

การบริหารจัดการ Source Code ด้วย



git

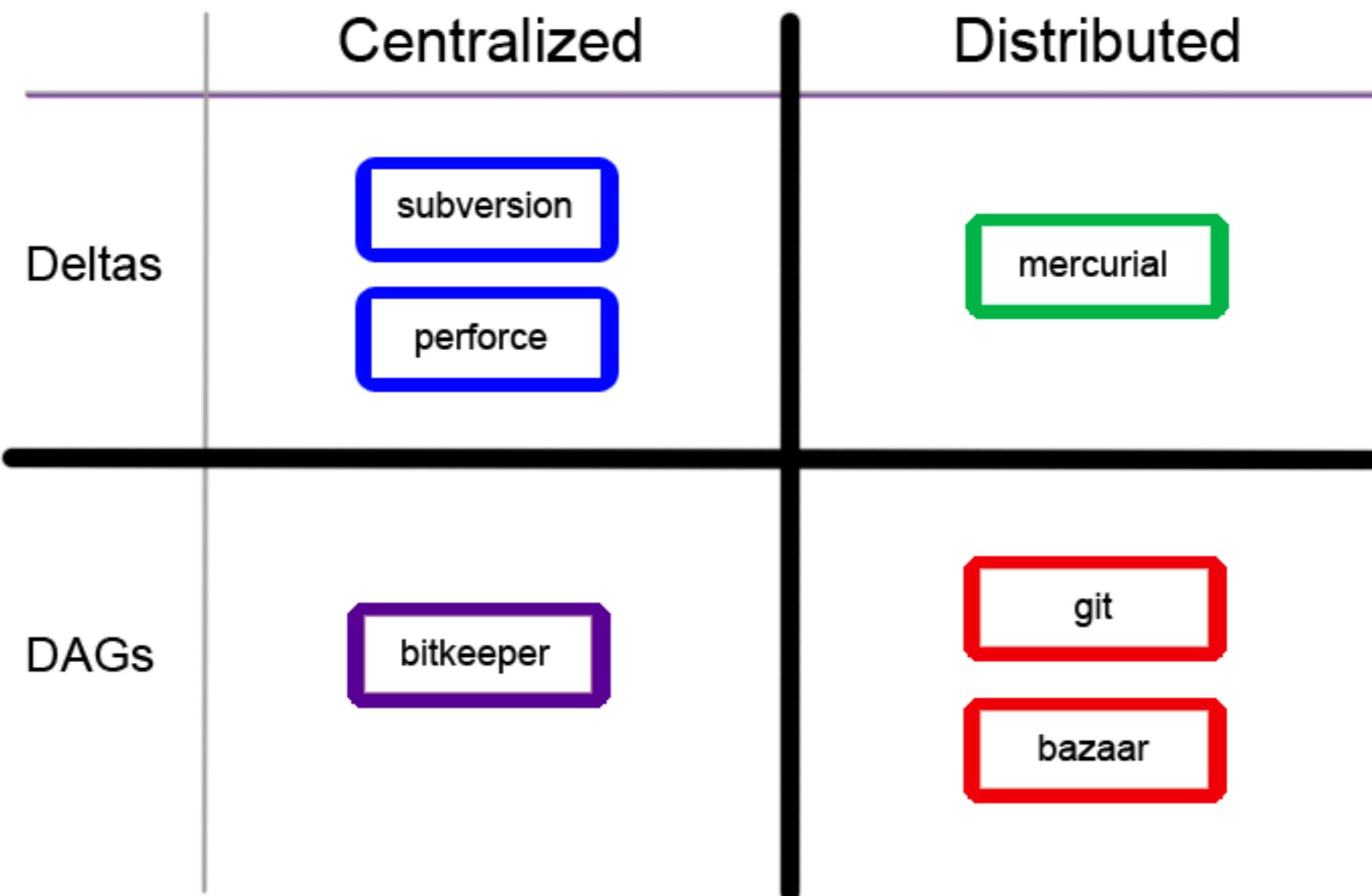


KEEP
CALM
AND
GIT PUSH
ORIGIN MASTER

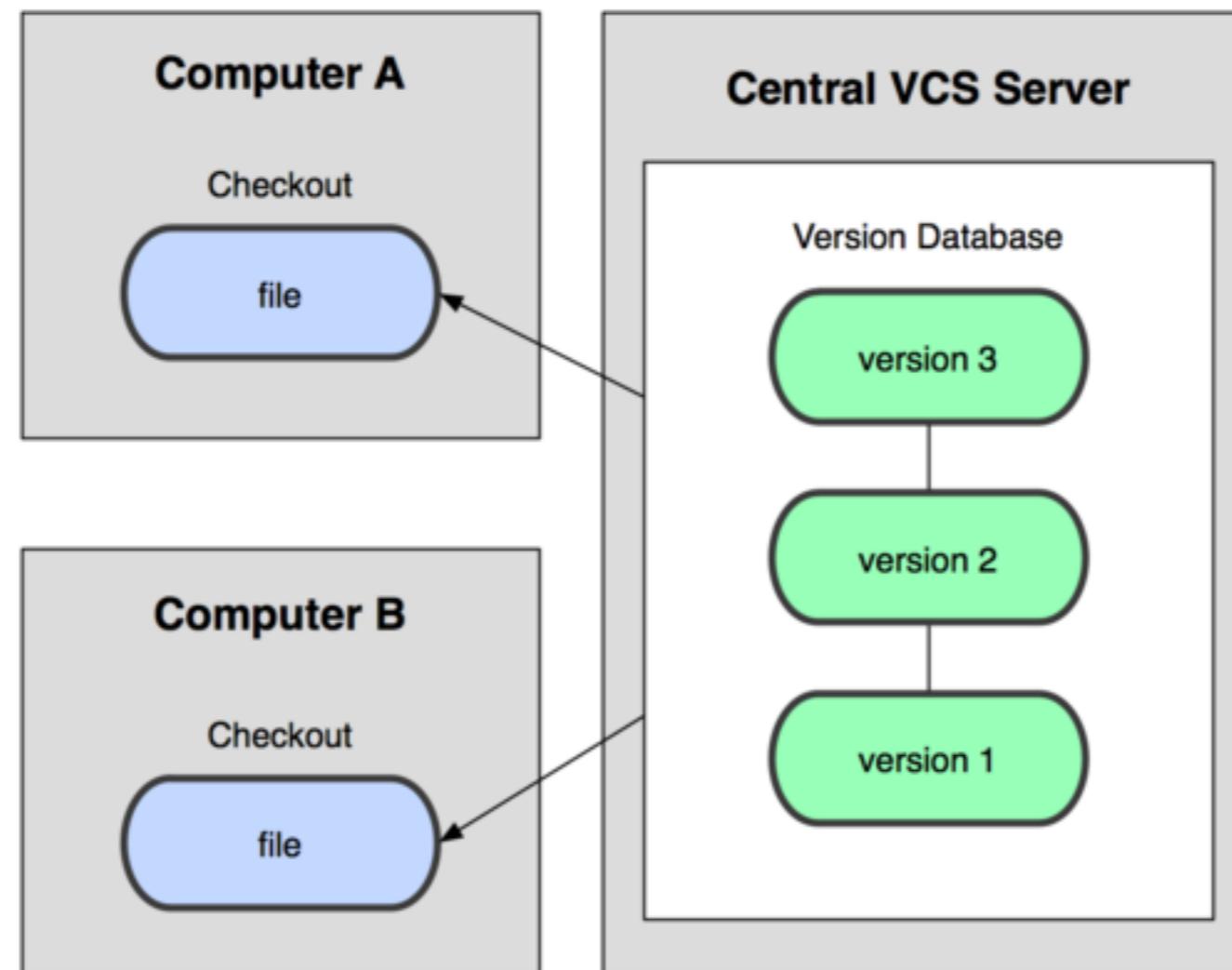
Start with People and Process

Before tools

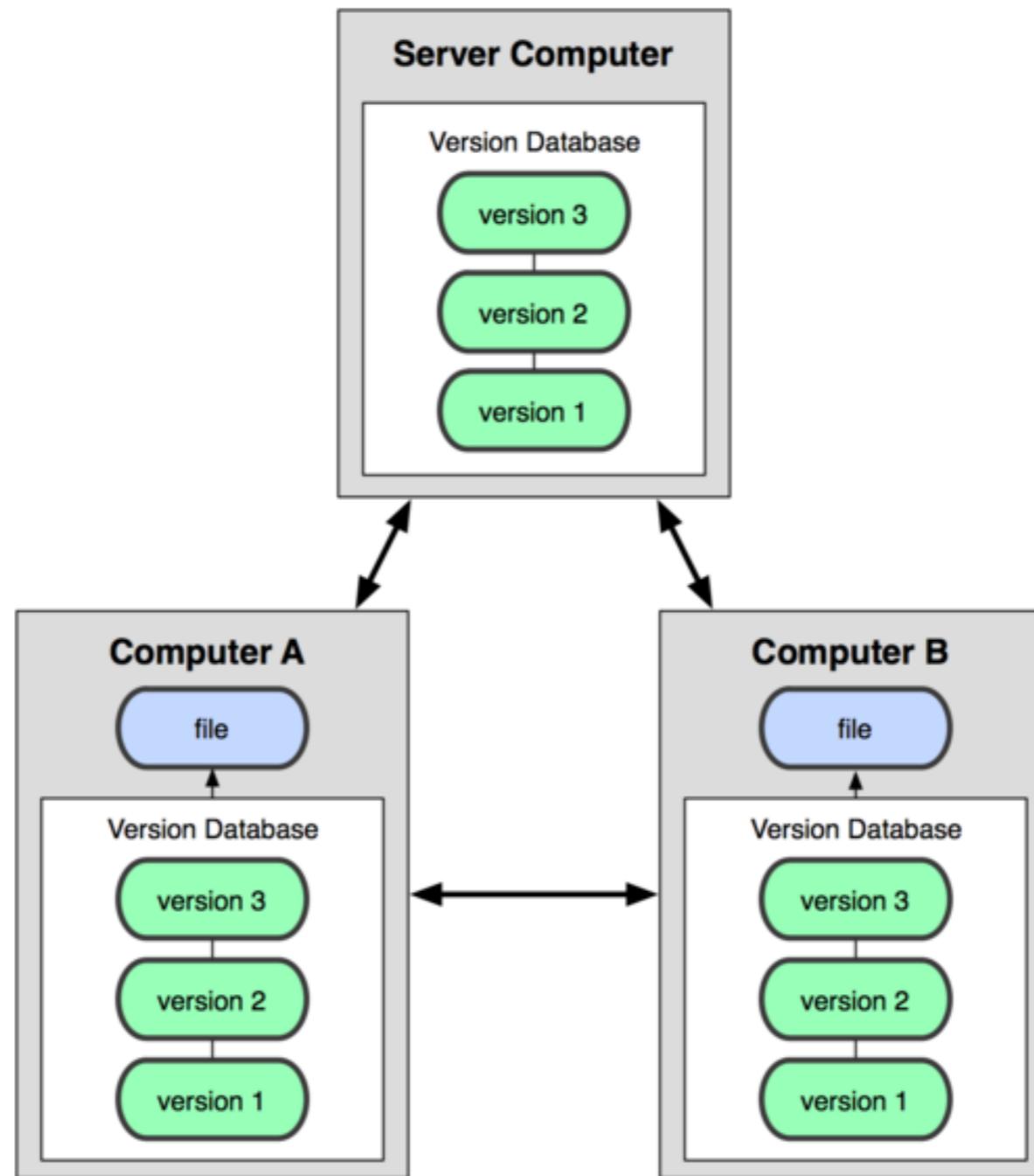
Version Control



Centralized Version Control

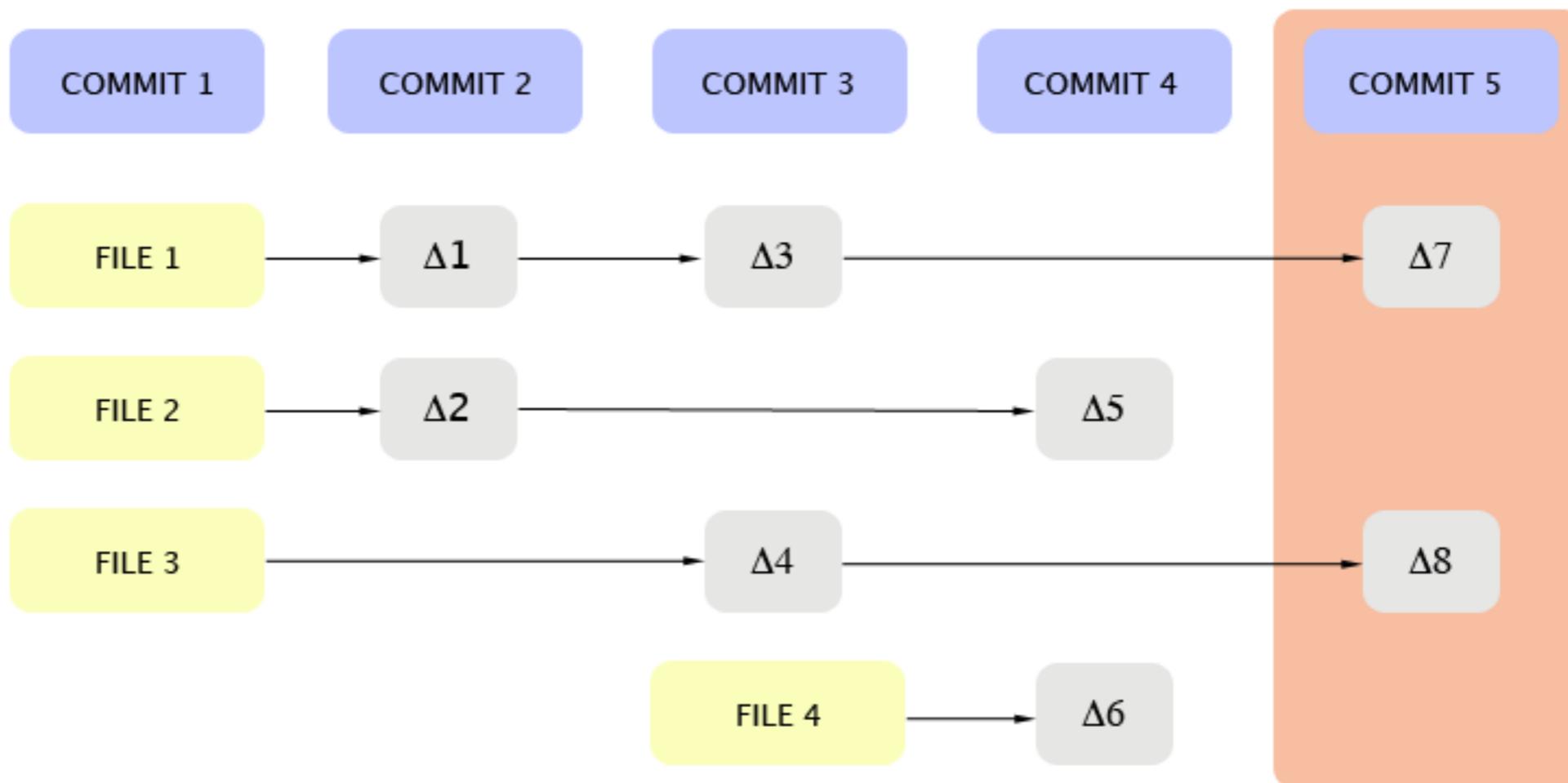


Distributed Version Control



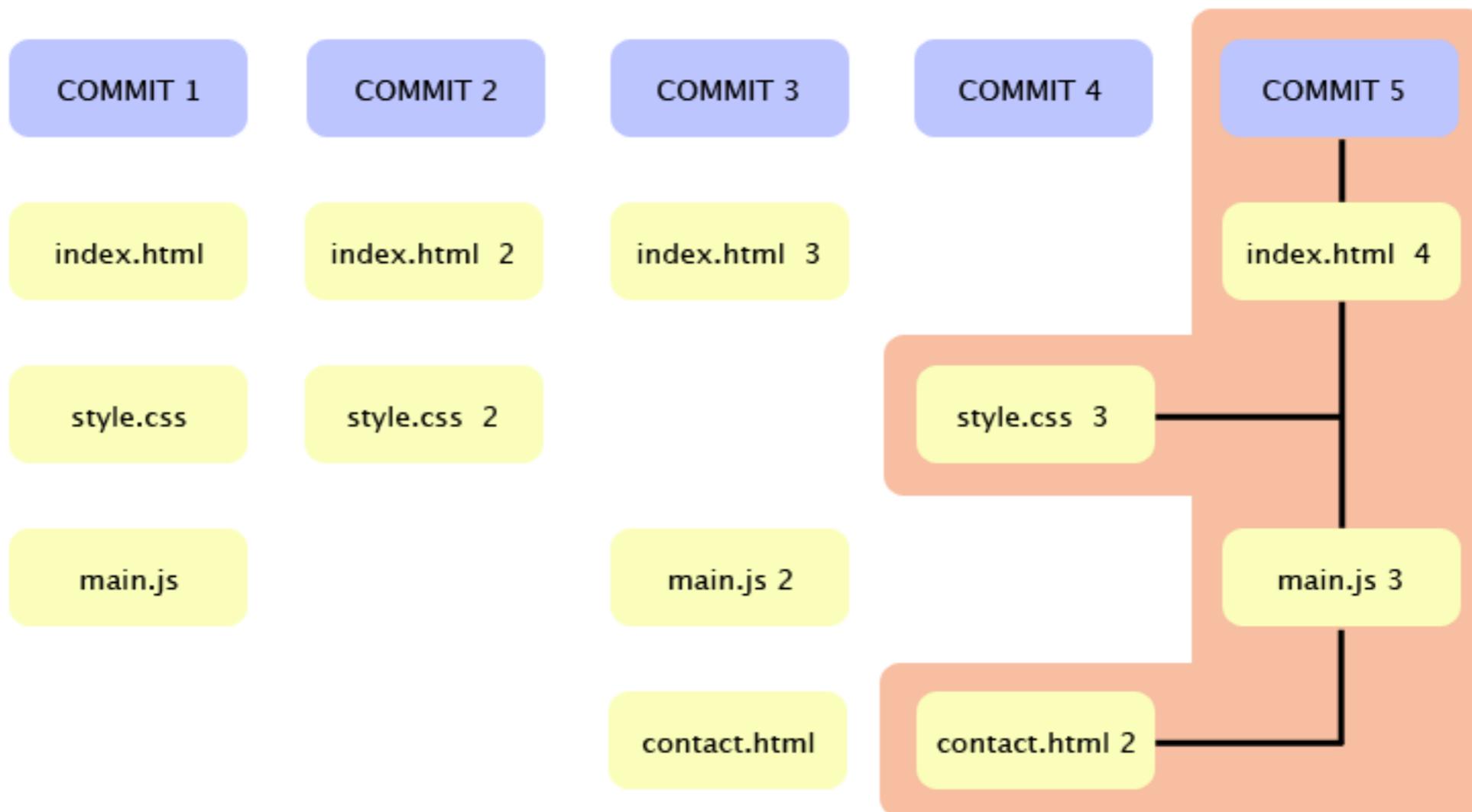
Delta Repository

DELTA-BASED REPOSITORY



DAG Repository

DAG-BASED REPOSITORY



เป้าหมายของการออกแบบ Git

Speed

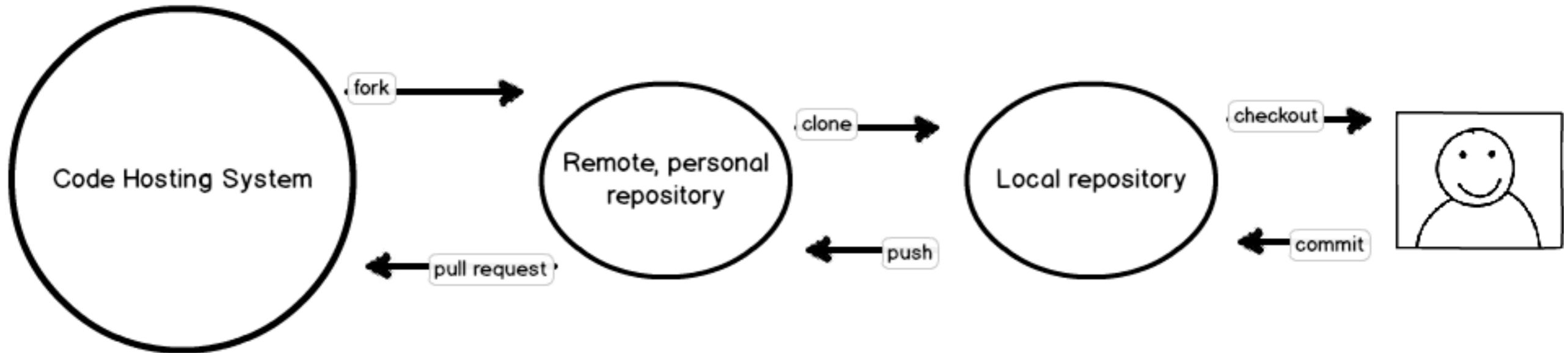
Simple design

Support for many parallel branches

Fully distributed

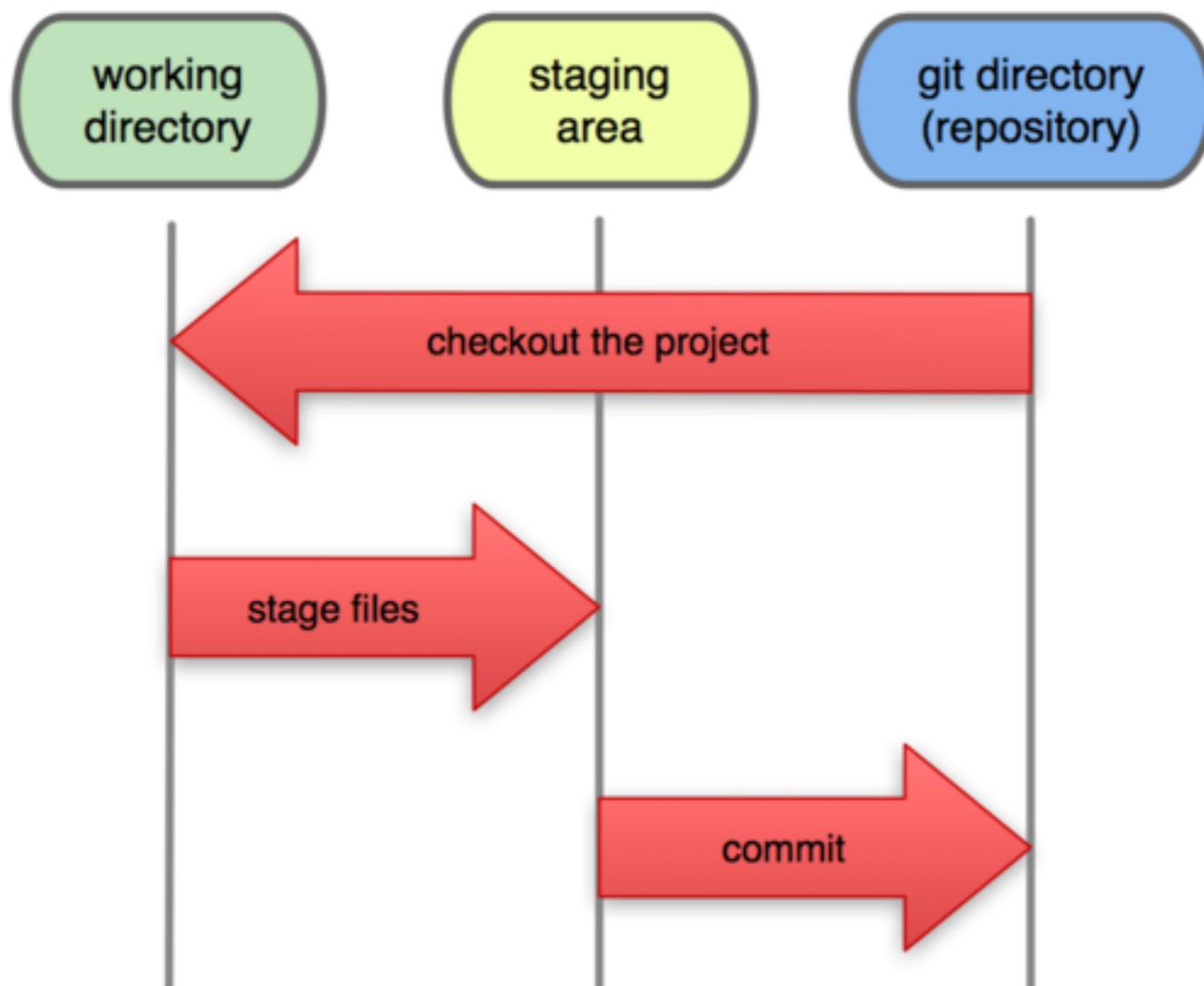
To handle large project like Linux kernel

Workflow



Git Local Workflow

Local Operations



Practice Git on Browser

<https://try.github.io>

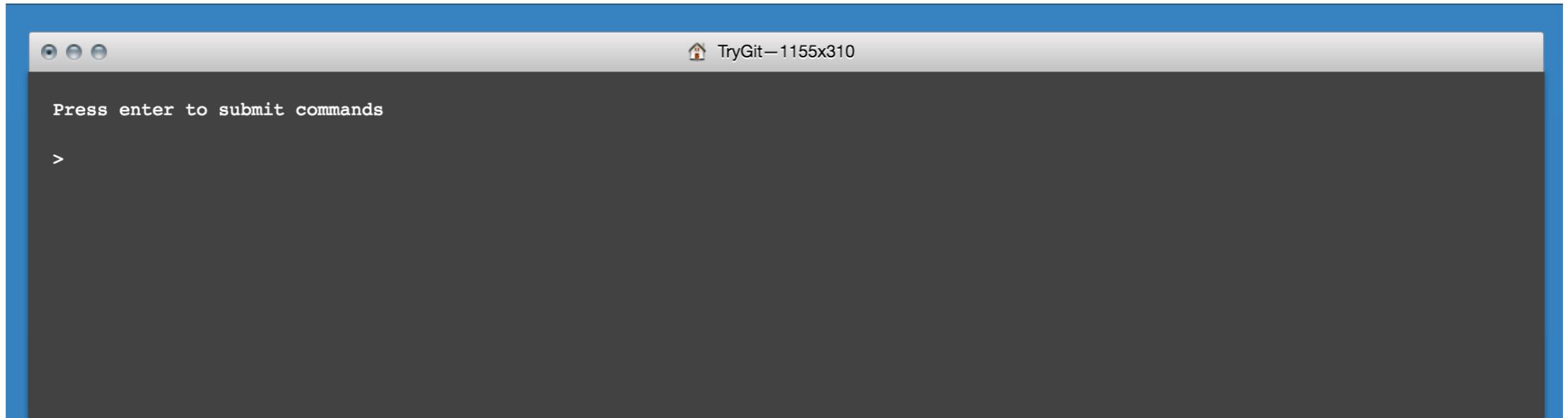


1.1 · Got 15 minutes and want to learn Git?

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in a directory we decided to name "octobox". To initialize a Git repository here, type the following command:

git init



Install Git

<http://git-scm.com/>

The screenshot shows the official website for Git (<http://git-scm.com/>). At the top left is the Git logo (a red diamond with a white 'g' and a black 'it'). Next to it is the slogan "git --distributed-even-if-your-workflow-isnt". A search bar is located at the top right. Below the header, there is a large diagram illustrating a distributed version control system with multiple repositories connected by various colored lines (red, blue, yellow) on a grid background. To the left of the diagram, text describes Git as a free and open source distributed version control system designed for efficiency. Another section highlights its ease of learning and performance, mentioning Subversion, CVS, Perforce, and ClearCase. A "Try Git" button with a GitHub icon is present. The main content area is divided into several sections: "About" (with a gear icon), "Documentation" (with a book icon), "Downloads" (with a download arrow icon), and "Community" (with a speech bubble icon). On the right side, a large callout box displays the "Latest source Release" (2.7.3), "Release Notes (2016-03-10)", and a "Downloads for Mac" button.

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 Learn Git in your browser for free with [Try Git](#).

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development

Latest source Release
2.7.3
Release Notes (2016-03-10)
Downloads for Mac

Configuration Git on Local Machine

```
$git config --global user.name "your name"
```

```
$git config --global user.email "your email"
```

```
$git config -l
```

Create a repository

Windows

Select git bash

Mac or
Linux

Open Terminal

\$mkdir *workspace*

\$git init

Repository structure

\$git **init**

```
.git
├── HEAD
├── branches
├── config
├── description
└── hooks
    ├── applypatch-msg.sample
    ├── commit-msg.sample
    ├── post-update.sample
    ├── pre-applypatch.sample
    ├── pre-commit.sample
    ├── pre-push.sample
    ├── pre-rebase.sample
    ├── prepare-commit-msg.sample
    └── update.sample
├── info
│   └── exclude
└── objects
    ├── info
    └── pack
└── refs
    ├── heads
    └── tags
```

Checking the status

\$touch *README*

\$git **status**

```
On branch master
```

```
Initial commit
```

```
Untracked files:
```

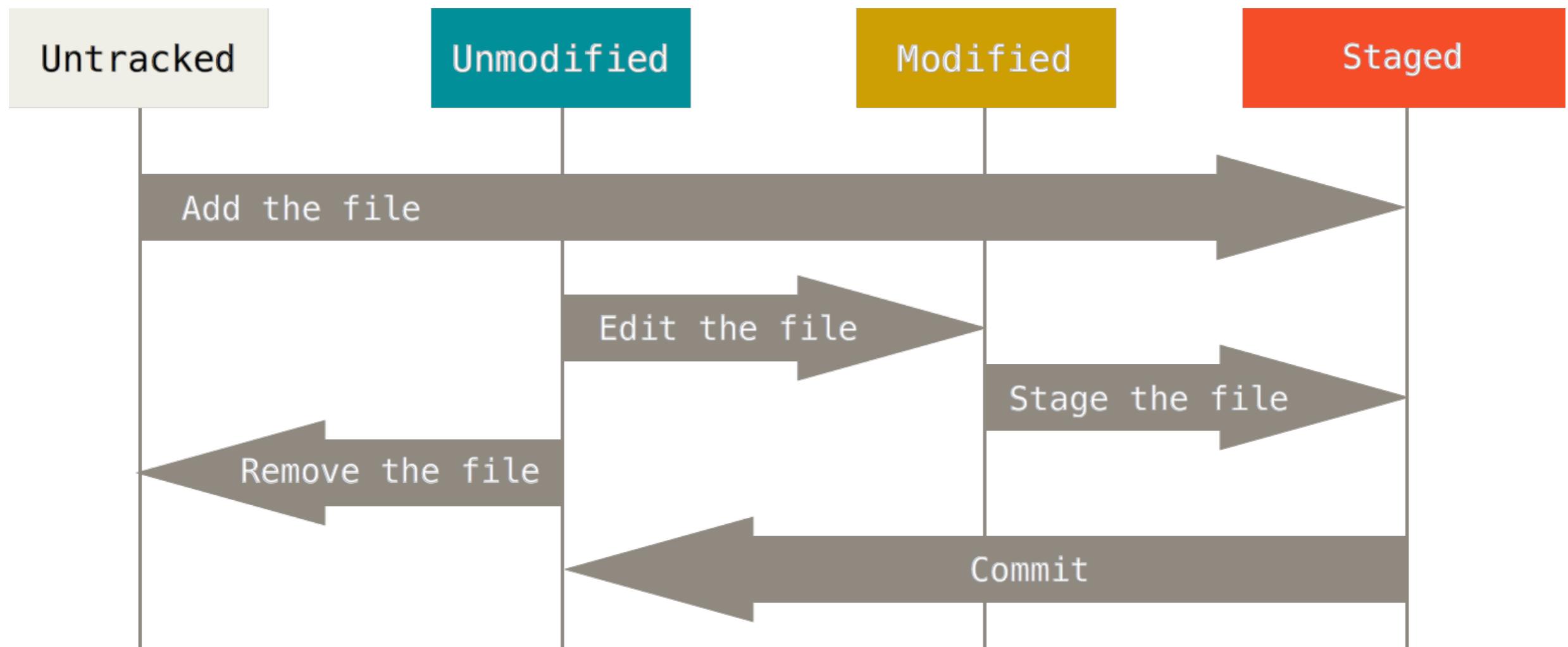
```
(use "git add <file>..." to include in what will be committed)
```

```
 README
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Lift Cycle of File Status

```
$touch README
```



Checking the status

\$git **add** *README*

\$git **status**

On branch master

Initial commit

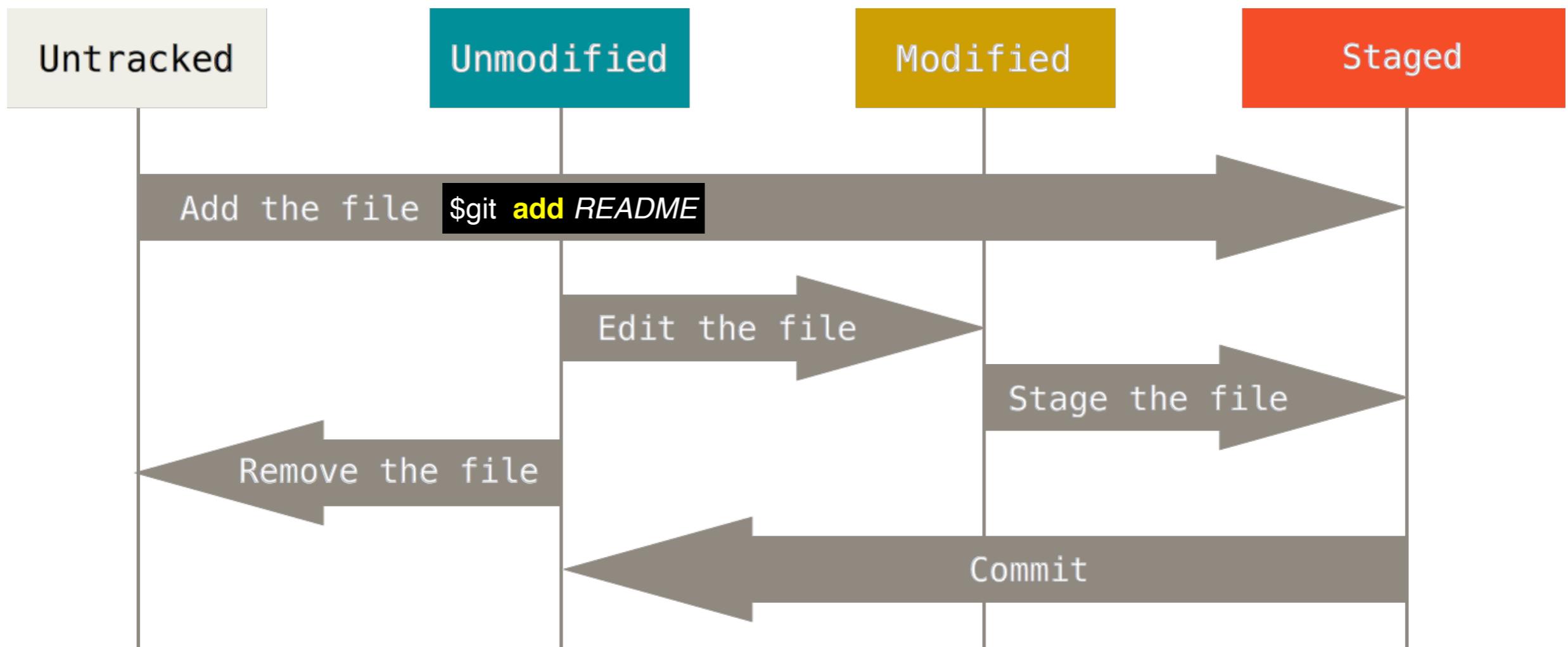
Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README

Lift Cycle of File Status

```
$touch README
```



Ignore files and folders

```
$ touch .gitignore
```

```
$ touch .gitignore
```

```
$ git add .gitignore
```

```
$ git status
```

View staged and unstaged change

```
$ echo "test" > README
```

```
$git status
```

```
On branch master  
Initial commit  
  
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  
  new file:  .gitignore  
  new file:  README  
  
Changes not staged for commit:  
(use "git add <file>..." to update what will be committed)  
(use "git checkout -- <file>..." to discard changes in working directory)  
  
  modified:   README
```

See difference of file

\$git **diff**

```
diff --git a/README b/README
index e69de29..9daeafb 100644
--- a/README
+++ b/README
@@ -0,0 +1 @@
+test
```

See difference of file with history

\$git **diff** --cached

\$git **diff** --staged

```
diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..e69de29
diff --git a/README b/README
new file mode 100644
index 0000000..e69de29
```

Commit your changes

\$git **commit** -m “*Message for this change*”

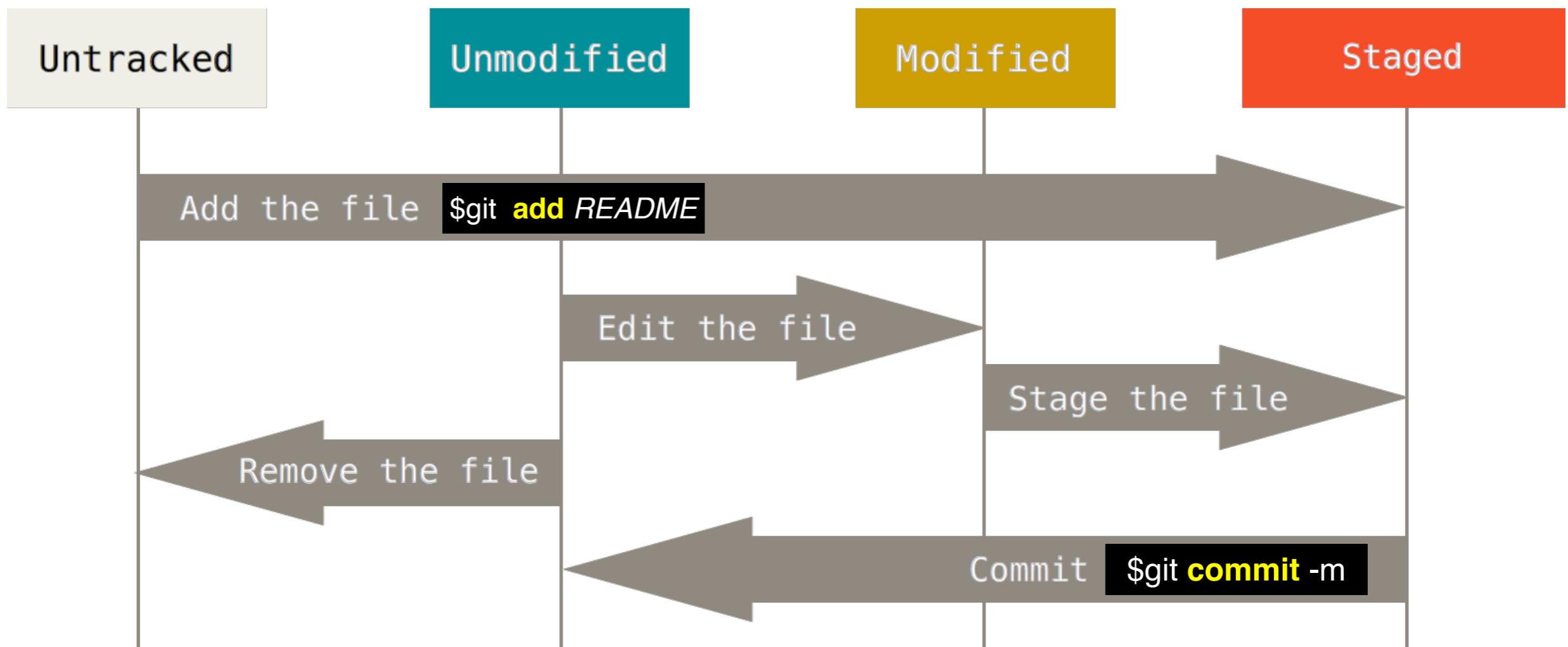
```
[master (root-commit) aed0048] Your message  
2 files changed, 1 insertion(+)  
create mode 100644 .gitignore  
create mode 100644 README
```

Tips

\$git **commit** -a -m “*Message for this change*”

Lift Cycle of File Status

```
$touch README
```



Remove your files

\$rm *README*

\$git **status**

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add/rm <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
    deleted:    README
```

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

Remove your files

\$git **rm** *README*

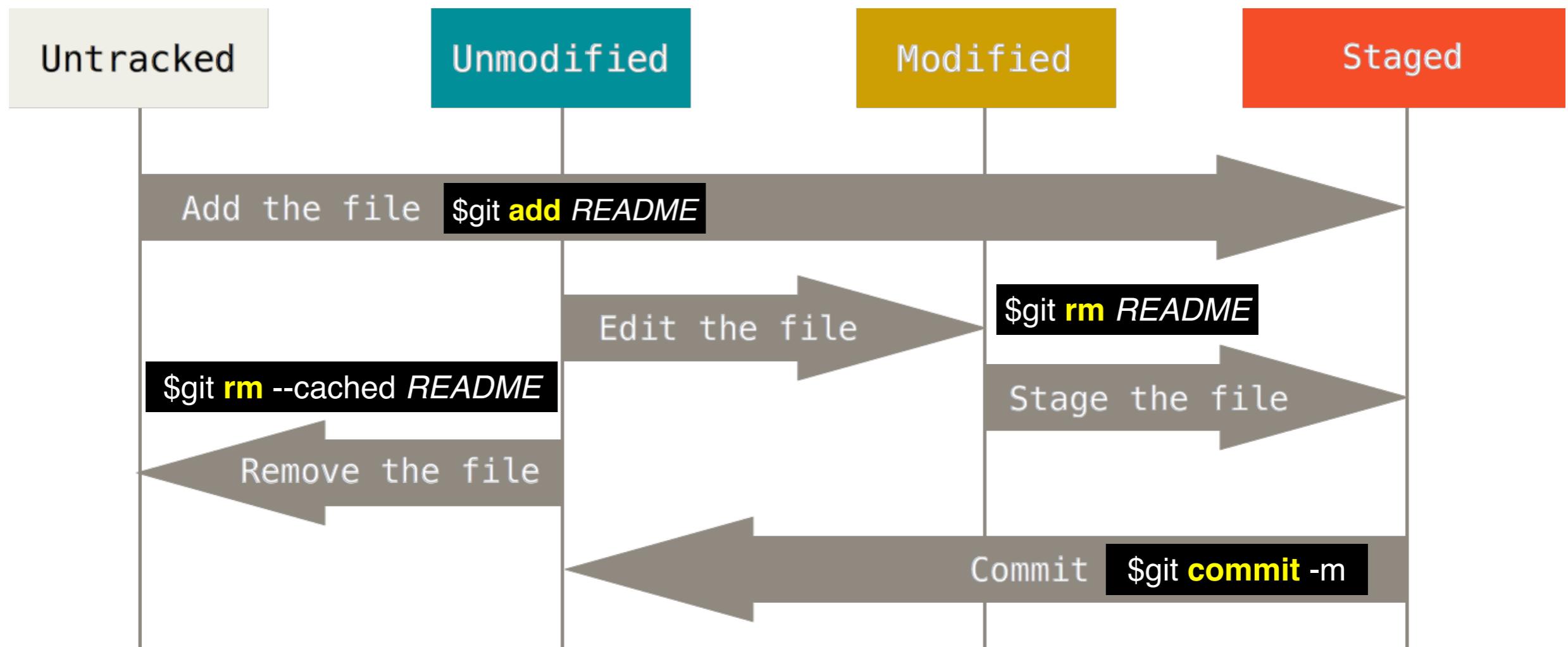
\$git **status**

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    deleted:    README
```

Lift Cycle of File Status

```
$touch README
```



Try to reset README file

\$git WHAT COMMAND?

\$git help

\$git reset HEAD README

\$git status

On branch master
nothing to commit, working directory clean

Move your files

\$git **mv** *README README.MD*

\$git **status**

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README -> README.md
```

Move your files

```
$git mv README README.MD
```

```
$mv README README.MD
```

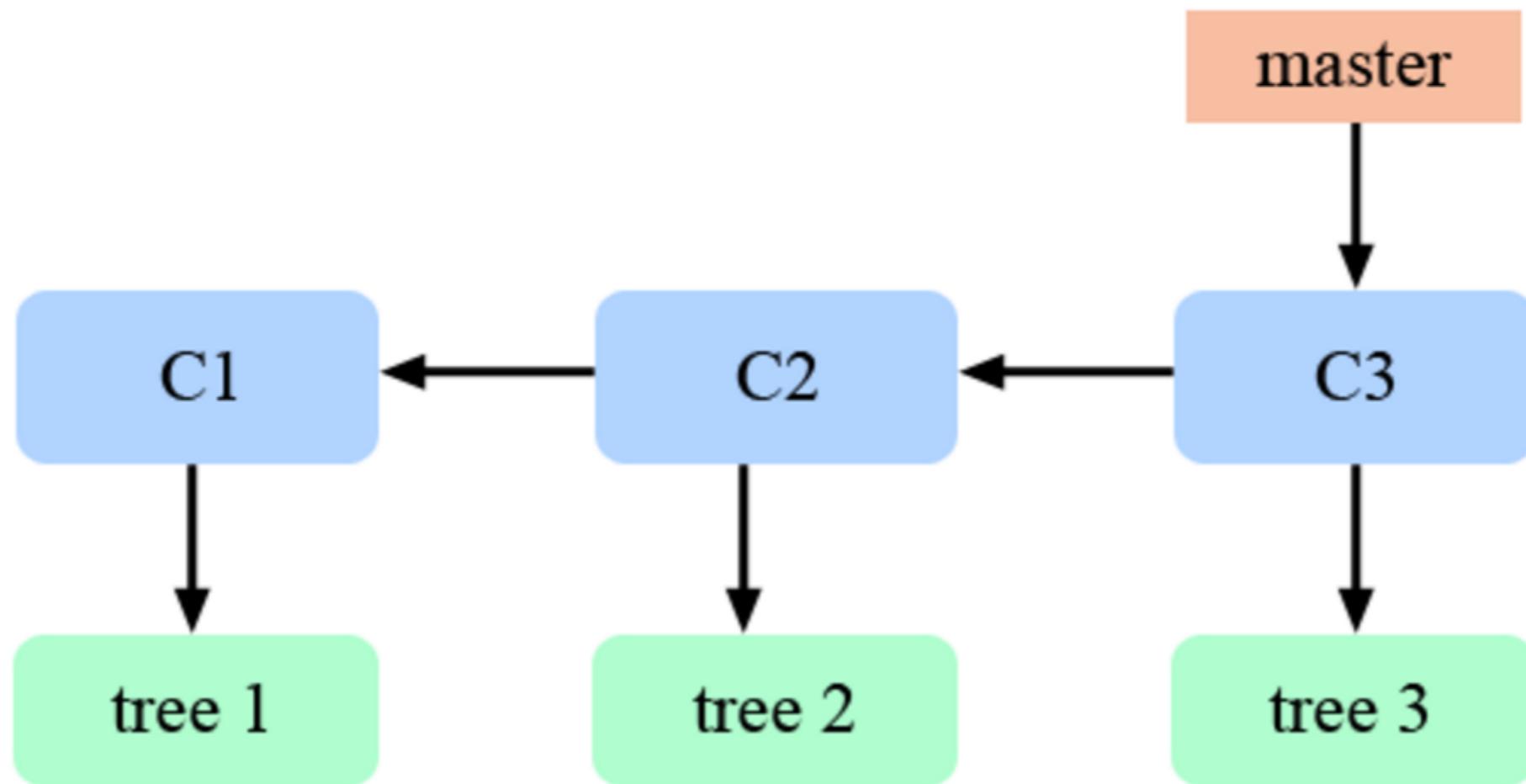
```
$git rm README
```

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



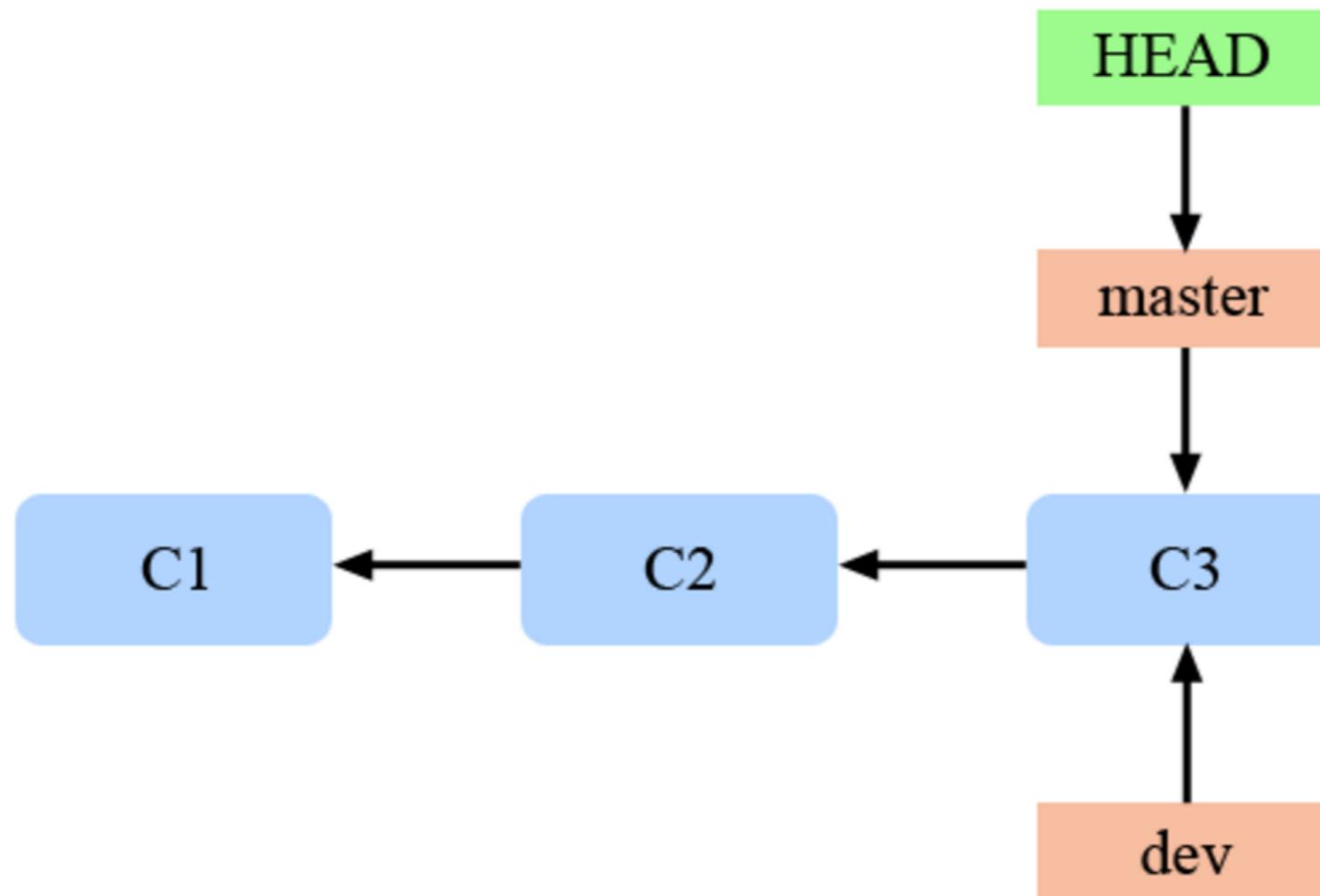
Working with Branch

Each branch points to a commit



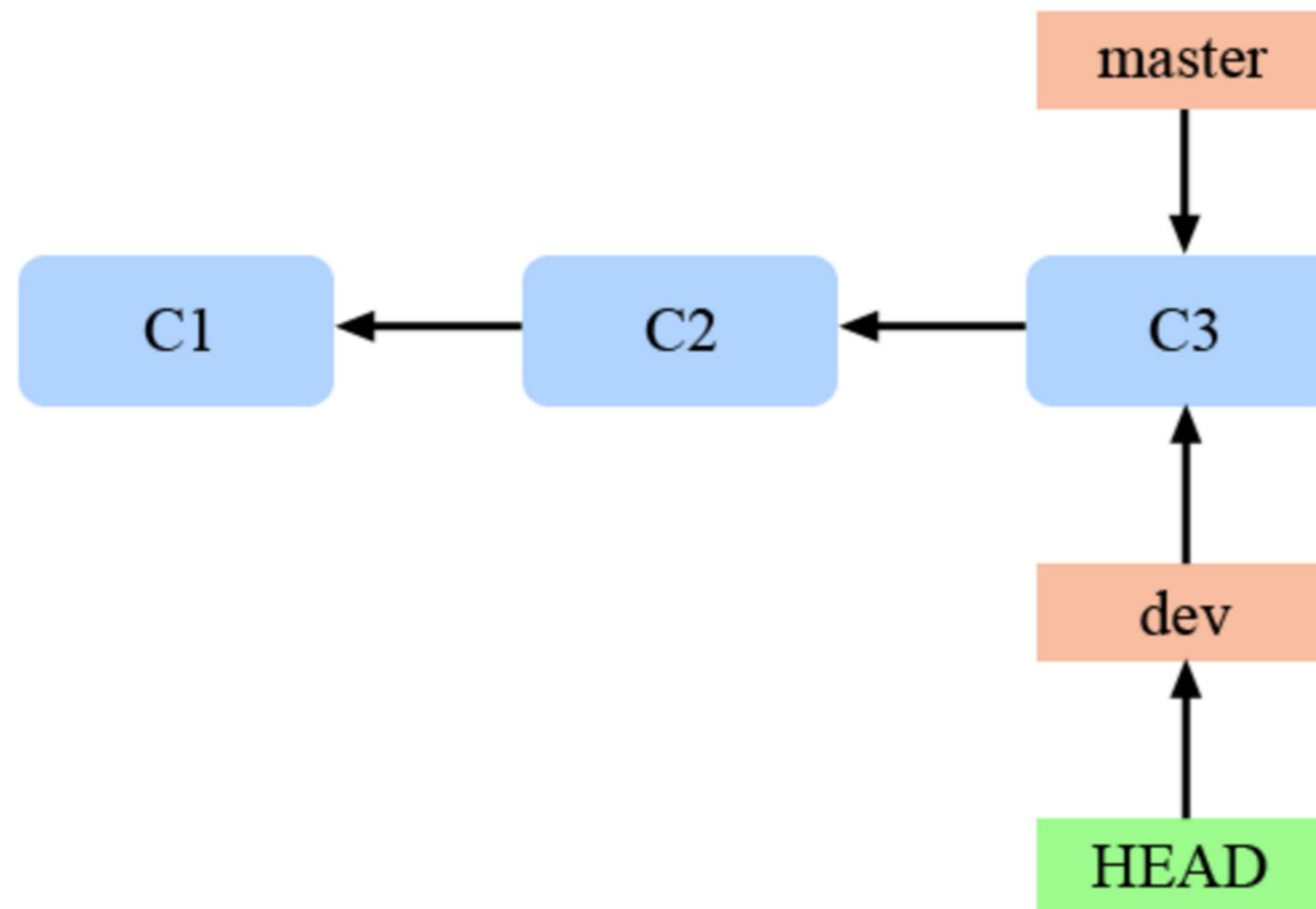
Create new branch

\$git **branch** *BRANCH NAME*



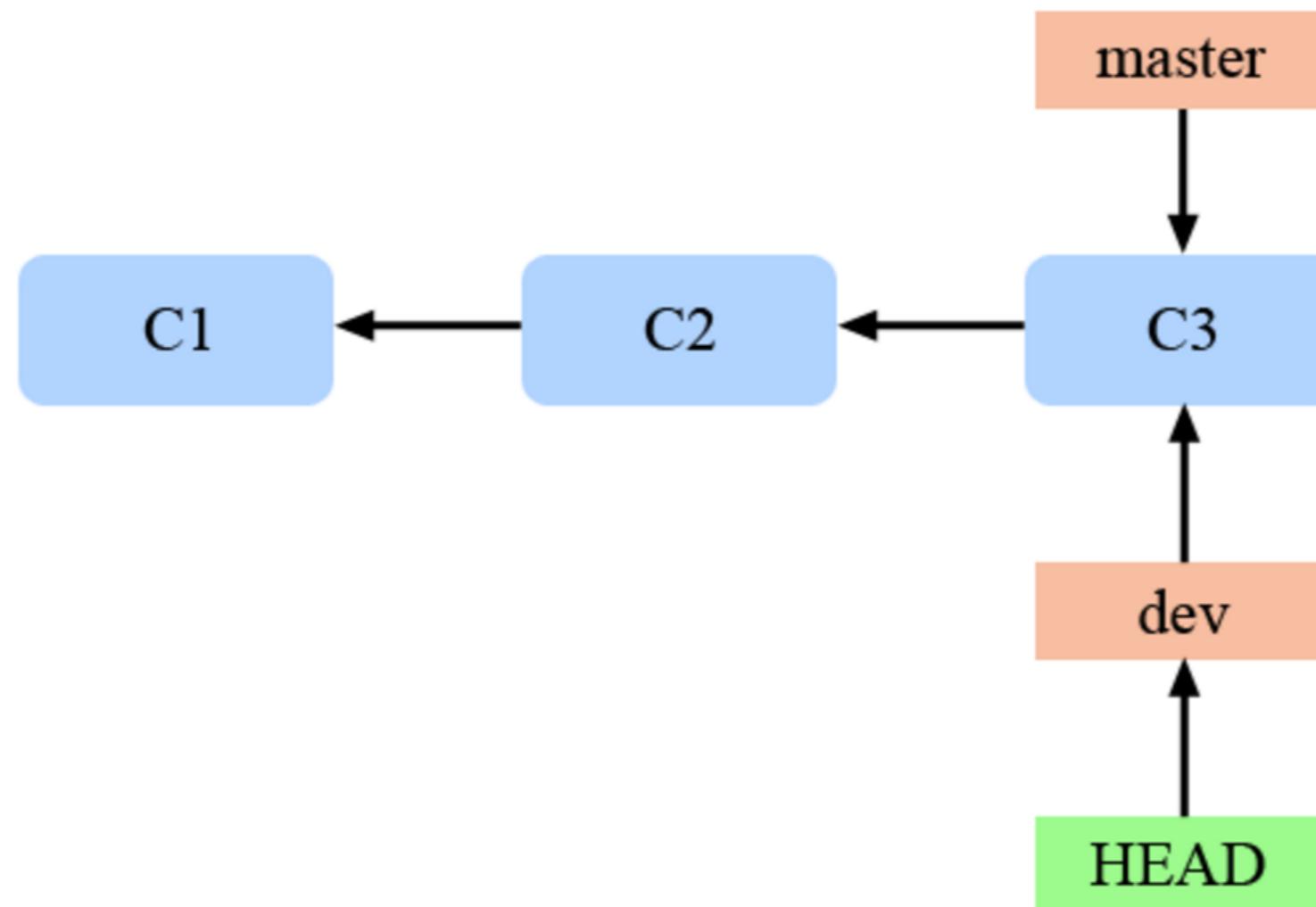
Switch branch

\$git **checkout** *BRANCH NAME*



Switch branch

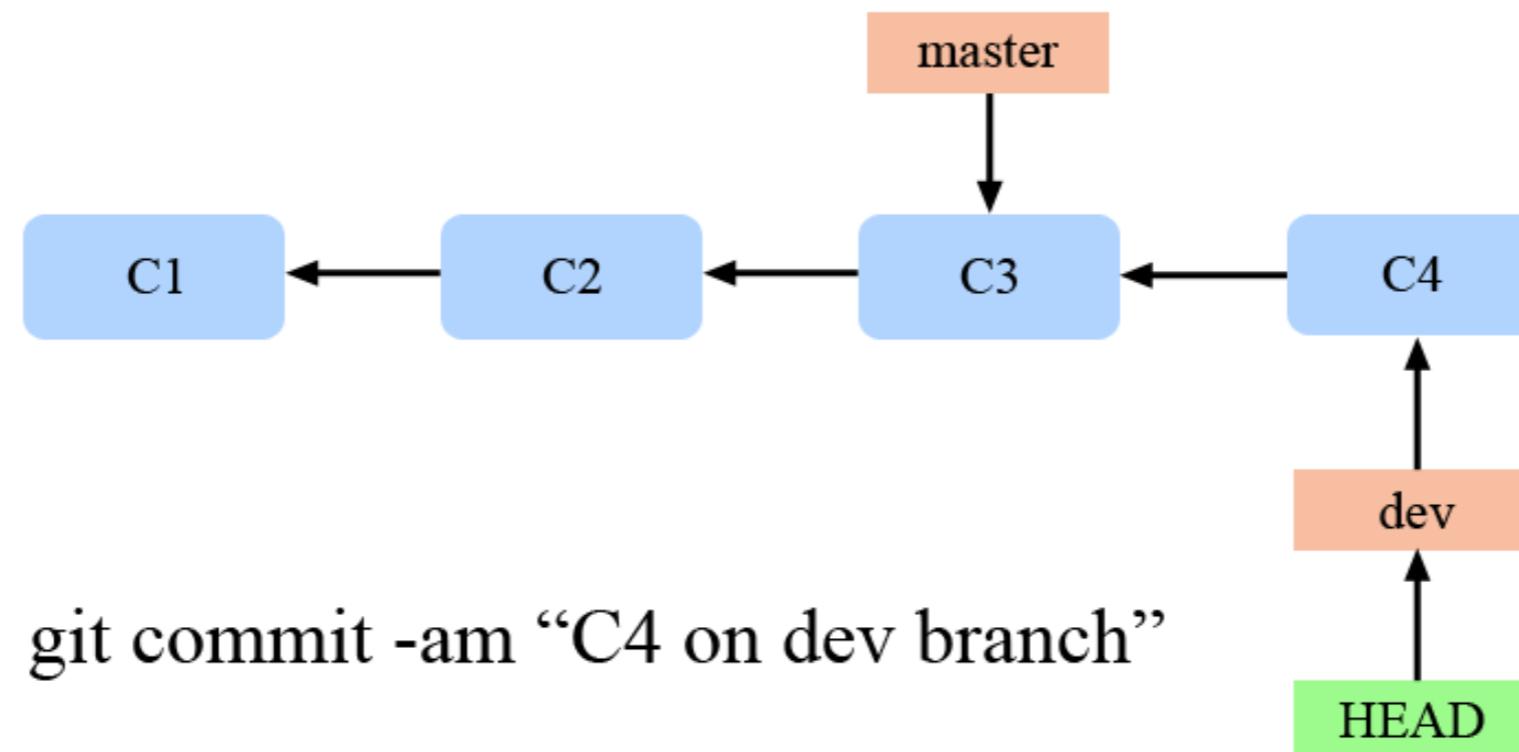
\$git **switch** *BRANCH NAME*



Modified and commit on branch

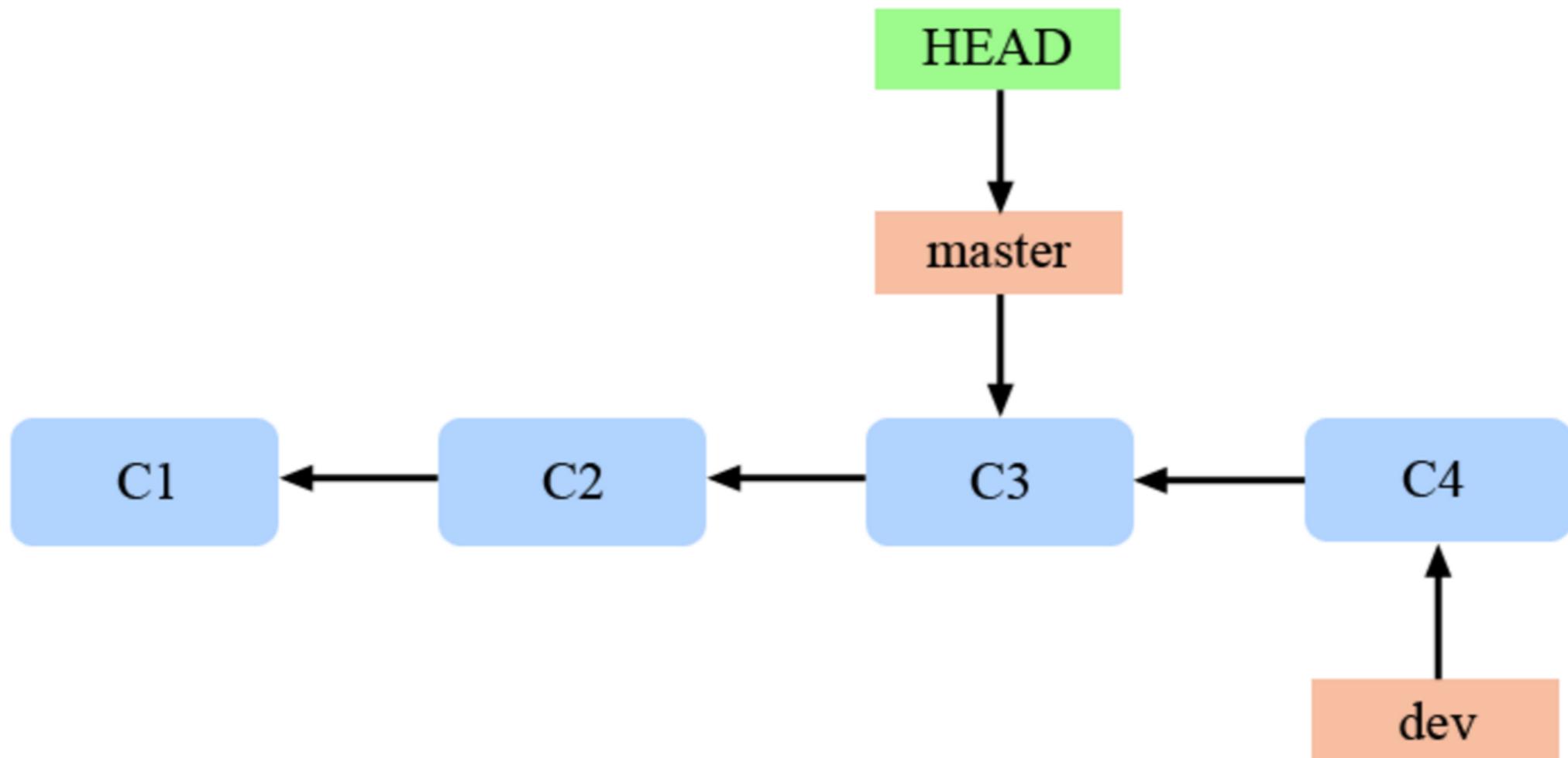
\$echo “On Branch” > *README*

\$git **commit** -am “C4 on dev”



Switch to master branch

\$git **checkout master**



Create and Switch branch

\$git **checkout** -b *BRANCH NAME*

\$git branch *BRANCH NAME*

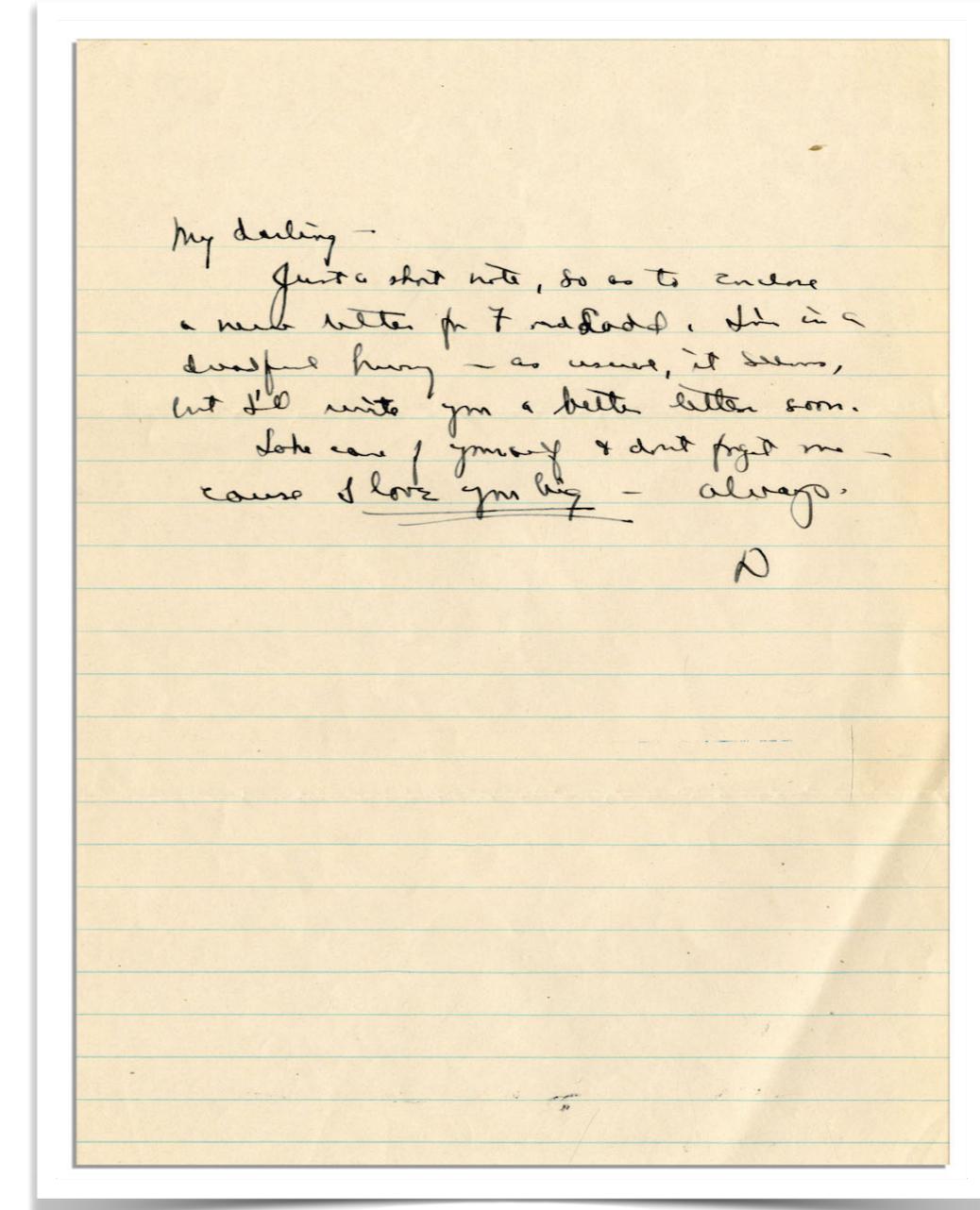
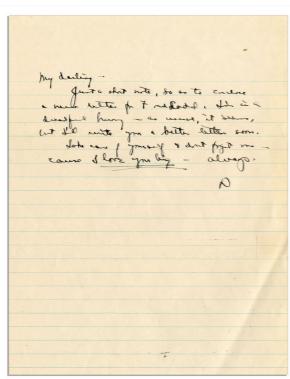
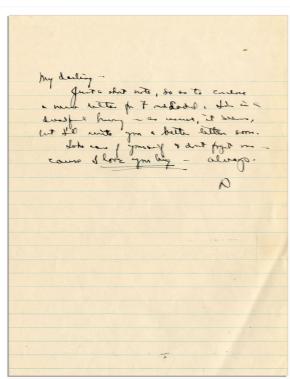
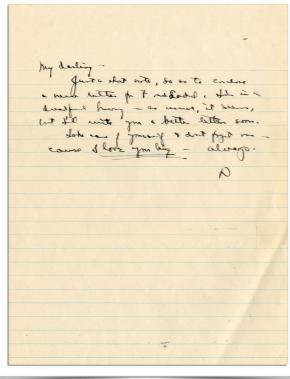
\$git checkout *BRANCH NAME*

Delete and Show all branch

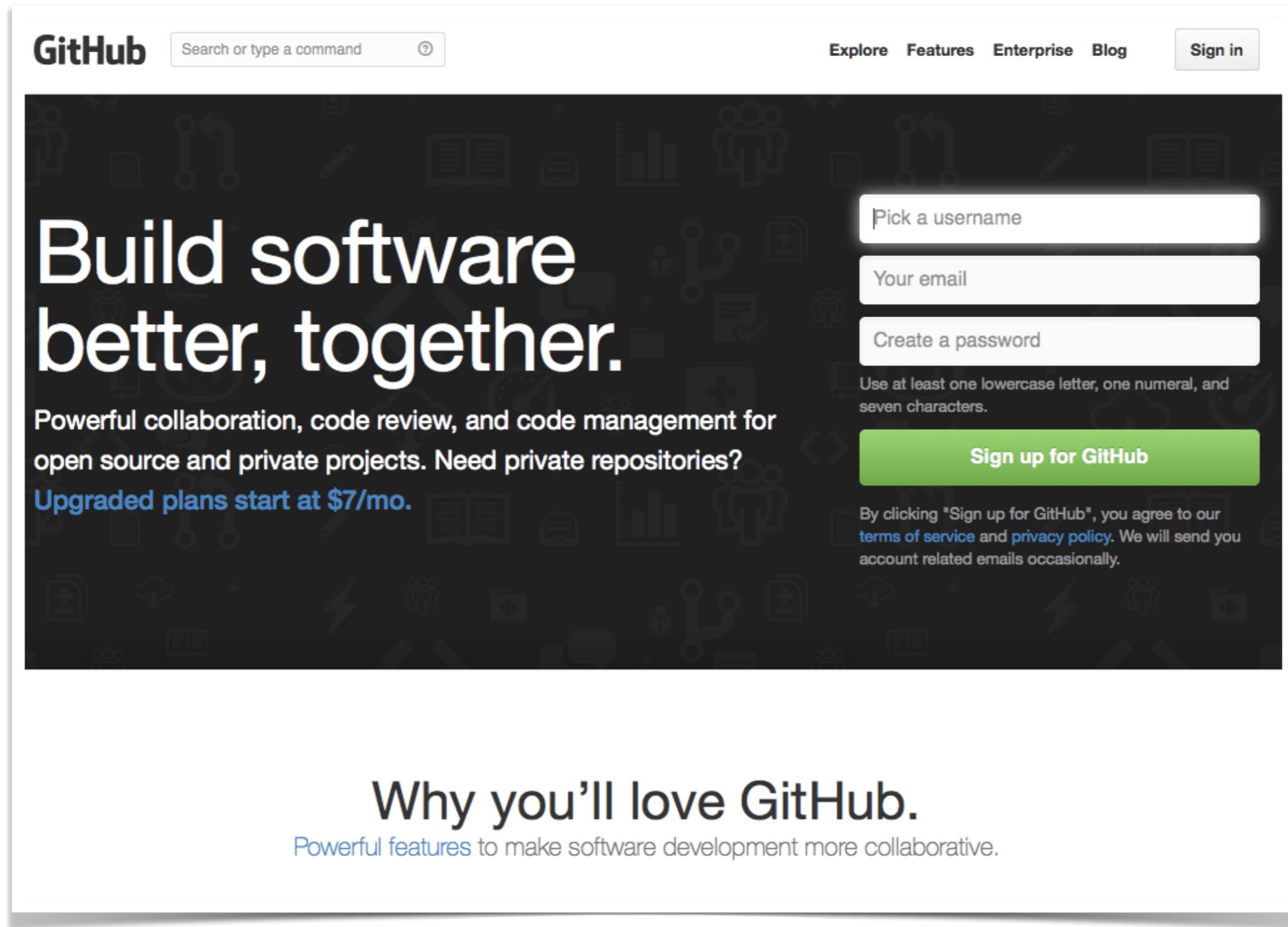
\$git **branch** -d *BRANCH NAME*

\$git **branch**

All become One



Working on GitHub



The screenshot shows the GitHub sign-up page. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for Explore, Features, Enterprise, Blog, and Sign in. Below the navigation is a large, bold headline: "Build software better, together." A sub-headline below it reads: "Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at \$7/mo." To the right of the headline is a sign-up form with three input fields: "Pick a username", "Your email", and "Create a password". Below the password field is a note: "Use at least one lowercase letter, one numeral, and seven characters." A green "Sign up for GitHub" button is positioned below the password field. A small disclaimer at the bottom states: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally." At the bottom of the page, under the heading "Why you'll love GitHub.", is the text: "Powerful features to make software development more collaborative."

Build software better, together.

Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? [Upgraded plans start at \\$7/mo.](#)

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally.

Why you'll love GitHub.

Powerful features to make software development more collaborative.

Generate ssh key and add to Github

\$ssh-keygen

\$pbcopy < ~/.ssh/id_rsa.pub

The screenshot shows the GitHub user interface. On the left, there's a sidebar with a bell icon, a plus sign, and a user icon. Below these are links for 'Signed in as octocat', 'Your profile', 'Your stars', 'Explore', 'Help', 'Settings' (which is highlighted), and 'Sign out'. In the center, there's a navigation bar with 'Billing', 'Payment history', 'SSH keys' (which is highlighted), and 'Security'. To the right, a modal window titled 'SSH keys' is open. It displays the message 'There are no SSH keys with access to your account.' There are fields for 'Title' and 'Key' (containing placeholder text about key formats). A green 'Add SSH key' button is at the bottom, and a note at the bottom of the modal says 'Check out our guide to generating SSH keys or troubleshoot common SSH Problems.'

Testing your SSH connection

```
$ ssh -T git@github.com
```

```
The authenticity of host 'github.com (192.30.252.1)' can't be established.  
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.  
Are you sure you want to continue connecting (yes/no)?
```

```
The authenticity of host 'github.com (192.30.252.1)' can't be established.  
RSA key fingerprint is nThbg6kXUpJWGl7E1IGOCspRomTxCARLviKw6E5SY8.  
Are you sure you want to continue connecting (yes/no)?
```

```
Hi username! You've successfully authenticated, but GitHub does not  
provide shell access.
```

Add and Rename remote

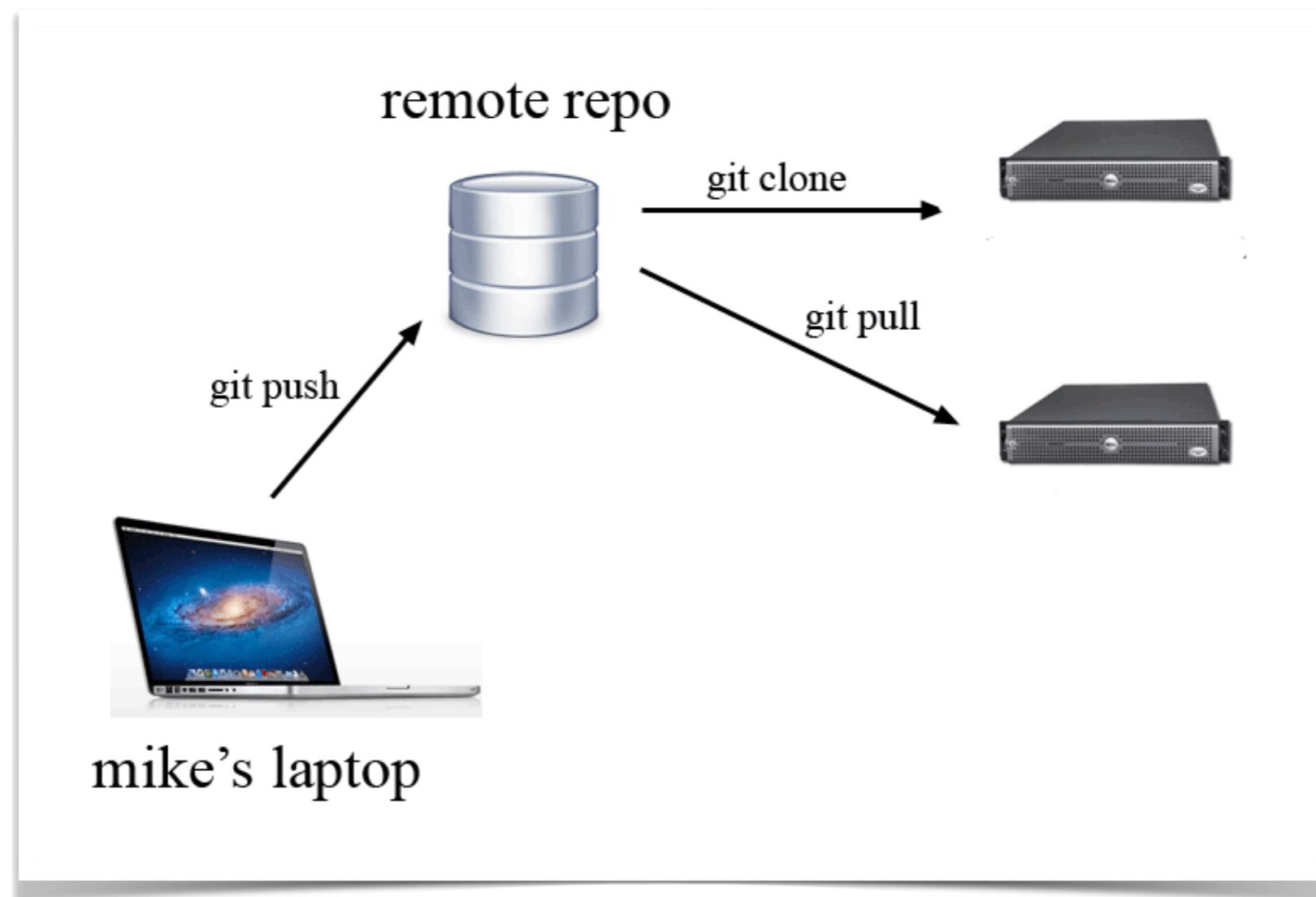
\$git remote add [NICKNAME] [REMOTE URL]

\$git remote rm [NICKNAME]

\$git remote rm [OLD NICKNAME] [NEW NICKNAME]

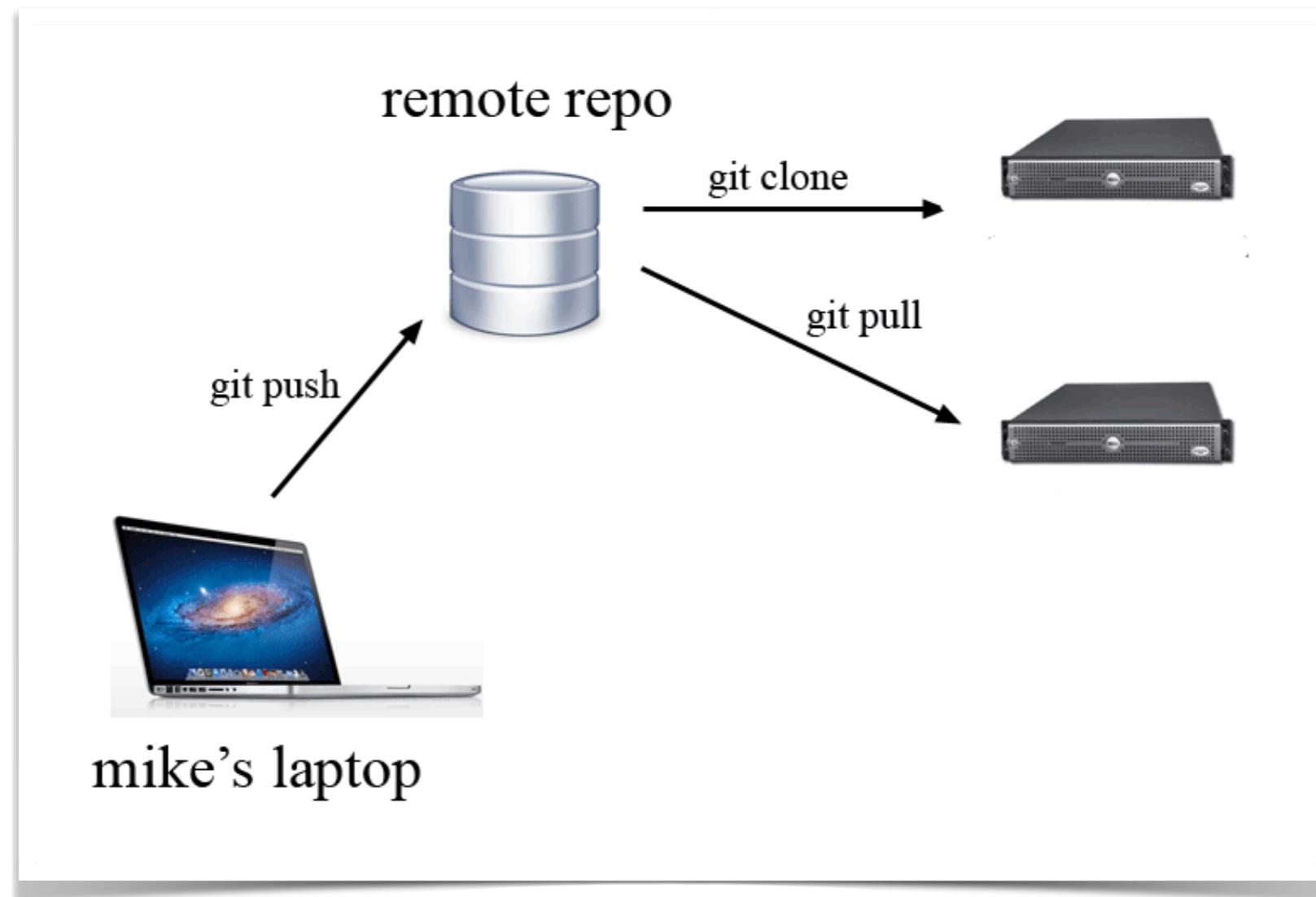
Push to remote

\$git **push** [NICKNAME] [BRANCH]



Fetch and Merge

\$git **pull** [NICKNAME] [BRANCH]



Fetch and Merge

\$git **pull** [NICKNAME] [BRANCH]

\$git fetch [NICKNAME]

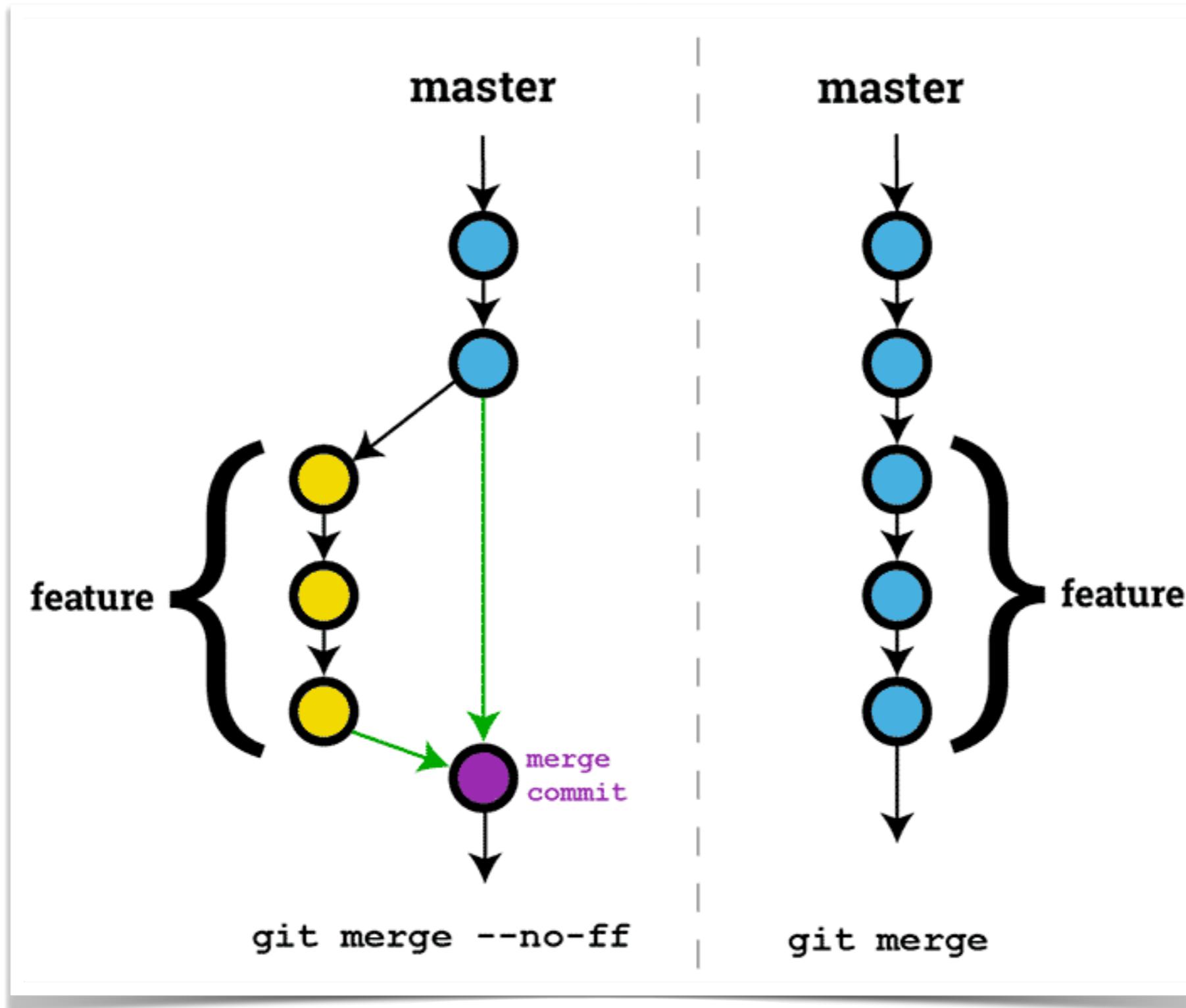
\$git merge [BRANCH]

**BRANCHED FROM MASTER 3
WEEKS AGO**

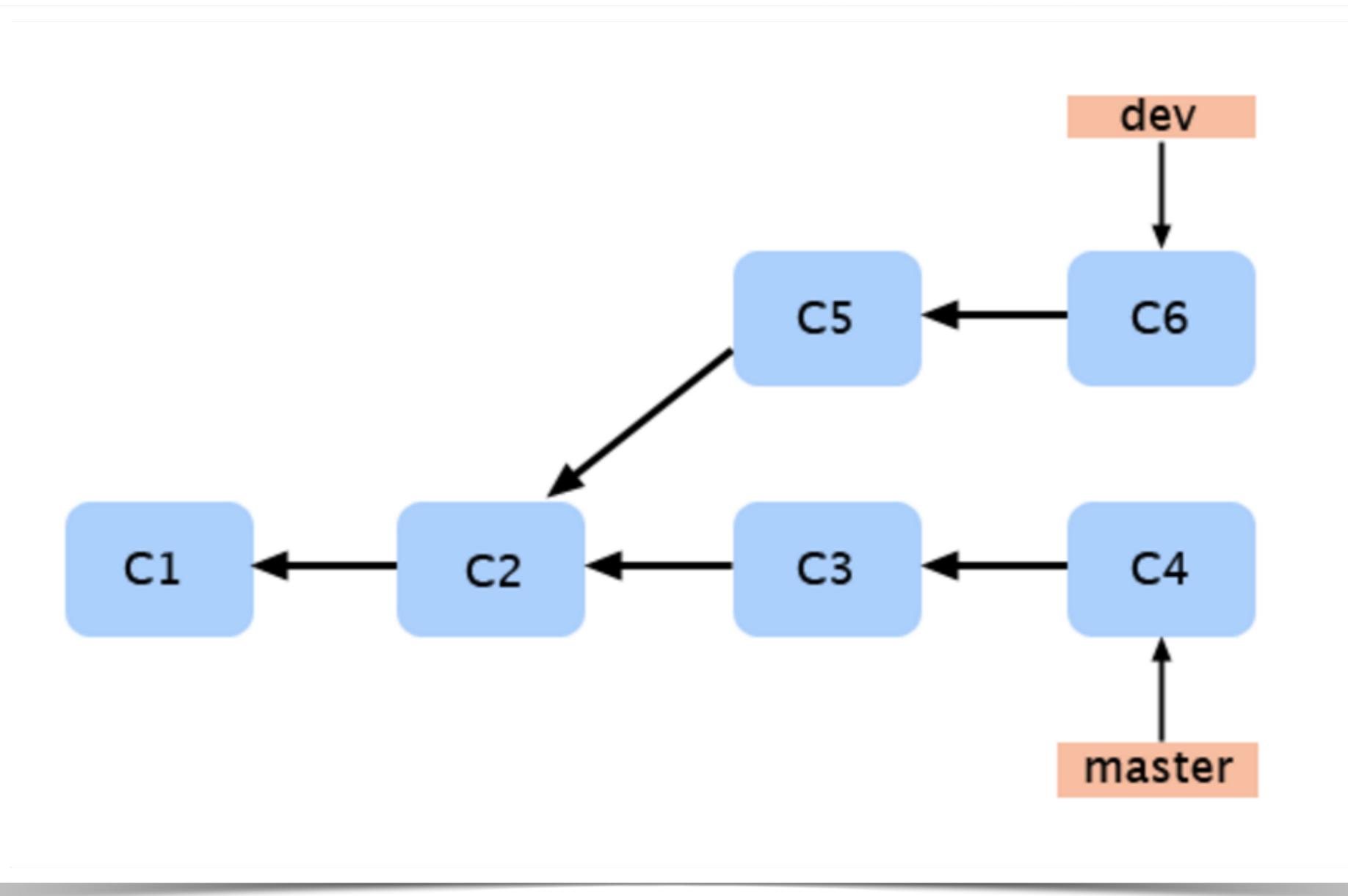


**MERGED BACK WITHOUT
ANY CONFLICTS**

Merge and Rebase

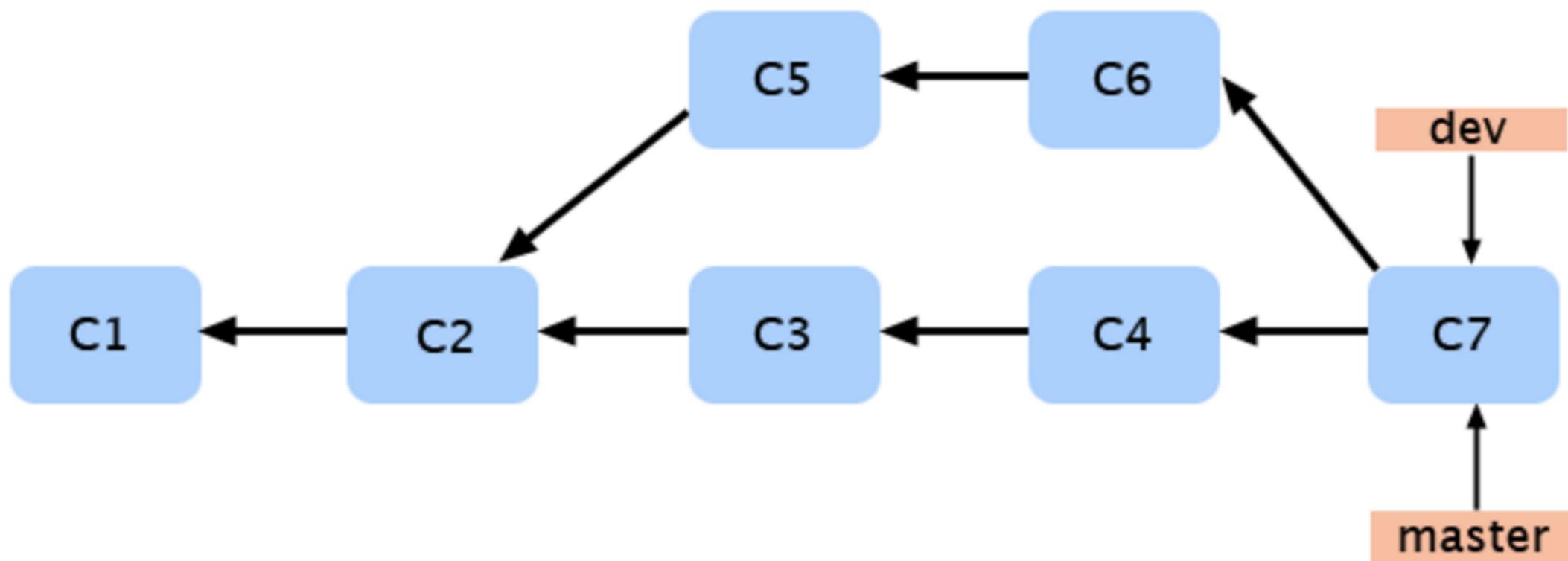


Start before Merge



After Merge

\$git **merge**



Avoid Merge Conflict

Small change and commit

Early merge

Single Responsibility Principle

Communication is a Key

Mob programming

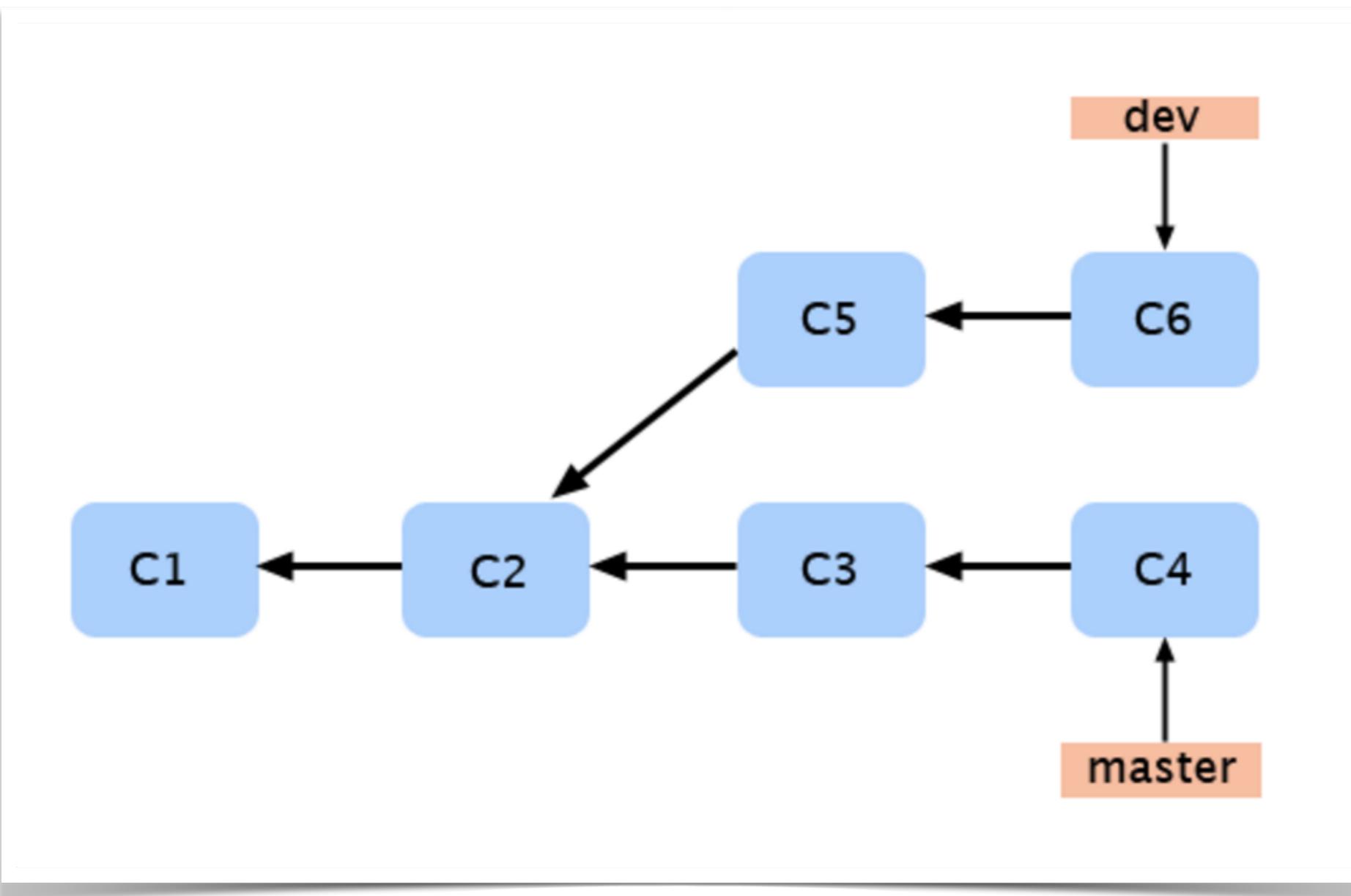
Rebase

Allows you to write history differently

Cleaner history

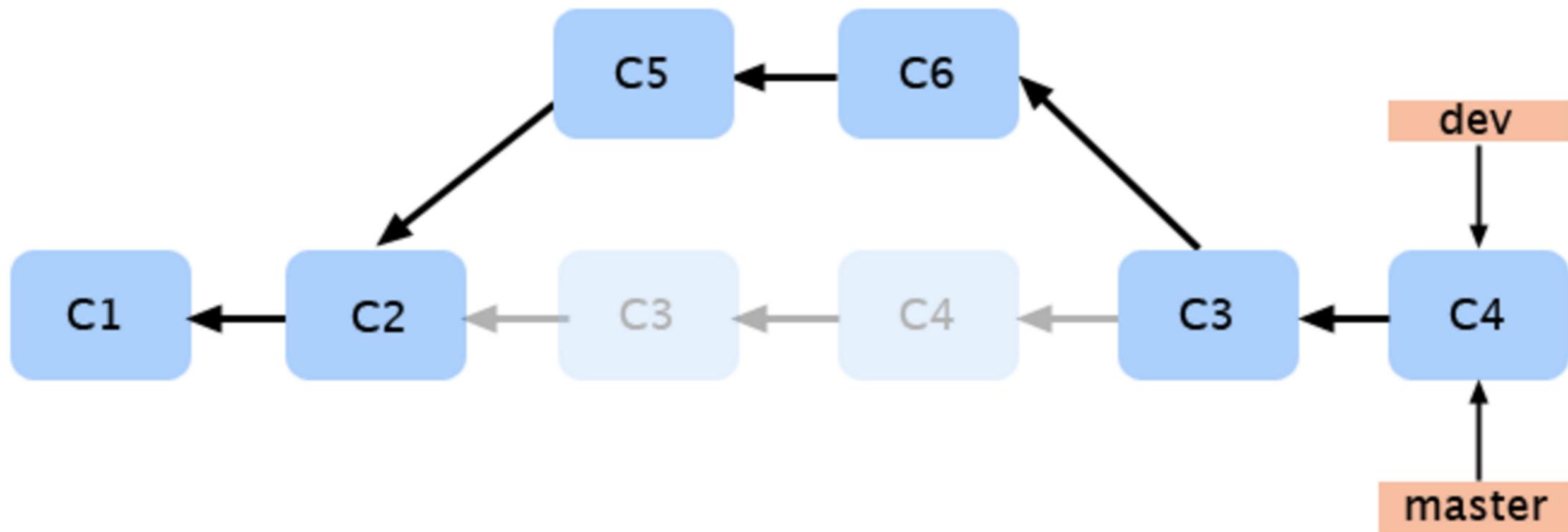
rebase -i (interactive)

Start before Rebase

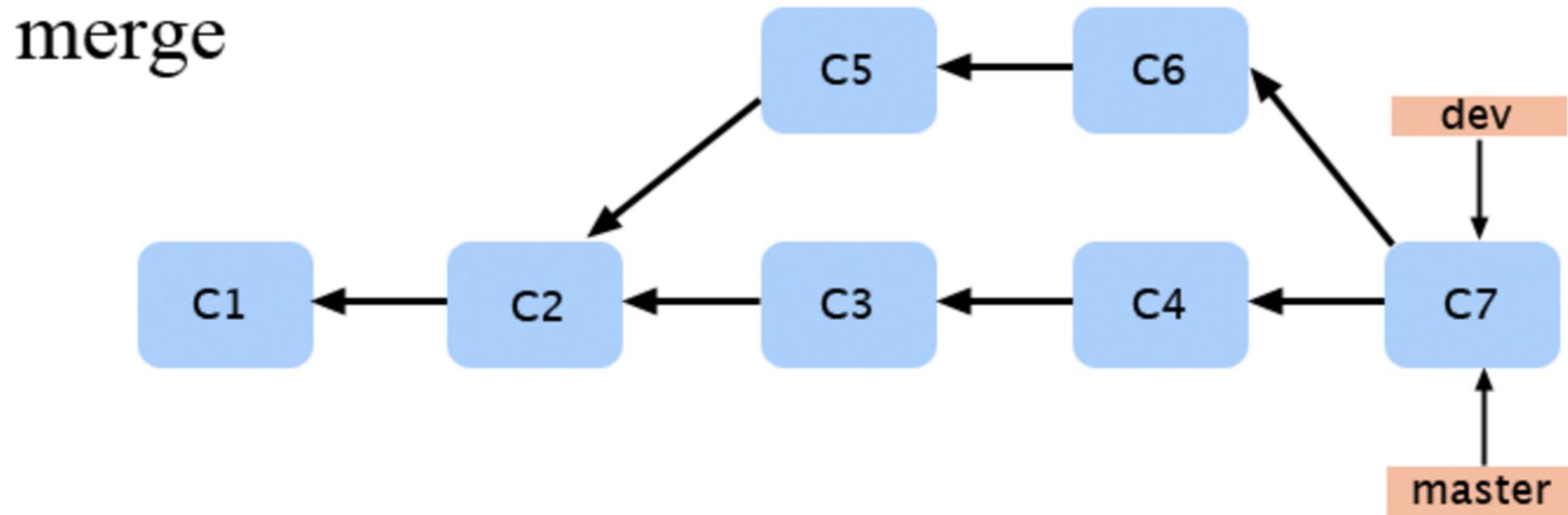
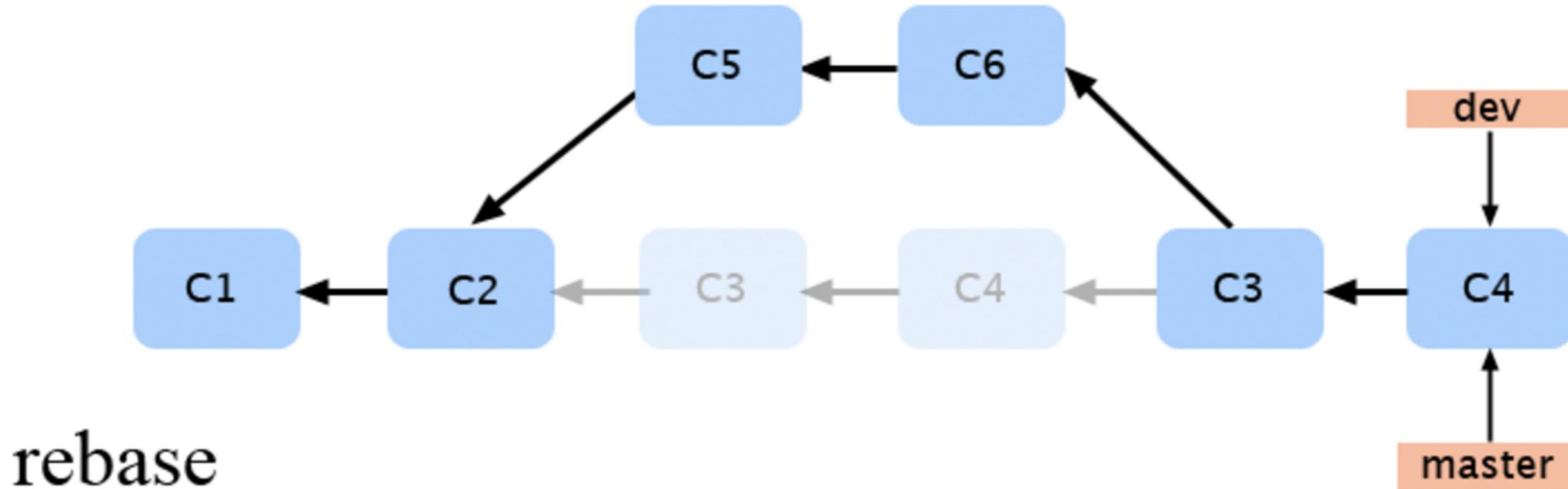


After Rebase

\$git **rebase** [BRANCH NAME]



Merge vs Rebase





Tips Working on Branch

1. Create your branch
2. Checkout branch
2. Make changes and save file
3. Use diff to see all changes
4. Add and commit on your branch
5. Merge and rebase or discard !!

Tips Learn More

\$git tag

\$git log

\$git stash

git hooks

Tips Workflow

1. initial repository
2. Pull (fetch and merge)
3. Branching in your local repository (optional)
4. Modify and improve your code
5. Commit to local repository
6. push to your branch on remote repository
- 7. Diff file => Review code**

Tips Learn More

Professional review code to find out **ERRORS**

List of tools for code review

From Wikipedia, the free encyclopedia

This is a list of [software](#) that helps [software developers](#) conduct and manage [code reviews](#).

Software	Maintainer	Development status	License	VCS supported	Platforms supported	Workflow
Swascan	Swascan	actively developed	Proprietary	Any	Java , PHP , csharp, css, erlang, flex, groovy, javascript, python, scmgit, scmsvn, web, xml	pre- and post-commit
Crucible	Atlassian	actively developed	Proprietary	CVS, Subversion, Git, Mercurial, Perforce	Java	pre- and post-commit
Gerrit	Shawn Pearce	actively developed	Apache v2	Git	Java EE	pre-commit
Upsource	JetBrains	actively developed	Proprietary	Git, Subversion, Mercurial, Perforce	Java	post-commit
GitLab	GitLab Inc.	actively developed	MIT	Git	Ruby on Rails	pre- and post-commit
Kallithea	kallithea-scm.org	actively developed	GPL v3	Git, Mercurial	Python	post-commit
Phabricator	Phacility	actively developed	Apache	Git, Subversion, Mercurial	PHP	pre- and post-commit
Review Board	reviewboard.org	actively developed	MIT	CVS, Subversion, Git, Mercurial, Bazaar, Perforce, ClearCase, Plastic SCM	Python , Java	mainly pre-commit
Rietveld	Guido van Rossum	actively developed	Apache v2	Git, Subversion, Mercurial, Perforce, CVS	Python	pre-commit
Understand	SciTools	actively developed	Proprietary	Any	Windows, Mac OSX, Linux	pre- and post-commit

https://en.wikipedia.org/wiki/List_of_tools_for_code_review

Tips Learn More

Professional review code to find out **ERRORS**

Review Board 2.5.3 alpha 0 (dev) [Jump to...](#) Log in Support ▾

✗ The demo server is reset every night at midnight PST.

[All Review Requests](#) [Users](#) [Groups](#) [Reports](#)

ALL REVIEW REQUESTS [Hide closed](#)

Summary	Submitter	Posted	Last Updated
Diff viewer demo	chipx86	March 31st, 2015, 4:28 a.m.	12 hours, 21 minutes ago
Change history demo	chipx86	March 31st, 2015, 4:28 a.m.	18 hours, 16 minutes ago
Test review	guest576	March 11th, 2016, 7:42 a.m.	1 day, 20 hours ago
Fix URL to web hook docs	guest4899	March 9th, 2016, 5:25 a.m.	3 days, 23 hours ago
Tesing	guest9247	March 8th, 2016, 7:18 a.m.	4 days, 21 hours ago
test1	guest8359	August 11th, 2015, 6:50 a.m.	5 days, 8 hours ago
Merge branch 'release-2.6.x'	guest8989	January 31st, 2016, 2:49 p.m.	5 days, 8 hours ago
Add a site setting for the static URL	guest6129	March 4th, 2016, 12:22 p.m.	6 days, 11 hours ago
Initial Review	guest7880	March 4th, 2016, 12:50 p.m.	1 week, 1 day ago
Merge branch 'release-2.6.x'	guest6129	March 4th, 2016, 10:54 a.m.	1 week, 1 day ago
PDF document review demo	chipx86	March 31st, 2015, 4:28 a.m.	1 week, 2 days ago
1111	guest4539	January 26th, 2016, 1:25 a.m.	1 week, 2 days ago
Submitted Merge branch 'release-2.5.x' into release-2.6.x	guest5371	March 3rd, 2016, 3:31 a.m.	1 week, 3 days ago
Test PDF review?	guest3369	March 3rd, 2016, 12:04 a.m.	1 week, 3 days ago

1 2 3 4 > » 5 pages

<https://www.reviewboard.org/>



Git Good

Git Status



Commit Message

...

TODO
asdasfdgafg
stuff
fixed bug
add feature

“Added a user object to the database.
currently only has a name and email.
no authentication yet ”

“Fixed the bug that would add something
to everything. Turn to not add something
to everything”

GitHub issue tracker integration

<https://guides.github.com/features/issues/>

bug 01 #1



Closed up1 opened this issue 5 minutes ago · 0 comments



up1 commented 5 minutes ago

Owner



No description provided.



up1 closed this issue from a commit 4 minutes ago



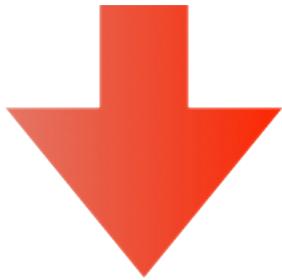
Fixed #1 to remove aomething

90db52f



up1 closed this in [90db52f](#) 4 minutes ago

“fixed #1234”



close issue 1234

fixed, fix, fixes

closed, close, closes

Reference to user

@up1

Git Log (commit history)



Log in one line

```
$git log --oneline
```

Log in pretty mode

```
$git log --oneline --decorate
```

Log in graph mode

```
$git log --graph --oneline --decorate
```

See diff from log

```
$git log --stat
```

```
$git log -p
```

Log by user

```
$git log --author=up1
```

By message

```
$git log --grep="message"
```

By content

```
$git log -S "content"
```

By content with regular expression

```
$git log -G“content”
```

Log in date range

```
$git log --after="2015-9-1"  
      --before="2015-9-12"
```

Log in date range

```
$git log --since=2.months.ago  
          --until=1.day.ago
```

Filter merge commit

\$git log --no-merges

\$git log --merges

Git branch

<http://nvie.com/posts/a-successful-git-branching-model/>

master

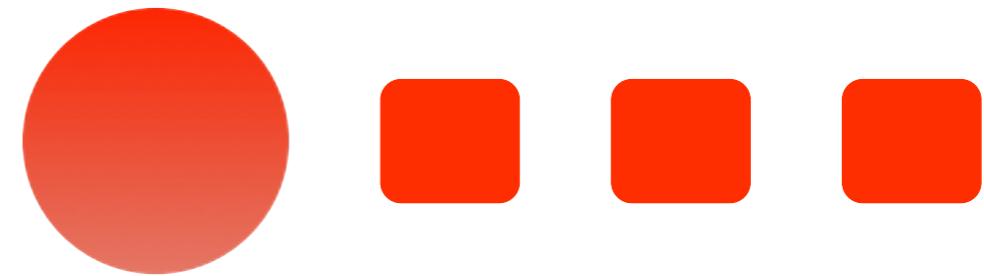


keep master running

Need to fix bug ?

\$git checkout feature

feature



master



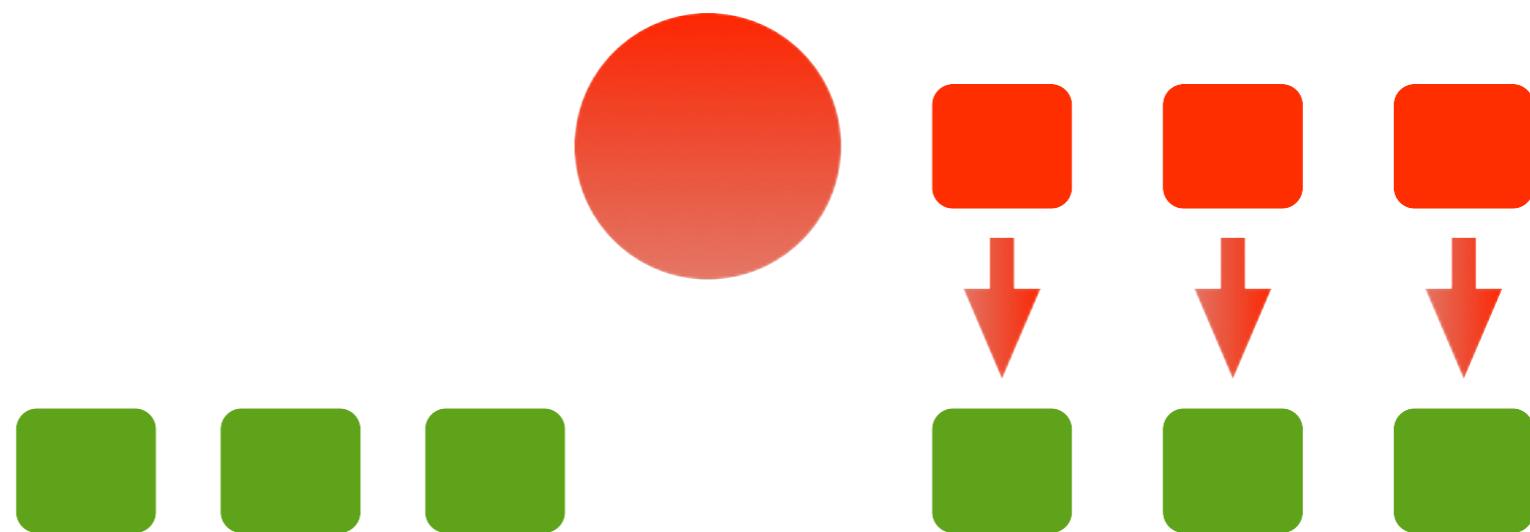
Git checkout -b feature

When ready to merge into master

How ?

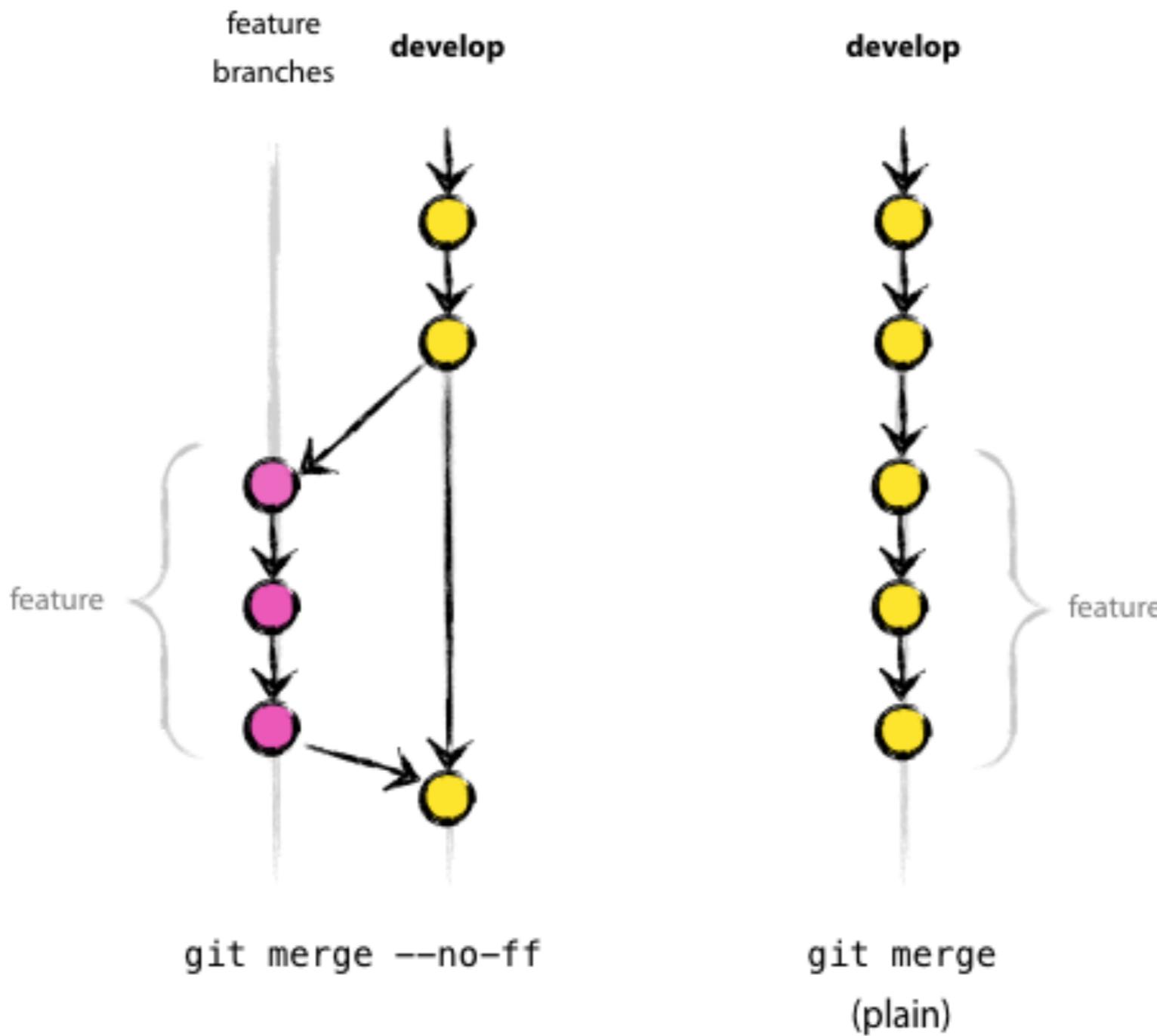
\$git merge feature master

feature



master

\$git merge --no-ff



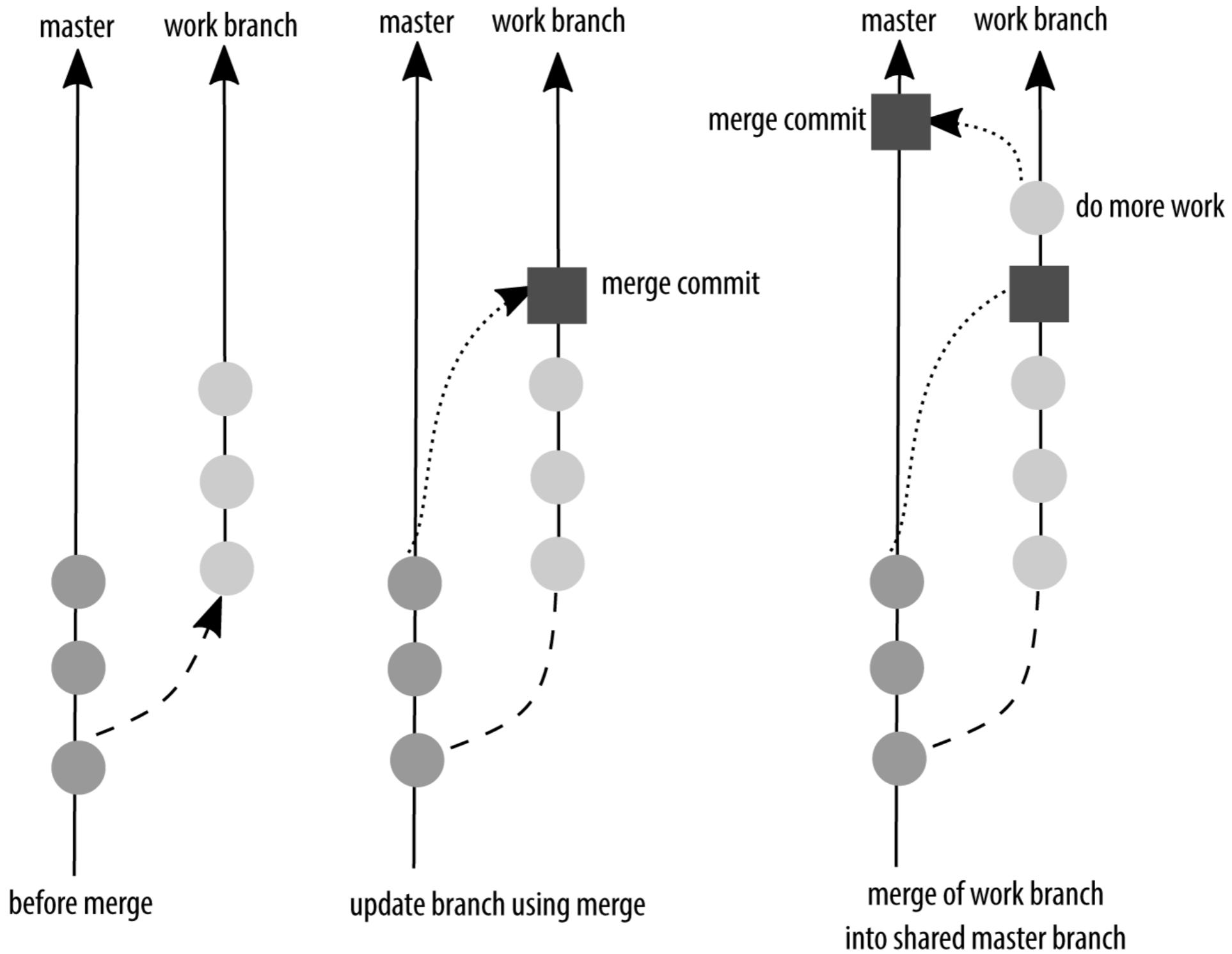
branching is cheap

painless development

delete branch if it's not work

delete branch if it's not use

Tips :: Merge to master branch





Branching Strategy

Maintainable Git



CODE is KING

Developer cost

\$\$\$\$\$\$\$\$\$

Branching cost
(~40 bytes)

\$

Developer cost

\$\$\$\$\$\$\$\$\$

Poor branching strategy

\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$

Branching strategies

Last good state ?
Mistake break everyone !!
Lead to big commits !!

What is deployed ?
Where do fixed ?



What is being worked on ?
How do you switch tasks ?
When do you integrated ?



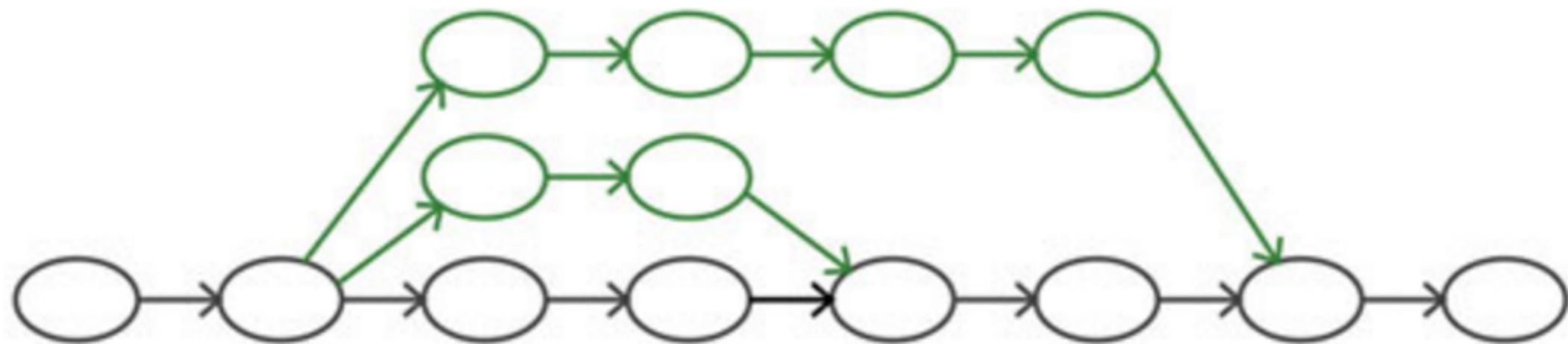
ทำอย่างไรดี ?

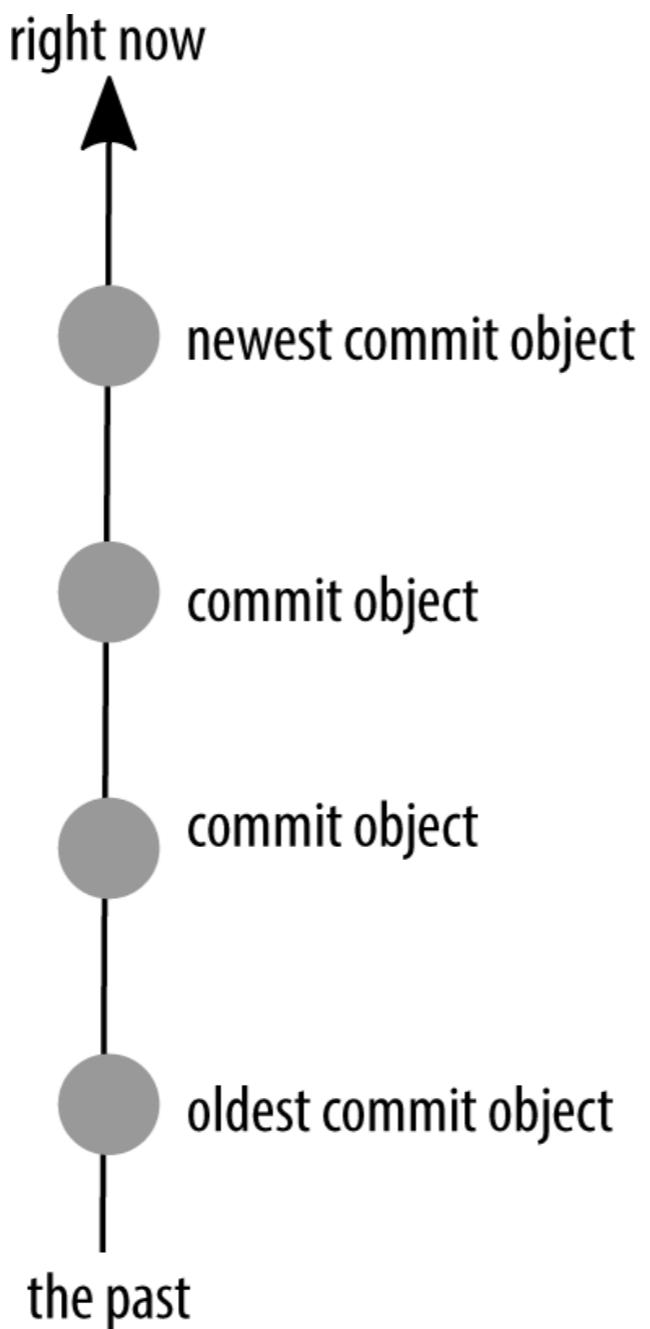
Release Process ?

Create a new branch

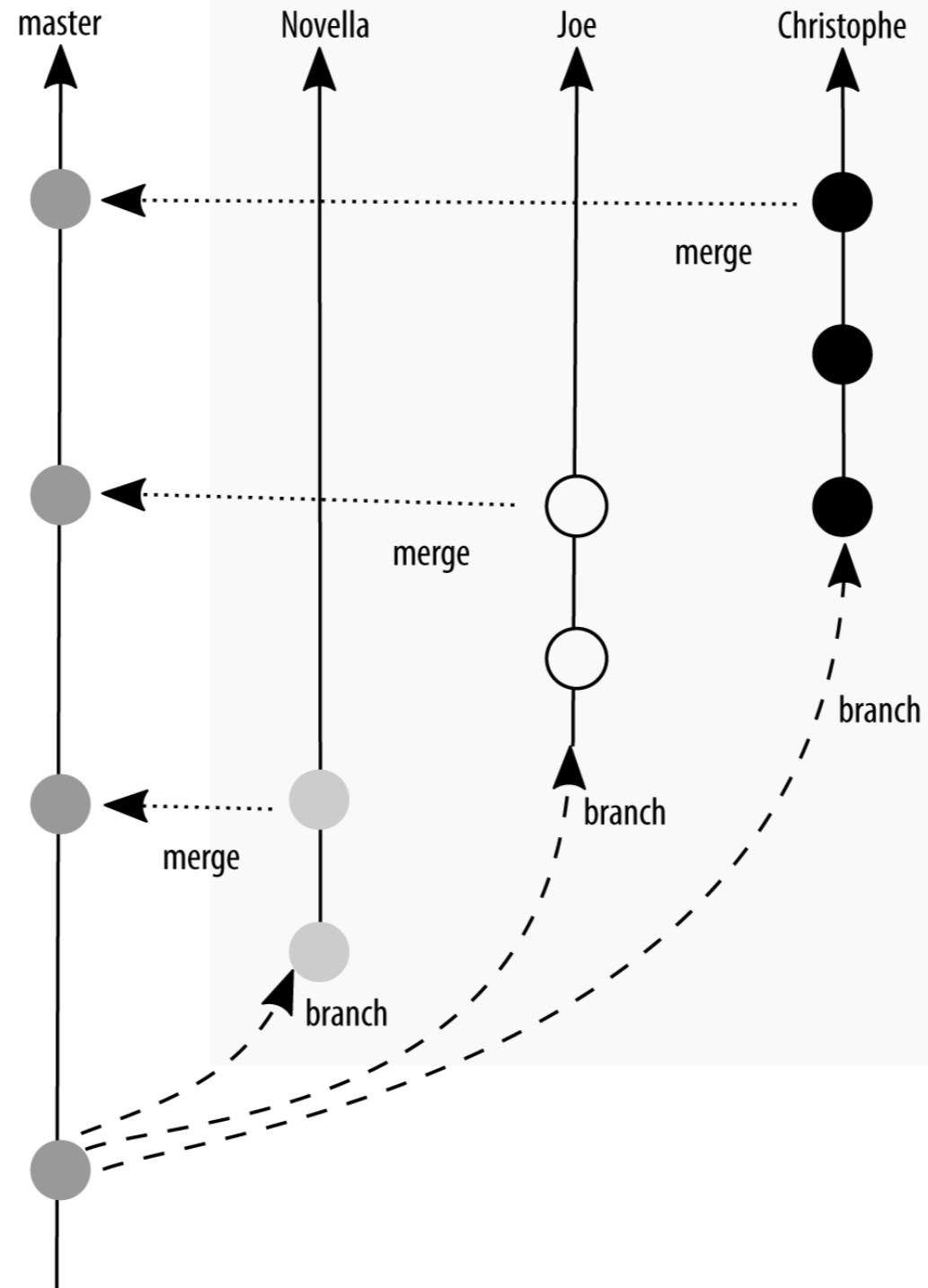
when working on any feature, issue or bug

Merged back when it is finished



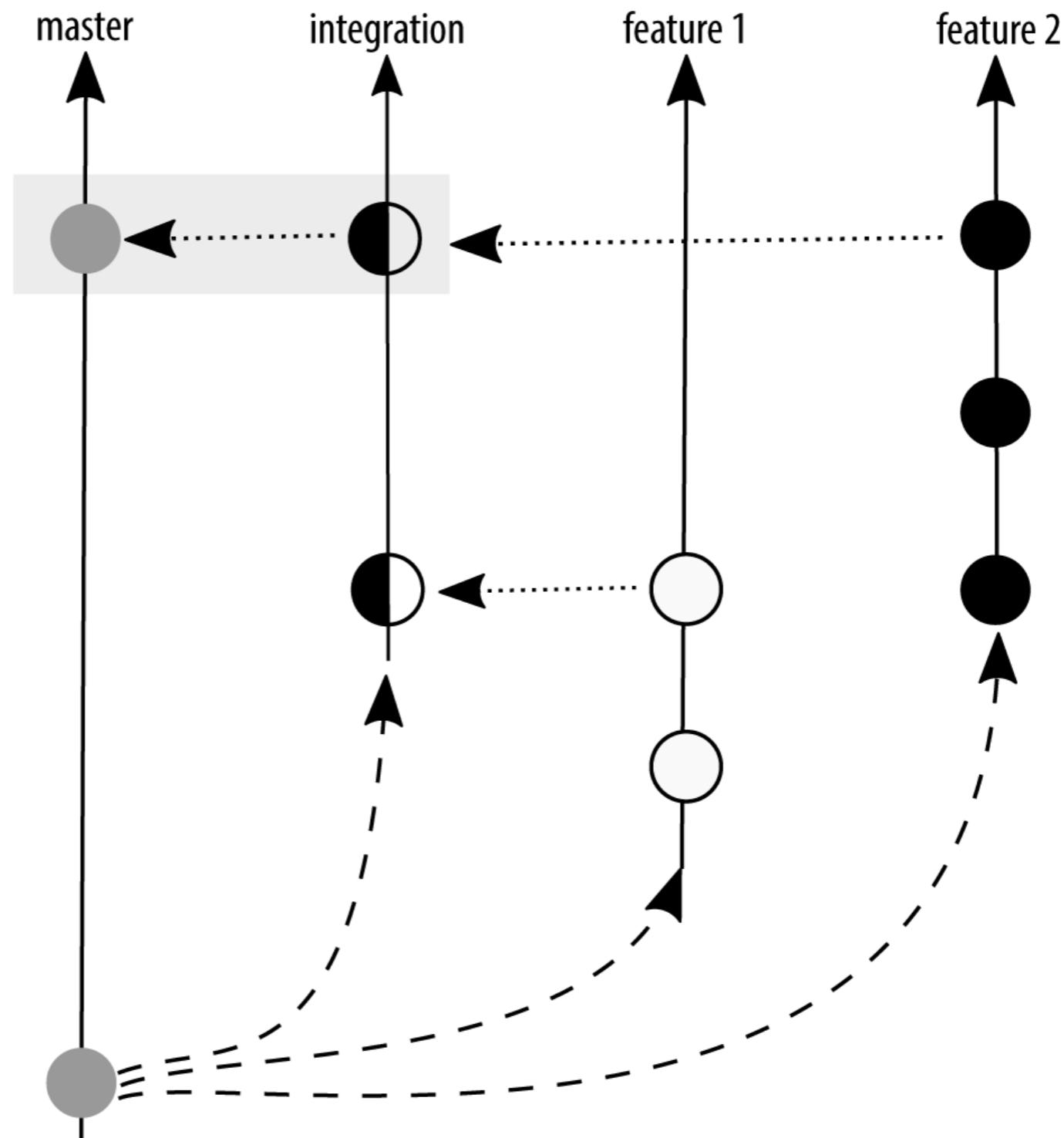


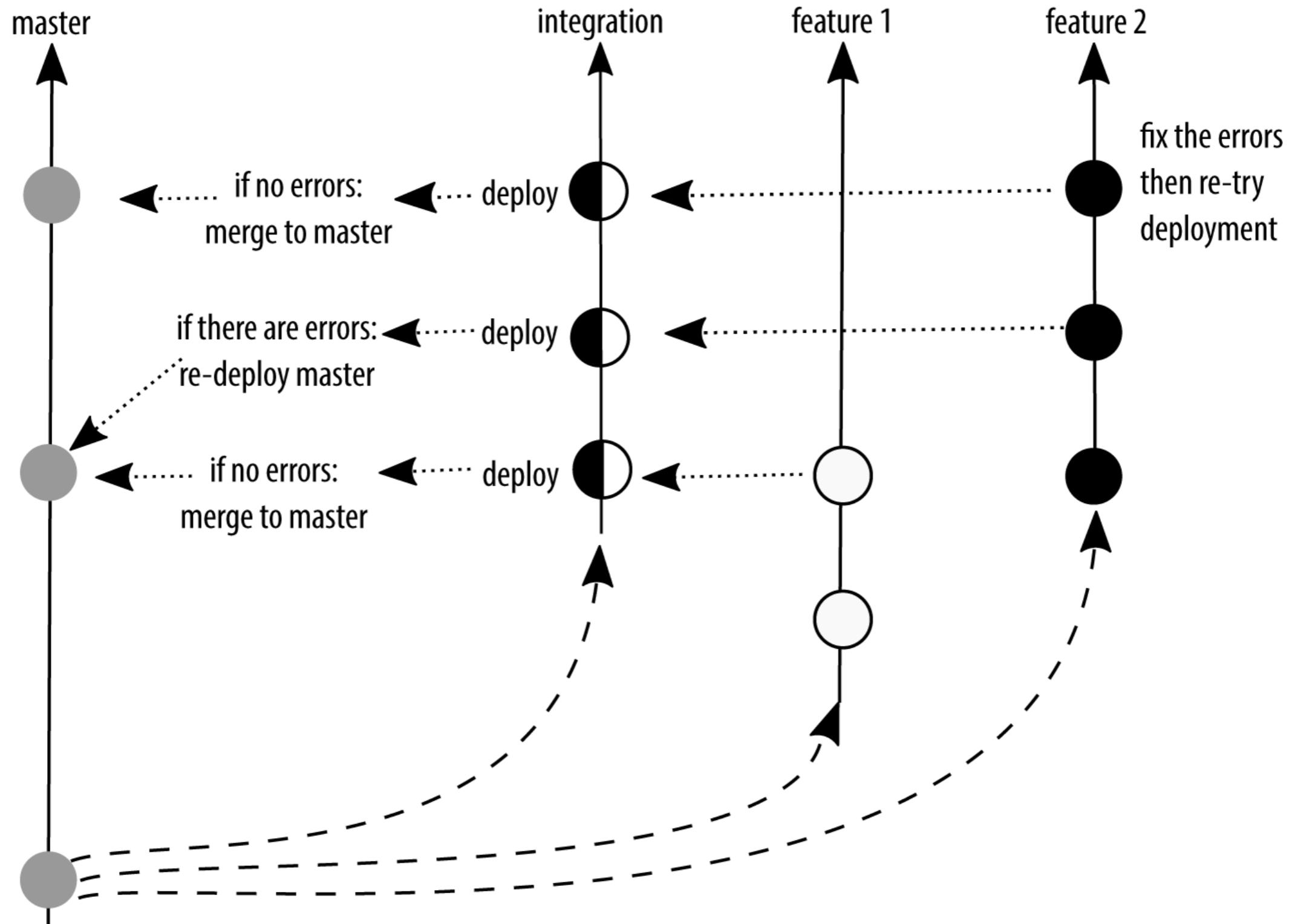
branches containing changes from editors



Let's discuss with Your team

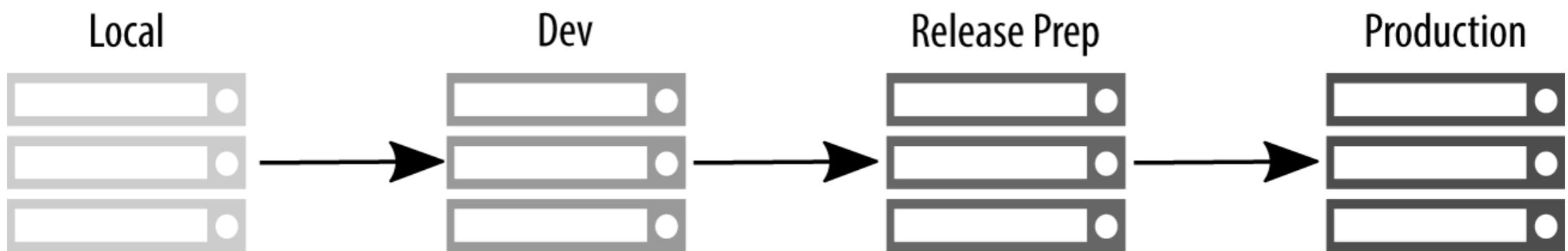
Assuming feature 1 and feature 2 integrate well: merge the combined changes in the integration branch into the master branch.

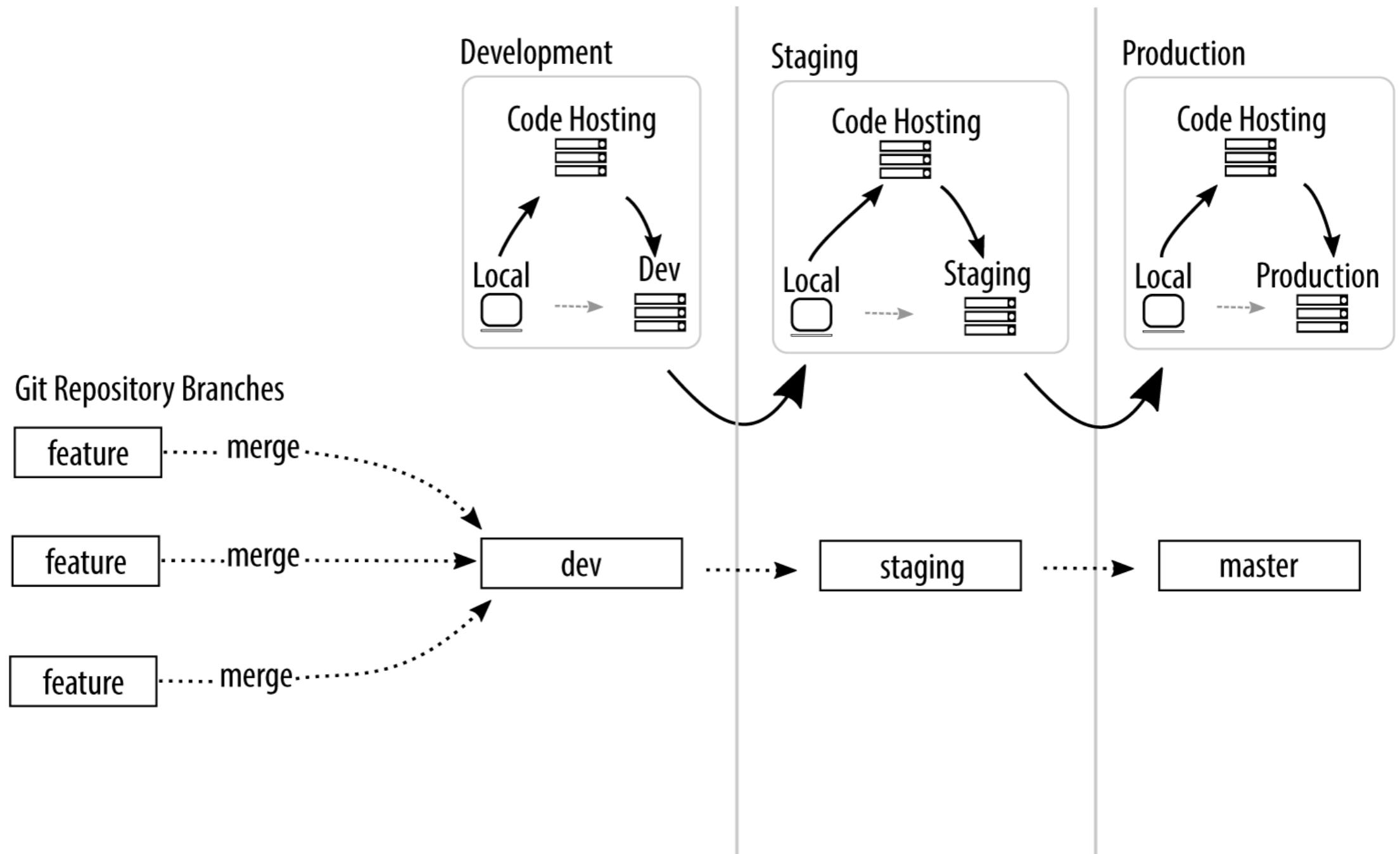




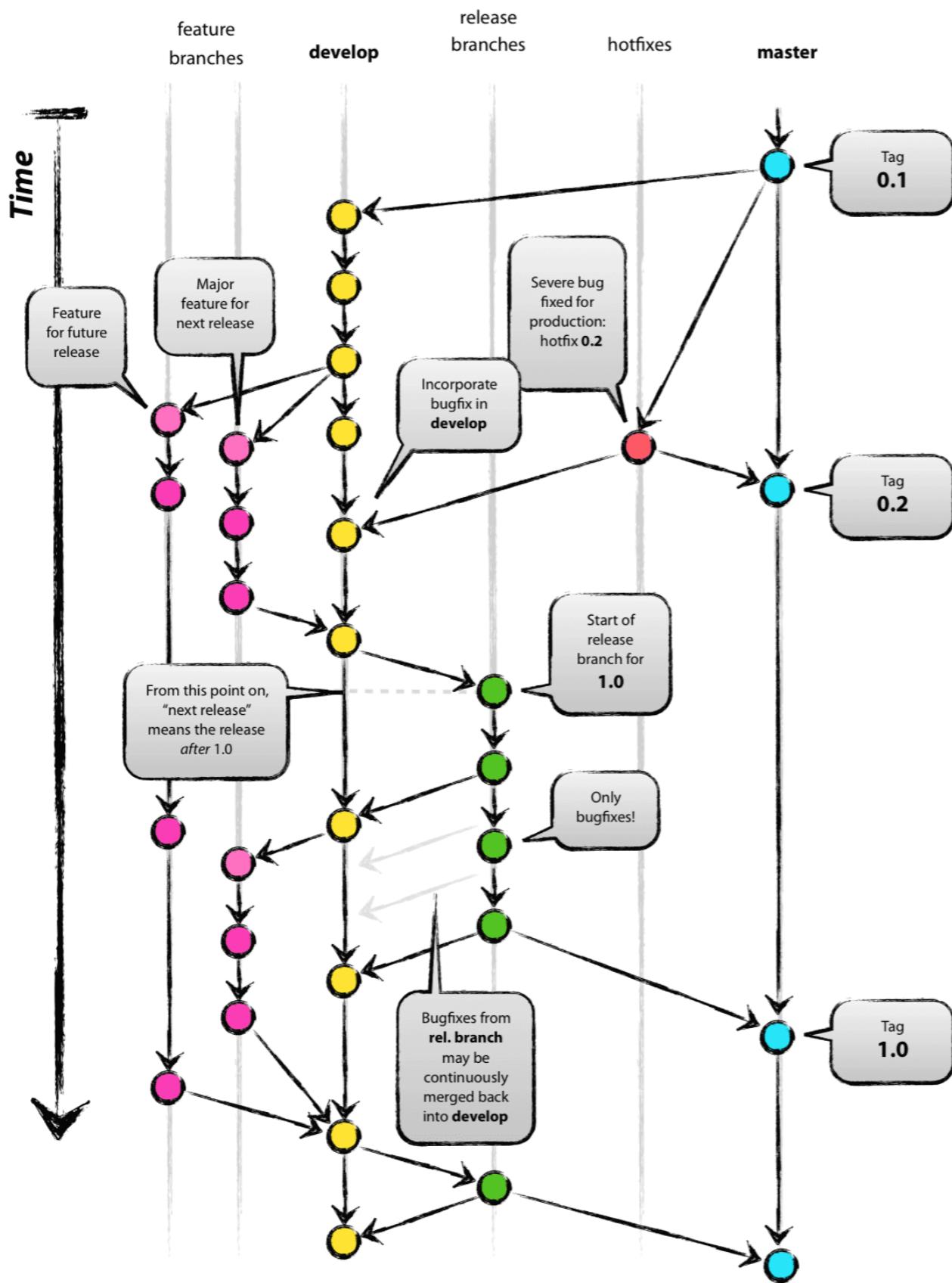
Let's discuss with Your team

One branch per platform

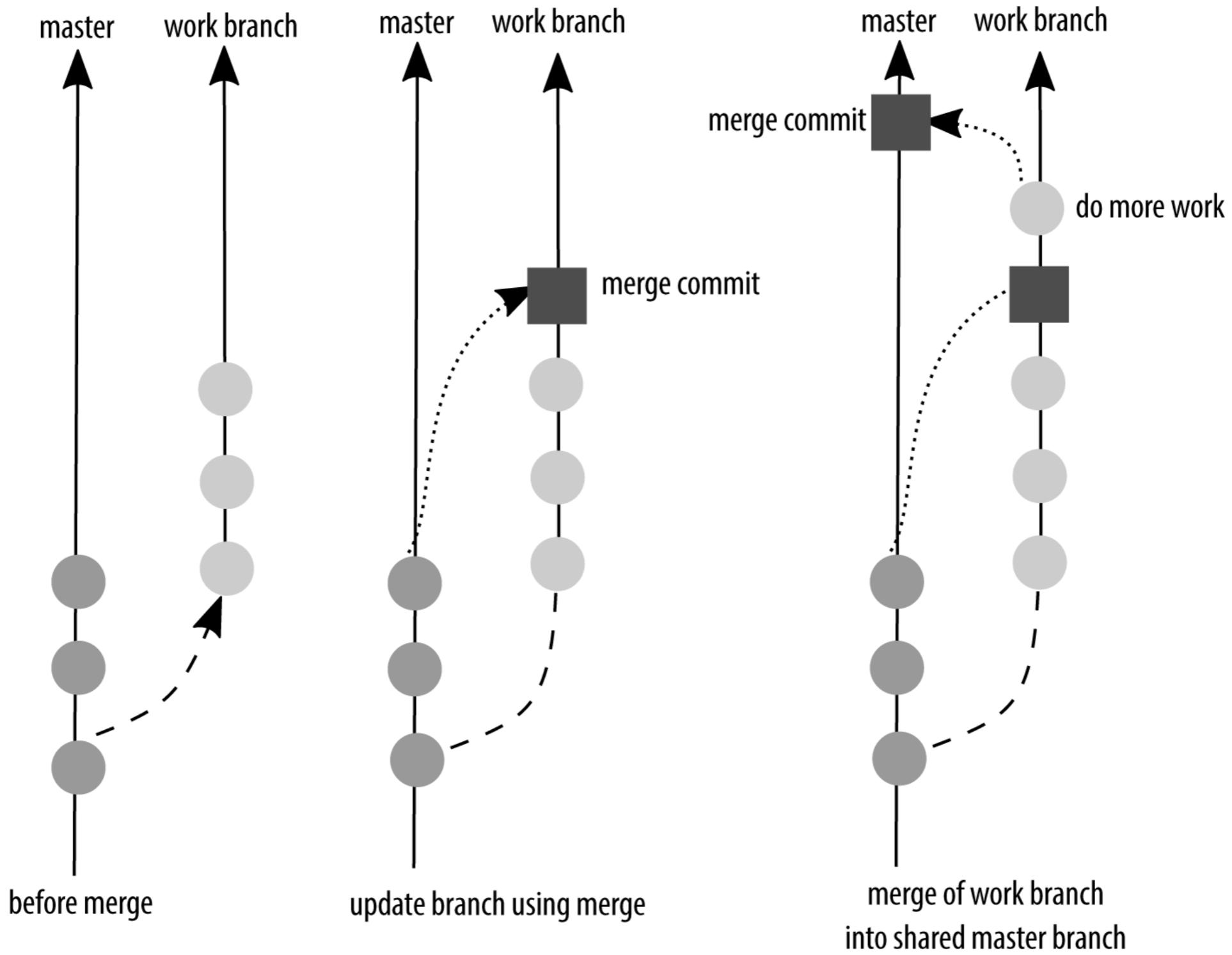




Let's discuss with Your team



Let's discuss with Your team



Maintainable Git



Developer cost

\$\$\$\$\$\$\$\$\$

Branching cost
(~40 bytes)

\$

Developer cost

\$\$\$\$\$\$\$\$\$

Poor branching strategy

\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$

Branching strategies

Everything on master

Last good state ?

Mistake break everyone !!

Lead to big commits !!

Trunk-based development

Branch per release

What is deployed ?
Where do fixed ?

Branch per developer

What is being worked on ?
How do you switch tasks ?
When do you integrated ?



Merge Hell !!



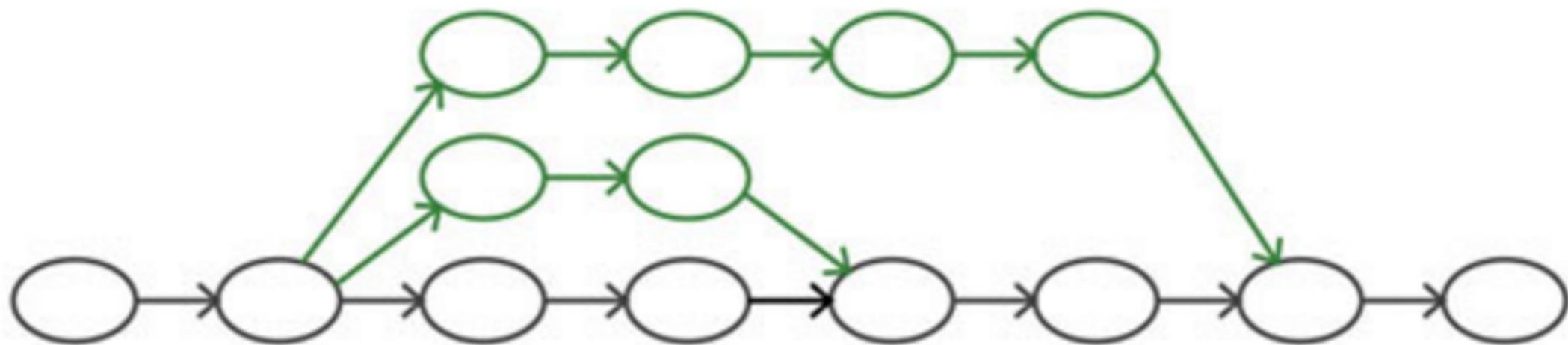
Release Process ?

Start simple

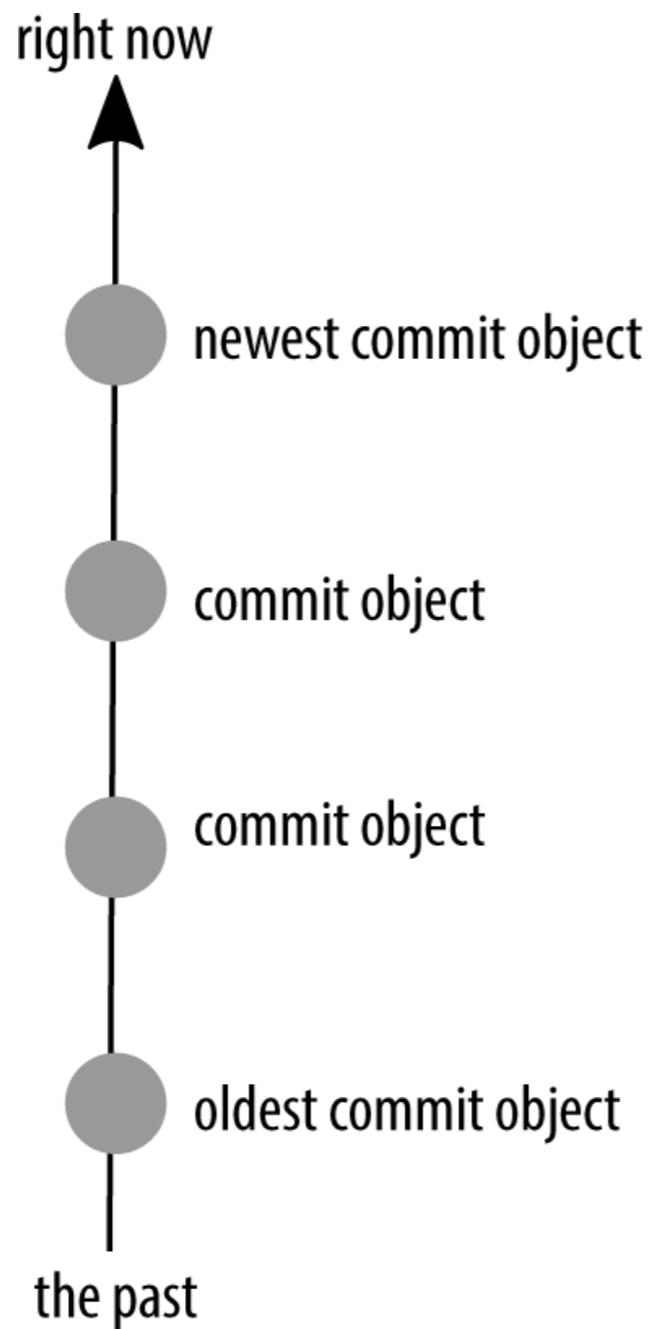
Create a new branch

when working on any feature, issue or bug

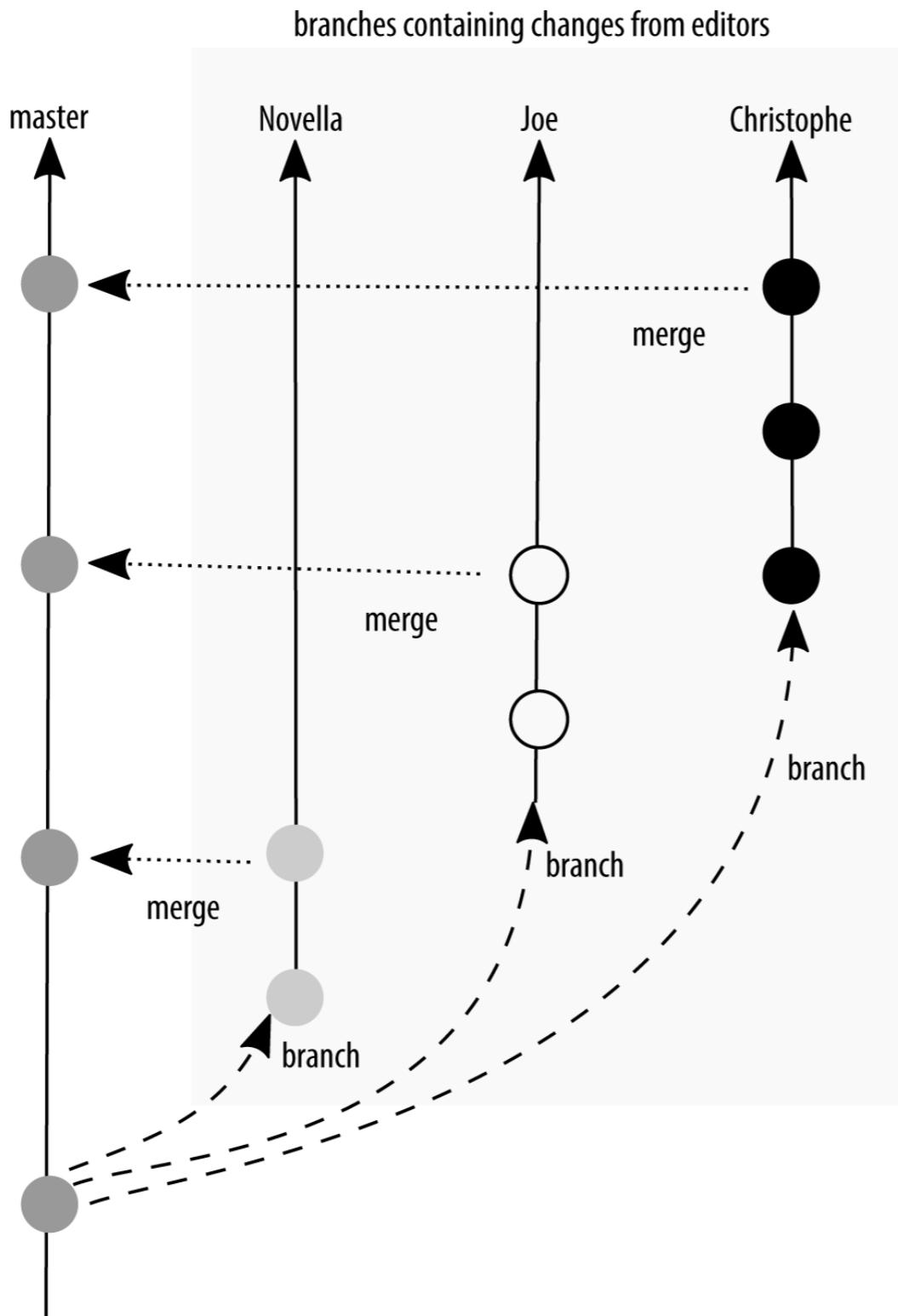
Merged back when it is finished



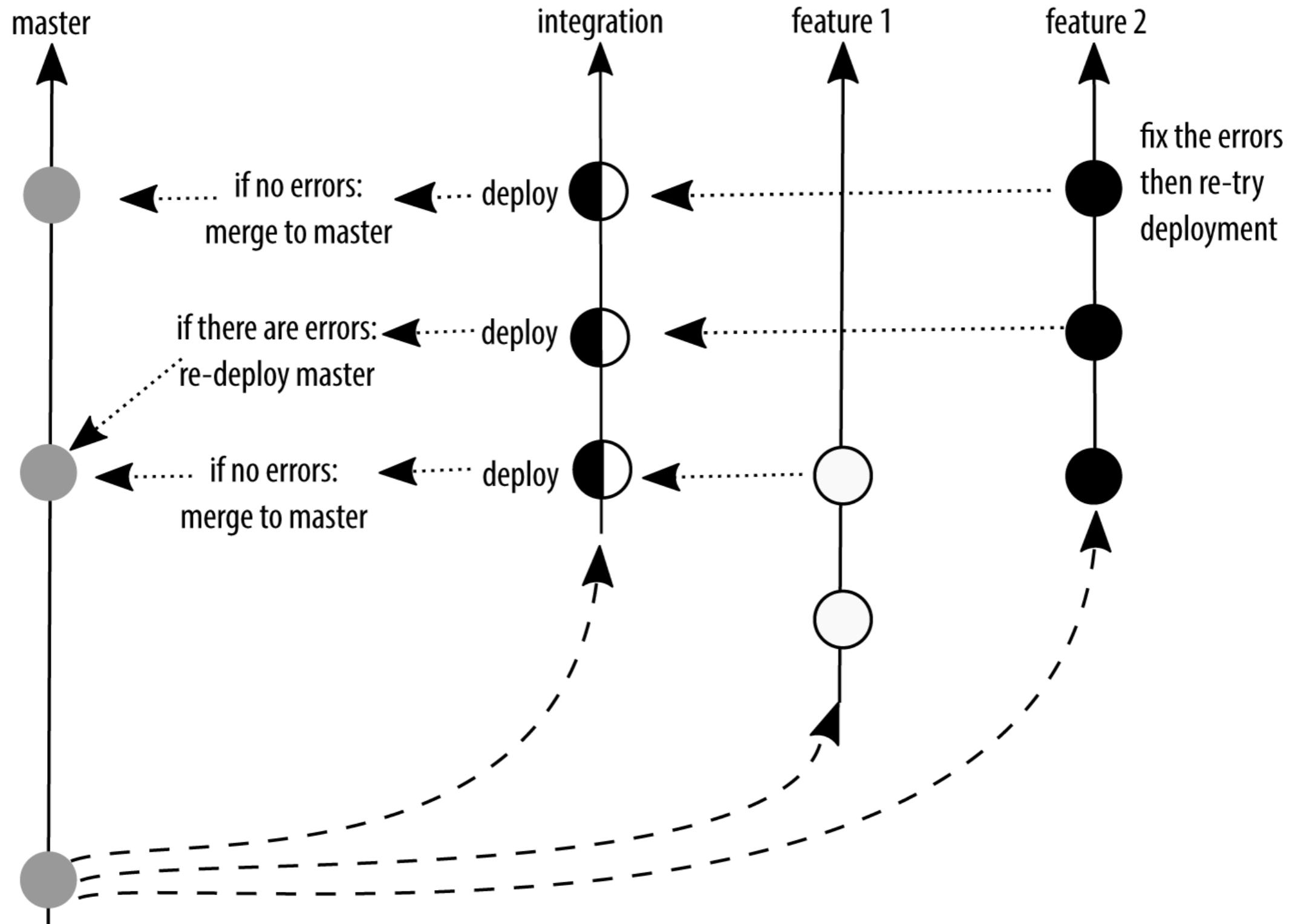
Mainline branching



Mainline branching

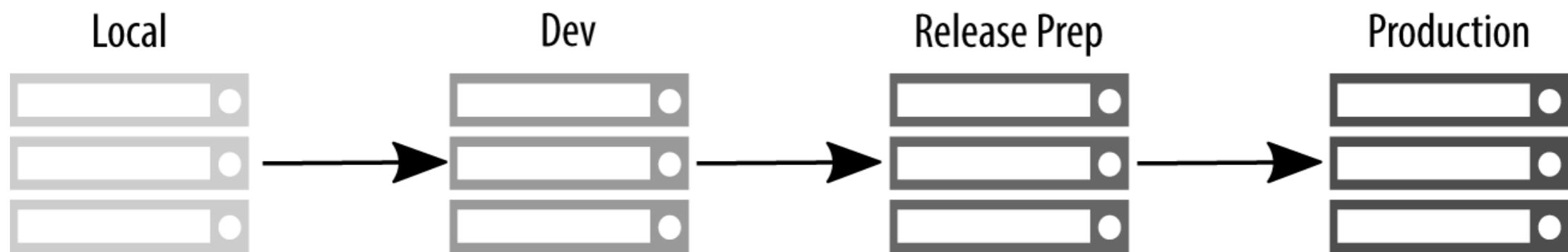


Feature branches

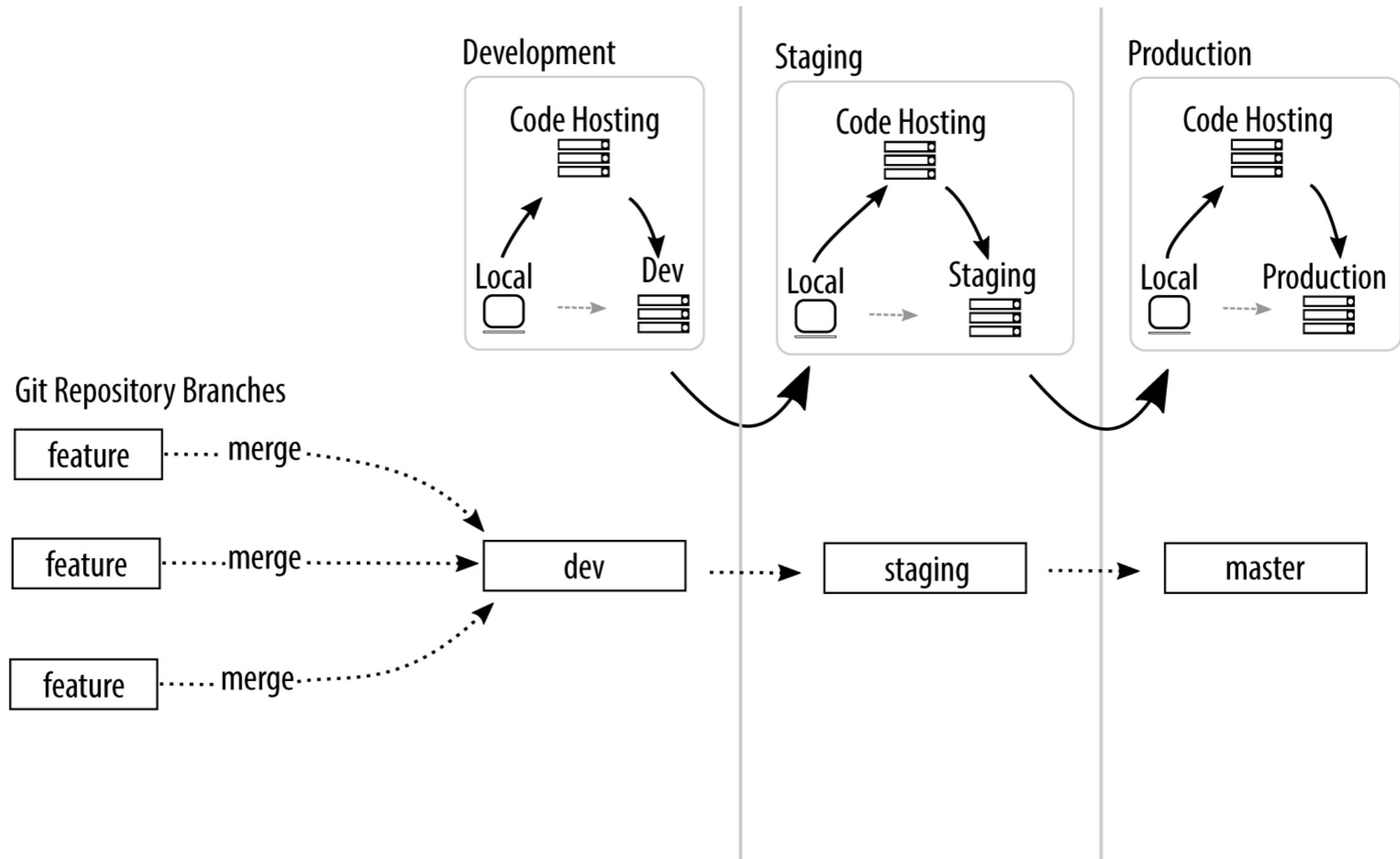


Environment branches

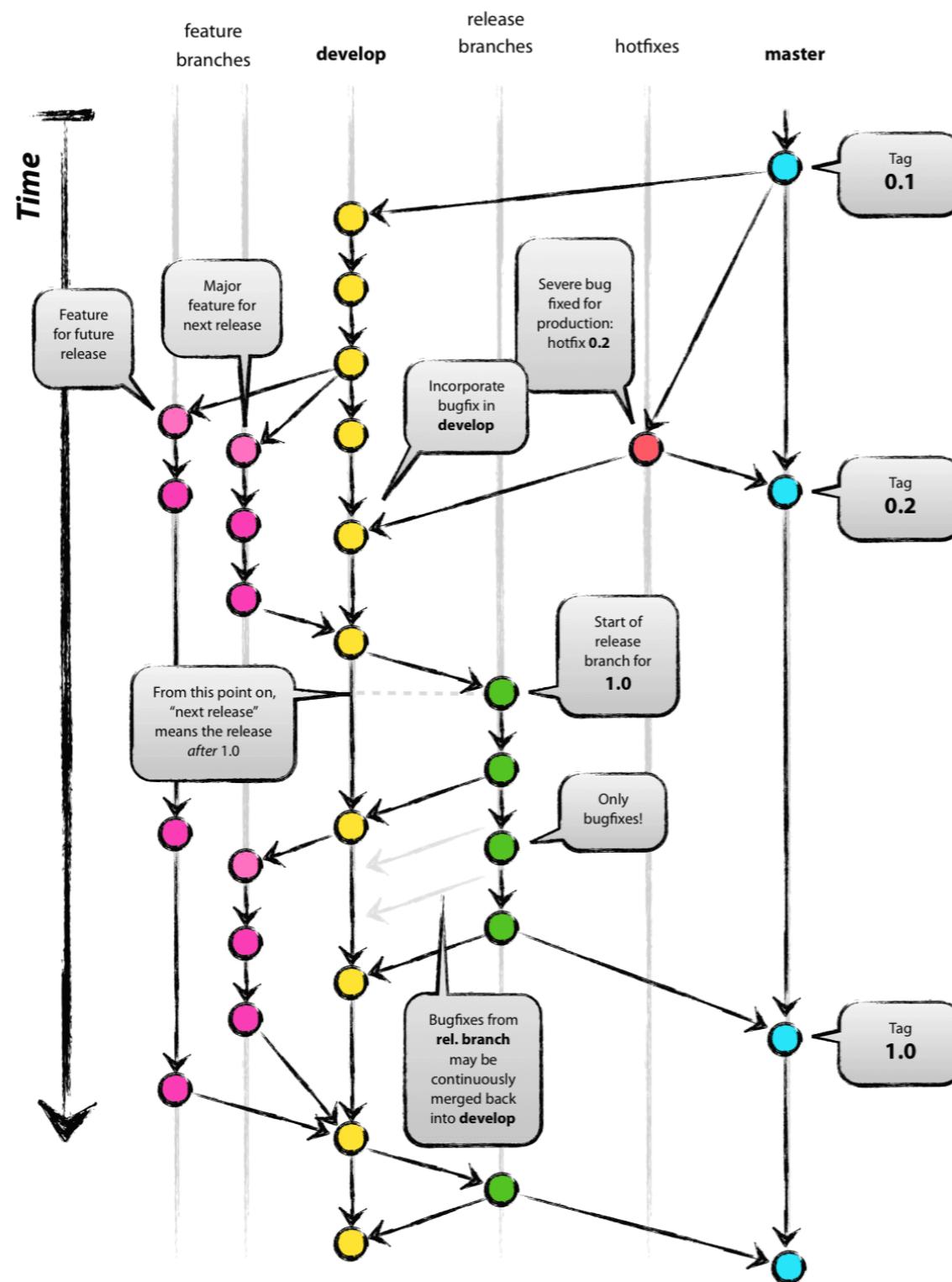
One branch per platform



Environment branches

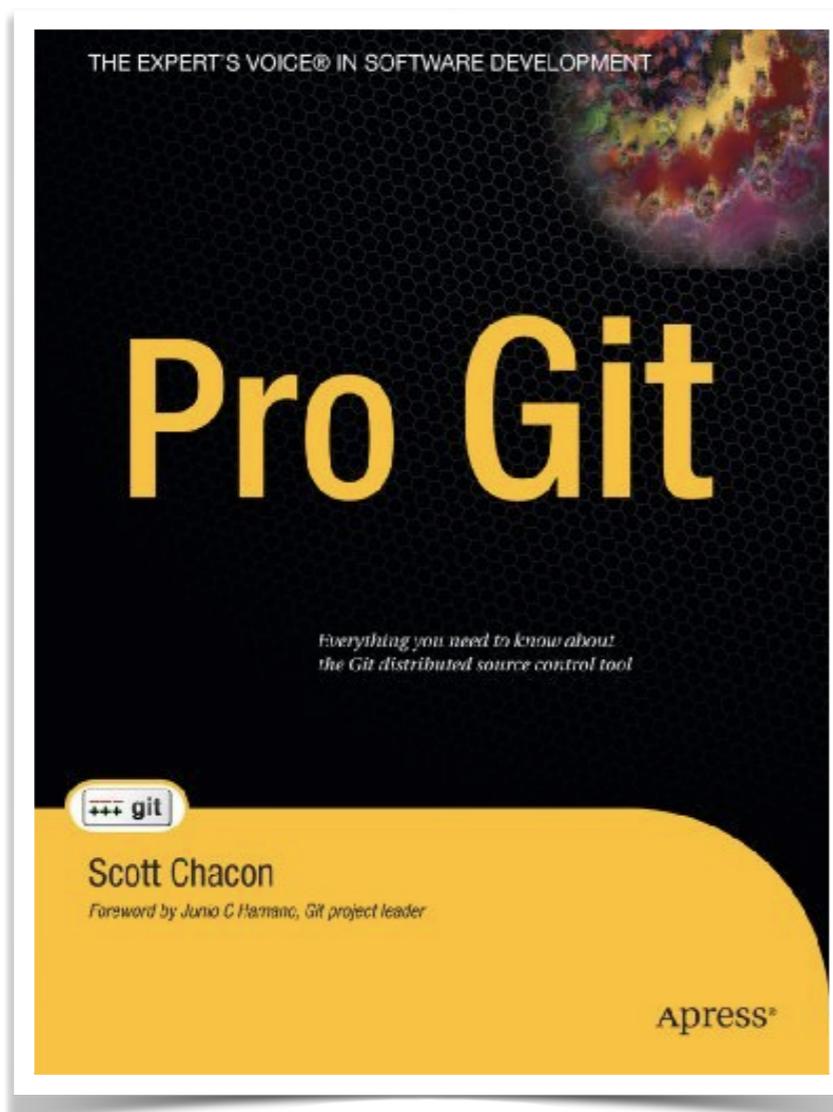


Git Flow

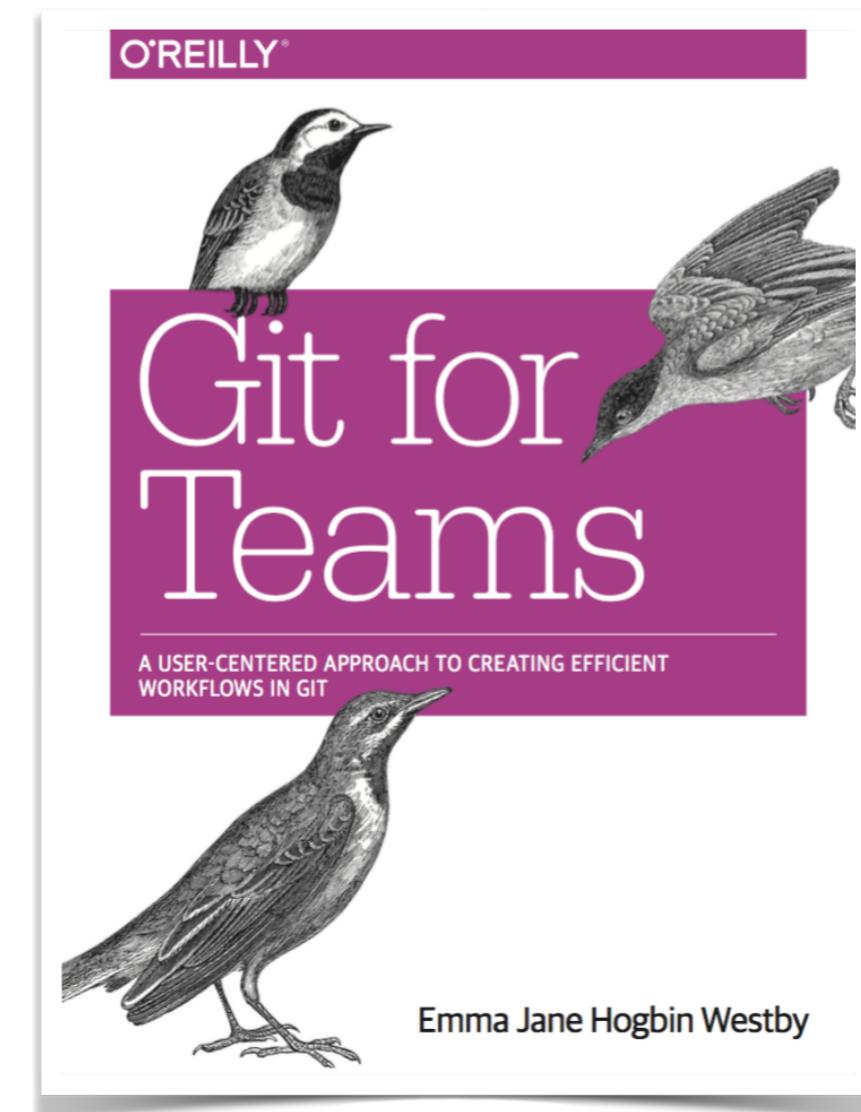


Let's discuss with Your team

Books to Read and Practice



<http://www.git-scm.com/book/en/v2>



<http://shop.oreilly.com/product/0636920034520.do>