

GIT WORKFLOWS

..for shaping your commit history





Philipp Albrecht




**WHY SHAPE YOUR COMMIT
HISTORY?**


COMMIT HISTORY = CONTEXT

FIXING A BUG

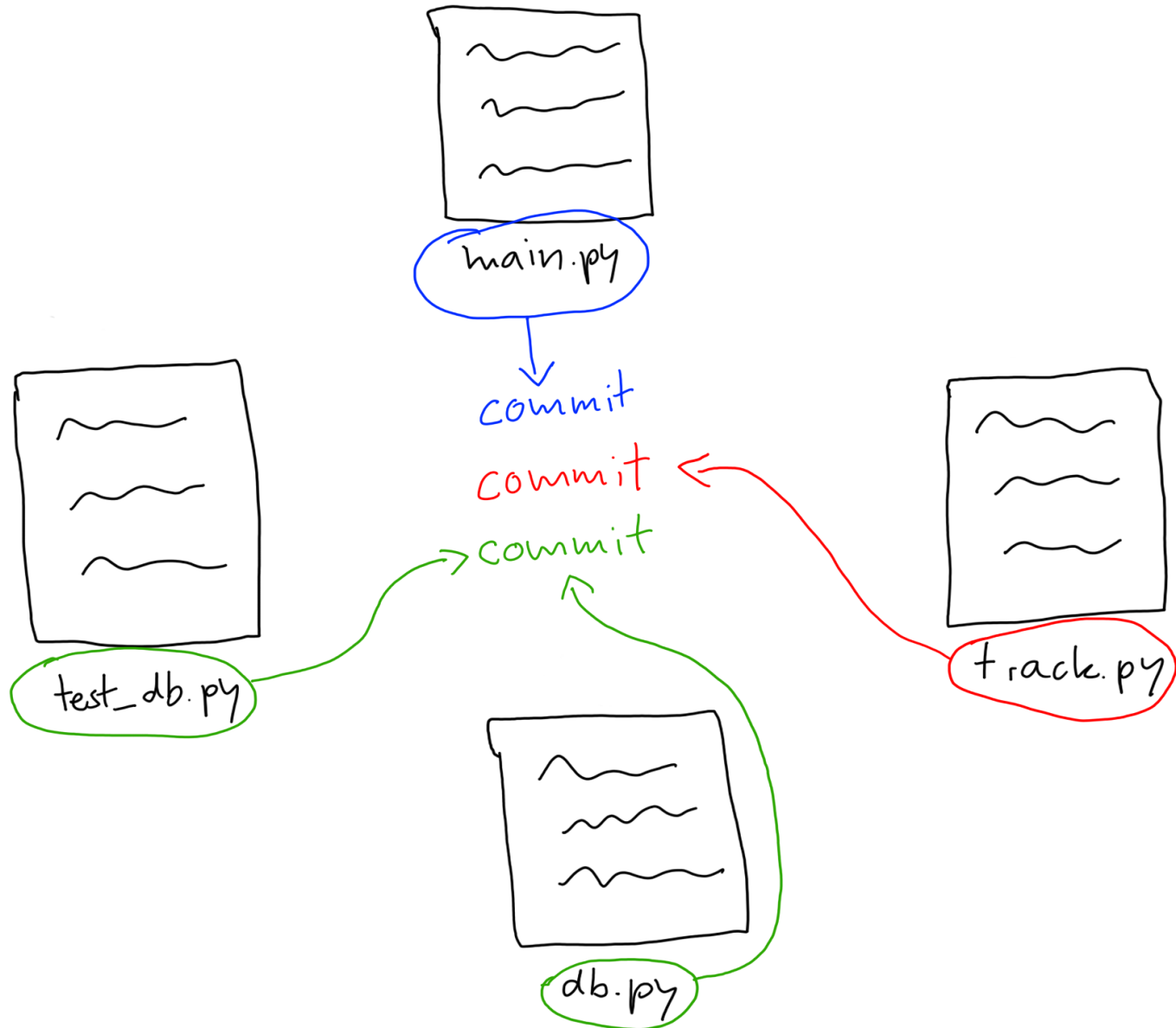
1. Dive into the bug related code
2. This variable name could be better, I'll rename that! 
3. Oh, I could extract that method here. Great! 
4. Done! Oh, cleaning up code feels so good 
5. Fix the actual bug 

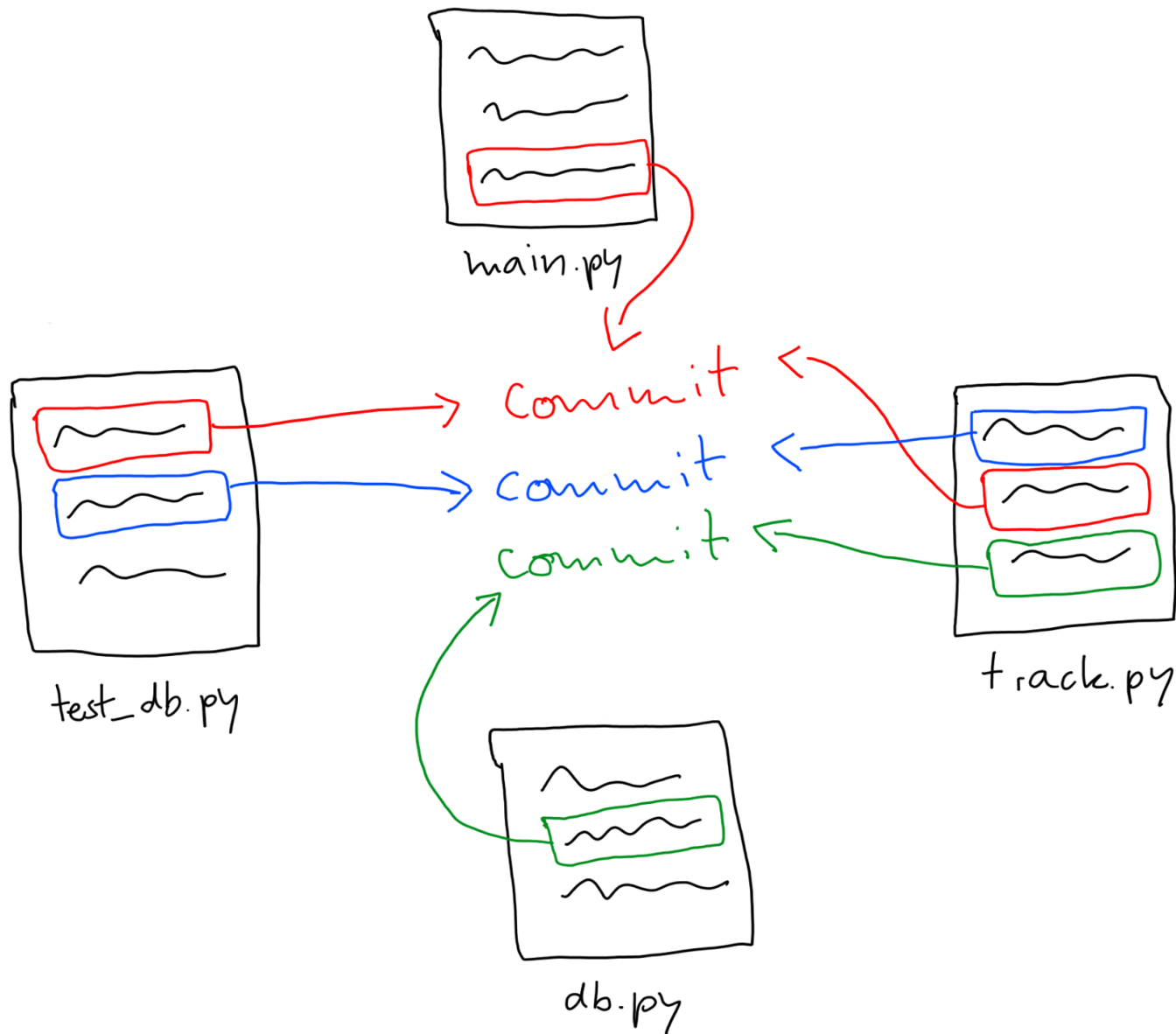
6. Damn, tests failing because of my refactoring..
7. Just need to rewrite these test cases a little 
8. Tests passing!
9. Finally..

```
$ git add .  
$ git commit -m "Fix bug"  
$ git push
```

A man with short dark hair, wearing a red and white plaid shirt, is smiling and talking to another man. The second man, wearing a white t-shirt, is seen from the side and back, with his hand on his hip. They are in a modern cafe or office setting with large windows and a copper-colored metal structure in the background.

```
git add .  
git commit -m "Fix bug"
```





INTERACTIVE HUNK STAGING

git add --patch

```
1 $ git add --patch
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
11
```

```
12
```

```
1 $ git add --patch
2 diff --git a/talk.py b/talk.py
3 index c7e0374..585dcd6 100644
4 --- a/talk.py
5 +++ b/talk.py
6 @@ -3,5 +3,5 @@ def give_talk(topic):
7
8     if __name__ == "__main__":
9 -         topic = "How to hone a straight razor"
10 +         topic = "Git workflows"
11         give_talk(topic)
12 (1/5) Stage this hunk [y,n,q,a,d,e,?]?
```

```
1 $ git add --patch
2 diff --git a/talk.py b/talk.py
3 index c7e0374..585dcd6 100644
4 --- a/talk.py
5 +++ b/talk.py
6 @@ -3,5 +3,5 @@ def give_talk(topic):
7
8     if __name__ == "__main__":
9 -         topic = "How to hone a straight razor"
10 +         topic = "Git workflows"
11         give_talk(topic)
12 (1/5) Stage this hunk [y,n,q,a,d,e,?]?
```

```
1 y - stage this hunk
2 n - do not stage this hunk
3 q - quit; do not stage this hunk or any of the remaining ones
4 a - stage this hunk and all later hunks in the file
5 d - do not stage this hunk or any of the later hunks in the file
6 e - manually edit the current hunk
7 ? - print help
```

Configure your editor of choice

```
$ git config --global core.editor "nvim"
```

WORKFLOW RECAP

~~Commit early, commit often.~~

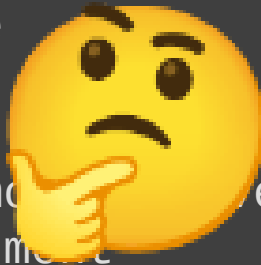
1. Hack out code like crazy!
2. Decide about commit boundaries in retrospect
3. Create commits with

```
$ git add --patch
```

4. Practice 1. - 3.
5. Commit early, commit often

fd8c33b	Encapsulate variable
ce52b63	Introduce parameter object
443d809	Replace conditional with polymorphism
d397f74	Extract interface
92335de	Reword log message
a45ea80	Inline function
62c3ac3	Rename class
da9adf3	Move function to module level
69d0584	Rename method argument
2f09c99	Fix failing tests
d28ba13	Fix bug
b4a6748	Extract method
36962d4	Rename variable

fd8c33b Encapsulate variable
ce52b63 Introduce parameter object
443d809 Replace conditional with polymorphism
d397f74 Extract interface
92335de Reword log message
a45ea80 Inline function
62c3ac3 Rename class
da9adf3 Move function to model
69d0584 Rename method argument
2f09c99 Fix failing tests
d28ba13 Fix bug
b4a6748 Extract method
36962d4 Rename variable



INTERACTIVE REBASING

git rebase --interactive

REWRITE HISTORY

- Reword commit messages
- Reorder commits
- Squash commits
- Delete commits
- ...



<https://sportshub.cbsistatic.com/i/2021/04/09/db00f67d-6819-4f7c-bc14-14d516b6431e/rick-and-morty-zeus-1222179.jpg>

Don't push your work until you're happy with it!

One of the cardinal rules of Git is that, since so much work is local within your clone, you have a great deal of freedom to rewrite your history locally. However, once you push your work, it is a different story entirely, and you should consider pushed work as final unless you have good reason to change it. In short, you should avoid pushing your work until you're happy with it and ready to share it with the rest of the world.

<https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History>

```
1 139e19d (HEAD -> bugfix) Fix failing tests
2 b5a0f1a Extract method
3 ce52b63 Inline function
4 a9c4c48 Fix bug
5 443d809 Move class into separate module
6 535e5b7 Rename variable
```



```
1 26550fe (HEAD -> bugfix) Fix bug
2 73a2261 Refactor applying the boyscout rule
3 |
4 +-----+
5 | Refactor applying the boyscout rule |
6 |                                     |
7 | - Extract method                   |
8 | - Inline function                  |
9 | - Move class into separate module  |
10 | - Rename variable                  |
11 | - Fix failing tests                 |
12 +-----+
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 pick 443d809 Move class into separate module
3 pick a9c4c48 Fix bug
4 pick ce52b63 Inline function
5 pick b5a0f1a Extract method
6 pick 139e19d Fix failing tests
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 pick 443d809 Move class into separate module
3 pick a9c4c48 Fix bug
4 pick ce52b63 Inline function
5 pick b5a0f1a Extract method
6 pick 139e19d Fix failing tests
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 pick 443d809 Move class into separate module
3 pick ce52b63 Inline function
4 pick b5a0f1a Extract method
5 pick 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```



```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 pick 443d809 Move class into separate module
3 pick ce52b63 Inline function
4 pick b5a0f1a Extract method
5 pick 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 pick 443d809 Move class into separate module
3 pick ce52b63 Inline function
4 pick b5a0f1a Extract method
5 pick 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 squash 443d809 Move class into separate module
3 squash ce52b63 Inline function
4 squash b5a0f1a Extract method
5 squash 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 s 443d809 Move class into separate module
3 s ce52b63 Inline function
4 s b5a0f1a Extract method
5 s 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```



```
1 pick 535e5b7 Rename variable
2 s 443d809 Move class into separate module
3 s ce52b63 Inline function
4 s b5a0f1a Extract method
5 s 139e19d Fix failing tests
6 pick a9c4c48 Fix bug
7
8 # Commands:
9 # p, pick <commit> = use commit
10 # r, reword <commit> = use commit, but edit the commit message
11 # e, edit <commit> = use commit, but stop for amending
12 # s, squash <commit> = use commit, but meld into previous commit
13 # d, drop <commit> = remove commit
14 # ...
```

```
$ git rebase -i HEAD~6
```

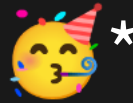


```
1 # This is a combination of 5 commits.
2 # This is the 1st commit message:
3
4 Rename variable
5
6 # This is the commit message #2:
7
8 Move class into separate module
9
10 # This is the commit message #3:
11
12 Inline function
13
14 ...
```

```
$ git rebase -i HEAD~6
```



```
1 # This is a combination of 5 commits.
2 # This is the 1st commit message:
3
4 Refactor applying the boyscout rule
5
6 - Extract method
7 - Inline function
8 - Move class into separate module
9 - Rename variable
10 - Fix failing tests
11 - Rename variable
12
13
14
```



```
$ git log --oneline  
26550fe (HEAD -> bugfix) Fix bug  
73a2261 Refactor applying the boyscout rule
```

```
$ git log 73a2261
```

Refactor applying the boyscout rule

- Extract method
- Inline function
- Move class into separate module
- Rename variable
- Fix failing tests

* MERGE CONFLICTS

Resolve conflicts manually and

```
$ git rebase --continue
```

..or get out

```
$ git rebase --abort
```

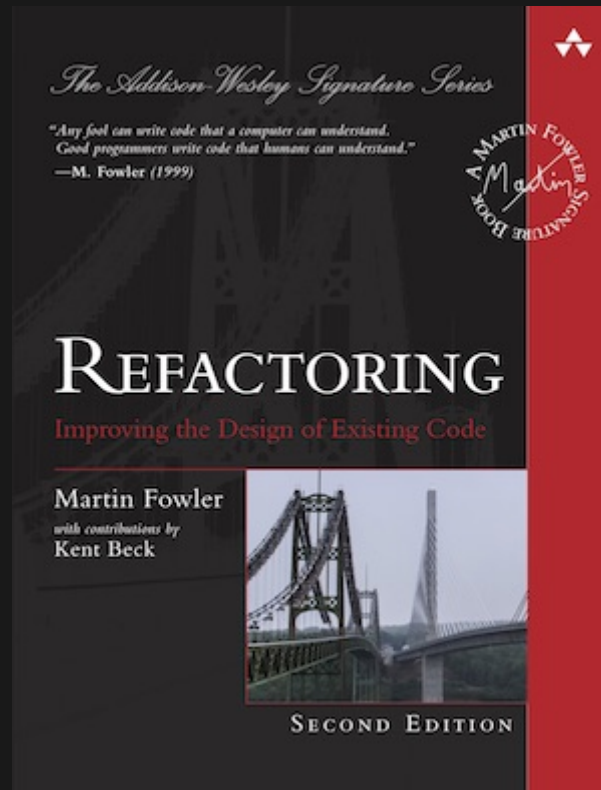
WORKFLOW RECAP

1. Commit early, commit often
..or --patch
2. Revisit your commit boundaries
3. Reshape history

```
$ git rebase -i
```

4. Publish

```
$ git push
```



...small steps are the key to moving quickly,
particularly when working with difficult code.

REFERENCES

- <https://nuclearsquid.com/writings/git-add/>
- <https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History>
- <https://krishansubudhi.github.io/git/2020/01/20/git-rebase-undo.html>
- <https://ohshitgit.com/>
- The wonder of good commit messages (Fawad Malik, Tech Talks Vol. 4, 09.04.2021)

QUESTIONS?

philipp.albrecht@momox.biz

Slack @lens