Git Tutorial

Let's git to work

- Version Control Overview
- Recommended Policy
- Basic Skills
  - beginner
  - intermediate
  - advanced
- Algorithmic Problems
- Low-Level Examples

Why Version Control?

Traditional Engineering

Development coordination btw people

Preserve progress

(separate bugs from master)

Track bug origins (blame)

What about rapid prototyping or data work?

Multiple scientists, not software engineers

Focus on rapid prototyping and experiments

Typically easier to rewrite code than read it

Track versions of data, code, & models

Various Group Development Policies

Guidelines

- CI/CD
- Release Strategy
- When to tag, when to version
- Branching/Merging
    - pull requests
    - peer review
    - work organization

Technologies

- CI/CD
- IDEs
- Prod Operations

Many of these technologies are not optimized or mature for data or ML code. A more relaxed dev policy is acceptable.

Common Conventions - **Branching Strategy**

- A `master` branch is "production" or closest thing to it
    - Code is gated by a testing process to reduce presence of bugs
    - Typically source of continuous build cycles and QA testing on pipeline to production
- Side branches are where all development occurs
    - Buggy development code is ok
    - Merge into `master` when feature and testing is complete
    - Can isolate different developer efforts from each other

Common Conventions - **Commit for *Verbosity***

- Document and preserve EVERYTHING!
- Constant checkpointing
- Know exactly where something went wrong
- Not easy to read

```
* d0276e4 - Wed, 9 Mar 2016 18:56:59 -0800 (17 hours ago) (HEAD -> master)
|           fixed a typo "hello wrold" - Michael Stewart
* b5eac1e - Wed, 9 Mar 2016 18:51:59 -0800 (17 hours ago)
|           named a class - Michael Stewart
* cbd35a0 - Wed, 9 Mar 2016 18:50:45 -0800 (17 hours ago)
|           started to write a comment - Michael Stewart
* 9bd3568 - Wed, 9 Mar 2016 17:40:30 -0800 (18 hours ago)
|           fixed indentation - Michael Stewart
* 6520c75 - Wed, 9 Mar 2016 17:36:29 -0800 (18 hours ago)
|           took a 5 minute walk - Michael Stewart
* 4fb0f58 - Wed, 9 Mar 2016 17:32:59 -0800 (18 hours ago)
|           wrote import statements - Michael Stewart
* 54ca0fc - Wed, 9 Mar 2016 17:30:16 -0800 (18 hours ago)
            initial commit - Michael Stewart
```

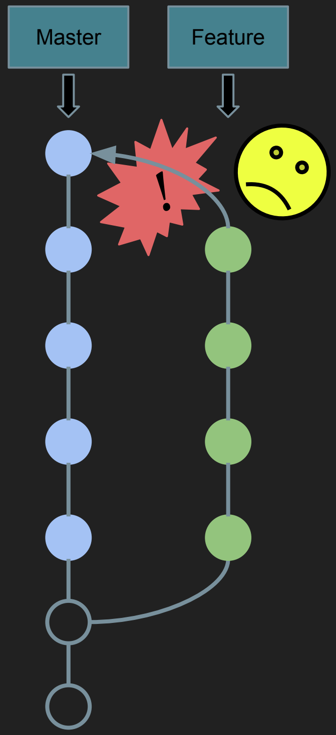
Common Conventions - Commit for *Readability*

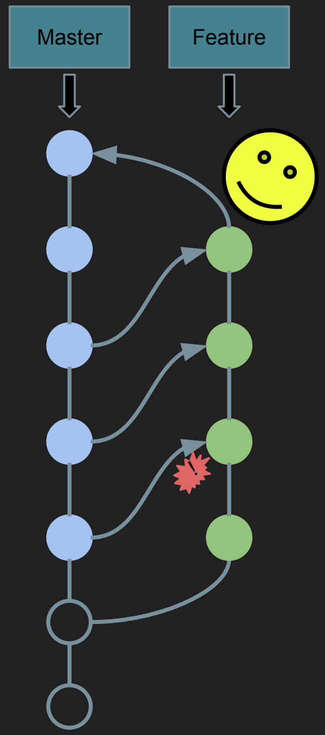
- Fewer checkpoints
- Fast comparison and branching
- Easy to read

```
* b5eac1e - Sun, 7 Feb 2016 17:15:47 -0800 (5 weeks ago)
|         refactoring to use object classes - Michael Stewart
* cbd35a0 - Sat, 6 Feb 2016 18:10:35 -0800 (5 weeks ago)
|         fixed various bugs - Michael Stewart
* acf472c - Fri, 5 Feb 2016 16:27:57 -0800 (5 weeks ago)
|         begun tests - Michael Stewart
* bf45d26 - Thu, 4 Feb 2016 11:33:14 -0800 (5 weeks ago)
|         feature: log printing - Michael Stewart
* 4c0de2c - Wed, 3 Feb 2016 13:15:26 -0700 (5 weeks ago)
|         simplified structure for testing and serialization - Michael Stewart
* 4fb0f58 - Tue, 2 Feb 2016 20:15:35 -0700 (6 weeks ago)
|         initial script format works - Michael Stewart
* 54ca0fc - Mon, 1 Feb 2016 17:01:44 -0700 (6 weeks ago)
          initial commit - Michael Stewart
```

Recommended Version Ctrl. Policy

1. Master branch represents combined group work. Assume it can be updated at any time by others!
2. Feature branch for work in progress
3. Include Jira ticket IDs in commit messages
4. **Stay in-touch!** Merge frequently to integrate master changes into your branch.
5. Always merge from master into your feature branch before going the other way.
6. Pass all tests before merging to master
7. When in doubt:
   a. make a new branch
   b. ask for help merging





frequent merging makes conflicts easier to manage.

Skill Levels

Basic

Minimum needed to work with assistance of others.

Intermediate

See the commits behind the branches.

Advanced

Break anything.

Fix almost anything.

Basic Version Ctrl.

Minimum to get by:

- How to branch
- Verbose/readable commit messages
- Handle conflicts
- How to push/pull

Merge Conflict: Don't Panic!

`git status` gives next steps:

1. Review conflicting files
   - look for "<<<<<<"
   - need your knowledge of code logic
2. mark resolved files with `**git add ...**`
3. `git ... --continue` when done

git help

```
usage: git [--version] [--help] [-C <path>] [-c name=value]

These are common Git commands used in various situations:

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

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

...
```

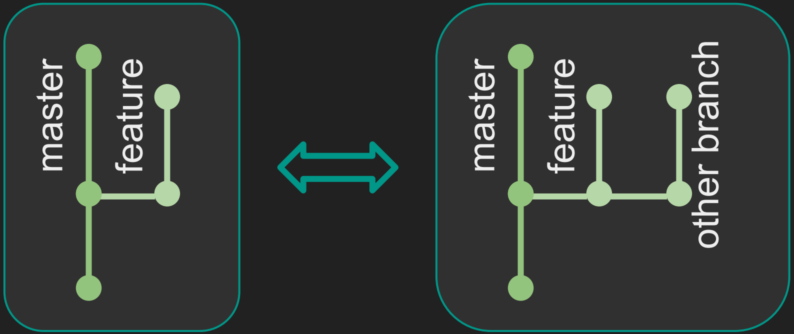
```
$ git help branch
```

```
GIT-BRANCH(1)     Git Manual       GIT-BRANCH(1)

NAME
       git-branch - List, create, or delete branches

SYNOPSIS
       git branch [--color[=<when>] | --no-color] [-r | -a]
               [--list] [-v [--abbrev=<length> | --no-abbrev]]
               [--column[=<options>] | --no-column]
               [(--merged | --no-merged | --contains) [<commit>]]
[<pattern>...]
       git branch [--set-upstream | --track | --no-track] [-l] [-f]
<branchname> [<start-point>]
       git branch (--set-upstream-to=<upstream> | -u <upstream>)
[<branchname>]
       git branch --unset-upstream [<branchname>]
       git branch (-m | -M) [<oldbranch>] <newbranch>
       git branch (-d | -D) [-r] <branchname>...
       git branch --edit-description [<branchname>]
```

Synchronizing between repositories

Everything can be done on one machine.

The final complication is differences between machines.



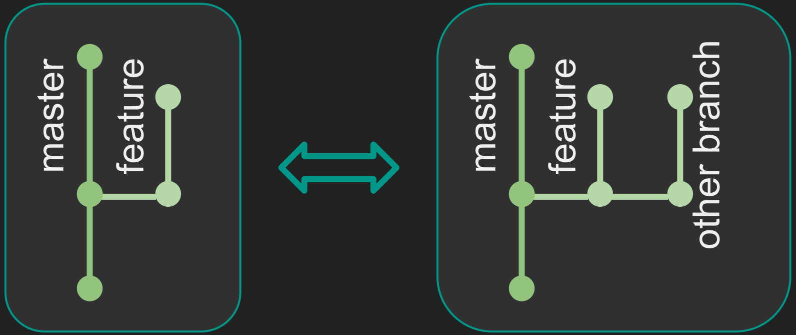
```
$ git branch
  R-correllation-analysis
  big-network-issue-query
* master
$ git branch -a
  R-correllation-analysis
  big-network-issue-query
* master
  remotes/origin/R-correllation-analysis
  remotes/origin/big-network-issue-query
  remotes/origin/master
  remotes/origin/other-branch
```

Synchronizing between repositories

Conflicts during a merge: celebrate and fix them.

Conflicts during a push/pull: gonna have a bad time.

Always solve by branching.



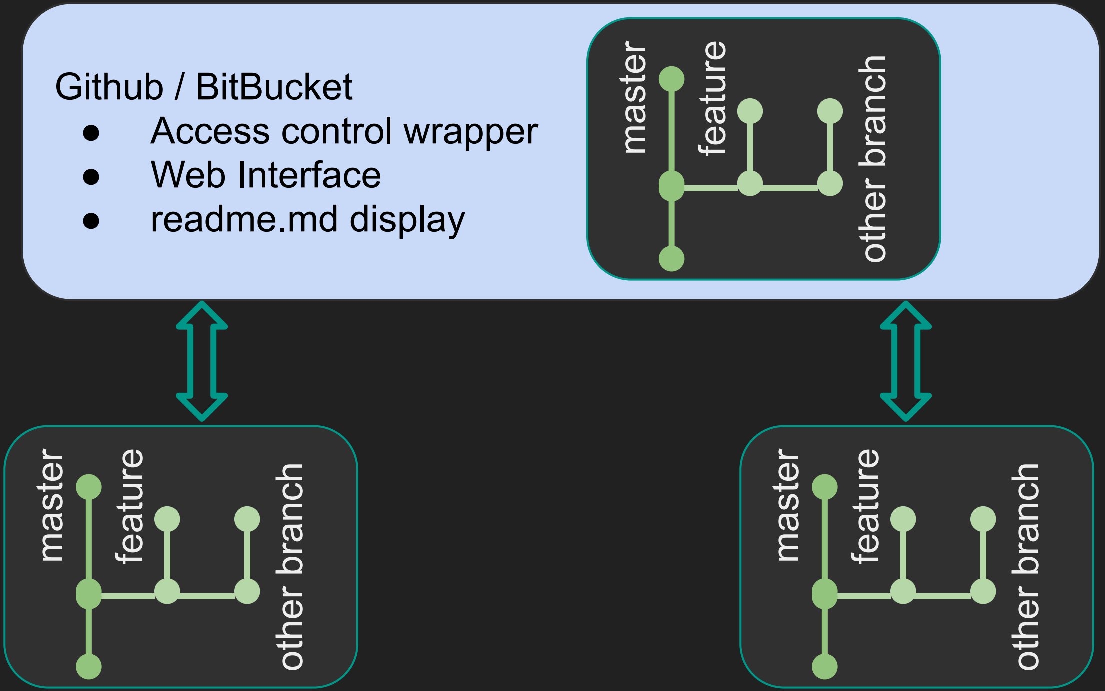
Local branches "track" remotes.

`git clone` sets up sane defaults for tracking:

- 'origin' is name of remote source
- remote branches tracked with same name
  - fetch & pull will update these

You can configure this manually if you *really* want.

Synchronizing between repositories



Github / BitBucket

- Access control wrapper
- Web Interface
- readme.md display

GUIs

- Ungit
- Gitea
- SourceTree
- Github Desktop
- Most IDEs e.g. Eclipse

They will do most everything for you. You won't learn.

Intermediate Version Ctrl. Deltas Everywhere

Checkout, branch, tag any commit.

Stash changes before committing.

Export/import any delta btw files, commits, branches to text file.

Senior dev who fixed a merge conflict
Will you please listen? I'm not the messiah!
Junior devs that caused it
HE IS THE MESSIAH!

Advanced Version Ctrl. Destruction

What they don't know can't hurt them.

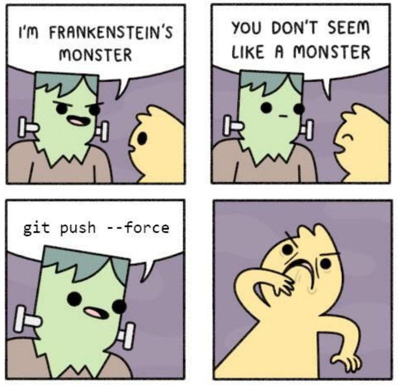
Perform destructive operations in your local branch.

No destructive behavior on branches shared with others.

Destructive changes:

- `git rebase`
- `git commit --amend`
- `git push --force`

  `(never do this)`



A real monster

Advanced Version Ctrl. Destruction

Commit strategies

Rewrite History

- Constant checkpointing
- Readable in detail during development
- At the end of the day, **`git rebase!`**
- Easy to read long-term

```
* d0276e4 - Wed, 9 Mar 2016 18:56:59 -0800 (17 hours ago)
|       fixed a typo "hello wrold" - Michael Stewart
* b5eac1e - Wed, 9 Mar 2016 18:51:59 -0800 (17 hours ago)
|       named a class - Michael Stewart
* cbd35a0 - Wed, 9 Mar 2016 18:50:45 -0800 (17 hours ago)
|       started to write a comment - Michael Stewart
* 9bd3568 - Wed, 9 Mar 2016 17:40:30 -0800 (18 hours ago)
|       fixed indentation - Michael Stewart
* 6520c75 - Wed, 9 Mar 2016 17:36:29 -0800 (18 hours ago)
|       took a 5 minute walk - Michael Stewart
* 4fb0f58 - Wed, 9 Mar 2016 17:32:59 -0800 (18 hours ago)
|       wrote import statements - Michael Stewart
* 54ca0fc - Wed, 9 Mar 2016 17:30:16 -0800 (18 hours ago)
        initial commit - Michael Stewart
```

`git rebase!`

```
* 4fb0f58 - Thu, 10 Mar 2016 11:25:35 -0800 (5 minutes ago)
|       initial script format works - Michael Stewart
* 54ca0fc - Wed, 9 Mar 2016 17:30:16 -0800 (18 hours ago)
        initial commit - Michael Stewart
```

destructive

Git Cheat Sheet

`git commit -am "comment"`	Make a commit with comment. Add all local changes first.
`git commit --amend`	Rewrite the the previous commit
`git rebase -i HEAD~2`	Rewrite the last two commits
`git push origin HEAD`	Share current branch with remote repository "origin"
`git pull`	Get current branch from remote repository (default branch)
`git merge otherbranch`	Merge **`otherbranch`** *into* the current branch
`git rebase otherbranch`	Rebase *onto* **`otherbranch`**
`git checkout -b newbranch`	Create **`newbranch`** and check it out (one-liner)
`git reset --hard HEAD`	Discard all local changes and go back to last commit

Configuration

Some mandatory to function
Some very helpful to function

Three levels of configuration

- System
    - `git config --system ...`
- Global
    - `git config --global ...`
- Repository
    - `git config ...`

Repository overrides Global

Global overrides System

Configuration tips: cache passwords

Via StackOverflow

Git 1.7.9+

```
git config --global credential.helper cache
```

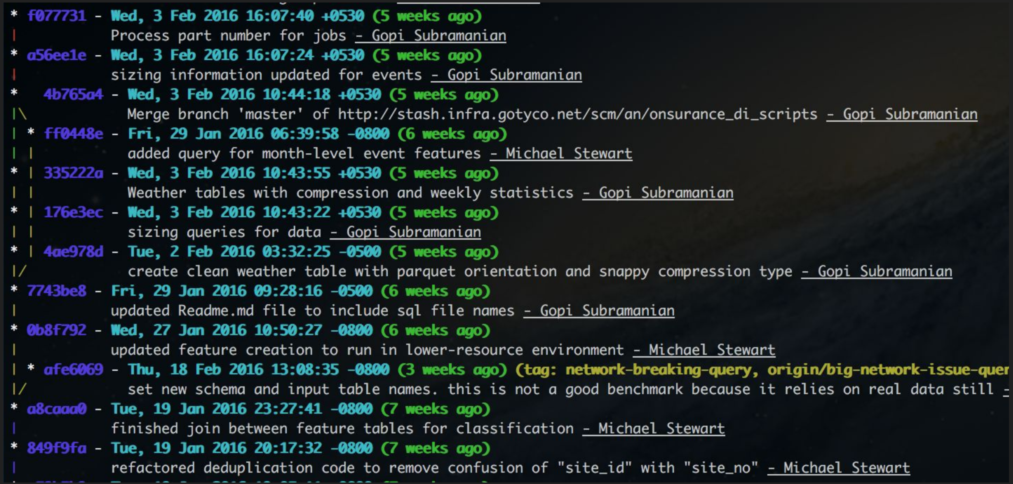
Mac Homebrew

```
git config --global credential.helper osxkeychain
```

```
michael@di-dev-gw1 12:54:44 $ git fetch
Password for 'you':
michael@di-dev-gw1 12:55:01 $ git config --global credential.helper cache
michael@di-dev-gw1 12:55:49 $ git fetch
Password for 'you':
michael@di-dev-gw1 12:55:55 $ git fetch
michael@di-dev-gw1 12:55:57 $
```

Configuration tips: `~/.gitconfig`

```
# This is Git's "global" configuration file.
[user]
        name = Michael Stewart
        email = mikestewart@tycoint.com
[credential]
        helper = osxkeychain
[core]
        editor = vim
[alias]
graph = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold
cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n"          %C(white)%s%C(reset)
%C(dim white)- %an%C(reset)' --all
```



```
* f077731 - Wed, 3 Feb 2016 16:07:40 +0530 (5 weeks ago)
|           Process part number for jobs - Gopi Subramanian
* a56ee1e - Wed, 3 Feb 2016 16:07:24 +0530 (5 weeks ago)
|           sizing information updated for events - Gopi Subramanian
*   4b765a4 - Wed, 3 Feb 2016 10:44:18 +0530 (5 weeks ago)
|\            Merge branch 'master' of http://stash.infra.gotyco.net/scm/an/onsurance_di_scripts - Gopi Subramanian
| * ff0448e - Fri, 29 Jan 2016 06:39:58 -0800 (6 weeks ago)
| |           added query for month-level event features - Michael Stewart
* | 335222a - Wed, 3 Feb 2016 10:43:55 +0530 (5 weeks ago)
| |           Weather tables with compression and weekly statistics - Gopi Subramanian
* | 176e3ec - Wed, 3 Feb 2016 10:43:22 +0530 (5 weeks ago)
| |           sizing queries for data - Gopi Subramanian
* | 4ae978d - Tue, 2 Feb 2016 03:32:25 -0500 (5 weeks ago)
|/            create clean weather table with parquet orientation and snappy compression type - Gopi Subramanian
* 7743be8 - Fri, 29 Jan 2016 09:28:16 -0500 (6 weeks ago)
|           updated Readme.md file to include sql file names - Gopi Subramanian
* 0b8f792 - Wed, 27 Jan 2016 10:50:27 -0800 (6 weeks ago)
|           updated feature creation to run in lower-resource environment - Michael Stewart
| * afe6069 - Thu, 18 Feb 2016 13:08:35 -0800 (3 weeks ago) (tag: network-breaking-query, origin/big-network-issue-quer
|/            set new schema and input table names. this is not a good benchmark because it relies on real data still -
* a8caaa0 - Tue, 19 Jan 2016 23:27:41 -0800 (7 weeks ago)
|           finished join between feature tables for classification - Michael Stewart
* 849f9fa - Tue, 19 Jan 2016 20:17:32 -0800 (7 weeks ago)
|           refactored deduplication code to remove confusion of "site_id" with "site_no" - Michael Stewart
```

Configuration tips: `./.git/config`

```
# This is a config file for a particular repository. It isn't tracked in the repository's version control!
[core]
	repositoryformatversion = 0
	filemode = true
	bare = false
	logallrefupdates = true
	ignorecase = true
	precomposeunicode = true
[remote "origin"]
	url = http://mikestewart_i@stash.infra.gotyco.net/scm/an/mastermind_di_scripts.git
	fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
	remote = origin
	merge = refs/heads/master
[branch "big-network-issue-query"]
	remote = origin
	merge = refs/heads/big-network-issue-query
```

Configuration tips: `./.gitignore`

```
# This is a list of files to ignore. Typically you include files your IDE creates that others won't want.
# This file is tracked within the repository's version control.
# many convenient defaults are available online e.g. Github or via Eclipse IDE

.Rproj.user
.Rhistory
.RData
*.Rproj
```

Recommended reading

The **`git help`** commands

- interactive support right at your fingertips
- **`git help <command>`** for separate manual pages
- Try **`git help workflows`**

Atlassian Git Tutorial on Merging and Rebasing - Well written. I copied their diagrams.

Official Git Documentation - Great details

Oh Shit Git

Git for the Lazy

Algorithmic Problems - Diffing

a.k.a. Longest common subsequence problem
(applications in bioinformatics, Gnu `diff`, version control)
NP-hard problem

```
Roses are red

Violets are blue

Git is easier for one than for two
```

```
Roses are red

Violets are dumb

I'm a manly man!

I don't need flowers!
```

Algorithmic Problems - Diffing

```
Roses are red
Violets are blue
Git is easier for one than for two
```

```
Roses are red
Violets are dumb
I'm a manly man!
I don't need flowers!
```

Algorithmic Problems - Diffing

```
SELECT *
FROM yourtable
WHERE true;
```

→

```
SELECT goodfield
FROM mytable
WHERE true;
```

Δ_1:

- line 1, delete '*'
- line 1, add 'goodfield'
- line 2, delete 'yourtable'
- line 2, add 'mytable'

Δ_2:

- line 1, delete '*\nFROM yourtable'
- line 1, add 'goodfield\n FROM mytable'

Which delta is more efficient to store?
Which delta is faster to identify?

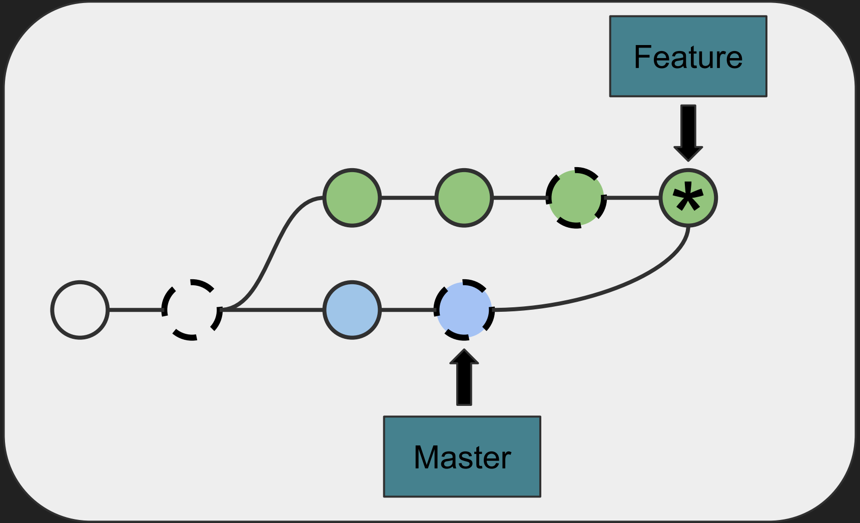
Algorithmic Problems - Diffing

```
Roses are red

Violets are blue

Git is easier for one than for two
```

```
Roses are red

Violets are dumb

I'm a manly man!

I don't need flowers!
```

Git just treats lines as atomic units, throwing out the whole line.
Trade off: better diffing execution time for more space used to store deltas.

Algorithmic Problems - Merging



Git: "resolve these conflicts"

```
<<<<<<< HEAD
=======
>>>>>>> master

<<<<<<< HEAD
same line
=======
same line
>>>>>>> master
```

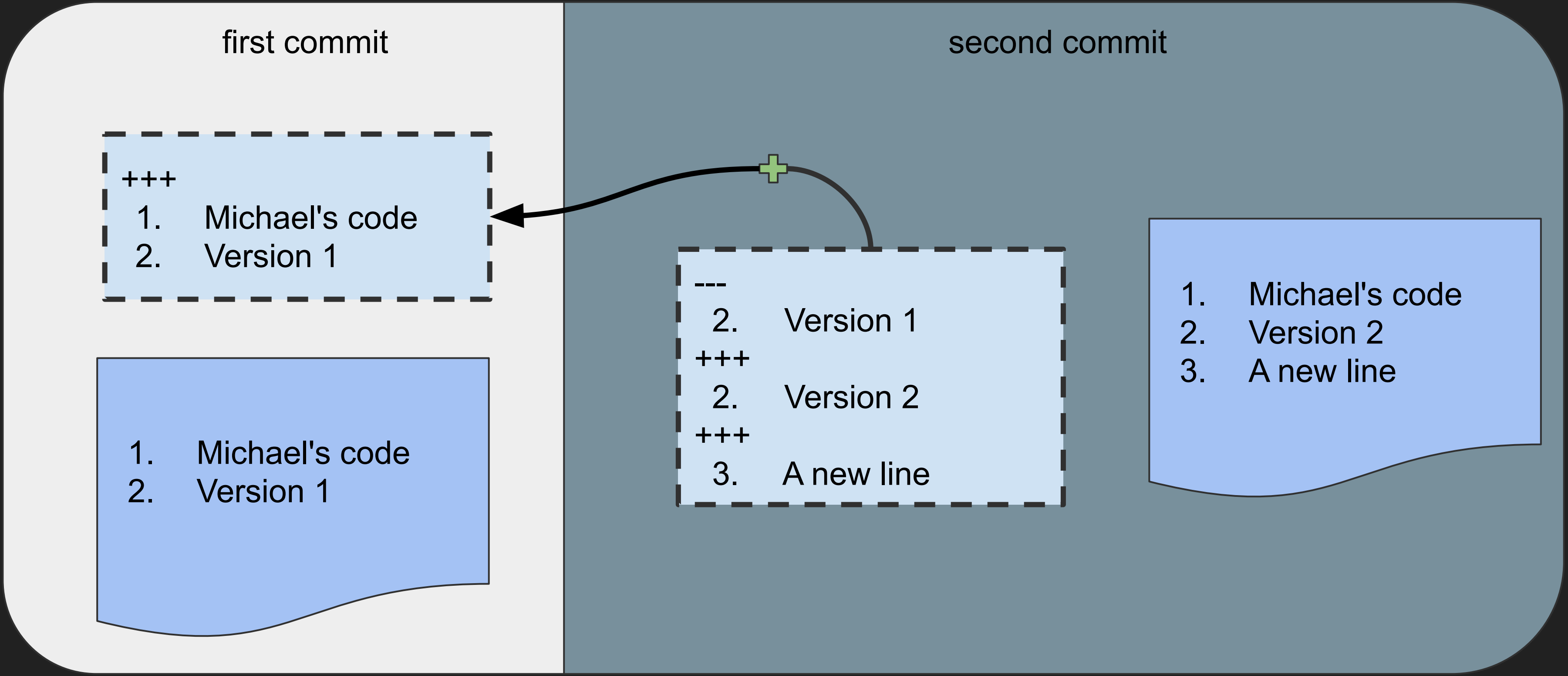
Which 3-way merge was that, again?

Low-Level Examples

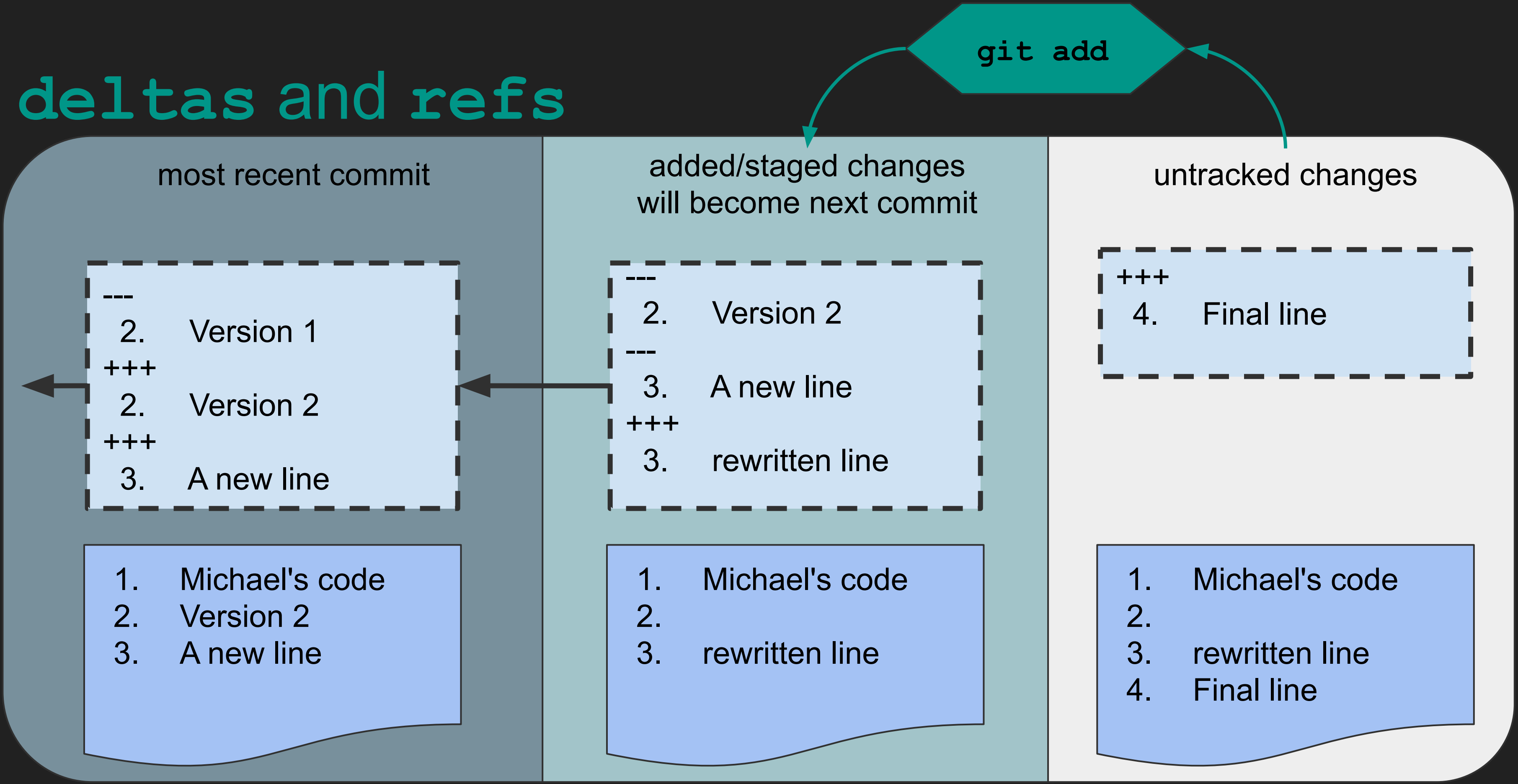
`deltas` and `refs` all the way down.

```
$ git init
$ printf "Michael's code\nVersion 1\n" > my_file.txt
$ git commit -am "initial commit"
```

deltas and refs

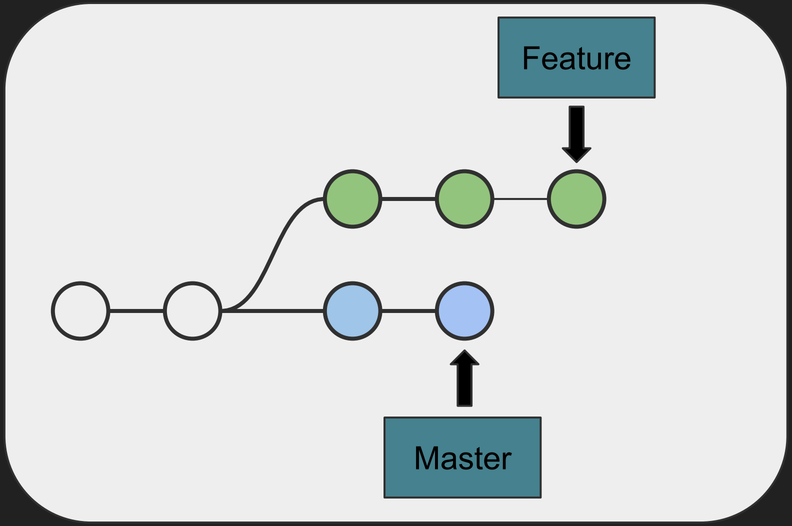


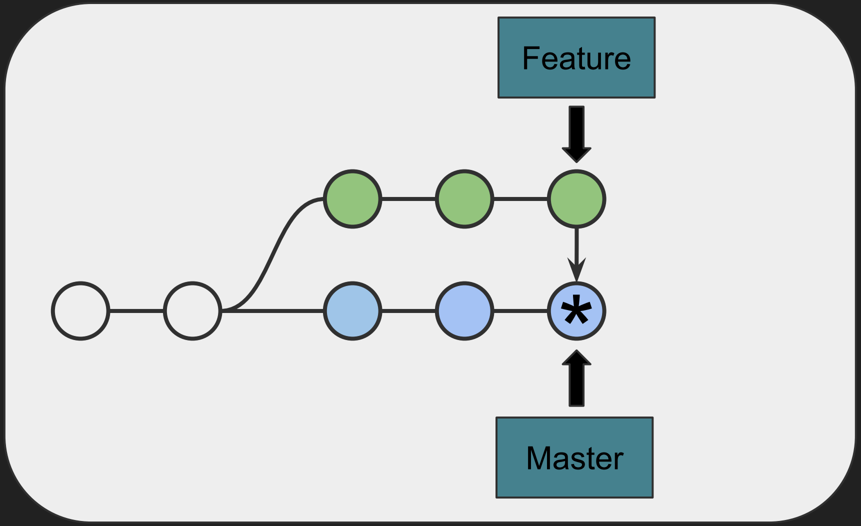
deltas and refs



Merging

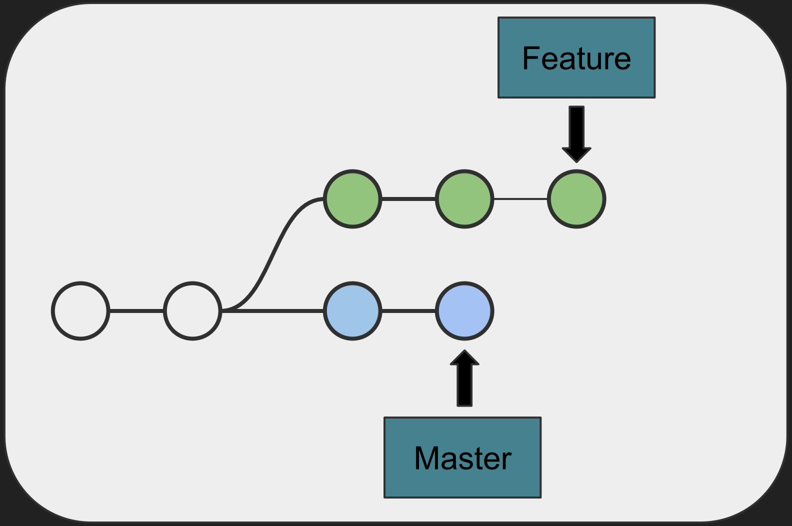
```
$ git checkout Master
$ git merge Feature
```

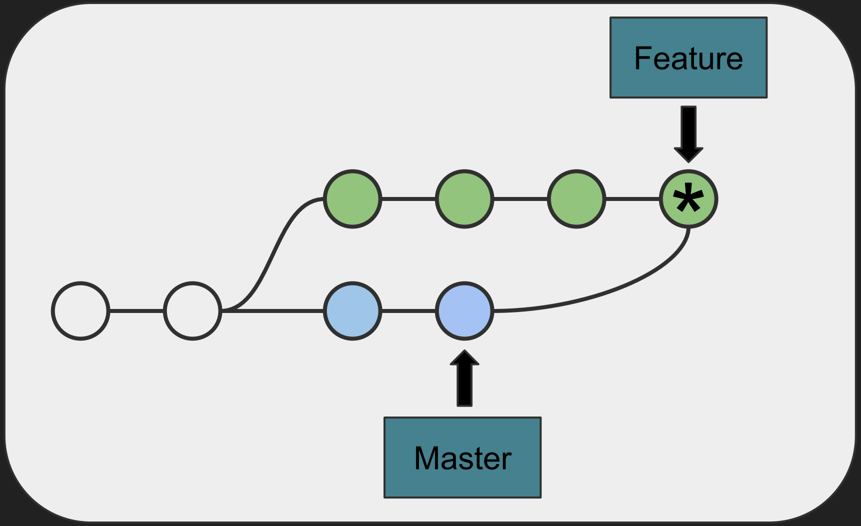




Merging

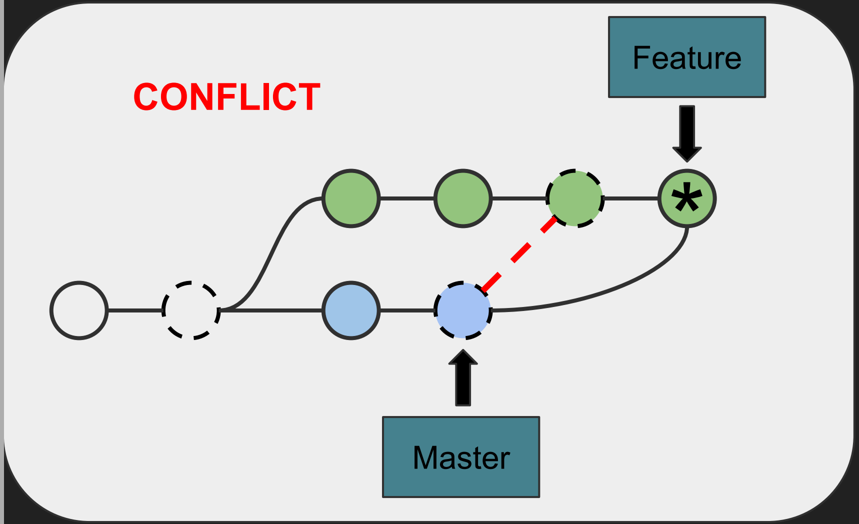
```
$ git checkout Feature
$ git merge Master
```





Merging Conflicts

```
* a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (14 minutes ago) (HEAD -> master)
|           changed to 'main line'
| * b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (15 minutes ago) (feature)
|/          changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (26 minutes ago)
            initial commit
```



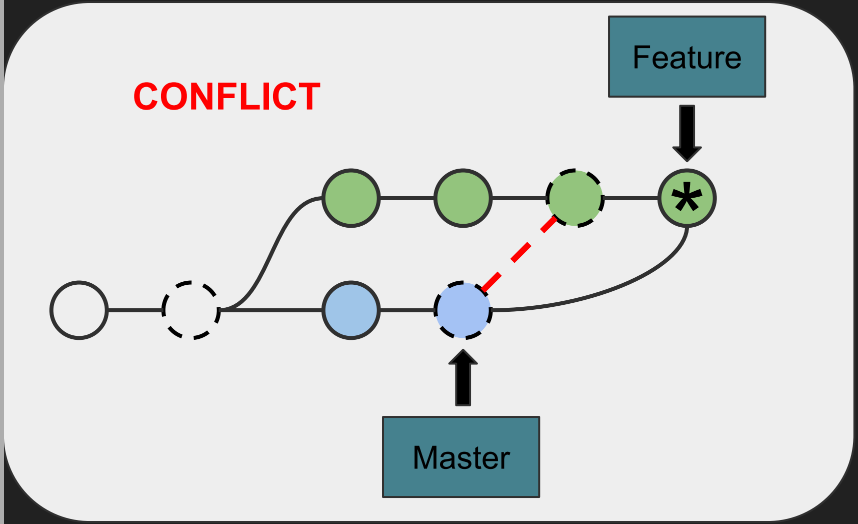
```
$ git checkout feature
$ git merge master
Auto-merging my_file.txt
CONFLICT (content): Merge conflict in my_file.txt
Automatic merge failed; fix conflicts and then commit the
result.


$ git status
On branch feature
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both modified:   my_file.txt
```

We are in an intermediate stage *BEFORE* the new commit is created.

Merging Conflicts

```
* a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (14 minutes ago) (HEAD -> master)
|           changed to 'main line'
| * b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (15 minutes ago) (feature)
|/          changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (26 minutes ago)
            initial commit
```

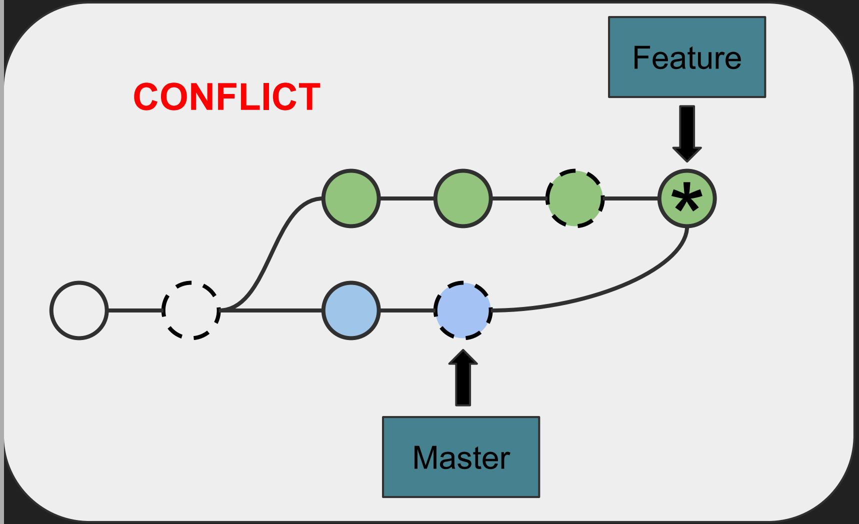
```
$ cat my_file.txt

<<<<<<< HEAD
only line
=======
main line
>>>>>>> master
```



Current changes under "<"

Other branch changes above ">"

Merging Conflicts

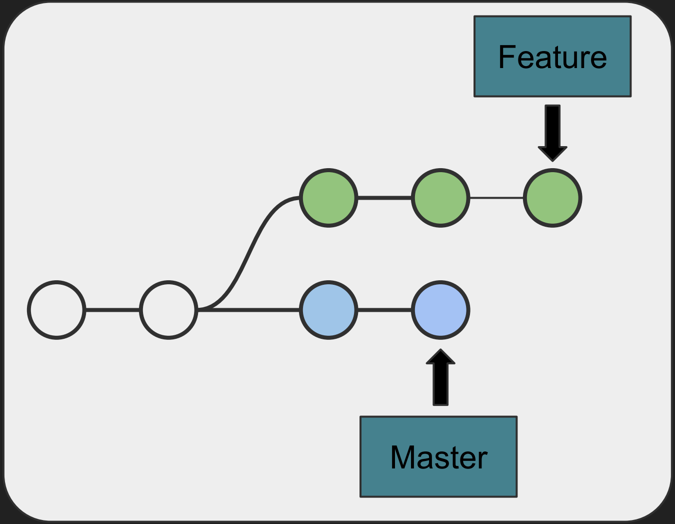


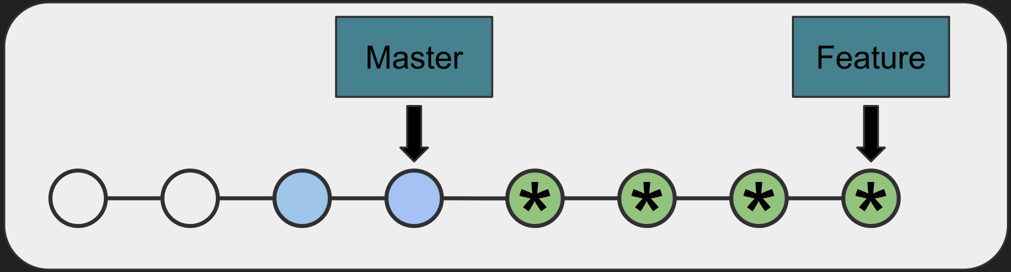
```
* a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (14 minutes ago) (HEAD -> master)
|           changed to 'main line'
| * b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (15 minutes ago) (feature)
|/          changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (26 minutes ago)
            initial commit

$ echo only main line > my_file.txt
$ git add my_file.txt
$ git status
On branch feature
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   my_file.txt
$ git commit -m "resolved conflict"


*   e481b10 - Sun, 13 Mar 2016 19:45:21 -0700 (6 seconds ago) (HEAD -> feature)
|\            resolved conflict
| * a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (56 minutes ago) (master)
| |           changed to 'main line'
* | b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (56 minutes ago)
|/            changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (68 minutes ago)
            initial commit
```

Rebase

Same merge issues, but this time for each commit.

```
* e43b215 - Sun, 13 Mar 2016 20:20:15 -0700 (1 second ago) (HEAD -> second-feature)
|           perfect documentation
* d6a7512 - Sun, 13 Mar 2016 20:19:30 -0700 (46 seconds ago)
|           the best one-liner
| *   e481b10 - Sun, 13 Mar 2016 19:45:21 -0700 (35 minutes ago) (master)
|/|             resolved conflict
* | a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (2 hours ago)
| |           changed to 'main line'
| * b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (2 hours ago)
|/            changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (2 hours ago)
            initial commit
```

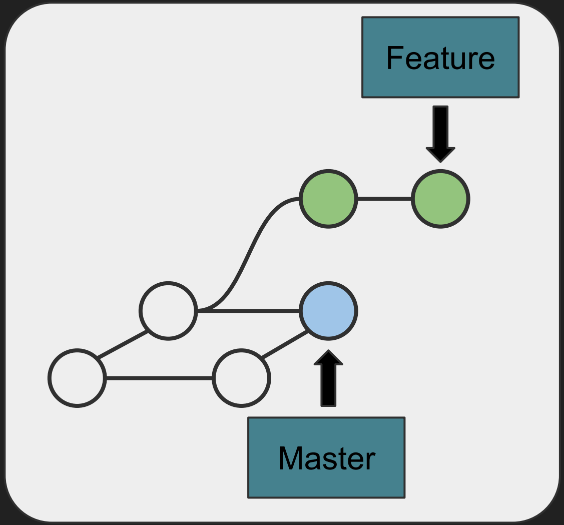




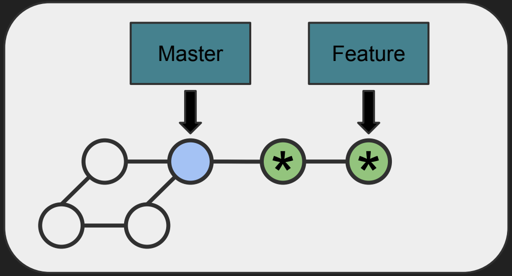
Rebase

```
$ git rebase master
...
CONFLICT (content): Merge conflict in my_file.txt
Failed to merge in the changes.
...
When you have resolved this problem, run "git rebase
--continue".

$ git status
rebase in progress; onto e481b10
You are currently rebasing branch 'second-feature' on
'e481b10'.
  (fix conflicts and then run "git rebase --continue")
  (use "git rebase --skip" to skip this patch)
  (use "git rebase --abort" to check out the original branch)
```

```
$ echo the best one-liner > my_file.txt
$ git add my_file.txt
$ git rebase --continue
Applying: the best one-liner
Applying: perfect documentation
$ cat my_file.txt
the best one-liner
#perfect documentation
$
```



```
* e43b215 - Sun, 13 Mar 2016 20:20:15 -0700 (1 second ago) (HEAD -> second-feature)
|           perfect documentation
* d6a7512 - Sun, 13 Mar 2016 20:19:30 -0700 (46 seconds ago)
|           the best one-liner
| *   e481b10 - Sun, 13 Mar 2016 19:45:21 -0700 (35 minutes ago) (master)
|/|             resolved conflict
* | a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (2 hours ago)
| |           changed to 'main line'
| * b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (2 hours ago)
|/            changed to 'only line'
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (2 hours ago)
            initial commit
```



```
* e275ba7 - Sun, 13 Mar 2016 20:20:15 -0700 (14 minutes ago) (HEAD -> second-feature)
|           perfect documentation - Michael Stewart
* 9189b89 - Sun, 13 Mar 2016 20:19:30 -0700 (15 minutes ago)
|           the best one-liner - Michael Stewart
*   e481b10 - Sun, 13 Mar 2016 19:45:21 -0700 (49 minutes ago) (master)
|\            resolved conflict - Michael Stewart
| * a9e6f14 - Sun, 13 Mar 2016 18:49:36 -0700 (2 hours ago)
| |           changed to 'main line' - Michael Stewart
* | b5a065f - Sun, 13 Mar 2016 18:49:09 -0700 (2 hours ago)
|/            changed to 'only line' - Michael Stewart
* ccac72d - Sun, 13 Mar 2016 18:37:56 -0700 (2 hours ago)
            initial commit
```