode 100644 logs/refs/heads/pe/whiskey_is_also_an_option
 create mode 100644 objects/00/0ce0a9703aebd0722e2ac3f285985b6b22
 create mode 100644 objects/0d/f4281955475551ad1a4232fce76a5fb6d3
 create mode 100644 objects/21/990ee9610d1601649ca9c669f7f51ecad5
 create mode 100644 objects/68/f2339b23c674d3b288411a975499114588
 create mode 100644 objects/7e/c764e3ac5af8a360fc2df5ac5c58aa5bff
 create mode 100644 objects/9c/8d69a8414db1654a6c725de0c670fa28df
 create mode 100644 objects/a2/8e0af61a8785cfec49e2ea707f8172d4b9
 create mode 100644 objects/d3/719373bb86bdd46c56e521135a1bd7f69d
 create mode 100644 refs/heads/pe/add_list_of_favorite_beers
 create mode 100644 refs/heads/pe/whiskey_is_also_an_option
```

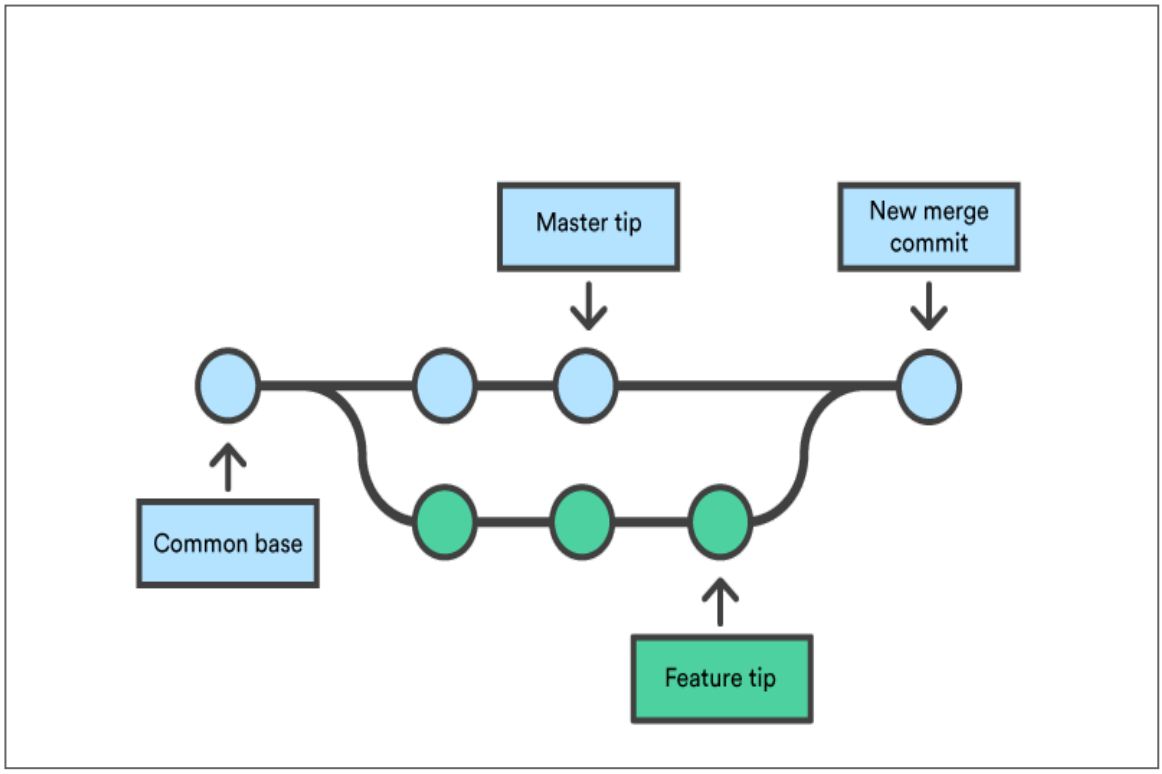
# BREAK

# ADDING FEATURES OF A BRANCH TO ANOTHER BRANCH

# WAIT... WHAT?

# CONFLICTS?!

# GIT MERGE




Join development histories

# LET'S MERGE

## Checkout a new branch for the merge

```
$ git checkout -b pe/merging pe/add_list_of_favorite_beers
Switched to a new branch 'pe/merging'
```

## Merge...

```
$ git merge pe/whiskey_is_also_an_option
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the re
```

## ...and run into conflicts!

# WHAT'S THE STATUS?

Let's take a look at the status:

```
$ git status
On branch pe/merging
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:

        new file:   whiskeys.md

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

        both modified:   README.md
```

As expected, a file was modified by both branches!

# MERGE CONFLICTS ARE FUN!

## How does Git handle merge conflicts?

```
$ git add . && git commit -m 'Commit during merge conflict'
[master cc2485f] Commit during merge conflict
 10 files changed, 14 insertions(+), 1 deletion(-)
 create mode 100644 MERGE_HEAD
 create mode 100644 MERGE_MODE
 create mode 100644 MERGE_MSG
 create mode 100644 ORIG_HEAD
 rewrite index (100%)
 create mode 100644 logs/refs/heads/pe/merging
 create mode 100644 objects/74/53e34d766307d5056d804f80e4cc2395fb
 create mode 100644 refs/heads/pe/merging
```

## A further object?

# TAKE A LOOK AT THAT OBJECT!

## We are getting used to this!

```
$ git cat-file -t 7453e34
blob
$ git cat-file -p 7453e34
# My awwwesome training
This training will make you better!
Buy me a beer if it made you better.
<<<<<<< HEAD
My list of [favorite beers](beers.md).
=======
Whiskey is also a good reward.
[These whiskeys](whiskeys.md) are great!
>>>>>>> pe/whiskey_is_also_an_option
```

## Looks like the README file

# CONFLICT RESOLUTION

## Take a look at the conflicting files:

```
$ git diff
diff --cc README.md
index d371937,a28e0af..0000000
--- a/README.md
+++ b/README.md
@@@ -1,4 -1,5 +1,9 @@@
  # My awwwesome training
  This training will make you better!
  Buy me a beer if it made you better.
++<<<<<<< HEAD
 +My list of [favorite beers](beers.md).
++=======
+ Whiskey is also a good reward.
+ [These](whiskeys.md) are great!
++>>>>>>> pe/whiskey_is_also_an_option
```

## This conflict is easily solved!

# FINISH MERGING

## ...once you resolved the conflicts:

```
$ git add README.md
$ git commit -m 'Add the list of beers first'
[pe/merging 3a1f82c] Add the list of beers first
```

## That was easy!

# TAKE A LOOK AT THE MERGE COMMIT

## Merge commits are special...

```
$ git cat-file -t 3a1f82c
commit
$ git cat-file -p 3a1f82c
tree d5a29e72348dd06004654c605f561d7d6fc32e6c
parent 000ce0a9703aebd0722e2ac3f285985b6b223312
parent 68f2339b23c674d3b288411a9754991145883e56
author Pablo Escodebar <escodebar@gmail.com> 1534559548 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1534559548 -

Add the list of beers first
```

## ...since they have more than one parent!

# CLEAN UP!

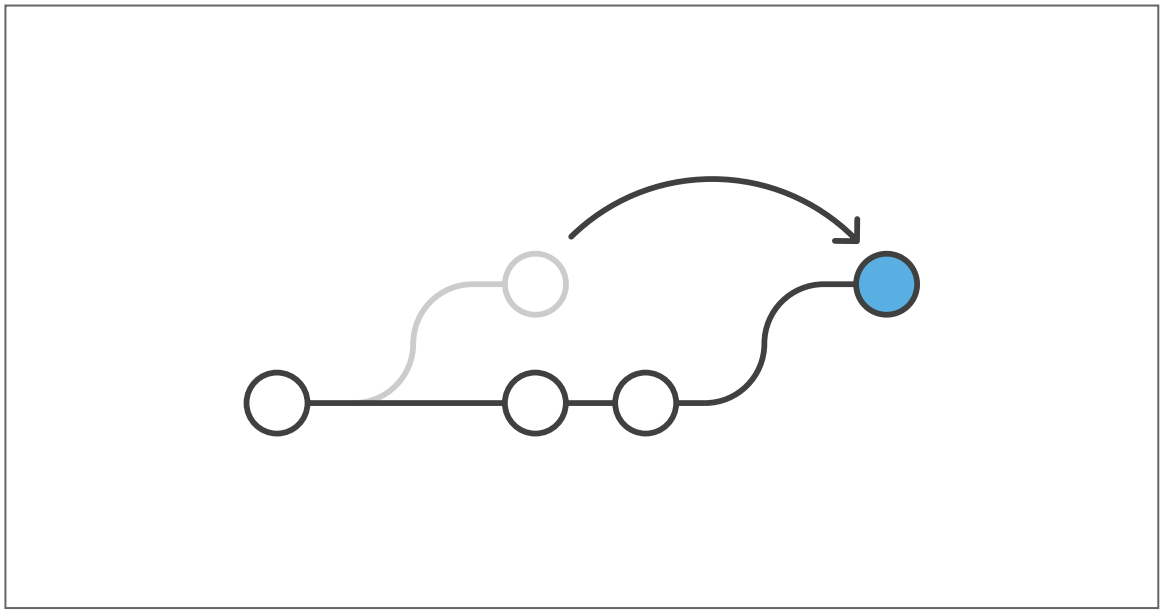
## Commit the changes into the repository's repository

```
$ git add . && git commit -m 'Add the merge'
[master 5dbb9d5] Add the merge
 11 files changed, 6 insertions(+), 7 deletions(-)
 delete mode 100644 MERGE_HEAD
 delete mode 100644 MERGE_MODE
 delete mode 100644 MERGE_MSG
 rewrite index (100%)
 create mode 100644 objects/3a/1f82cf3b89824878b6844f5752a0b28ab
 create mode 100644 objects/93/d56bde8cd7e1ac44d1f4f454a189b71b7
 create mode 100644 objects/d5/a29e72348dd06004654c605f561d7d6fc
```

## The merge files are gone!

# BREAK

# GIT CHERRY-PICK



Apply the changes introduced by some existing commits

# PICK A CHERRY

## Let's add another branch for cherry picking

```
$ git checkout -b pe/cherry_picking pe/add_list_of_favorite_
Switched to branch 'pe/cherry_picking'
```

## Find the hash of the cherry (commit) to be picked

```
$ git log --oneline --abbrev-commit pe/whiskey_is_also_an_o
68f2339 (pe/whiskey_is_also_an_option) Accept whiskey as rev
a894a8e (pe/new_branch, master) Motivate the speaker
113b2fe Motivate the participant
78d7aa6 Describe the training
```

# ...then pick it up!

```
$ git cherry-pick 68f2339
error: could not apply 68f2339... Accept whiskey as reward
hint: after resolving the conflicts, mark the corrected patl
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
```

# ENJOY CHERRY PICK CONFLICTS!

## Dig, dig, dig, dig

```
$ git add . && git commit -m 'Commit a cherry pick conflict'
[master 8bb838d] Commit during cherry pick conflict
 8 files changed, 11 insertions(+), 1 deletion(-)
 create mode 100644 CHERRY_PICK_HEAD
 create mode 100644 MERGE_MSG
 rewrite index (100%)
 create mode 100644 logs/refs/heads/pe/cherry_picking
 create mode 100644 objects/77/047a805c055408b0f9a6ef2a96ef932d677
 create mode 100644 refs/heads/pe/cherry_picking
```

## Another object!

# FINISH CHERRY-PICKING

## ...once you resolved the conflicts:

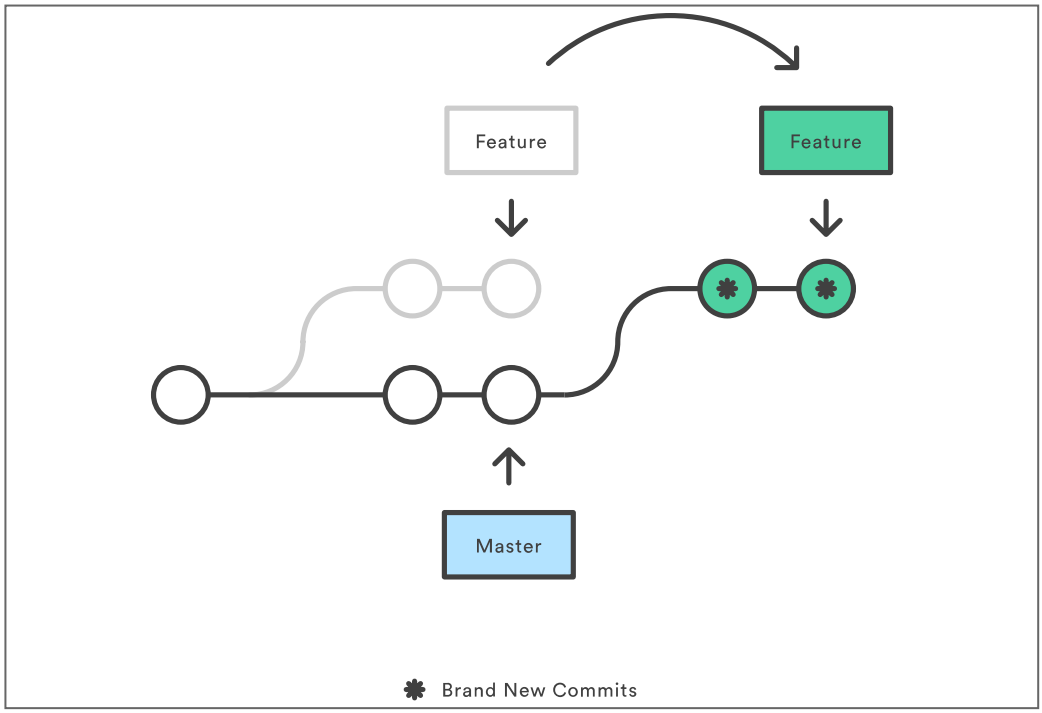
```
$ git add README.md
$ git cherry-pick --continue
[pe/cherry_picking b7083bf] Accept whiskey as reward
 Date: Fri Jun 29 19:47:30 2018 +0200
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

# CLEAN UP!

## Once again...

```
$ git add . && git commit -m 'Add the cherry-pick'
[master a5f2863] Add the cherry-pick
 8 files changed, 26 insertions(+), 7 deletions(-)
 delete mode 100644 CHERRY_PICK_HEAD
 rewrite COMMIT_EDITMSG (100%)
 delete mode 100644 MERGE_MSG
 rewrite index (100%)
 create mode 100644 objects/b7/083bff4b2604a3d5669cdf73ed3629d53b
```

# GIT REBASE




*Reapply commits on top of another branch*

# REBASE YOURSELF!

```
$ git checkout -b pe/rebasing pe/whiskey_is_also_an_option
Switched to branch 'pe/rebasing'
```

```
$ git rebase pe/add_list_of_favorite_beers
First, rewinding head to replay your work on top of it...
Applying: Let people know, what beer to buy
Using index info to reconstruct a base tree...
M       README.md
Falling back to patching base and 3-way merge...
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: Failed to merge in the changes.
Patch failed at 0001 Let people know, what beer to buy
The copy of the patch that failed is found in: .git/rebase-apply/patch

Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --ab
```

# REBASING

## ...is slightly more complicated:

```
$ git add . && git commit -m 'Commit a rebase conflict'
[master cd7d697] Commit a rebase conflict
 29 files changed, 80 insertions(+), 2 deletions(-)
 create mode 100644 REBASE_HEAD
 rewrite index (100%)
 create mode 100644 logs/refs/heads/pe/rebasing
 create mode 100644 objects/85/c2d4e6fc7a00c7e49bd1d851f54863fd3
 create mode 100644 rebase-apply/0001
 create mode 100644 rebase-apply/abort-safety
 create mode 100644 rebase-apply/apply-opt
 create mode 100644 rebase-apply/author-script
 create mode 100644 rebase-apply/final-commit
 create mode 100644 rebase-apply/head-name
 create mode 100644 rebase-apply/keep
 create mode 100644 rebase-apply/last
 create mode 100644 rebase-apply/messageid
 create mode 100644 rebase-apply/next
 create mode 100644 rebase-apply/onto
 create mode 100644 rebase-apply/orig-head
 create mode 100644 rebase-apply/original-commit
 create mode 100644 rebase-apply/patch
 create mode 100644 rebase-apply/patch-merge-index
 create mode 100644 rebase-apply/quiet
 create mode 100644 rebase-apply/rebasing
 create mode 100644 rebase-apply/scissors
```

# FINISH REBASING

## ...once you resolved the conflicts:

```
$ git add README.md && git rebase --continue
Applying: Accept whiskey as reward
```

# WHAT ABOUT THE REBASE FILES?

```
$ git add . && git commit -m 'Add the rebase'
[master 46da891] Add the rebase
 28 files changed, 5 insertions(+), 76 deletions(-)
 delete mode 100644 REBASE_HEAD
 rewrite index (100%)
 create mode 100644 objects/c9/6224a8837cfd2996396e0e523032e09eb
 delete mode 100644 rebase-apply/0001
 delete mode 100644 rebase-apply/abort-safety
 delete mode 100644 rebase-apply/apply-opt
 delete mode 100644 rebase-apply/author-script
 delete mode 100644 rebase-apply/final-commit
 delete mode 100644 rebase-apply/head-name
 delete mode 100644 rebase-apply/keep
 delete mode 100644 rebase-apply/last
 delete mode 100644 rebase-apply/messageid
 delete mode 100644 rebase-apply/next
 delete mode 100644 rebase-apply/onto
 delete mode 100644 rebase-apply/orig-head
 delete mode 100644 rebase-apply/original-commit
 delete mode 100644 rebase-apply/patch
 delete mode 100644 rebase-apply/patch-merge-index
 delete mode 100644 rebase-apply/quiet
 delete mode 100644 rebase-apply/rebasing
 delete mode 100644 rebase-apply/scissors
 delete mode 100644 rebase-apply/sign
```

They are gone!

# BREAK

# REBASING IN INTERACTIVE MODE

The interactive mode allows to "change" the commit history before rebasing!


COMMENT
DATE
CREATED MAIN LOOP & TIMING CONTROL
14 HOURS AGO
ENABLED CONFIG FILE PARSING
9 HOURS AGO
MISC BUGFIXES
5 HOURS AGO
CODE ADDITIONS/EDITS
4 HOURS AGO
MORE CODE
4 HOURS AGO
HERE HAVE CODE
4 HOURS AGO
AAAAAAAA
3 HOURS AGO
ADKFJSLKDFJSDKLFJ
3 HOURS AGO
MY HANDS ARE TYPING WORDS
2 HOURS AGO
HAAAAAAAANDS
2 HOURS AGO
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

# PREPARING THE INTERACTIVE REBASE

## Add a branch with a few commits

```
$ git checkout -b pe/interactive_rebasing pe/rebasing
$ echo 'I would also love some feedback.' >> README.md
$ git commit -am 'Ask for feedback'
[pe/interactive_rebasing c16a824] Ask for feedback
 1 file changed, 1 insertion(+)
$ echo 'Personal feedback is the best.' >> README.md
$ git commit -am 'Ask for personal feedback'
[pe/interactive_rebasing 450dc77] Ask for personal feedback
 1 file changed, 1 insertion(+)
$ echo 'Helpful feedback is awarded with great coffee.' >> READ
$ git commit -am 'Trade feedback for coffee'
[pe/interactive_rebasing 1603ef5] Trade feedback for coffee
 1 file changed, 1 insertion(+)
```

# REEEEBAAAASE!

Start the interactive rebase with the `-i` flag:

```
$ git rebase -i pe/rebasing
```

Your editor now lists all the commits of your branch!

```
pick c16a824 Ask for feedback
pick 450dc77 Ask for personal feedback
pick 1603ef5 Trade feedback for coffee

# Rebase c96224a..1603ef5 onto c96224a (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log m
# x, exec <command> = run command (the rest of the line) using shell
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to botto
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
#   However, if you remove everything, the rebase will be aborted.
#
```

# COMMIT THE CHANGES IN THE REPOSITORY'S REPOSITORY

```
$ git add . && git commit -m 'Add interactive rebase'
[master f1fc09c] Add interactive rebase
 17 files changed, 32 insertions(+), 25 deletions(-)
 rewrite COMMIT_EDITMSG (100%)
 create mode 100644 logs/refs/heads/pe/interactive_rebasing
 create mode 100644 objects/01/9ece38ca9ad05589d46c853faf4d24bb6
 [...]
 create mode 100644 objects/c3/2f3348d4d95fed6ff7c80054daf3690c5
 create mode 100644 refs/heads/pe/interactive_rebasing
```

# BREAK

# TAKE A LOOK AT YOUR WORK

```
$ git log --oneline --abbrev-commit --all --graph
* 019ece3 (HEAD -> pe/interactive_rebasing) Ask for feedback
* c96224a (pe/rebasing) Accept whiskey as reward
| * b7083bf (pe/cherry_picking) Accept whiskey as reward
|/
| *   3a1f82c (pe/merging) Add the list of beers first
| |\
|/ /
| * 68f2339 (pe/whiskey_is_also_an_option) Accept whiskey as reward
* | 000ce0a (pe/add_list_of_favorite_beers) Let people know, what beer
|/
* a894a8e (pe/new_branch, master) Motivate the speaker
* 113b2fe Motivate the participant
* 78d7aa6 Describe the training
```

The graph is now more complex!

# GIT RESET

## Reset current HEAD to the specified state

# UPDATE YOUR MASTER BRANCH

```
$ git checkout master && git reset --hard pe/interactive_reba
HEAD is now at 019ece3 Ask for feedback
```

# RESOURCES

- **Git Magic**
- **Git Reference**
- **Git is simpler**
- **Oh shit Git!**
- **Pro Git**
- **The thing about Git**
- **Think like a Git**
- **Why Git is Better than X**

# QUESTIONS & FEEDBACK

# THANK YOU!