# My Favorite Advanced Git Features

1) git bisect
2) git commit --amend
3) git add -p
4) git stash
5) git rebase -i

- Samuel Gélineau

1) git bisect
2)
3)
4)
5)

```
*-------------------------*
|       git bisect        |
*-------------------------*
```

Suppose you need to debug a crash.

```
  make
```

```
*-------------------------*
|       git bisect        |
*-------------------------*
```

Suppose you need to debug a crash.

```
  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)
```

```
*------------------------*
|       git bisect       |
*------------------------*
```

But wait, was it crashing before?

```
  git checkout v1.0

  make
```

Didn’t think so.

```
*------------------------*
|       git bisect       |
*------------------------*
```

But wait, was it crashing before?

```
  git checkout v1.0
Note: checking out ’v1.0’.
HEAD is now at bcb064f... simple hello world

  make
```

Didn’t think so.

```
*-------------------------*
|       git bisect        |
*-------------------------*
```

But wait, was it crashing before?

```
  git checkout v1.0
Note: checking out ’v1.0’.
HEAD is now at bcb064f... simple hello world

  make
gcc -W -Wall -o a.out hello.c
./a.out
hello world
```

Didn’t think so.

```
*------------------------*
|       git bisect       |
*------------------------*
```

What has changed since?

```
  git log --oneline v1.0..master
```

```
*------------------------*
|       git bisect       |
*------------------------*
```

What has changed since?

```
  git log --oneline v1.0..master
645581b my commit #30
f7e5658 my commit #29
f133427 my commit #28
4736c17 my commit #27
6cc2629 my commit #26
1e8afd8 my commit #25
8a73009 my commit #24
d7dcfef my commit #23
ce11172 my commit #22
5359641 my commit #21
3fd939c my commit #20
b22eb43 my commit #19
cb94352 my commit #18
c399993 my commit #17
65afaea my commit #16
13c8e33 my commit #15
8b1d5b7 my commit #14
c9a9943 my commit #13
3ad4eaf my commit #12
7849790 my commit #11
306649c my commit #10
4206a79 my commit #9
d3c7183 my commit #8
bbfec29 my commit #7
b5dfaed my commit #6
225c011 my commit #5
```

```
*-------------------------*
|       git bisect        |
*-------------------------*

"git bisect" to the rescue!

                 now broken
                     |   worked
                     |      |
                     v      v
  git bisect start master v1.0
```

```
*------------------------*
|       git bisect       |
*------------------------*

"git bisect" to the rescue!

                 now broken
                     |   worked
                     |      |
                     v      v
  git bisect start master v1.0
Bisecting: 14 revisions left to test after this (roughly 4 steps)
[13c8e33af1371e62b7b4142d188f9e2ab12d8f5a] my commit #15
```

```
*------------------------*
|       git bisect       |
*------------------------*
```

It’s just binary search.

```
  git bisect start master v1.0
Bisecting: 14 revisions left to test after this (roughly 4 steps)
[13c8e33af1371e62b7b4142d188f9e2ab12d8f5a] my commit #15

  make

  git bisect bad

  make

  git bisect good

  make

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

It’s just binary search.

```
  git bisect start master v1.0
Bisecting: 14 revisions left to test after this (roughly 4 steps)
[13c8e33af1371e62b7b4142d188f9e2ab12d8f5a] my commit #15

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad

  make

  git bisect good

  make

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
  git bisect start master v1.0
Bisecting: 14 revisions left to test after this (roughly 4 steps)
[13c8e33af1371e62b7b4142d188f9e2ab12d8f5a] my commit #15

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[bbfec2911b3b14a6eb1aa0a4776411a79f10ed21] my commit #7

  make

  git bisect good

  make

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

```
[13c8e33af1371e62b7b4142d188f9e2ab12d8f5a] my commit #15

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[bbfec2911b3b14a6eb1aa0a4776411a79f10ed21] my commit #7

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good

  make

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[bbfec2911b3b14a6eb1aa0a4776411a79f10ed21] my commit #7

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

```
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[bbfec2911b3b14a6eb1aa0a4776411a79f10ed21] my commit #7

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
  git bisect bad
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[bbfec2911b3b14a6eb1aa0a4776411a79f10ed21] my commit #7

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 1 revision left to test after this (roughly 1 step)
[4206a791a2f2ebc2e6e02953bb1711d094357940] my commit #9

  make

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 1 revision left to test after this (roughly 1 step)
[4206a791a2f2ebc2e6e02953bb1711d094357940] my commit #9

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

```
./a.out
hello world

  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 1 revision left to test after this (roughly 1 step)
[4206a791a2f2ebc2e6e02953bb1711d094357940] my commit #9

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[d3c71832584f38b3aa85b24f2106d438e30a90f5] my commit #8

  make

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
  git bisect good
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[784979017329026ab026f8dfa354e9f4b7c7cae9] my commit #11

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 1 revision left to test after this (roughly 1 step)
[4206a791a2f2ebc2e6e02953bb1711d094357940] my commit #9

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[d3c71832584f38b3aa85b24f2106d438e30a90f5] my commit #8

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

```
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 1 revision left to test after this (roughly 1 step)
[4206a791a2f2ebc2e6e02953bb1711d094357940] my commit #9

  make
gcc -o a.out hello.c
./a.out
Makefile:5: recipe for target ‘test’ failed
make: *** [test] Segmentation fault (core dumped)

  git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[d3c71832584f38b3aa85b24f2106d438e30a90f5] my commit #8

  make
gcc -o a.out hello.c
./a.out
hello world

  git bisect good
4206a791a2f2ebc2e6e02953bb1711d094357940 is the first bad commit
commit 4206a791a2f2ebc2e6e02953bb1711d094357940
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Tue Mar 18 17:14:30 2014 -0400

    my commit #9

:100644 100644 920b26cabe5e8ae160351454fdb27d93162e2f70 9b0f1e9c4e07670469bf5faeaa12d36

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
```

```
hello world

  git bisect good
4206a791a2f2ebc2e6e02953bb1711d094357940 is the first bad commit
commit 4206a791a2f2ebc2e6e02953bb1711d094357940
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Tue Mar 18 17:14:30 2014 -0400

    my commit #9

:100644 100644 920b26cabe5e8ae160351454fdb27d93162e2f70 9b0f1e9c4e07670469bf5faeaa12d36

  git show 4206a791a2f2ebc2e6e02953bb1711d094357940
commit 4206a791a2f2ebc2e6e02953bb1711d094357940
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Tue Mar 18 17:14:30 2014 -0400

    my commit #9

diff --git a/hello.c b/hello.c
index 920b26c..9b0f1e9 100644
--- a/hello.c
+++ b/hello.c
@@ -1,7 +1,7 @@
 #include <stdio.h>

 void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
+  printf("hello %s\n", *to_whom);
 }

 int main() {
```

## Addendum

```
*------------------------*
|       git bisect       |
*------------------------*

Sometimes you want to find the commit which *fixed* a crash.

  # not crashing is "bad" (i.e., the commit you want is earlier)
  git bisect bad

  # crashing is "good" (i.e., the commit you want is later)
  git bisect good

...
```

## Three New Places

in addition to:

* your working copy
* the remote repo

```
Each new place corresponds to a new trick:

      1)
      2) git commit --amend
      3) git add -p
      4) git stash
      5)
```

```
*-------------------------*
|   git commit --amend    |
*-------------------------*

If you know "git push",
you know one of the new places:

       in addition to:

       * your working copy

       * the remote repo
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

If you know "git push",
you know one of the new places:

       in addition to:

       * your working copy
       * your local repo
       * the remote repo
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Same example as usual:

```
  ls

  make
```

```
*-------------------------*
|   git commit --amend    |
*-------------------------*
```

Same example as usual:

```
  ls
main.cpp
Makefile

  make
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Same example as usual:

```
  ls
main.cpp
Makefile

  make
g++ -W -Wall -o a.out main.cpp
./a.out
hello world
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp

  cat hello.h

  cat hello.cpp
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world");
  return 0;
}

  cat hello.h

  cat hello.cpp
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp
[[
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}
]]:!refactor-to-file hello

int main() {
  say_hello("world");
  return 0;
}

  cat hello.h

  cat hello.cpp
```

```
*-------------------------*
|   git commit --amend    |
*-------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}

  cat hello.h

  cat hello.cpp
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}

  cat hello.h
#ifndef HELLO_H
#define HELLO_H

void say_hello(char* to_whom);

#endif

  cat hello.cpp
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

This time, instead of fixing a bug, let’s refactor:

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}

  cat hello.h
#ifndef HELLO_H
#define HELLO_H

void say_hello(char* to_whom);

#endif

  cat hello.cpp
#include "hello.h"
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

git add, git commit...

  git add main.cpp

  git commit -m "refactor to file"
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

git add, git commit...

  git add main.cpp
(no output)

  git commit -m "refactor to file"
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

git add, git commit...

  git add main.cpp
(no output)

  git commit -m "refactor to file"
[master 3478ad7] refactor to file
 1 files changed, 1 insertions(+), 5 deletions(-)
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Or, equivalently:

```
  git commit -am "refactor to file"
```

```
*-------------------------*
|   git commit --amend    |
*-------------------------*
```

Or, equivalently:

```
  git commit -am "refactor to file"
[master 988ed97] refactor to file
 1 files changed, 1 insertions(+), 5 deletions(-)
```

```
*-------------------------*
|   git commit --amend   |
*-------------------------*
```

All right! What’s next?

```
  git status
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

All right! What’s next?

```
  git status
# Untracked files:
#
#       hello.cpp
#       hello.h
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Oops :(

```
  git add hello.*

  git commit -am "forgot to add the refactored files"

  git log --stat
```

```
*-------------------------*
|   git commit --amend    |
*-------------------------*

Oops :(

  git add hello.*
(no output)

  git commit -am "forgot to add the refactored files"

  git log --stat
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

Oops :(

  git add hello.*
(no output)

  git commit -am "forgot to add the refactored files"
[master dafd96b] forgot to add the refactored files
 2 files changed, 16 insertions(+), 0 deletions(-)
 create mode 100644 hello.cpp
 create mode 100644 hello.h

  git log --stat
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

Oops :(

  git add hello.*
(no output)

  git commit -am "forgot to add the refactored files"
[master dafd96b] forgot to add the refactored files
 2 files changed, 16 insertions(+), 0 deletions(-)
 create mode 100644 hello.cpp
 create mode 100644 hello.h

  git log --stat
commit dafd96b13bb67c32edf76101902e2aef0556ce5a
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Wed Mar 19 17:31:34 2014 -0400

    forgot to add the refactored files

 hello.cpp |    8 ++++++++
 hello.h   |    8 ++++++++
 2 files changed, 16 insertions(+), 0 deletions(-)

commit 1f53d7e0a4e42675c4b2c75ee3067430878199fb
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Thu Mar 13 17:31:00 2014 -0400

    refactor to file

 main.cpp |    6 +-----
```

But wait! There is a better way.

But wait! There is a better way.
We haven’t pushed yet.
There is still time to rewrite history!

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Rewrite the commit instead:

```
  git add hello.*

  git commit --amend -am "refactor to hello.{h,c}"

  git log --stat
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

Rewrite the commit instead:

  git add hello.*
(no output)

  git commit --amend -am "refactor to hello.{h,c}"

  git log --stat
```

```
*------------------------*
|   git commit --amend   |
*------------------------*
```

Rewrite the commit instead:

```
  git add hello.*
(no output)

  git commit --amend -am "refactor to hello.{h,c}"
[master 1402c4f] refactor to hello.{h,c}
 3 files changed, 17 insertions(+), 5 deletions(-)
 create mode 100644 hello.cpp
 create mode 100644 hello.h

  git log --stat
```

```
*------------------------*
|   git commit --amend   |
*------------------------*

Rewrite the commit instead:

  git add hello.*
(no output)

  git commit --amend -am "refactor to hello.{h,c}"
[master 1402c4f] refactor to hello.{h,c}
 3 files changed, 17 insertions(+), 5 deletions(-)
 create mode 100644 hello.cpp
 create mode 100644 hello.h

  git log --stat
commit 1402c4f09b7b30129f7249c77c3c379f2cbda033
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Thu Mar 13 17:31:00 2014 -0400

    refactor to hello.{h,c}

 hello.cpp |    8 ++++++++
 hello.h   |    8 ++++++++
 main.cpp  |    6 +-----
 3 files changed, 17 insertions(+), 5 deletions(-)

commit 1444ea5c65d37e40c3ce56427376d197bd3fd3d0
Author: Samuel Gélineau <sgelineau@innobec.com>
Date:   Thu Mar 13 16:16:51 2014 -0400

    simple hello world
```

```
1)
2)
3) git add -p
4)
5)
```

```
*-------------------------*
|       git add -p        |
*-------------------------*

If you know "git add",
you know the second of the new places:

       in addition to:

       * your working copy

       * your local repo

       * the remote repo
```

```
*------------------------*
|       git add -p       |
*------------------------*

If you know "git add",
you know the second of the new places:

       in addition to:

       * your working copy
       * the staging area
       * your local repo

       * the remote repo
```

```
*------------------------*
|       git add -p       |
*------------------------*
```

Same example as before:

```
  cat main.cpp
```

```
*------------------------*
|       git add -p       |
*------------------------*
```

Same example as before:

```
  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("worlp");
  return 0;
}
```

```
*------------------------*
|       git add -p       |
*------------------------*

Same example as before:

  cat main.cpp
[[
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}
]]:!refactor-to-file hello

int main() {
  say_hello("worlp");
  return 0;
}
```

```
*------------------------*
|       git add -p       |
*------------------------*

Same example as before:

  cat main.cpp
[[
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}
]]:!refactor-to-file hello

int main() {
  say_hello("world");
  return 0;
}
```

```
*------------------------*
|       git add -p       |
*------------------------*
```

Only add the typo:

```
  git add -p
```

```
*------------------------*
|       git add -p       |
*------------------------*
```

Only add the typo:

```
  git add -p
diff --git a/main.cpp b/main.cpp
index 39e58d2..9a6d7cf 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]?
```

```
*------------------------*
|       git add -p       |
*------------------------*
```

Only add the typo:

```
  git add -p
diff --git a/main.cpp b/main.cpp
index 39e58d2..9a6d7cf 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
```

Only add the typo:

```
  git add -p
diff --git a/main.cpp b/main.cpp
index 39e58d2..9a6d7cf 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]?
```

Only add the typo:

```
  git add -p
diff --git a/main.cpp b/main.cpp
index 39e58d2..9a6d7cf 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
```

```
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -6,5 +2,5 @@

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,K,g,e,?]?
```

```
@@ -1,10 +1,6 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -6,5 +2,5 @@

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,K,g,e,?]? y
```

```
-}
+#include "hello.h"

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -6,5 +2,5 @@

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,K,g,e,?]? y
```

```
  git commit -m "typo"

  git push
```

```
 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -6,5 +2,5 @@

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,K,g,e,?]? y

  git commit -m "typo"
[master a213819] typo
 1 files changed, 1 insertions(+), 1 deletions(-)

  git push
 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,7 +1,3 @@
-#include <stdio.h>
-
-void say_hello(char* to_whom) {
-  printf("hello %s\n", to_whom);
-}
+#include "hello.h"

 int main() {
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -6,5 +2,5 @@

 int main() {
-  say_hello("worlp");
+  say_hello("world");
   return 0;
 }
Stage this hunk [y,n,q,a,d,/,K,g,e,?]? y

  git commit -m "typo"
[master a213819] typo
 1 files changed, 1 insertions(+), 1 deletions(-)

  git push
To ../example-remote
   0b7a22f..a213819  master -> master
1)
2)
3)
4) git stash
5)
```

```
*-------------------------*
|        git stash        |
*-------------------------*

The last place is less glamorous:

       in addition to:

       * your working copy
       * the staging area
       * your local repo
       * the remote repo
```

```
*------------------------*
|        git stash       |
*------------------------*

The last place is less glamorous:

       in addition to:

       * your working copy    |
       * the staging area     |  more perfect
       * your local repo      |
       * the remote repo      V
```

```
*------------------------*
|        git stash       |
*------------------------*

The last place is less glamorous:

       in addition to:
       * the stash
       * your working copy    |
       * the staging area     |  more perfect
       * your local repo      |
       * the remote repo      V
```

```
*------------------------*
|        git stash       |
*------------------------*
```

Same example as before:

```
  cat main.cpp
```

```
*-------------------------*
|        git stash        |
*-------------------------*

Same example as before:

  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world");
  return 0;
}
```

```
*------------------------*
|        git stash       |
*------------------------*

Same example as before:

  cat main.cpp
[[
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}
]]:!refactor-to-file hello

int main() {
  say_hello("world");
  return 0;
}
```

```
*-------------------------*
|        git stash        |
*-------------------------*
```

Same example as before:

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}
```

```
ALERT - ALERT - ALERT - ALERT - ALERT

  *******************************
  *     Showstopper bug!!1!     *
  * Drop everything and fix it! *
  *******************************

ALERT - ALERT - ALERT - ALERT - ALERT
```

```
*------------------------*
|        git stash       |
*------------------------*
```

But we’re in the middle of something!

```
  cat main.cpp

  git status
```

```
*------------------------*
|        git stash       |
*------------------------*
```

But we’re in the middle of something!

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}

  git status
```

```
*------------------------*
|        git stash       |
*------------------------*
```

But we’re in the middle of something!

```
  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world");
  return 0;
}

  git status
# Untracked files:
#
#       hello.cpp
#       hello.h
```

```
*-----------------------*
|        git stash      |
*-----------------------*
```

"git stash" to the rescue.

```
  git stash

  cat main.cpp
```

```
*------------------------*
|        git stash       |
*------------------------*
```

"git stash" to the rescue.

```
  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world

  cat main.cpp
```

```
*------------------------*
|        git stash       |
*------------------------*
```

"git stash" to the rescue.

```
  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world

  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world");
  return 0;
}
```

```
*------------------------*
|        git stash       |
*------------------------*

"git stash" to the rescue.

  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world

  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world");
  return 0;
}

  (we’re back to the a clean state)
```

```
*------------------------*
|        git stash       |
*------------------------*

"git stash" to the rescue.

  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world

  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world!");
  return 0;
}

  (unlike "git add -p", we can *test* our fix)
```

```
*------------------------*
|        git stash       |
*------------------------*
```

Save the day!

```
  git commit -am "add exclamation mark, fixes showstopper"

  git push
```

```
*------------------------*
|        git stash       |
*------------------------*
```

Save the day!

```
  git commit -am "add exclamation mark, fixes showstopper"
[master a37a39e] add exclamation mark, fixes showstopper
 1 files changed, 1 insertions(+), 1 deletions(-)

  git push
```

```
*------------------------*
|        git stash       |
*------------------------*
```

Save the day!

```
  git commit -am "add exclamation mark, fixes showstopper"
[master a37a39e] add exclamation mark, fixes showstopper
 1 files changed, 1 insertions(+), 1 deletions(-)

  git push
To ../example-remote
   1444ea5..a37a39e  master -> master
```

```
*-------------------------*
|        git stash        |
*-------------------------*
```

...then go back to work.

```
  git stash pop

  cat main.cpp
```

```
*------------------------*
|        git stash       |
*------------------------*
```

...then go back to work.

```
  git stash pop
# Changes not staged for commit:
#
#         modified:   main.cpp
#
# Untracked files:
#
#         hello.cpp
#         hello.h
Dropped refs/stash@{0} (5207fcdedcc03cd299c82c79cb2529120a7cc3b7)

  cat main.cpp
```

```
*------------------------*
|        git stash       |
*------------------------*

...then go back to work.

  git stash pop
# Changes not staged for commit:
#
#         modified:   main.cpp
#
# Untracked files:
#
#         hello.cpp
#         hello.h
Dropped refs/stash@{0} (5207fcdedcc03cd299c82c79cb2529120a7cc3b7)

  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world!");
  return 0;
}
```

```
*------------------------*
|        git stash       |
*------------------------*

...then go back to work.

  git stash pop
# Changes not staged for commit:
#
#         modified:   main.cpp
#
# Untracked files:
#
#         hello.cpp
#         hello.h
Dropped refs/stash@{0} (5207fcdedcc03cd299c82c79cb2529120a7cc3b7)

  cat main.cpp
#include "hello.h"

int main() {
  say_hello("world!");
  return 0;
}

  (our refactoring was applied on top of the exclamation fix)
```

## Addendum

```
*------------------------*
|        git stash       |
*------------------------*
```

When we did this:

```
  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world

  cat main.cpp
#include <stdio.h>

void say_hello(char* to_whom) {
  printf("hello %s\n", to_whom);
}

int main() {
  say_hello("world");
  return 0;
}
```

```
*------------------------*
|        git stash       |
*------------------------*
```

We forgot the two new files.

```
  ls

  git status
```

```
*------------------------*
|        git stash       |
*------------------------*
```

We forgot the two new files.

```
  ls
hello.cpp
hello.h
main.cpp
Makefile

  git status
```

```
*------------------------*
|        git stash       |
*------------------------*
```

We forgot the two new files.

```
  ls
hello.cpp
hello.h
main.cpp
Makefile

  git status
# Untracked files:
#
#       hello.cpp
#       hello.h
```

```
*------------------------*
|        git stash       |
*------------------------*
```

We should have done this instead:

```
  git add .

  git stash
```

```
*------------------------*
|        git stash       |
*------------------------*

We should have done this instead:

  git add .
(no output)

  git stash
```

```
*-------------------------*
|        git stash        |
*-------------------------*

We should have done this instead:

  git add .
(no output)

  git stash
Saved working directory and index state WIP on master: 1444ea5 simple hello world
HEAD is now at 1444ea5 simple hello world
```

```
*------------------------*
|        git stash       |
*------------------------*
```

This time we did stash everything.

```
  ls

  git status
```

```
*-------------------------*
|        git stash        |
*-------------------------*
```

This time we did stash everything.

```
  ls
main.cpp
Makefile

  git status
```

```
*------------------------*
|        git stash       |
*------------------------*
```

This time we did stash everything.

```
  ls
main.cpp
Makefile

  git status
# On branch master
nothing to commit (working directory clean)
```

Branches

```
*------------------------*
|       git branch       |
*------------------------*
```

Each stash is a branch in disguise!

Let’s visualize the branching...

```
  git log --graph --oneline --all --decorate
```

```
*-------------------------*
|        git branch       |
*-------------------------*
```

Each stash is a branch in disguise!

Let’s visualize the branching...

```
  git log --graph --oneline --all --decorate
* 50aa4fd (HEAD, master) add exclamation mark, fixes showstopper
| * 25d90bc (stash) refactor to hello.{h,c}
|/
* 1444ea5 simple hello world
```

```
*------------------------*
|       git branch       |
*------------------------*
```

Here is how to create a branch explicitly:

```
  git checkout -b mybranch
```

And here’s what the new branching looks like:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|       git branch       |
*------------------------*

Here is how to create a branch explicitly:

  git checkout -b mybranch
Switched to a new branch ’mybranch’


And here’s what the new branching looks like:

  git log --graph --oneline --all --decorate
```

```
*------------------------*
|       git branch       |
*------------------------*
```

Here is how to create a branch explicitly:

```
  git checkout -b mybranch
Switched to a new branch ’mybranch’
```

And here’s what the new branching looks like:

```
  git log --graph --oneline --all --decorate
* 50aa4fd (HEAD, mybranch, master) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
*------------------------*
|       git branch       |
*------------------------*
```

Here is how to create a branch explicitly:

```
  git checkout -b mybranch
Switched to a new branch ’mybranch’
```

And here’s what the new branching looks like:

```
  git log --graph --oneline --all --decorate
* 50aa4fd (HEAD, mybranch, master) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

  ("mybranch" and "master" point to the same commit!)

```
*------------------------*
|       git branch       |
*------------------------*

Here is how to create a branch explicitly:

  git checkout -b mybranch
Switched to a new branch ’mybranch’


And here’s what the new branching looks like:

  git log --graph --oneline --all --decorate
* 50aa4fd (HEAD, mybranch, master) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world

  ("mybranch" and "master" point to the same commit!)
  (in svn, creating a branch would have required a folder-copying commit)
```

```
*------------------------*
|       git branch       |
*------------------------*
```

A few commits later, the branches diverge:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|       git branch       |
*------------------------*
```

A few commits later, the branches diverge:

```
  git log --graph --oneline --all --decorate
* 6e16484 (HEAD, master) master commit #3
* 556977b master commit #2
* 8b1d6b7 master commit #1
| * 9d569fe (mybranch) mybranch commit #3
| * 1418975 mybranch commit #2
| * 67e9927 mybranch commit #1
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
*------------------------*
|       git branch       |
*------------------------*
```

Branches are often local and temporary.
Some of us commit each time we save!

But now the history looks a bit silly:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|       git branch       |
*------------------------*
```

Branches are often local and temporary.
Some of us commit each time we save!

But now the history looks a bit silly:

```
  git log --graph --oneline --all --decorate
* b1a1b04 (HEAD, master) master commit #3
* 2b22dbd master commit #2
* c762807 master commit #1
| * 0ede61a (mybranch) WIP add example
| * cedde0f WIP add example
| * 3af17dd WIP add example
| * 3641626 WIP add example
| * 4a121a9 WIP add example
| * ac2096f WIP add example
| * 77e5b57 WIP add example
| * d24a955 WIP add example
| * e1d1bc8 WIP add example
| * 732d475 WIP add comment
| * a4698d2 WIP add comment
| * 03639a0 WIP add comment
| * fd2d53e WIP add comment
| * 166bf0f WIP add comment
| * f472b4c WIP add comment
| * 191f7f1 WIP add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
1)
2)
3)
4)
5) git rebase --interactive
```

```
*------------------------*
|      git rebase -i     |
*------------------------*
```

```
It looks like this:

  git rebase -i branching_point
```

```
*-------------------------*
|      git rebase -i      |
*-------------------------*
```

It looks like this:

```
  EDITOR=cat git rebase -i branching_point

pick 191f7f1 WIP add comment
pick f472b4c WIP add comment
pick 166bf0f WIP add comment
pick fd2d53e WIP add comment
pick 03639a0 WIP add comment
pick a4698d2 WIP add comment
pick d8ea61e WIP add comment
pick 732d475 WIP add comment
pick e1d1bc8 WIP add example
pick d24a955 WIP add example
pick 0a95d99 WIP add example
pick 77e5b57 WIP add example
pick ac2096f WIP add example
pick 4a121a9 WIP add example
pick 3641626 WIP add example
pick 3af17dd WIP add example
pick cedde0f WIP add example
pick 0ede61a WIP add example

# Rebase 50aa4fd..0ede61a onto 50aa4fd
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
```

```
pick 191f7f1 WIP add comment
pick f472b4c WIP add comment
pick 166bf0f WIP add comment
pick fd2d53e WIP add comment
pick 03639a0 WIP add comment
pick a4698d2 WIP add comment
pick d8ea61e WIP add comment
pick 732d475 WIP add comment
pick e1d1bc8 WIP add example
pick d24a955 WIP add example
pick 0a95d99 WIP add example
pick 77e5b57 WIP add example
pick ac2096f WIP add example
pick 4a121a9 WIP add example
pick 3641626 WIP add example
pick 3af17dd WIP add example
pick cedde0f WIP add example
pick 0ede61a WIP add example

# Rebase 0bc1f06..0ede61a onto 0bc1f06
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit’s log message
#  x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

```
pick 191f7f1 WIP add comment
s f472b4c WIP add comment
s 166bf0f WIP add comment
s fd2d53e WIP add comment
s 03639a0 WIP add comment
s a4698d2 WIP add comment
s d8ea61e WIP add comment
s 732d475 WIP add comment
pick e1d1bc8 WIP add example
s d24a955 WIP add example
s 0a95d99 WIP add example
s 77e5b57 WIP add example
s ac2096f WIP add example
s 4a121a9 WIP add example
s 3641626 WIP add example
s 3af17dd WIP add example
s cedde0f WIP add example
s 0ede61a WIP add example

# Rebase 0bc1f06..0ede61a onto 0bc1f06
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit’s log message
#  x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

```
*------------------------*
|      git rebase -i     |
*------------------------*
```

```
The rewritten history is easier to read:

  git log --graph --oneline --all --decorate
```

```
*------------------------*
|      git rebase -i     |
*------------------------*
```

The rewritten history is easier to read:

```
  git log --graph --oneline --all --decorate
* e723dbc (HEAD, master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (mybranch) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Merging the branch into master

Merging the branch into master:
Two Schools of Thought

1) linear history
2) bubble history

```
Merging the branch into master:
    Two Schools of Thought

      1) linear history
      2) bubble history

    VS the default:
      3) spaghetti history
```

```
*------------------------*
|        git merge       |
*------------------------*
```

How to create a spaghetti history:

```
git push

git pull

git push
```

The result:

```
git log --graph --oneline --all --decorate
```

```
*------------------------*
|        git merge       |
*------------------------*
```

How to create a spaghetti history:

```
  git push
To ../example-remote
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to '../example-remote'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

  git pull

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|        git merge       |
*------------------------*
```

How to create a spaghetti history:

```
  git push
To ../example-remote
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to '../example-remote'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

  git pull
Already up-to-date!
Merge made by the 'recursive' strategy.

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|        git merge       |
*------------------------*
```

How to create a spaghetti history:

```
  git push
To ../example-remote
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to '../example-remote'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

  git pull
Already up-to-date!
Merge made by the 'recursive' strategy.

  git push
To ../example-remote
   cd11f27..acbdf11  master -> master
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*-------------------------*
|        git merge        |
*-------------------------*
```

How to create a spaghetti history:

```
  git push
To ../example-remote
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to '../example-remote'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

  git pull
Already up-to-date!
Merge made by the 'recursive' strategy.

  git push
To ../example-remote
   cd11f27..acbdf11  master -> master
```

The result:

```
  git log --graph --oneline --all --decorate
*   acbdf11 (HEAD, origin/master, master) Merge branch 'master' of ../example-remote
|\
| * cd11f27 colleague's commit #1
* | 3da1271 my commit #1
|/
* 1ab7fbc add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
*------------------------*
|        git merge       |
*------------------------*
```

As you and your colleague pull then push,
knit and purl,
knit and purl,
a complex tapestry is woven:

```
  git log --graph --oneline --all --decorate
```

As you and your colleague pull then push,
knit and purl,
knit and purl,
a complex tapestry is woven:

```
  git log --graph --oneline --all --decorate
*   347f605 (HEAD, origin/master, master) Merge branch ’master’ of ../example-remote
|\
| *   c4cedc5 Merge branch ’master’ of /home/Samuel/working/torvalds/slides/example-rem
| |\
| * | 5ba3368 colleague’s commit #3
* | | 4599bfd my commit #3
| |/
|/|
* |   dd2b26f Merge branch ’master’ of ../example-remote
|\ \
| |/
| *   beffa3c Merge branch ’master’ of /home/Samuel/working/torvalds/slides/example-rem
| |\
| * | 1740f4a colleague’s commit #2
* | | 5ac4c30 my commit #2
| |/
|/|
* |   f4d137c Merge branch ’master’ of ../example-remote
|\ \
| |/
| * cd11f27 colleague’s commit #1
* | 3da1271 my commit #1
|/
* 1ab7fbc add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Merging the branch into master:
Two Schools of Thought

1) linear history
2)

```
*------------------------*
|   git merge --rebase   |
*------------------------*
```

Before the merge:

```
  git log --graph --oneline --all --decorate
```

Linearize the history:

```
  git pull --rebase

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --rebase   |
*------------------------*

Before the merge:

  git log --graph --oneline --all --decorate
* e723dbc (origin/master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (HEAD, master) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world


Linearize the history:

  git pull --rebase

  git push


The result:

  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --rebase   |
*------------------------*

Before the merge:

  git log --graph --oneline --all --decorate
* e723dbc (origin/master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (HEAD, master) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world


Linearize the history:

  git pull --rebase
First, rewinding head to replay your work on top of it...
Applying: add comment
Applying: add example

  git push


The result:

  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --rebase   |
*------------------------*

Before the merge:

  git log --graph --oneline --all --decorate
* e723dbc (origin/master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (HEAD, master) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world


Linearize the history:

  git pull --rebase
First, rewinding head to replay your work on top of it...
Applying: add comment
Applying: add example

  git push
To ../example-remote
   e723dbc..7928fed  master -> master


The result:

  git log --graph --oneline --all --decorate
```

```
* e723dbc (origin/master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (HEAD, master) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Linearize the history:

```
  git pull --rebase
First, rewinding head to replay your work on top of it...
Applying: add comment
Applying: add example

  git push
To ../example-remote
   e723dbc..7928fed  master -> master
```

The result:

```
  git log --graph --oneline --all --decorate
* 7928fed (HEAD, origin/master, master) add example
* d05159f add comment
* e723dbc master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
*------------------------*
|   git merge --rebase   |
*------------------------*
```

```
As you and your colleague rebase then push,
detach and append,
detach and append,
the history remains linear:

  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --rebase   |
*------------------------*

As you and your colleague rebase then push,
detach and append,
detach and append,
the history remains linear:

  git log --graph --oneline --all --decorate
* 5ee3352 (HEAD, origin/master, origin/HEAD, master) my commit #3
* cb23350 colleague’s commit #3
* 3a0aa8f my commit #2
* 601ac01 colleague’s commit #2
* 7f20f9f my commit #1
* 1cd3e95 colleague’s commit #1
* 1ab7fbc add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Merging the branch into master:
Two Schools of Thought

1)
2) bubble history

```
*------------------------*
|   git merge --no-ff    |
*------------------------*
```

Always work on a branch:

```
  git log --graph --oneline --all --decorate
```

Create a feature bubble:

```
  git merge --no-ff topic/myfeature

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --no-ff    |
*------------------------*
```

Always work on a branch:

```
  git log --graph --oneline --all --decorate
* e723dbc (HEAD, origin/master, master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (topic/myfeature) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Create a feature bubble:

```
  git merge --no-ff topic/myfeature

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --no-ff    |
*------------------------*
```

Always work on a branch:

```
  git log --graph --oneline --all --decorate
* e723dbc (HEAD, origin/master, master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (topic/myfeature) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Create a feature bubble:

```
  git merge --no-ff topic/myfeature
Merge made by the ’recursive’ strategy.
 main.cpp |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

  git push
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
*------------------------*
|   git merge --no-ff    |
*------------------------*
```

Always work on a branch:

```
  git log --graph --oneline --all --decorate
* e723dbc (HEAD, origin/master, master) master commit #3
* f6c8d9c master commit #2
* 66c586a master commit #1
| * d159449 (topic/myfeature) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Create a feature bubble:

```
  git merge --no-ff topic/myfeature
Merge made by the ’recursive’ strategy.
 main.cpp |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

  git push
To ../example-remote
   e723dbc..5e08a39  master -> master
```

The result:

```
  git log --graph --oneline --all --decorate
```

```
| * d159449 (topic/myfeature) add example
| * 4e461db add comment
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

Create a feature bubble:

```
  git merge --no-ff topic/myfeature
Merge made by the 'recursive' strategy.
 main.cpp |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

  git push
To ../example-remote
   e723dbc..5e08a39  master -> master
```

The result:

```
  git log --graph --oneline --all --decorate
*   5e08a39 (HEAD, origin/master, master) Merge branch 'topic/myfeature'
|\
| * d159449 (topic/myfeature) add example
| * 4e461db add comment
* | e723dbc master commit #3
* | f6c8d9c master commit #2
* | 66c586a master commit #1
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
```

```
*------------------------*
|   git merge --no-ff    |
*------------------------*
```

As you and your colleague
branch then merge,
branch then merge,
the history remains linear, with one bubble per feature:

```
  git log --graph --oneline --all --decorate
  git log --graph --oneline --all --decorate
*   b9c9437 (HEAD, origin/master, master) Merge branch ’topic/myfeature3’
|\
| * f76bc70 (topic/myfeature3) my commit #3
* |   39bc001 Merge branch ’topic/otherfeature3’
|\ \
| |/
|/|
| * 0d4c002 colleague’s commit #3
* |   61045c8 Merge branch ’topic/myfeature2’
|\ \
| |/
|/|
| * 811c400 (topic/myfeature2) my commit #2
* |   5e063e9 Merge branch ’topic/otherfeature2’
|\ \
| |/
|/|
| * d3cf62c colleague’s commit #2
* |   6c56996 Merge branch ’topic/myfeature1’
|\ \
| |/
|/|
| * b6eb7b1 (topic/myfeature1) my commit #1
* |   a582e91 Merge branch ’topic/otherfeature1’
|\ \
| |/
|/|
| * b1af0d6 colleague’s commit #1
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world
*   b9c9437 (HEAD, origin/master, master) Merge branch ’topic/myfeature3’
|\
| * f76bc70 (topic/myfeature3) my commit #3
* |   39bc001 Merge branch ’topic/otherfeature3’
|\ \
| |/
|/|
| * 0d4c002 colleague’s commit #3
* |   61045c8 Merge branch ’topic/myfeature2’
|\ \
| |/
|/|
| * 811c400 (topic/myfeature2) my commit #2
* |   5e063e9 Merge branch ’topic/otherfeature2’
|\ \
| |/
|/|
| * d3cf62c colleague’s commit #2
* |   6c56996 Merge branch ’topic/myfeature1’
|\ \
| |/
|/|
| * b6eb7b1 (topic/myfeature1) my commit #1
* |   a582e91 Merge branch ’topic/otherfeature1’
|\ \
| |/
|/|
| * b1af0d6 colleague’s commit #1
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world

  (notice how the trunk only contains merges)
```

```
|\
| * f76bc70 (topic/myfeature3) my commit #3
* |   39bc001 Merge branch ’topic/otherfeature3’
|\ \
| |/
|/|
| * 0d4c002 colleague’s commit #3
* |   61045c8 Merge branch ’topic/myfeature2’
|\ \
| |/
|/|
| * 811c400 (topic/myfeature2) my commit #2
* |   5e063e9 Merge branch ’topic/otherfeature2’
|\ \
| |/
|/|
| * d3cf62c colleague’s commit #2
* |   6c56996 Merge branch ’topic/myfeature1’
|\ \
| |/
|/|
| * b6eb7b1 (topic/myfeature1) my commit #1
* |   a582e91 Merge branch ’topic/otherfeature1’
|\ \
| |/
|/|
| * b1af0d6 colleague’s commit #1
|/
* 50aa4fd (tag: branching_point) add exclamation mark, fixes showstopper
* 1444ea5 simple hello world

  (notice how the trunk only contains merges)
  (and each feature is a "bubble", starting and ending on the trunk)
```

```
Those were my favorite advanced git features:

      1) git bisect
      2) git commit --amend
      3) git add -p
      4) git stash
      5) git rebase -i

Questions? Other tricks to share?
```