DR. GIT-LOVE

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

A FEW WORDS ABOUT

THIS TRAINING

SCHEDULE

- Gitception: Reaching into the substructure
- Of trees, branches and pieces of fruit
- It's a backup system... It's a patch system... It's Git!

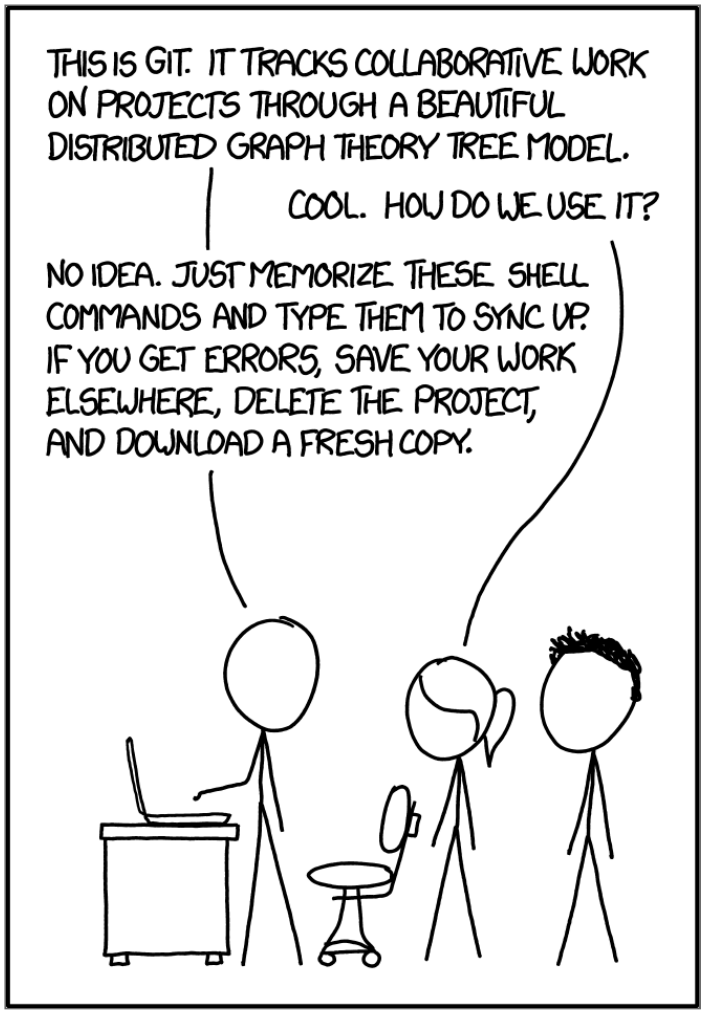
REQUIREMENTS

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

SLIDES

https://escodebar.github.io/trainings/git/meetup/

ARE YOU READY?


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.

SETUP

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 Sep 18 14:08 .
drwx------ 3 escodebar escodebar  60 Sep 18 14:08 ..
drwxr-xr-x 7 escodebar escodebar 200 Sep 18 14:08 .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 Sep 18 14:08 .
drwxr-xr-x 3 escodebar escodebar  60 Sep 18 14:08 ..
drwxr-xr-x 2 escodebar escodebar  40 Sep 18 14:08 branches
-rw-r--r-- 1 escodebar escodebar  92 Sep 18 14:08 config
-rw-r--r-- 1 escodebar escodebar  73 Sep 18 14:08 descriptio
-rw-r--r-- 1 escodebar escodebar  23 Sep 18 14:08 HEAD
drwxr-xr-x 2 escodebar escodebar 260 Sep 18 14:08 hooks
drwxr-xr-x 2 escodebar escodebar  60 Sep 18 14:08 info
drwxr-xr-x 4 escodebar escodebar  80 Sep 18 14:08 objects
drwxr-xr-x 4 escodebar escodebar  80 Sep 18 14:08 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 -A && git commit -m "Add the reposito
[master (root-commit) e2d03aa] 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

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!

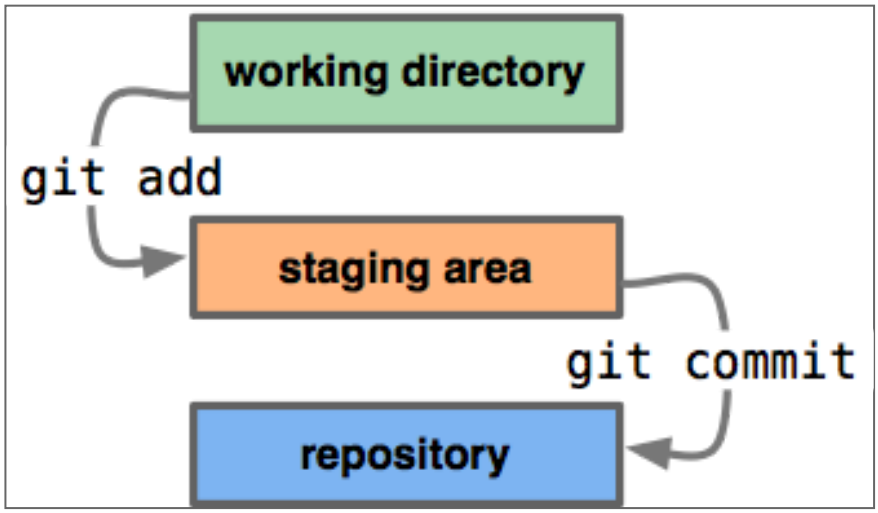
```
$ echo "# My awwwesome training" > README.md
```

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
```

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 9d4baf6] Add files to index
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index
 create mode 100644 objects/b2/7501ade65f39bc91a5e6eb0d707903ba2
```

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
```

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
```

Are you ready to commit them?

A COMMIT

aka. *a change*

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

SAVE THE CHANGES

...by commiting them to the repository

```
$ git commit -m "Describe the training"
[master (root-commit) b4aec28] Describe the training
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

TIME TO DIG DEEPER

The repository's content must have changed...

```
$ git add . && git commit -m "Commit file"
[master d775dc2] Commit file
 7 files changed, 8 insertions(+)
 create mode 100644 COMMIT_EDITMSG
 create mode 100644 logs/HEAD
 create mode 100644 logs/refs/heads/master
 create mode 100644 objects/a4/4f211c376b94d122c6429ef8e87ffa785
 create mode 100644 objects/b4/aec28dcbca565485e754378a4e5b4f610
 create mode 100644 refs/heads/master
```

COMMIT OBJECTS!

What is the object with the commit's hash?

```
$ git cat-file -t b4aec28
commit
```

```
$ git cat-file -p b4aec28
tree a44f211c376b94d122c6429ef8e87ffa7856419d
author Pablo Escodebar <escodebar@gmail.com> 1568808496 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1568808496 -

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!

WHAT WAS THE LAST COMMIT?

Take a look at a change using:

```
$ git show
commit b4aec28dcbca565485e754378a4e5b4f610221c1
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:16 2019 +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
```

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 b05598c] 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 b05598c
tree 10d06a676fb65acc4b1a2e57454039d904318393
parent b4aec28dcbca565485e754378a4e5b4f610221c1
author Pablo Escodebar <escodebar@gmail.com> 1568808497 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1568808497 -

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 bc03912] Motivate the speaker
 1 file changed, 1 insertion(+)
```

WHAT DID WE DO SO FAR?

Take a look back at your work using:

```
$ git log --oneline
bc03912 Motivate the speaker
b05598c Motivate the participant
b4aec28 Describe the training
```

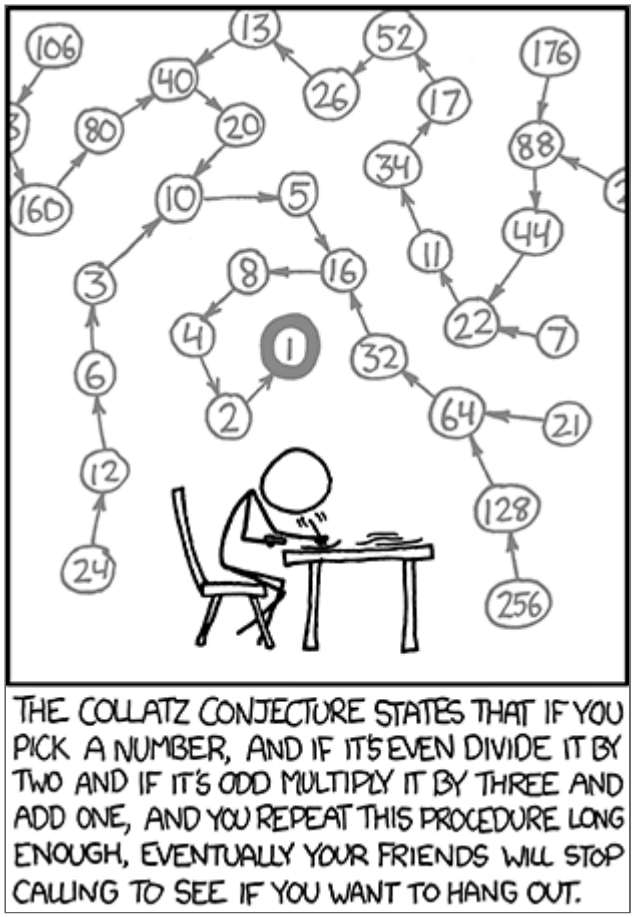
...so this is why we want short commit titles?

COMMIT THE REPOSITORY'S CHANGES

Add the new objects to the repository's repository:

```
$ git add . && git commit -m "Use the patch mode"
[master 8f45085] Use the patch mode
 11 files changed, 8 insertions(+), 2 deletions(-)
 create mode 100644 objects/10/d06a676fb65acc4b1a2e57454039d904318393
 create mode 100644 objects/22/d2d6223474b8b442b8aae05d4deab6f57a4a2a
 create mode 100644 objects/38/52d81df67551ce4174a25ce844cf690499f55c
 create mode 100644 objects/3f/652ededa8ed2a054ffa2c02bb34f99b53e94dd
 create mode 100644 objects/b0/5598cc258916a3e46724a0b573875c0017b390
 create mode 100644 objects/bc/03912a337adecda41bba5fddc4bb48b2308e9f
```

TIME FOR QUESTIONS



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.

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 a40ed00] Add a new branch
 2 files changed, 2 insertions(+)
 create mode 100644 logs/refs/heads/pe/new_branch
 create mode 100644 refs/heads/pe/new_branch
```

```
$ cat refs/heads/pe/new_branch
bc03912a337adecda41bba5fddc4bb48b2308e9f
```

It's just a file with a hash

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

REMEMBER GIT SHOW?

Let's see what the hash is

```
$ git show bc03912a337adecda41bba5fddc4bb48b2308e9f
commit bc03912a337adecda41bba5fddc4bb48b2308e9f
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:17 2019 +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.
```

A branch is just a hash of a commit

NOW LET'S CHECK, CHECK, CHECK IT OUT!

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

... to add further commit to it!

WHAT HAPPENED IN THE REPO?

```
$ git add . && git commit -m "Check out the branch"
[master ca534e6] Check out the branch
 2 files changed, 2 insertions(+), 1 deletion(-)
```

2 files changed... but what changed?

LET'S TAKE A CLOSER LOOK

`git show` comes with many options:

```
$ git show --name-only
commit ca534e60c872c46405eac1600f4ffbe76d534f0d
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:17 2019 +0200

    Check out the branch

HEAD
logs/HEAD
```

...ah! the `HEAD` changed!

CREATE AND CHECKOUT BRANCHES

...IN ONE STEP!

Switch to a *new* branch using checkout:

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

...one command is faster than two!

WAIT... WHAT?

CONFLICTS?!

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 2a81529] Let people know, what beer to
 2 files changed, 6 insertions(+)
 create mode 100644 beers.md
```

CREATE ANOTHER BRANCH

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

...with another commit

```
$ 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 -am "Accept whiskey as reward"
[pe/whiskey_is_also_an_option bf4d8e2] Accept whiskey as rew
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

CLEAN UP!

We do not want to have uncommitted changes

```
$ git add . && git commit -m "Add branches with conflicting commi
[master 6c9cd77] Add branches with conflicting commits
 16 files changed, 17 insertions(+), 2 deletions(-)
 create mode 100644 logs/refs/heads/pe/add_list_of_favorite_beers
 create mode 100644 logs/refs/heads/pe/whiskey_is_also_an_option
 create mode 100644 objects/0d/f4281955475551ad1a4232fce76a5fb6d3
 create mode 100644 objects/21/990ee9610d1601649ca9c669f7f51ecad5
 create mode 100644 objects/2a/81529a13736bd7c79f6e45d27a47b89789
 create mode 100644 objects/7e/c764e3ac5af8a360fc2df5ac5c58aa5bff
 create mode 100644 objects/9c/8d69a8414db1654a6c725de0c670fa28df
 create mode 100644 objects/a2/8e0af61a8785cfec49e2ea707f8172d4b9
 create mode 100644 objects/bf/4d8e2953b9d8067fc3a67d9831c7720e9b
 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
```

WHAT A BEAUTIFUL TREE

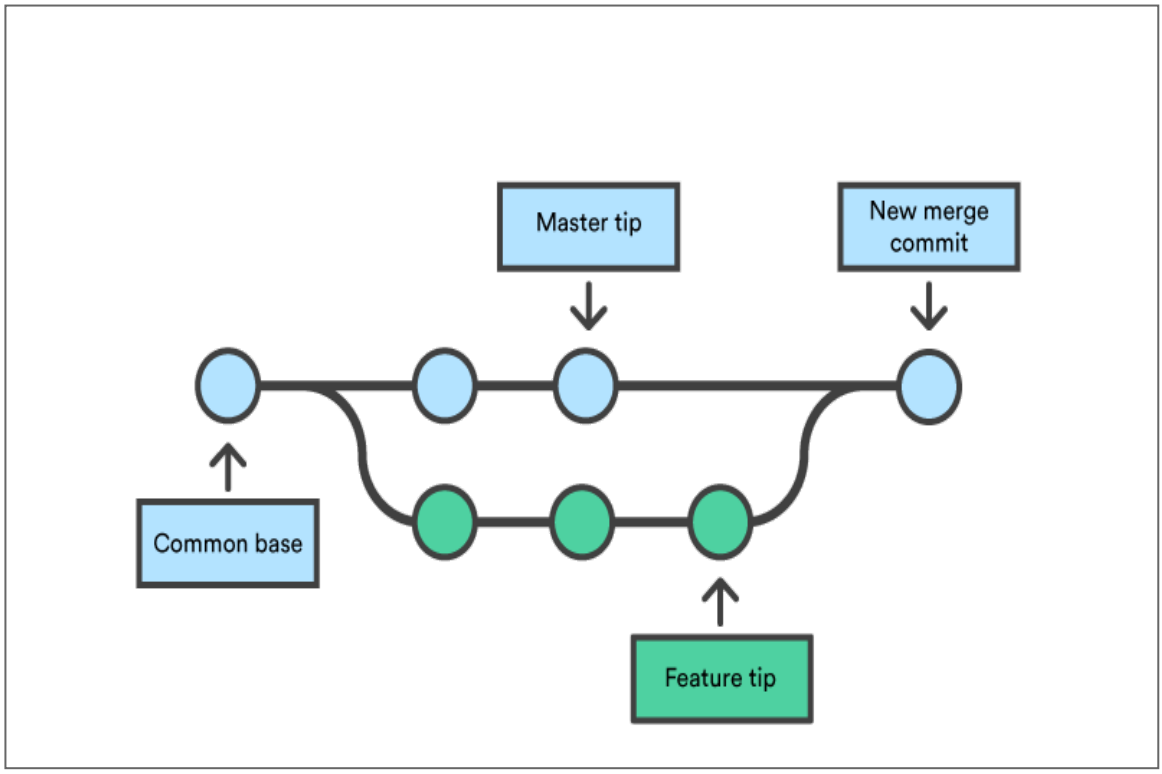
Take a look at the graph of the repository using:

```
$ git log --oneline --all --graph
* 2a81529 Let people know, what beer to buy
| * bf4d8e2 Accept whiskey as reward
|/
* bc03912 Motivate the speaker
* b05598c Motivate the participant
* b4aec28 Describe the training
```

Our tree starts growing branches!


I'M HEARING THAT NONE OF YOUR CODE HAS BEEN IMPLEMENTED. WHY ARE YOU SO UNPRODUCTIVE?
DILBERT.COM @SCOTTADAMSSAYS
YOUR NEW LEAD DEVELOPER DOESN'T KNOW HOW TO USE GIT AND HE KEEPS OVERWRITING MY PATCHES.
4-18-19 2019 Scott Adams, Inc./Dist. by Andrews McMeel
I DON'T KNOW WHAT ANY OF THAT MEANS.
WELL, THANK YOU FOR STOPPING BY.

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!

MERGE CONFLICTS ARE FUN!

How does Git handle merge conflicts?

```
$ git add . && git commit -m "Commit during merge conflict"
[master 9b3b168] 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...

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!

UNDERSTANDING THE CONFLICT

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](whiskeys.md) are great!
++>>>>>>> pe/whiskey_is_also_an_option
```

This conflict is easily solved!

CONFLICT RESOLUTION

Just remove the 4th, 6th and last line.

```
$ sed -i '4d;6d;$d' README.md
```

```
$ cat README.md
# My awwwesome training
This training will make you better!
Buy me a beer if it made you better.
My list of [favorite beers](beers.md).
Whiskey is also a good reward.
[These whiskeys](whiskeys.md) are great!
```

Use your favorite editor to do so!

FINISH MERGING

...once you resolved the conflicts:

```
$ git add README.md
```

and run

```
$ git merge --continue
```

to open your editor to write the commit's message

or commit yourself directly

```
$ git commit -m "Add the list of beers first"
[pe/merging 5d19f7c] Add the list of beers first
```

That was easy!

TAKE A LOOK AT THE MERGE COMMIT

Merge commits are special...

```
$ git cat-file -t 5d19f7c
commit
```

```
$ git cat-file -p 5d19f7c
tree d5a29e72348dd06004654c605f561d7d6fc32e6c
parent 2a81529a13736bd7c79f6e45d27a47b89789fa30
parent bf4d8e2953b9d8067fc3a67d9831c7720e9b749e
author Pablo Escodebar <escodebar@gmail.com> 1568808497 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1568808497 -

Add the list of beers first
```

...since they have more than one parent!

MEET THE PARENTS

```
$ git show HEAD^1
commit 2a81529a13736bd7c79f6e45d27a47b89789fa30
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:17 2019 +0200

    Let people know, what beer to buy

diff --git a/README.md b/README.md
index 3f652ed..d371937 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,4 @@
 # My awwwesome training
 This training will make you better!
 Buy me a beer if it made you better.
+My list of [favorite beers](beers.md).
diff --git a/beers.md b/beers.md
new file mode 100644
index 0000000..7ec764e
--- /dev/null
+++ b/beers.md
@@ -0,0 +1,5 @@
+* To Øl - 1 ton of Happiness
+* Rokki - Muikea
+* Felsenau - Bärner Müntschi
```

MEET THE PARENTS 2

```
$ git show HEAD^2
commit bf4d8e2953b9d8067fc3a67d9831c7720e9b749e
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:17 2019 +0200

    Accept whiskey as reward

diff --git a/README.md b/README.md
index 3f652ed..a28e0af 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,5 @@
 # My awwwesome training
 This training will make you better!
 Buy me a beer if it made you better.
+Whiskey is also a good reward.
+[These whiskeys](whiskeys.md) are great!
diff --git a/whiskeys.md b/whiskeys.md
new file mode 100644
index 0000000..0df4281
--- /dev/null
+++ b/whiskeys.md
@@ -0,0 +1,5 @@
+* Lagavulin - 16
+* Ledaig - 10
```

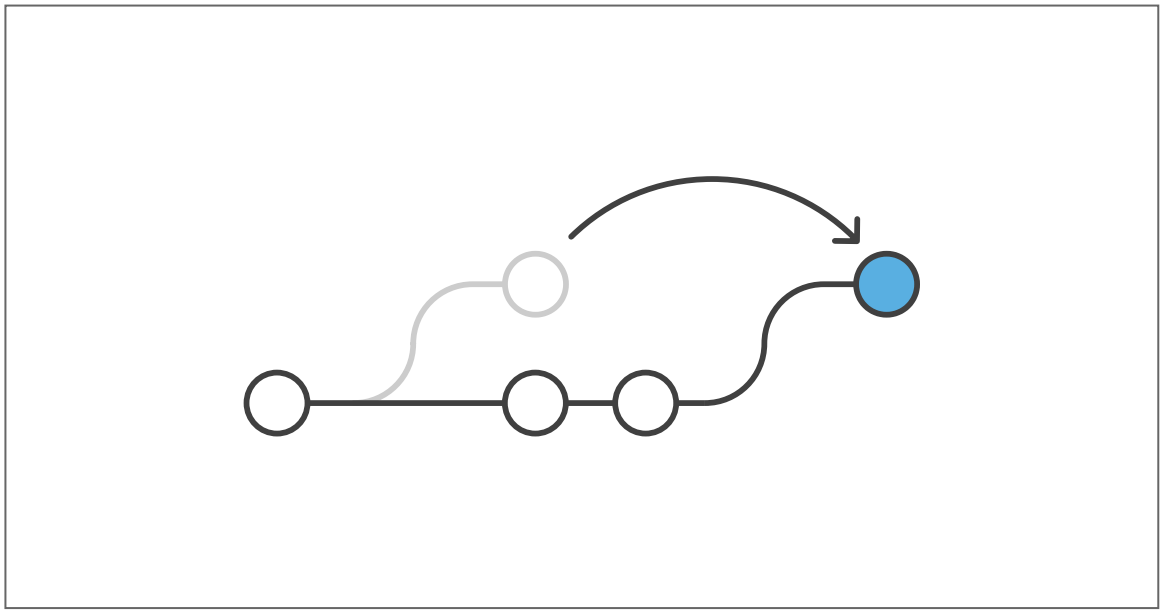
CLEAN UP!

Commit the changes into the repository's repository

```
$ git add . && git commit -m "Add the merge"
[master ef5f14f] 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/5d/19f7c065889bc9945592b19044346f323
 create mode 100644 objects/93/d56bde8cd7e1ac44d1f4f454a189b71b7
 create mode 100644 objects/d5/a29e72348dd06004654c605f561d7d6fc
```

The merge files are gone!

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 a new branch 'pe/cherry_picking'
```

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

```
$ git log --oneline pe/whiskey_is_also_an_option
bf4d8e2 Accept whiskey as reward
bc03912 Motivate the speaker
b05598c Motivate the participant
b4aec28 Describe the training
```

...then pick it up!

```
$ git cherry-pick pe/whiskey_is_also_an_option
error: could not apply bf4d8e2... Accept whiskey as reward
hint: after resolving the conflicts, mark the corrected pat
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
```

SWEET SWEET CONFLICTS

Dig, dig, dig, dig

```
$ git add . && git commit -m "Commit a cherry pick conflict"
[master f3ec7b6] Commit a cherry pick conflict
 8 files changed, 9 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/42/107595c8a0cc81f103bc87d240456d2965c
 create mode 100644 refs/heads/pe/cherry_picking
```

Another object!

SO WHAT'S THE CONFLICT NOW?

```
$ 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](whiskeys.md) are great!
++>>>>>>> bf4d8e2... Accept whiskey as reward
```

as expected, the conflict looks almost the same!

USE A MERGETOOL

...to fix the conflict!

```
$ git mergetool
```

Conflict resolution with assistance!

Or fix the conflict manually if you prefer

```
$ sed -i '4d;6d;$d' README.md && git add README.md
```

CONTINUE CHERRY-PICKING

...once you finished fixing the conflict

```
$ git cherry-pick --continue
```

...or commit the staged changes with an existing commit message

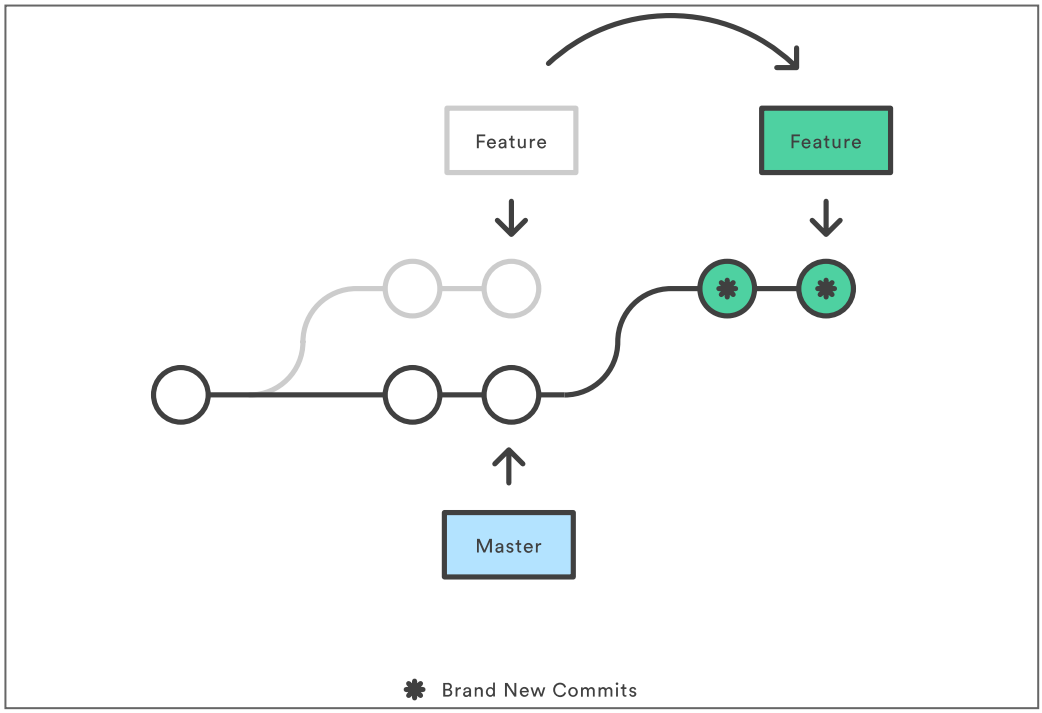
```
$ git commit -C pe/whiskey_is_also_an_option
[pe/cherry_picking 4ceca1f] Accept whiskey as reward
 Date: Wed Sep 18 14:08:17 2019 +0200
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

CLEAN UP!

Once again...

```
$ git add . && git commit -m "Add the cherry-pick"
[master b4bee90] Add the cherry-pick
 8 files changed, 6 insertions(+), 7 deletions(-)
 delete mode 100644 CHERRY_PICK_HEAD
 delete mode 100644 MERGE_MSG
 rewrite index (100%)
 create mode 100644 objects/4c/eca1f33df657e4a10e3df0145e63808bd9
```

GIT REBASE



Reapply commits on top of another branch

REBASE YOURSELF!

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

```
$ git rebase pe/add_list_of_favorite_beers
First, rewinding head to replay your work on top of it...
Applying: Accept whiskey as reward
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
Patch failed at 0001 Accept whiskey as reward
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
```

...that's a lot of output!

REBASING

...is "slightly" more complicated:

```
$ git add . && git commit -m "Commit a rebase conflict"
[master c2fc60d] Commit a rebase conflict
 29 files changed, 79 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
```

TAKE A CLOSER LOOK AT THE CONFLICT

```
$ 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](whiskeys.md) are great!
++>>>>>>> Accept whiskey as reward
```

The reference here is the commit's title!

FINISH REBASING

...once you resolved the conflicts:

```
$ sed -i '4d;6d;$d' README.md && git add README.md && git rebase --cor
Applying: Accept whiskey as reward
```

WHAT ABOUT THE REBASE FILES?

```
$ git add . && git commit -m "Add the rebase"
[master c747c7d] Add the rebase
 27 files changed, 5 insertions(+), 75 deletions(-)
 delete mode 100644 REBASE_HEAD
 rewrite index (100%)
 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
 delete mode 100644 rebase-apply/threeway
 delete mode 100644 rebase-apply/utf8
```

They are gone!

NON OBVIOUS CONFLICTS

Sometimes conflicts are introduced, which are not obvious to Git:

If one of the branches changes the behavior of a part of the code and the old behavior is required for the other branch, but there is no conflict within the files... then Git won't alert you!

How do you solve this?

TESTS TO THE RESCUE!

You can run your tests while rebasing:

```
$ git rebase -x "pytest" <newbase>
```

If the tests fail, Git will stop the rebase and allow you to fix your code

TIME FOR QUESTIONS


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 BRANCH OF BACKUPS

Add a branch with a few commits

```
$ git checkout -b pe/backups pe/rebasing
Switched to a new branch 'pe/backups'
$ echo "I would also love some feedback." >> README.md
$ git commit -am "Ask for feedback"
[pe/backups e4aefa6] Ask for feedback
 1 file changed, 1 insertion(+)
$ echo "Personal feedback is the best." >> README.md
$ git commit -am "Ask for personal feedback"
[pe/backups b80fbb0] 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/backups fd3ec39] Trade feedback for coffee
 1 file changed, 1 insertion(+)
```

CLEAN UP!

```
$ git add . && git commit -m "Add the branch of backups"
[master f004433] Add the branch of backups
 15 files changed, 19 insertions(+), 2 deletions(-)
 create mode 100644 logs/refs/heads/pe/backups
 create mode 100644 objects/34/96ebedc52ee70e2b129e315084623c7dee
 create mode 100644 objects/44/09126657b95c9c83a73ac6d730ae7353b6
 create mode 100644 objects/4f/56584f94dc324de2c2ffd66b4e145a6912
 create mode 100644 objects/58/ad30ba9d86b178e9c878ac031b1217d89a
 create mode 100644 objects/b8/0fbb090f78308fb83a3938aa2c3d2ac8de
 create mode 100644 objects/c3/2f3348d4d95fed6ff7c80054daf3690c50
 create mode 100644 objects/e4/aefa6a92877e1c8e4f6b95a7c93acb804a
 create mode 100644 objects/f1/c24400178f30d7f56ac2967d2fd7969681
 create mode 100644 objects/fd/3ec399f16eb5b5078e306e3a4f71a12687
 create mode 100644 refs/heads/pe/backups
```

REBASING IN INTERACTIVE MODE

"Change" the commit history during rebase!

```
$ git checkout -b pe/interactive_rebase pe/backups
Switched to a new branch 'pe/interactive_rebase'
```

This is like rebasing

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

...on steroids!

Your editor now lists all the commits of your branch!

```
pick e4aefa6 Ask for feedback
pick b80fbb0 Ask for personal feedback
pick fd3ec39 Trade feedback for coffee

# Rebase 4ceca1f..fd3ec39 onto 4ceca1f (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.
#
```

THERE'S A SHORTCUT!

...to avoid having to move commits around

```
$ git help commit | grep "autosquash" -C 2

       --fixup=<commit>
           Construct a commit message for use with rebase --autosquash. The commit message w
           the subject line from the specified commit with a prefix of "fixup! ". See git-re
           for details.

       --squash=<commit>
           Construct a commit message for use with rebase --autosquash. The commit message s
           line is taken from the specified commit with a prefix of "squash! ". Can be used
           additional commit message options (-m/-c/-C/-F). See git-rebase(1) for details.
```

COMMIT THE CHANGES IN THE REPOSITORY

```
$ git add . && git commit -m "Rebase in interactive mode"
[master 4394851] Rebase in interactive mode
 4 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 logs/refs/heads/pe/interactive_rebase
 create mode 100644 refs/heads/pe/interactive_rebase
```

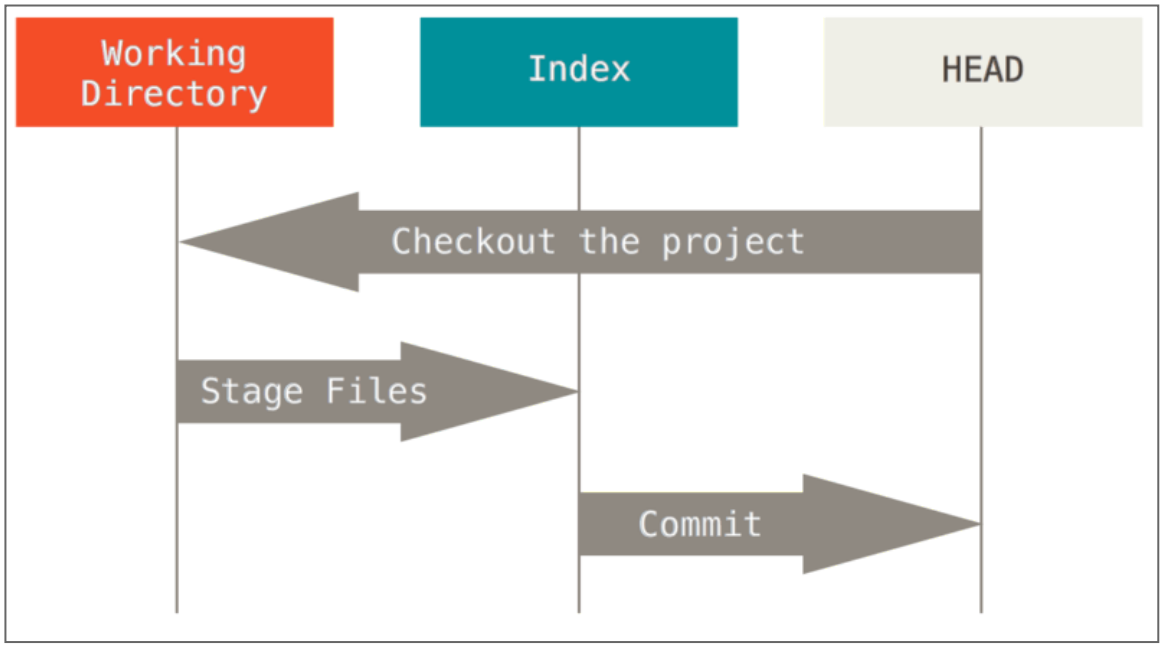
BUT WHAT IF

- ...it is more complex?
- ...commits need to be split?
- ...part of a commit sneaked into another commit?

`reset`, `checkout` and `stash` to the help!

THE THREE TREES

- the `HEAD`
- the index
- Working Directory


Working Directory
Index
HEAD
Checkout the project
Stage Files
Commit

RESET VS CHECKOUT VS STASH

- Reset will move what your `HEAD` is pointing to
- Checkout will move your `HEAD`
- Stash will move your index and change the working directory

RESET IN DETAIL

The following steps are executed when resetting:

```
1. Move the branch HEAD points to (--soft)
2. Make the index look like HEAD (--mixed, default)
3. Make the working directory look like the index (--hard)
```

Here's a less compact explanation!

LET'S SEE IT IN ACTION!

Create a new branch

```
$ git checkout -b pe/reset pe/backups
Switched to a new branch 'pe/reset'
```

...and reset to where your "backup history" starts

```
$ git reset pe/rebasing
Unstaged changes after reset:
M    README.md
```

WHAT'S THE STATUS NOW?

```
$ git status
On branch pe/reset
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory
    modified:   README.md

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

...but we did not modify anything?

WHAT'S MODIFIED?

```
$ git diff
diff --git a/README.md b/README.md
index 93d56bd..58ad30b 100644
--- a/README.md
+++ b/README.md
@@ -4,3 +4,6 @@ Buy me a beer if it made you better.
 My list of [favorite beers](beers.md).
 Whiskey is also a good reward.
 [These whiskeys](whiskeys.md) are great!
+I would also love some feedback.
+Personal feedback is the best.
+Helpful feedback is awarded with great coffee.
```

All the changes from the `backups` branch are now in the working directory!

SO WHAT HAPPENED?

```
$ git add . && git commit -m "Commit the reset"
[master 6c43224] Commit the reset
 6 files changed, 7 insertions(+), 2 deletions(-)
 create mode 100644 logs/refs/heads/pe/reset
 create mode 100644 refs/heads/pe/reset
```

...this is not enough information!

LET'S TAKE A CLOSER LOOK!

```
$ git show --name-only
commit 6c4322471f8ec4fa68217d00c89cd338171e1e7a
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Wed Sep 18 14:08:17 2019 +0200

    Commit the reset

HEAD
ORIG_HEAD
index
logs/HEAD
logs/refs/heads/pe/reset
refs/heads/pe/reset
```

LET'S LOOK EVEN CLOSER!

```
$ cat ORIG_HEAD
fd3ec399f16eb5b5078e306e3a4f71a1268761ff
```

This is the last commit in the `pe/backups` branch!

```
$ cat logs/refs/heads/pe/reset
0000000000000000000000000000000000000000 fd3ec399f16eb5b507
fd3ec399f16eb5b5078e306e3a4f71a1268761ff 4ceca1f33df657e4a1
```

These where `pe/reset` pointed to

(

You can access this information using

```
$ git reflog pe/reset
4ceca1f pe/reset@{0}: reset: moving to pe/rebasing
fd3ec39 pe/reset@{1}: branch: Created from pe/backups
```

...parsing it for readability

)

CREATE A NEW COMMIT

...by commiting the change

```
$ git commit -am "Trade coffee for personal feedback"
[pe/reset 07de835] Trade coffee for personal feedback
 1 file changed, 3 insertions(+)
```

with a more meaninful message.

COMMIT THE CHANGES IN THE REPOSITORY

```
$ git add . && git commit -m "Reset to improve the log"
[master d5d95cc] Reset to improve the log
 6 files changed, 5 insertions(+), 2 deletions(-)
 create mode 100644 objects/07/de835d9f91b4efca492546900fff8d502
```

CHECKOUT IN DETAIL

The following steps are executed when checking out:

```
1. Try a trivial merge with chosen commit
2. Move the HEAD
3. Make the working directory look like the HEAD
```

But used with a path the following happens:

```
1. Make the index look like chosen commit
2. Make the working directory look like the index
```

USING THE CHECKOUT COMMAND

...to clean up the backup history

```
$ git checkout -b pe/checkingout pe/rebasing
Switched to a new branch 'pe/checkingout'
```

CHECKOUT THE CHANGES

...which you want to commit

```
$ git checkout -p pe/backups
diff --git b/README.md a/README.md
index 93d56bd..58ad30b 100644
--- b/README.md
+++ a/README.md
@@ -4,3 +4,6 @@ Buy me a beer if it made you better.
 My list of [favorite beers](beers.md).
 Whiskey is also a good reward.
 [These whiskeys](whiskeys.md) are great!
+I would also love some feedback.
+Personal feedback is the best.
+Helpful feedback is awarded with great coffee.
Apply this hunk to index and worktree [y,n,q,a,d,e,?]?
```

WHAT DID THE CHECKOUT DO?

```
$ git status
On branch pe/checkingout
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   README.md
```

...it put the files in the index!

CREATE THE NEW COMMIT

```
$ git commit -m "Trade coffee for personal feedback"
[pe/checkingout 07de835] Trade coffee for personal feedback
 1 file changed, 3 insertions(+)
```

AND WHAT HAPPENED IN THE REPOSITORY?

```
$ git add . && git commit -m "Checkout to improve the log"
[master 5f0748a] Checkout to improve the log
 5 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 logs/refs/heads/pe/checkingout
 create mode 100644 refs/heads/pe/checkingout
```

THE CHANGE STASH

...can also be used for changes which

- are "work in progress"
- do not belong to the current branch
- do not require a commit but need to be kept

LET'S SEE WHAT STASH DOES

Create a new branch

```
$ git checkout -b pe/stashing pe/rebasing
Switched to a new branch 'pe/stashing'
```

...as we did in the other steps

ADD CHANGES NOT WORTH COMMITTING

```
$ echo "[Beerpay](TODO: activate beerpay) is cool!" >> READM
```

Let's have some fun staging part of it

```
$ git add README.md
```

...and keeping another part in the working directory

```
$ echo "I also love mechanical keyboards" >> README.md
```

PUT IN THE STASH

...what prevents you from continue working

```
$ git stash push -m "WIP beerpay"
Saved working directory and index state On stashing: WIP bee
```

and the working directory is clean again!

```
$ git status
On branch pe/stashing
nothing to commit, working tree clean
```

NO MUSCLE SORENESS YET

...so let's check what happened in the repository:

```
$ git add . && git commit -m "Commit the stashed changes"
[master 40e386c] Commit the stashed changes
 14 files changed, 9 insertions(+), 2 deletions(-)
 create mode 100644 logs/refs/heads/pe/stashing
 create mode 100644 logs/refs/stash
 create mode 100644 objects/1a/47eb533a085da1065dc3e773fa144b803
 create mode 100644 objects/32/e961b3293f7a2690fed0d9507475f5447
 create mode 100644 objects/59/530508363fe3f9101f68e1a50eff14ac1
 create mode 100644 objects/a4/ac931e9ee88677b00fda33dd3ddd4d704
 create mode 100644 objects/be/da045e9cdb824a63f9c41adf38b4d43c9
 create mode 100644 objects/c4/b191b4e8a51a73ece4609decfc5c6871f
 create mode 100644 refs/heads/pe/stashing
 create mode 100644 refs/stash
```

6 new objects?! But we only changed one file!

GETTING STASHED CHANGES BACK

...as soon as you need them

```
$ git stash list
stash@{0}: On stashing: WIP beerpay
```

```
$ git stash pop
On branch pe/stashing
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working dire
    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (c4b191b4e8a51a73ece4609decfc5c6871f8d832)
```

NO BLISTERS EITHER

...so check the repository again!

```
$ git add . && git commit -m "Commit the stash pop"
[master b0715b2] Commit the stash pop
 3 files changed, 2 deletions(-)
 delete mode 100644 logs/refs/stash
 delete mode 100644 refs/stash
```

COMBINING THEM ALL!

You can use

`git checkout` and `git stash`

during a `git rebase -i`

TIME FOR QUESTIONS

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!