LEARNING GIT

THE HARD WAY

WHO AM I

SCHEDULE

- The Good, the Bad and the Ugly
- Gitception: Reaching into the substructure
- Of trees, branches and pieces of fruit
- Gitbreak: Try to keep the holistic view
- Workfellas: Five ways to contribute to the code base

REQUIREMENTS

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

THE TANGLED WORKING COPY PROBLEM

VERSION CONTROL SYSTEMS

What is a VCS and why do I need it?

- Version Control
- Collaboration
- Run automated tasks
- Error analysis

SOME VCS

ArX, Bazaar, BitKeeper, Codeville, CVS, Darcs, DCVS, Fossil, Git, GNU arch, Mercurial, Monotone, Perforce, Subversion, TFVCS, Veracity, ...

GIT


il buono

GIT

"... is oddly liberal with how and when you use it."

- is a powerful analysis tool
- allows running automated tasks
- allows many collaboration workflows
- allows patch management


il frutto

GIT

... is utterly complex

- has crazy command line syntax
- has a leaky abstraction
- requires many different concepts
- requires a holistic picture


il cattivo

GIT

... is really hard to learn

- has a complex information model
- comes with complex workflows
- has crappy documentation

...but once you're there

GIT

WILL MAKE YOU

A BETTER

PROGRAMMER

EXERCISES

- How long have been programming?
- What kind of software do you work on?
- What kind of tasks do you solve?
- What challenges you the most when writing software?
- Have you been using other version control systems?
- How long have been using Git?
- What challenges you the most when using Git?

DISCUSSION

SLIDES

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

SETUP

Install Git and tell it who you are

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

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 Jun 15 18:15 .
drwx------. 3 escodebar escodebar  60 Jun 15 18:15 ..
drwxrwxr-x. 7 escodebar escodebar 200 Jun 15 18:15 .git
```

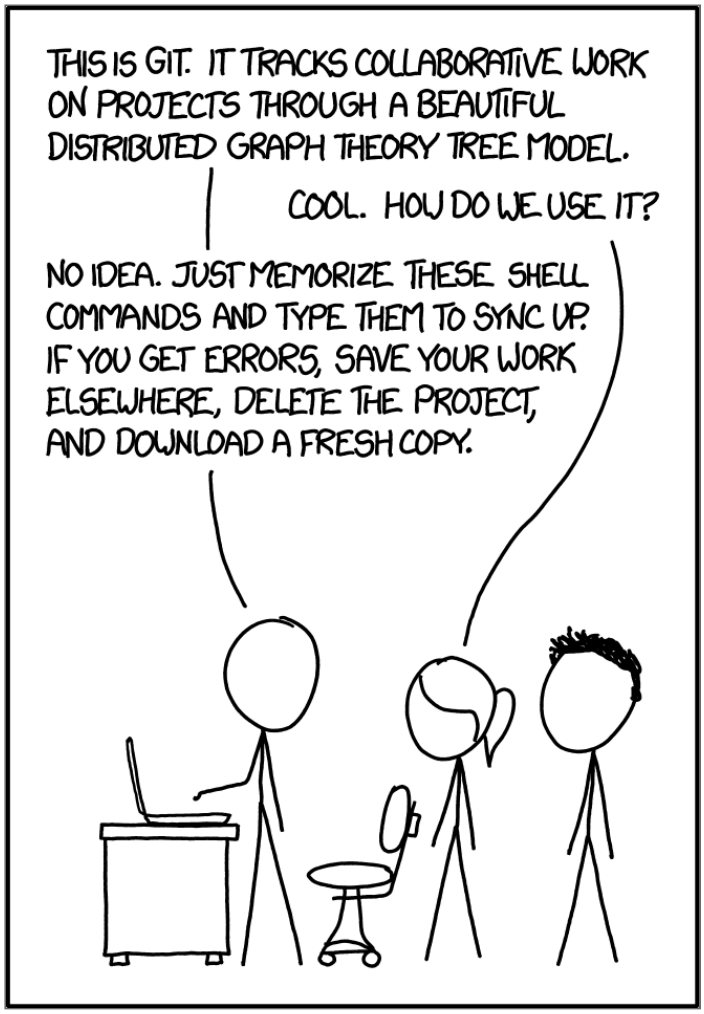
See that `.git` folder there? That's the repository.

DIGGING DEEPER

Don't panic!

```
$ ls -blah .git
total 12K
drwxrwxr-x. 7 escodebar escodebar 200 Jun 15 18:15 .
drwxr-xr-x. 3 escodebar escodebar  60 Jun 15 18:15 ..
drwxrwxr-x. 2 escodebar escodebar  40 Jun 15 18:15 branches
-rw-rw-r--. 1 escodebar escodebar  92 Jun 15 18:15 config
-rw-rw-r--. 1 escodebar escodebar  73 Jun 15 18:15 descript
-rw-rw-r--. 1 escodebar escodebar  23 Jun 15 18:15 HEAD
drwxrwxr-x. 2 escodebar escodebar 280 Jun 15 18:15 hooks
drwxrwxr-x. 2 escodebar escodebar  60 Jun 15 18:15 info
drwxrwxr-x. 4 escodebar escodebar  80 Jun 15 18:15 objects
drwxrwxr-x. 4 escodebar escodebar  80 Jun 15 18:15 refs
```

This is deep enough for now!


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.

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) aa92d21] 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!

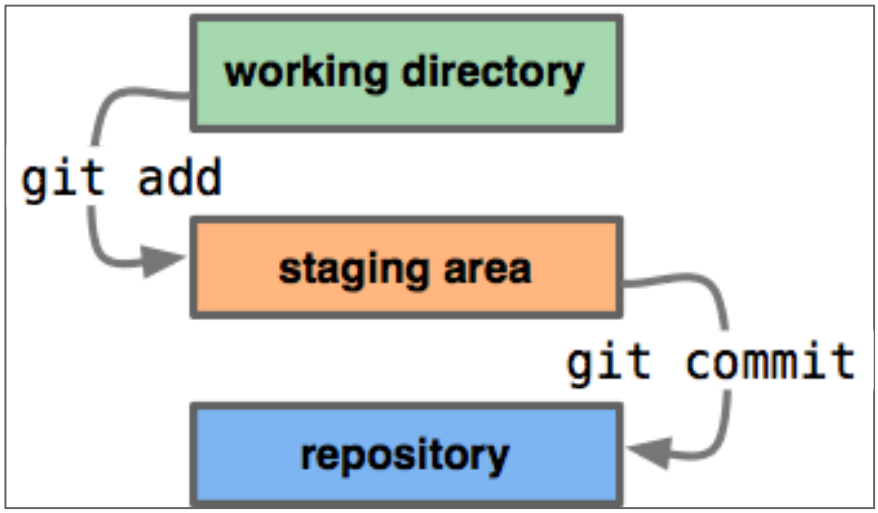
```
$ echo "# My awwwesome training" > README.md
```

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 2231cc5] 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) 1a72aa2] 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 c6fae32] Commit file
 7 files changed, 6 insertions(+)
 create mode 100644 COMMIT_EDITMSG
 create mode 100644 logs/HEAD
 create mode 100644 logs/refs/heads/master
 create mode 100644 objects/1a/72aa298e80003e7b0a297d3c797ff0672
 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 1a72aa2
commit
```

```
$ git cat-file -p 1a72aa2
tree a44f211c376b94d122c6429ef8e87ffa7856419d
author Pablo Escodebar <escodebar@gmail.com> 1592237756 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1592237756 -

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 1a72aa298e80003e7b0a297d3c797ff0672d4de6
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:15:56 2020 +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!
(1/1) Stage this hunk [y,n,q,a,d,e,?]?
```

```
$ git commit -m "Motivate the participant"
[master 7e68af6] 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 7e68af6
tree 10d06a676fb65acc4b1a2e57454039d904318393
parent 1a72aa298e80003e7b0a297d3c797ff0672d4de6
author Pablo Escodebar <escodebar@gmail.com> 1592237756 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1592237756 -

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.
(1/1) Stage this hunk [y,n,q,a,d,e,?]?
```

Once all hunks are decided, a commit will be created

```
[master 136082c] 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
136082c Motivate the speaker
7e68af6 Motivate the participant
1a72aa2 Describe the training
```

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

COMMIT THE REPOSITORY'S CHANGES

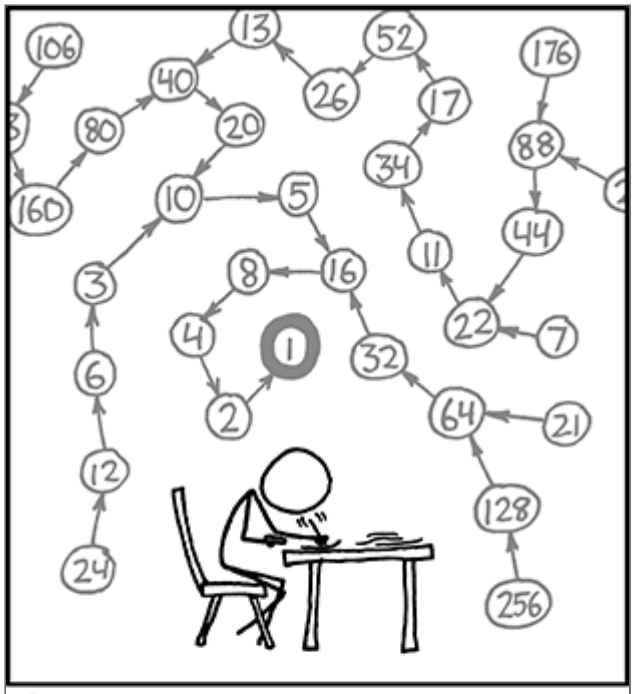
Add the new objects to the repository's repository:

```
$ git add . && git commit -m "Use the patch mode"
[master f39e882] Use the patch mode
 11 files changed, 9 insertions(+), 2 deletions(-)
 create mode 100644 objects/10/d06a676fb65acc4b1a2e57454039d904318393
 create mode 100644 objects/13/6082c01f1a06f705b7414ad92b64f289831abd
 create mode 100644 objects/22/d2d6223474b8b442b8aae05d4deab6f57a4a2a
 create mode 100644 objects/38/52d81df67551ce4174a25ce844cf690499f55c
 create mode 100644 objects/3f/652ededa8ed2a054ffa2c02bb34f99b53e94dd
 create mode 100644 objects/7e/68af6cdcddc12f3276742e20253fdf8ef4c018
```

EXERCISES

- Run the commands in the previous slides
- What's your mindset? Do you commit backups or changes? Or both?
- Why can we add a repository in the repository?
- Add two different changes to a file and commit them separatedly
- Restore a file from a blob object
- Create a commit with two files but only one blob object

DISCUSSION



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 fc0b16e] 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
136082c01f1a06f705b7414ad92b64f289831abd
```

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 136082c01f1a06f705b7414ad92b64f289831abd
commit 136082c01f1a06f705b7414ad92b64f289831abd
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:15:56 2020 +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 SWITCH TO IT!

```
$ git switch 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 0feaa00] 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 0feaa00c254b9304af040ddfe1b530d6d2cc39b7
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:15:56 2020 +0200

    Check out the branch

HEAD
logs/HEAD
```

...ah! the `HEAD` changed!

CREATE AND SWITCH TO BRANCHES

...IN ONE STEP!

Switch to a *new* branch:

```
$ git switch -c pe/add_list_of_favorite_beers master
Switched to a new 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 e6c1717] Let people know, what beer to
 2 files changed, 6 insertions(+)
 create mode 100644 beers.md
```

CREATE ANOTHER BRANCH

```
$ git switch -c 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 61b15c3] 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 bfd75ef] Add branches with conflicting commits
 16 files changed, 23 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/61/b15c3bc3152543b29c9069edafe85d5624
 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 objects/e6/c1717b9a80f504b890dd3130febc666467
 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
* e6c1717 Let people know, what beer to buy
| * 61b15c3 Accept whiskey as reward
|/
* 136082c Motivate the speaker
* 7e68af6 Motivate the participant
* 1a72aa2 Describe the training
```

Our tree starts growing branches!

EXERCISES

- Run the commands in the previous slides
- What is stored in the `HEAD` file?
- Take a look at the log files
- Can you switch to a specific commit?
- Delete and recover a branch with some commits
- Create a new branch based on a commit
- Take a look at the graph of a more mature project
- Create a new branch using then `--orphan` option

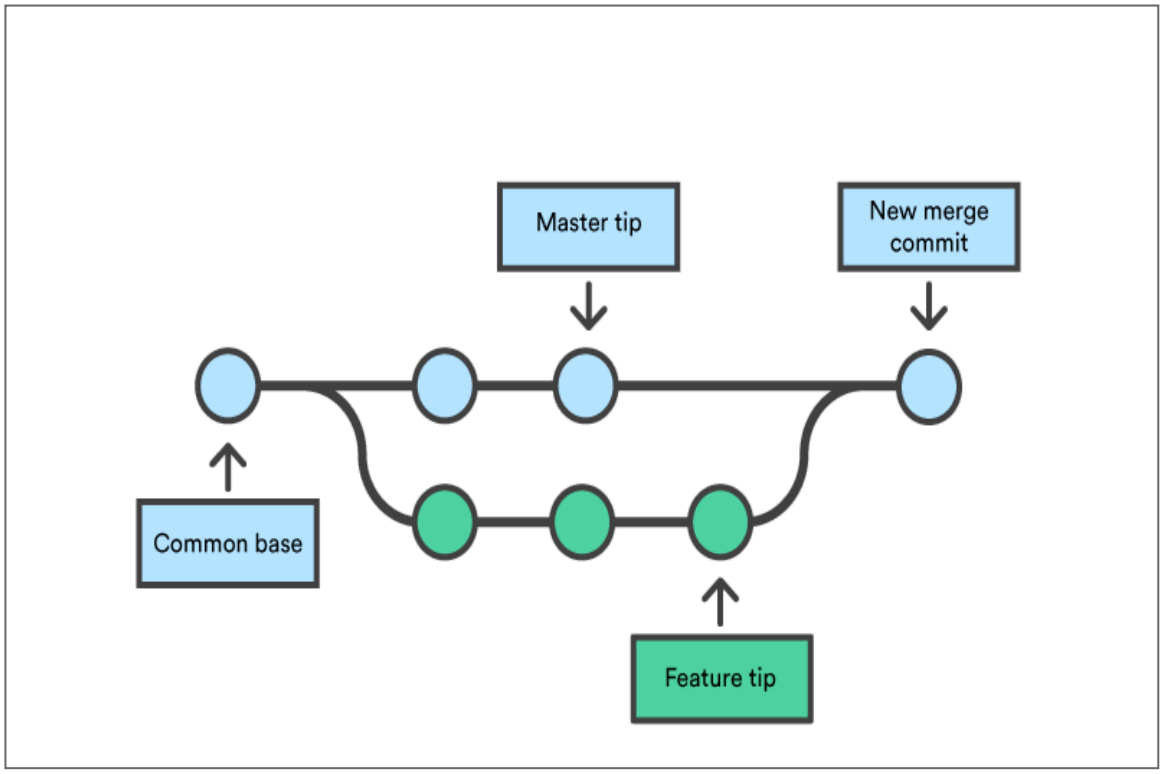
DISCUSSION


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.

WAIT... WHAT?

CONFLICTS?!

GIT MERGE



Join development histories

LET'S MERGE

Checkout a new branch for the merge

```
$ git switch -c 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 0cf5ac8] 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 d4691a8] Add the list of beers first
```

That was easy!

TAKE A LOOK AT THE MERGE COMMIT

Merge commits are special...

```
$ git cat-file -t d4691a8
commit
```

```
$ git cat-file -p d4691a8
tree d5a29e72348dd06004654c605f561d7d6fc32e6c
parent e6c1717b9a80f504b890dd3130febc666467063a
parent 61b15c3bc3152543b29c9069edafe85d5624bbd9
author Pablo Escodebar <escodebar@gmail.com> 1592237756 +02(
committer Pablo Escodebar <escodebar@gmail.com> 1592237756 -

Add the list of beers first
```

...since they have more than one parent!

MEET THE PARENTS

```
$ git show HEAD^1
commit e6c1717b9a80f504b890dd3130febc666467063a
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:15:56 2020 +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 61b15c3bc3152543b29c9069edafe85d5624bbd9
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:15:56 2020 +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
```

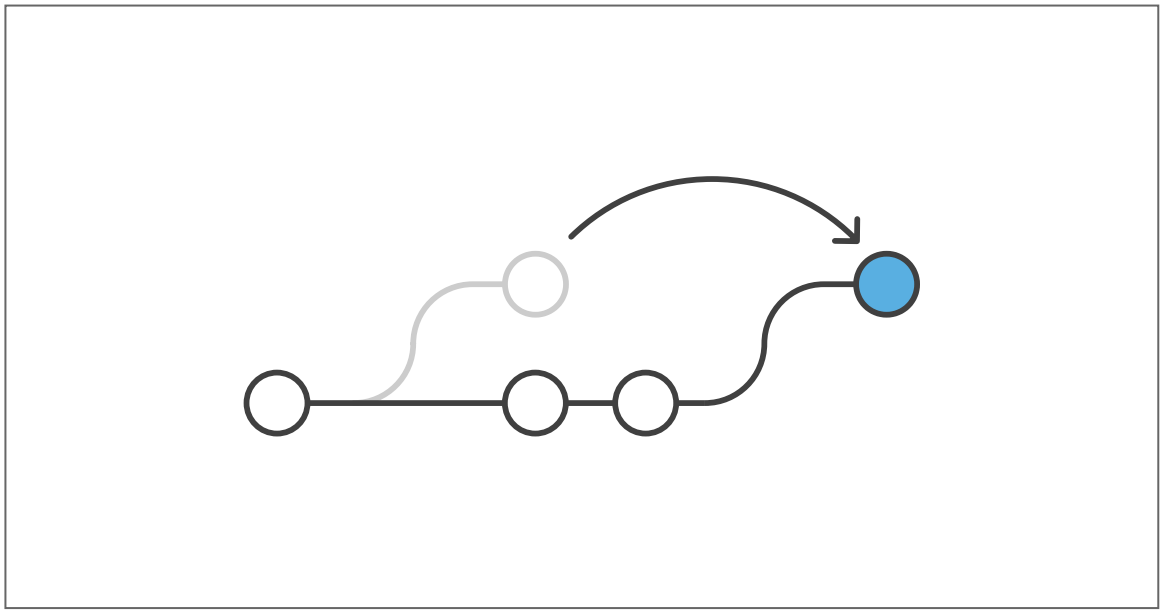
CLEAN UP!

Commit the changes into the repository's repository

```
$ git add . && git commit -m "Add the merge"
[master c5d8cfc] Add the merge
 11 files changed, 5 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/93/d56bde8cd7e1ac44d1f4f454a189b71b7
 create mode 100644 objects/d4/691a8ded5b5f913c33bdf6f8c90f41caf
 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 switch -c pe/cherry_picking pe/add_list_of_favorite_be
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
61b15c3 Accept whiskey as reward
136082c Motivate the speaker
7e68af6 Motivate the participant
1a72aa2 Describe the training
```

...then pick it up!

```
$ git cherry-pick pe/whiskey_is_also_an_option
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
```

SWEET SWEET CONFLICTS

Dig, dig, dig, dig

```
$ git add . && git commit -m "Commit a cherry pick conflict"
[master 5b2c78e] Commit a cherry pick conflict
 8 files changed, 10 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/26/567c2a8c50735bffa0ef39ddf7c6f9bbd67
 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!
++>>>>>>> 61b15c3... 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

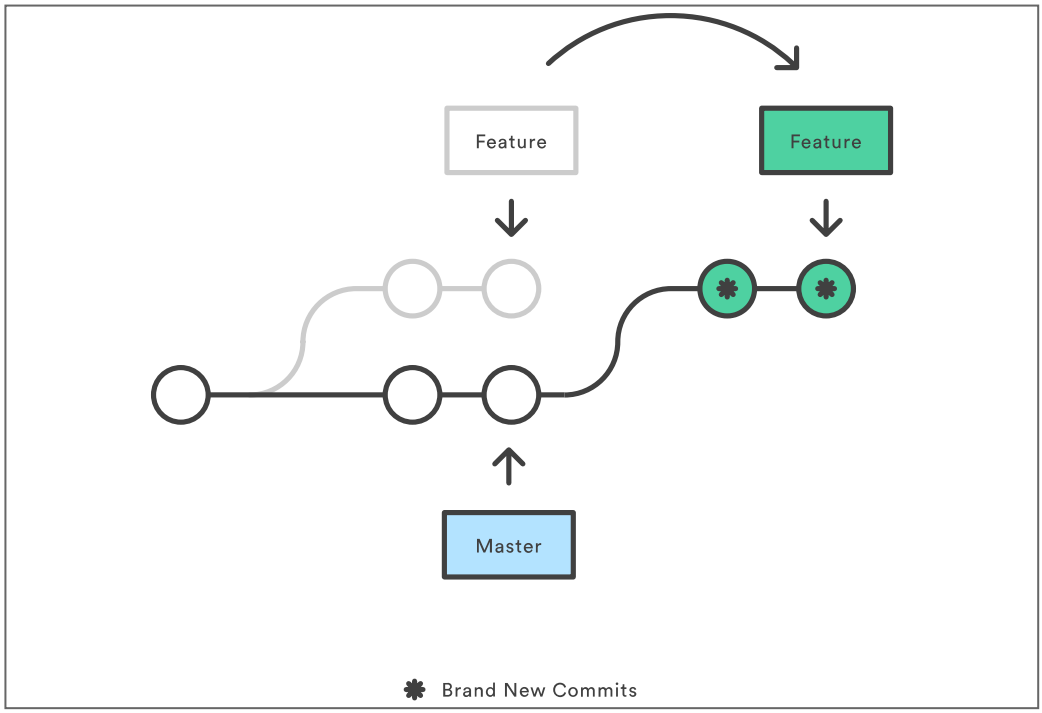
```
$ git commit -C pe/whiskey_is_also_an_option
[pe/cherry_picking 4e80c6b] Accept whiskey as reward
 Date: Mon Jun 15 18:15:56 2020 +0200
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

CLEAN UP!

Once again...

```
$ git add . && git commit -m "Add the cherry-pick"
[master 29b0efe] Add the cherry-pick
 8 files changed, 4 insertions(+), 7 deletions(-)
 delete mode 100644 CHERRY_PICK_HEAD
 delete mode 100644 MERGE_MSG
 rewrite index (100%)
 create mode 100644 objects/4e/80c6b4bed5881b2a4a5e66168bf10ca8ca
```

GIT REBASE



Reapply commits on top of another branch

REBASE YOURSELF!

```
$ git switch -c pe/rebasing pe/whiskey_is_also_an_option
Switched to a new branch 'pe/rebasing'
```

```
$ git rebase pe/add_list_of_favorite_beers
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
```

REBASING

...is "slightly" more complicated:

```
$ git add . && git commit -m "Commit a rebase conflict"
[master 1a2d457] Commit a rebase conflict
 22 files changed, 70 insertions(+), 2 deletions(-)
 create mode 100644 MERGE_MSG
 create mode 100644 REBASE_HEAD
 rewrite index (100%)
 create mode 100644 logs/refs/heads/pe/rebasing
 create mode 100644 rebase-merge/author-script
 create mode 100644 rebase-merge/done
 create mode 100644 rebase-merge/drop_redundant_commits
 create mode 100644 rebase-merge/end
 create mode 100644 rebase-merge/git-rebase-todo
 create mode 100644 rebase-merge/git-rebase-todo.backup
 create mode 100644 rebase-merge/head-name
 create mode 100644 rebase-merge/interactive
 create mode 100644 rebase-merge/message
 create mode 100644 rebase-merge/msgnum
 create mode 100644 rebase-merge/onto
 create mode 100644 rebase-merge/orig-head
 create mode 100644 rebase-merge/patch
 create mode 100644 rebase-merge/stopped-sha
 create mode 100644 refs/heads/pe/rebasing
```

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!
++>>>>>>> 61b15c3... 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 commit -m 'Accept whiskey as reward'
[detached HEAD 4e80c6b] Accept whiskey as reward
 2 files changed, 7 insertions(+)
 create mode 100644 whiskeys.md
```

```
$ git rebase --continue
¤[KSuccessfully rebased and updated refs/heads/pe/rebasing.
```

WHAT ABOUT THE REBASE FILES?

```
$ git add . && git commit -m "Add the rebase"
[master 2711516] Add the rebase
 20 files changed, 5 insertions(+), 65 deletions(-)
 delete mode 100644 MERGE_MSG
 rewrite index (100%)
 delete mode 100644 rebase-merge/author-script
 delete mode 100644 rebase-merge/done
 delete mode 100644 rebase-merge/drop_redundant_commits
 delete mode 100644 rebase-merge/end
 delete mode 100644 rebase-merge/git-rebase-todo
 delete mode 100644 rebase-merge/git-rebase-todo.backup
 delete mode 100644 rebase-merge/head-name
 delete mode 100644 rebase-merge/interactive
 delete mode 100644 rebase-merge/message
 delete mode 100644 rebase-merge/msgnum
 delete mode 100644 rebase-merge/onto
 delete mode 100644 rebase-merge/orig-head
 delete mode 100644 rebase-merge/patch
 delete mode 100644 rebase-merge/stopped-sha
```

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

EXERCISES

- Run the commands in the previous slides
- Abort a `merge`, `cherry-pick` or a `rebase`
- Merge two branches with different roots
- Create a commit with 3 parents
- Cherry-pick two commits at once
- Cherry-pick a merge commit
- Use `git rebase --onto`
- Take a look at the `--interactive` option of `git rebase`

DISCUSSION


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.

BACKUPS VS. PATCHES

Version control is a system that records changes to a file or set of files over time ...so that you can recall specific *states* later.

Version control is a system that allows to write and manage sets of *patches* ...so that you can recall specific *versions* later.

PREPARING THE BRANCH OF BACKUPS

Add a branch with a few commits

```
$ git switch -c 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 fee43a1] 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 4ca63e4] 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 affca42] Trade feedback for coffee
 1 file changed, 1 insertion(+)
```

CLEAN UP!

```
$ git add . && git commit -m "Add the branch of backups"
[master 2a038c4] Add the branch of backups
 15 files changed, 18 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/4c/a63e426c0d0e0632de59783bb88933b78a
 create mode 100644 objects/4f/56584f94dc324de2c2ffd66b4e145a6912
 create mode 100644 objects/58/ad30ba9d86b178e9c878ac031b1217d89a
 create mode 100644 objects/af/fca42f1026eadffb8129ca0ab0b01c800c
 create mode 100644 objects/c3/2f3348d4d95fed6ff7c80054daf3690c50
 create mode 100644 objects/f1/c24400178f30d7f56ac2967d2fd7969681
 create mode 100644 objects/fe/e43a1c5ec6ba250b8f2bb0052dcb7083d0
 create mode 100644 refs/heads/pe/backups
```

REBASING IN INTERACTIVE MODE

"Change" the commit history during rebase!

```
$ git switch -c 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 fee43a1 Ask for feedback
pick 4ca63e4 Ask for personal feedback
pick affca42 Trade feedback for coffee

# Rebase 4e80c6b..affca42 onto 4e80c6b (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 a31210f] 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
```

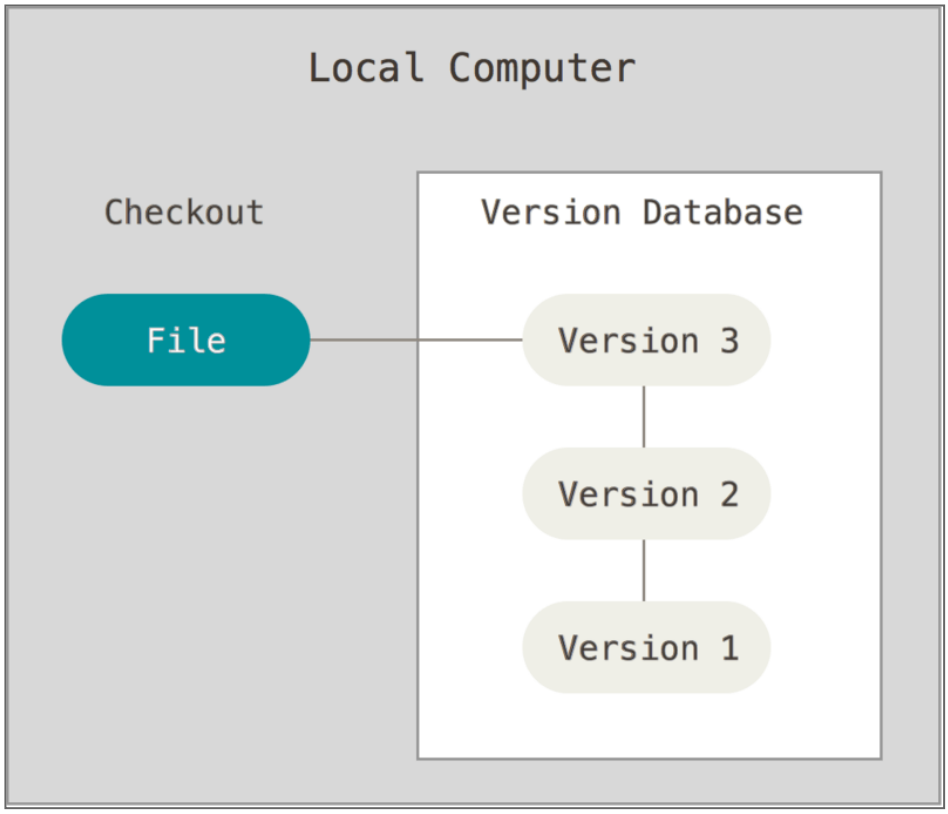
EXERCISES

- Change the order of commits
- Create a `--fixup` or `--squash` commit
- Rebase a branch using the `--autosquash` flag
- Run an interactive rebase with the `-x` option
- Edit a commit during a rebase
- Study `git reset`, `git stash` and `git checkout`

DISCUSSION

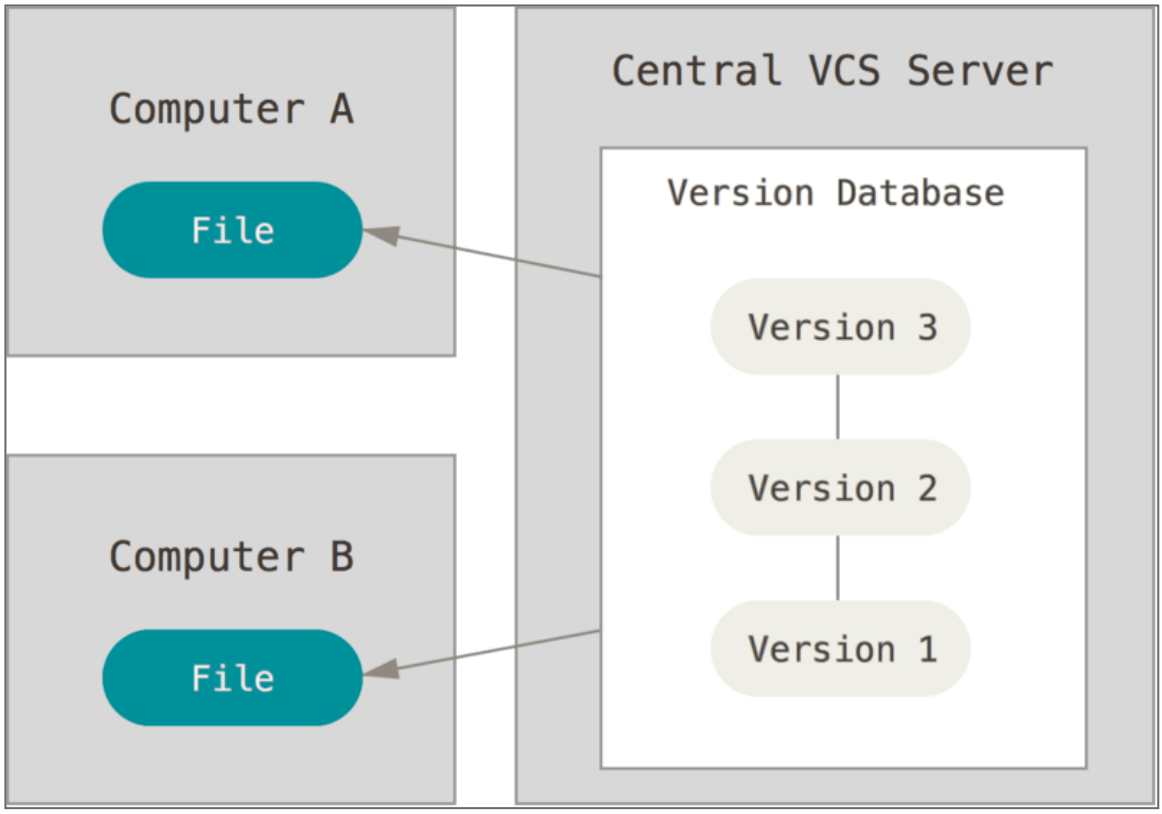
I DIDN'T FIND A DECENT COMIC FOR DISCUSSING DISTRIBUTED GIT

SO YOU ONLY GET THIS LOWSY SLIDE


Local Computer
Checkout
File
Version Database
Version 3
Version 2
Version 1

Let's add some...

collaboration

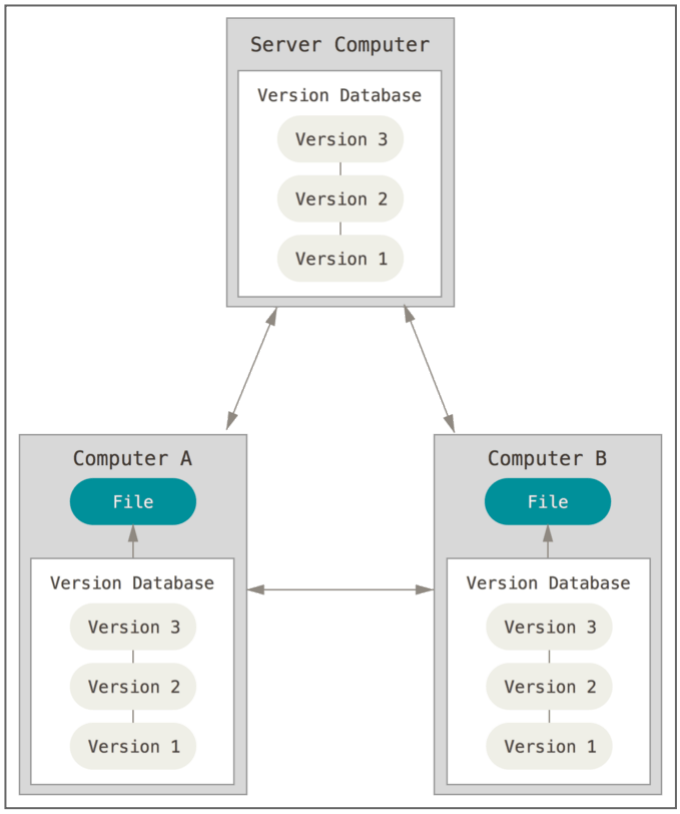

Computer A
File
Computer B
File
Central VCS Server
Version Database
Version 3
Version 2
Version 1

DRAWBACKS OF *CENTRALIZED* VCS

- Connection to central server is required
- Remote commits are slow
- Worflows can be very complicated
- A central server introduces a single point of failure

What if version control was...

distributed?


Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

Development becomes *complex!*

ADVANTAGES OF *DISTRIBUTED* VCS

- Committing changesets can be done locally
- Committing changesets is extremely fast
- One can share the changes with others without having to publish them

...also: A centralized solution is not less complex.

Complexity is reduced if patches are well...

- self contained
- documented
- tested
- small

Other conveniences of using patches:

- simpler error analysis
- simpler code review
- better code reusability
- faster development

ADDING A REMOTE

...is simple as running

```
$ git remote add gitception .git/
```

STARTING TO FEEL LIKE HOME

Let's see how Git deals with remotes:

```
$ git add . && git diff --cached
diff --git a/config b/config
index 515f483..6bc950f 100644
--- a/config
+++ b/config
@@ -3,3 +3,6 @@
     filemode = true
     bare = false
     logallrefupdates = true
+[remote "gitception"]
+    url = .git/
+    fetch = +refs/heads/*:refs/remotes/gitception/*
```

Aha! They're stored in the repository's configuration!

COMMIT THE CHANGES

```
$ git commit -m "Add Gitception as remote"
[master 7100849] Add Gitception as remote
 1 file changed, 3 insertions(+)
```

SYNC THE REPOSITORIES

...using

```
$ git fetch gitception
From .git
 * [new branch]      master     -> gitception/master
```

This will download all branches and their associated objects!

WE COULD START BUILDING A BUNKER!

```
$ git add . && git commit -m "Fetch the gitception remote"
[master 4d79136] Fetch the gitception remote
 5 files changed, 3 insertions(+)
 create mode 100644 FETCH_HEAD
 create mode 100644 logs/refs/remotes/gitception/master
 create mode 100644 objects/pack/pack-3384d99cb099abff0c211a16b628d44702b65249
 create mode 100644 objects/pack/pack-3384d99cb099abff0c211a16b628d44702b65249
 create mode 100644 refs/remotes/gitception/master
```

What the $%#@! What are pack files?

LET'S ADD A PUBLIC REPOSITORY

...to make our work accessible to others

```
$ git remote add github git@github.com:escodebar/trainings.g
```

PUSHING ALL BRANCHES

...and changing their name on the remote

```
$ git push -u github "refs/heads/*:refs/heads/git/unibe/repo
Branch 'master' set up to track remote branch 'git/unibe/re
Branch 'pe/add_list_of_favorite_beers' set up to track remo
Branch 'pe/backups' set up to track remote branch 'git/unib
Branch 'pe/cherry_picking' set up to track remote branch 'g
Branch 'pe/interactive_rebase' set up to track remote branc
Branch 'pe/merging' set up to track remote branch 'git/unib
Branch 'pe/new_branch' set up to track remote branch 'git/u
Branch 'pe/rebasing' set up to track remote branch 'git/uni
Branch 'pe/whiskey_is_also_an_option' set up to track remot
```

REMOTE BRANCHES IN OUR REPOSITORY?

...how does that work?

```
$ git add . && git commit -m "Push the branch"
[master 0b1c9b2] Push the branch
 19 files changed, 48 insertions(+)
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/master
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/add_list_of_favorit
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/backups
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/cherry_picking
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/interactive_rebase
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/merging
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/new_branch
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/rebasing
 create mode 100644 logs/refs/remotes/github/git/unibe/repo/pe/whiskey_is_also_an_
 create mode 100644 refs/remotes/github/git/unibe/repo/master
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/add_list_of_favorite_bee
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/backups
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/cherry_picking
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/interactive_rebase
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/merging
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/new_branch
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/rebasing
 create mode 100644 refs/remotes/github/git/unibe/repo/pe/whiskey_is_also_an_optic
```

BUT HOW...

...does it know which branches belong together?

```
$ git show
commit 0b1c9b25f9b0998694ef206ca2108b2edb2f558c
Author: Pablo Escodebar <escodebar@gmail.com>
Date:   Mon Jun 15 18:16:01 2020 +0200

    Push the branch

diff --git a/config b/config
index 6bc950f..4503e49 100644
--- a/config
+++ b/config
@@ -6,3 +6,33 @@
 [remote "gitception"]
     url = .git/
     fetch = +refs/heads/*:refs/remotes/gitception/*
+[remote "github"]
+    url = git@github.com:escodebar/trainings.git
+    fetch = +refs/heads/*:refs/remotes/github/*
+[branch "master"]
+    remote = github
+    merge = refs/heads/git/unibe/repo/master
+[branch "pe/add_list_of_favorite_beers"]
+    remote = github
+    merge = refs/heads/git/unibe/repo/pe/add_list_of_favo
```

LET'S ALSO PUSH THE GITCEPTION BRANCH

...for the sake of completeness

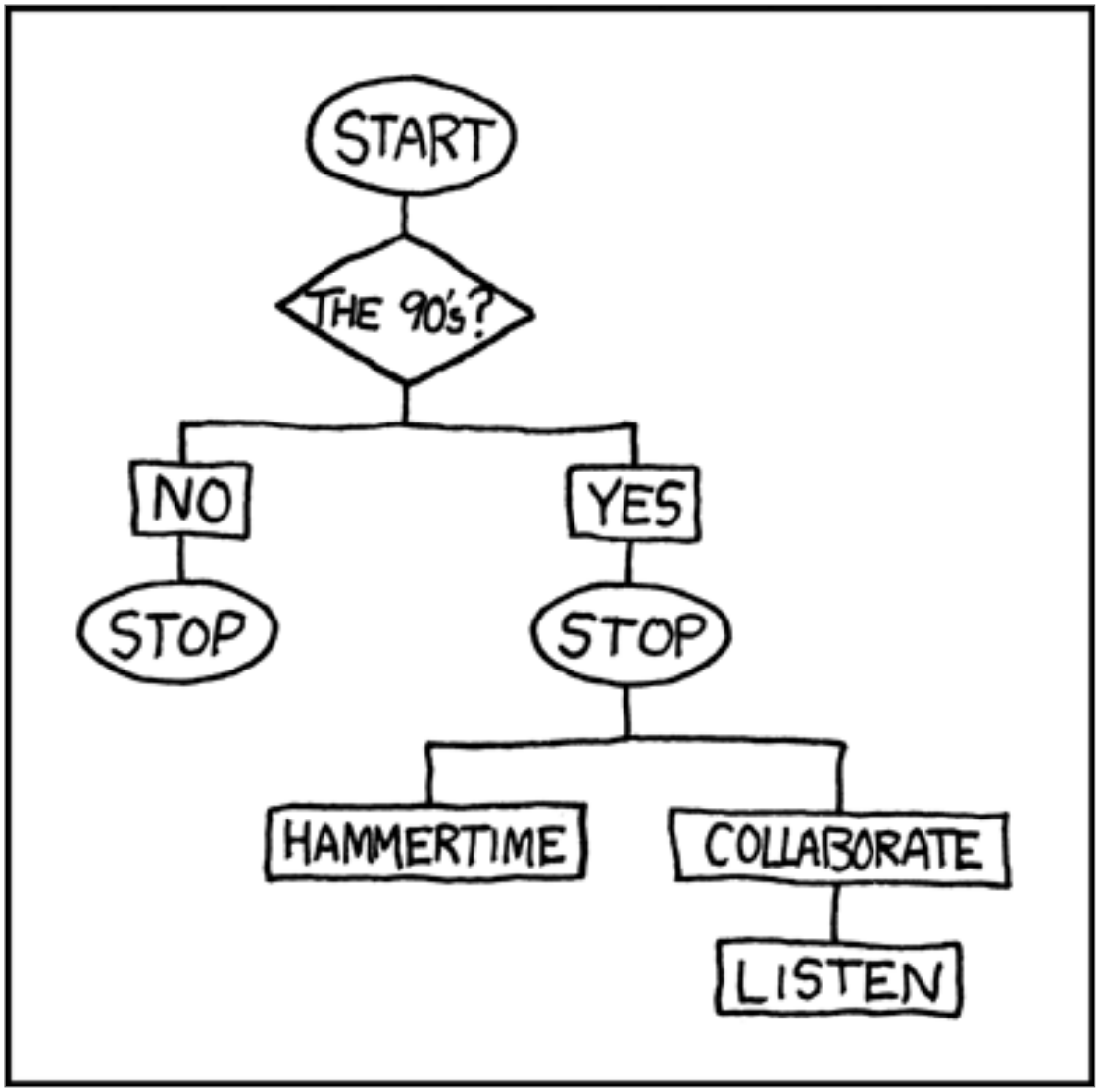
```
$ git fetch && git push -u github refs/remotes/gitception/master:refs/heads/git/unibe/
remote:
remote: Create a pull request for 'git/unibe/gitception' on GitHub by visiting:
remote:      https://github.com/escodebar/trainings/pull/new/git/unibe/gitception
remote:
To github.com:escodebar/trainings.git
 * [new branch]      gitception/master -> git/unibe/gitception
```

Now you know where to get the repositories for later analysis!

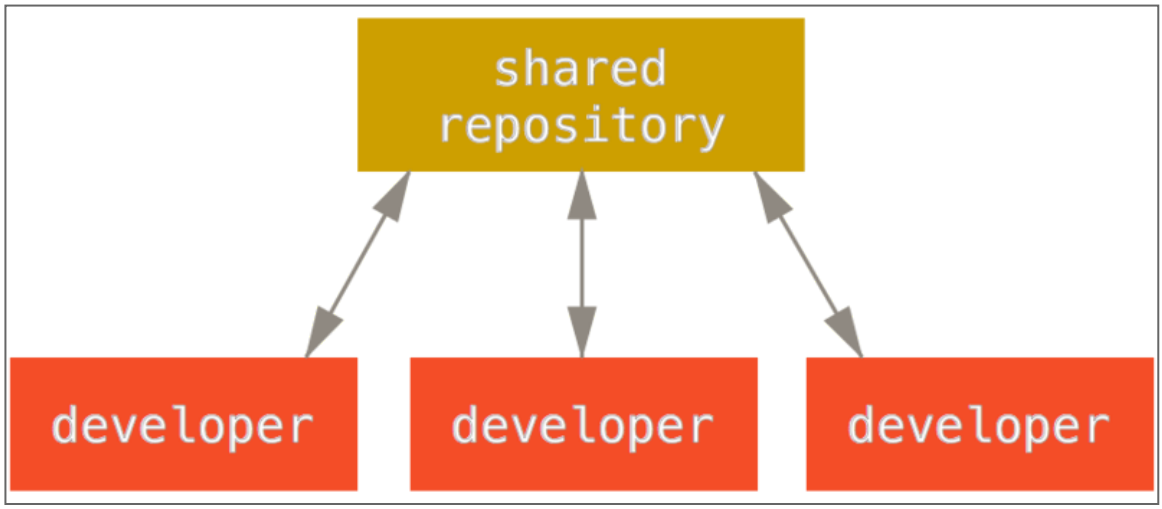
EXCERCISES

- Run the commands in the previous slides
- Checkout a remote branch
- Create a new branch based on a remote branch
- Think about it: Do you need a local master branch?
- Add a second remote and push a branch to it
- Explain how `git pull` is different from `git fetch`
- Push a local commit to a remote branch
- Read about `git push --mirror`

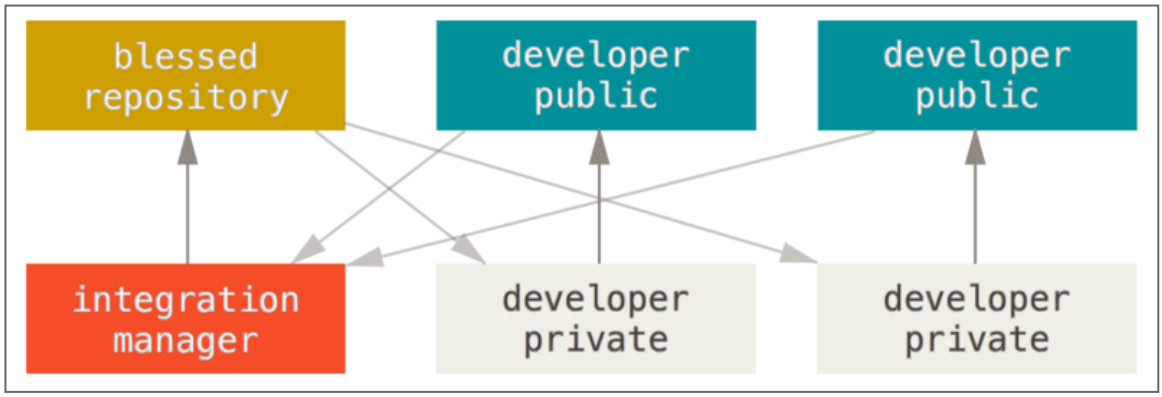
DISCUSSION


START
THE 90's?
NO
YES
STOP
STOP
HAMMERTIME
COLLABORATE
LISTEN

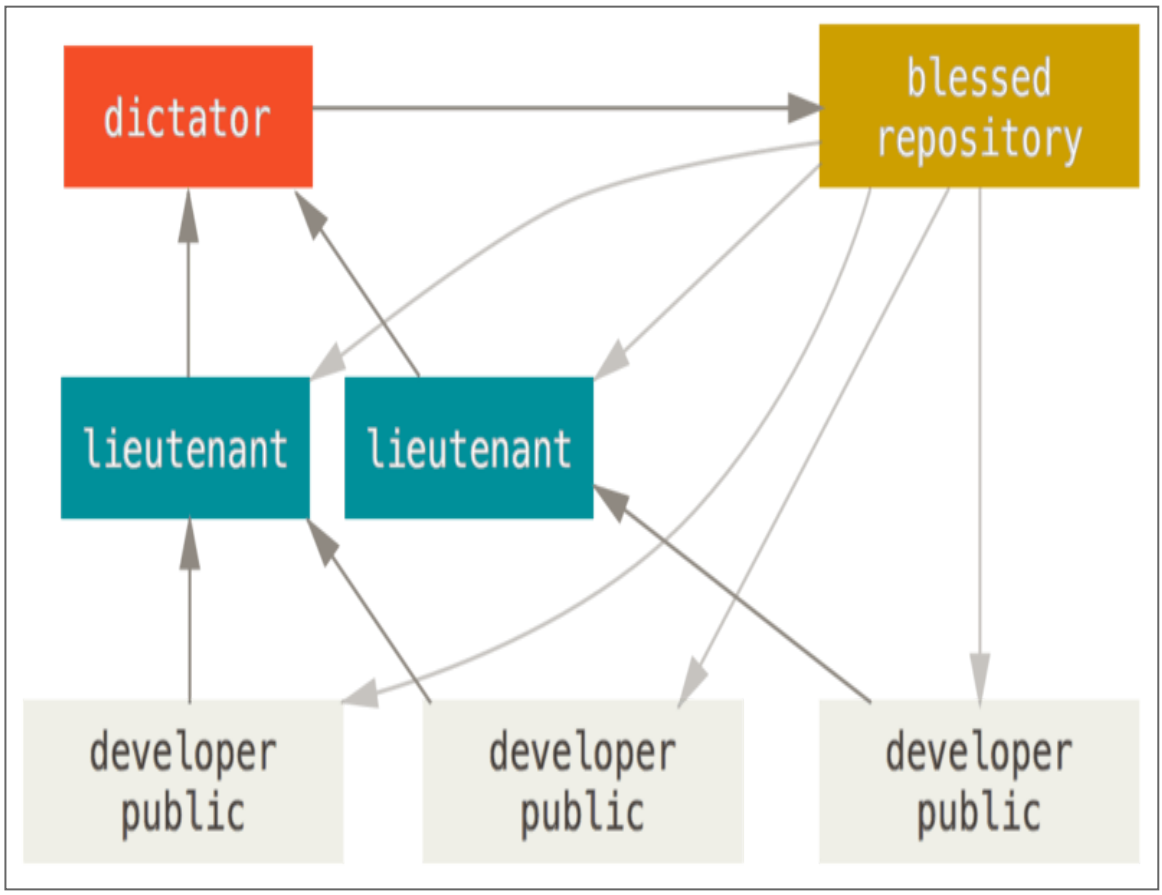
CENTRALIZED WORKFLOW



INTEGRATION-MANAGER WORKFLOW



DICTATOR AND LIEUTENANTS WORKFLOW



GITHUB FLOW

The **GitHub Flow** is a simple yet elegant workflow!


DID YOU FINISH THE SOFTWARE YET?
NO, I'M STILL PAYING OFF THE TECHNICAL DEBT FROM THE LAST PROGRAMMER YOU RUSHED.
I DON'T KNOW WHAT THAT MEANS.
WELL, THAT EXPLAINS A LOT.
Dilbert.com @ScottAdamsSays
1-3-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel

GITFLOW WORKFLOW

And then there is the **Gitflow Workflow**!

EXERCICES

- Discuss in a group your current workflows
- Think of possible scenarios where these workflows may be useful
- Discuss where and who manages project complexity with these workflows
- Choose one of the workflows to work within your current group
- Make a contribution to a toy repository using the chosen workflow

DISCUSSION

RESOURCES

- **10 things I hate about Git**
- **Cynefin**
- **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**

ONE LAST TASK

Fill out this survey

THANK YOU!