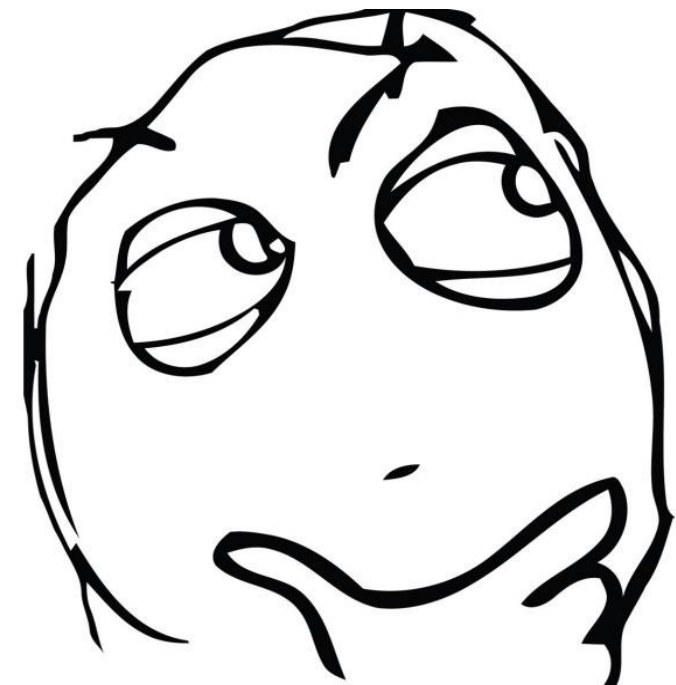


*GIT is not a VCS.*

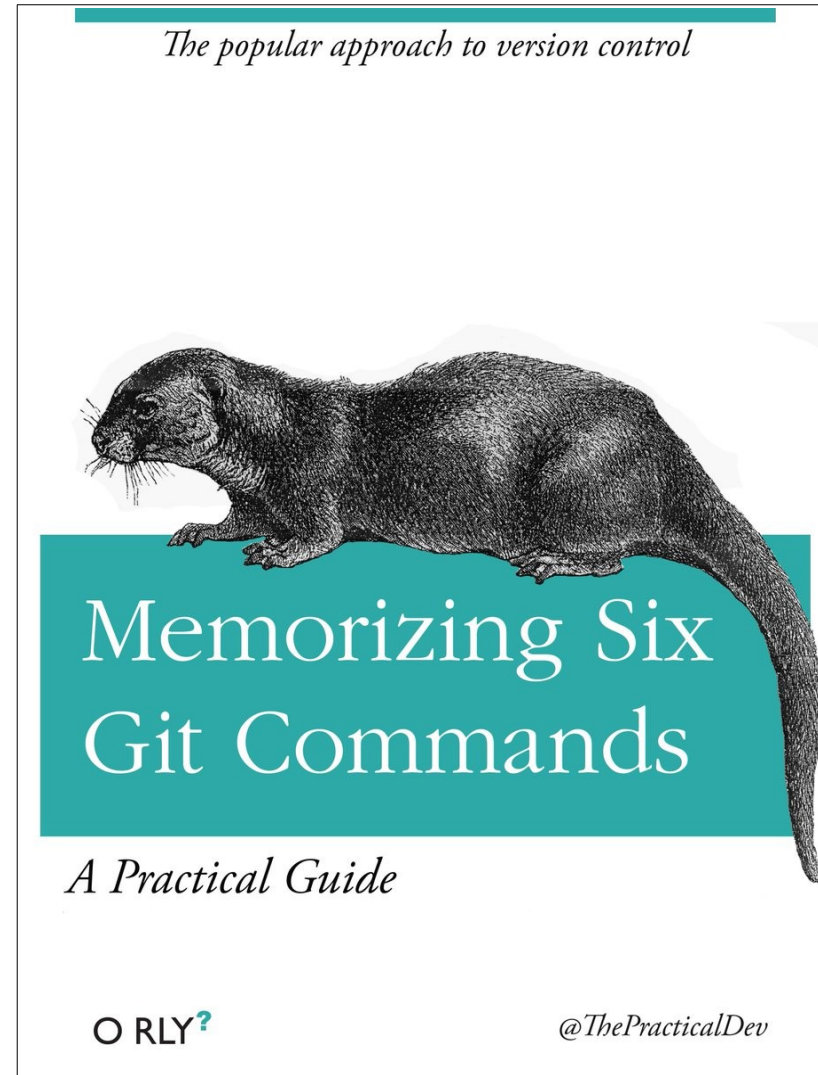
*That's why GIT is a great VCS*



*ern0@linkbroker.hu*

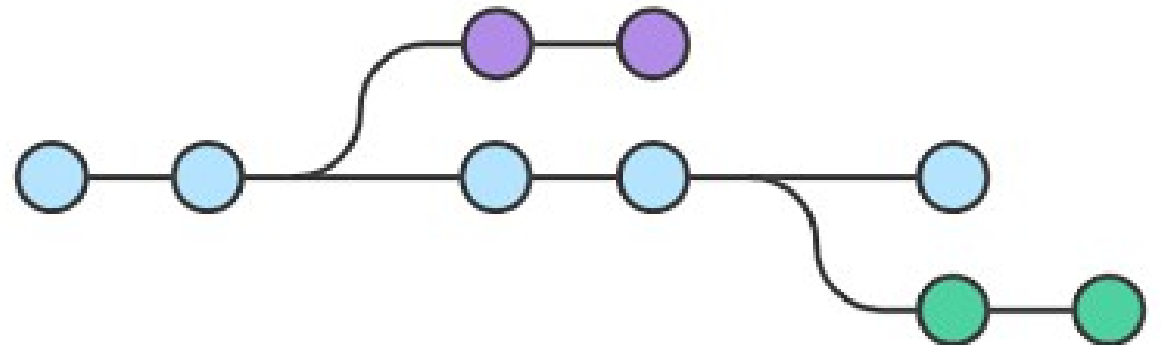
*V2 – 2023.04.17*

# This is not a Git course.



# This is not a GIT course.

- Definiton of VCS
- Living without VCS
- Version controlling concepts
- How GIT works
- ~~GIT commands, tips'n'tricks~~



# Version Control System (VCS)

A component of [software configuration management](#), **version control**, also known as **revision control** or **source control**,<sup>[1]</sup> is the management of changes to documents, [computer programs](#), large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a [timestamp](#) and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

(Wikipedia)

# VCS features:

- Central file repository
- Store all changes
- Recall any snapshot
- Organize snapshots
- Teamwork support
- Side effect: backup



*Thank you, Captain Obvious!*

w/o VCS



# Without VCS: editor backup

- myprg.bak
- myprogram.pas.bak
- myprogram.pas~
- myprg.pa~



# Without VCS: naming convention

- Logo2.psd
- Logo3b.psd
- Logo4-palettefix-veryfinal2.psd





# Without VCS: manual snapshot

- app\_old\_2/
- app\_1.3rc/
- app\_1.4\_20180402/
- app\_bak\_20180402.zip



# Without VCS: automatic backup

- local media
- network



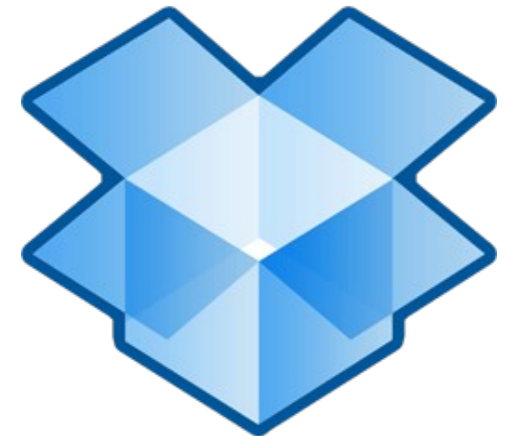
# Almost VCS:

- file versioning (VAX / VMS)
- journaling filesystem (AIX JFS)
- Time Machine (Mac OS)



# Poor man's VCS: Dropbox

- cloud sync (repo)
- file history (revert)
- pause sync (commit)



*GIT is not a VCS. That's why GIT is a great VCS*

---

**vcs**



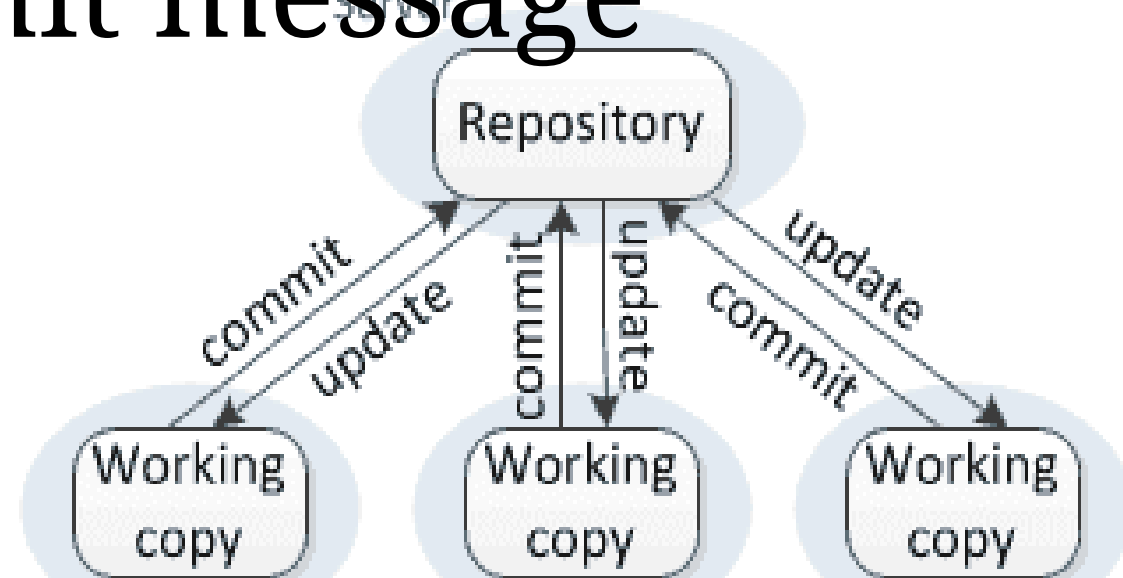
# VCS infrastructure

- server
- repository
- client



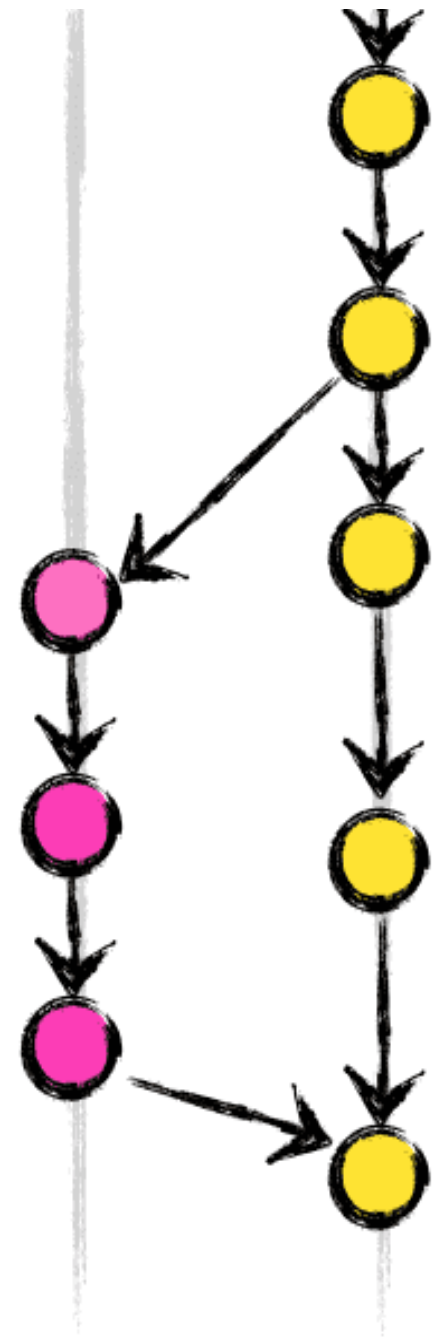
## VCS terms (basic)

- checkout
- working copy
- file is under version control
- **commit**, commit message
- revert
- changelog



## VCS terms (advanced)

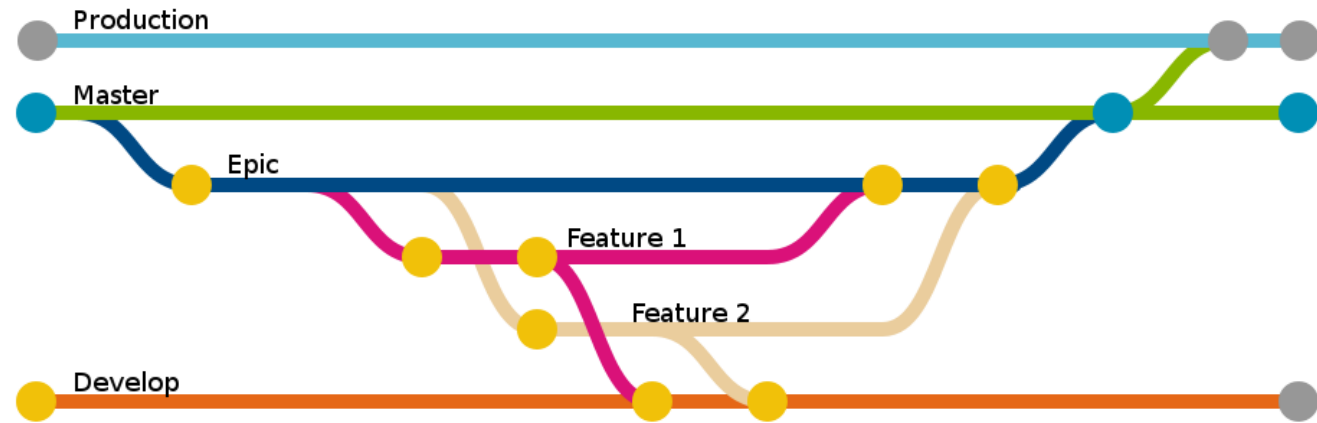
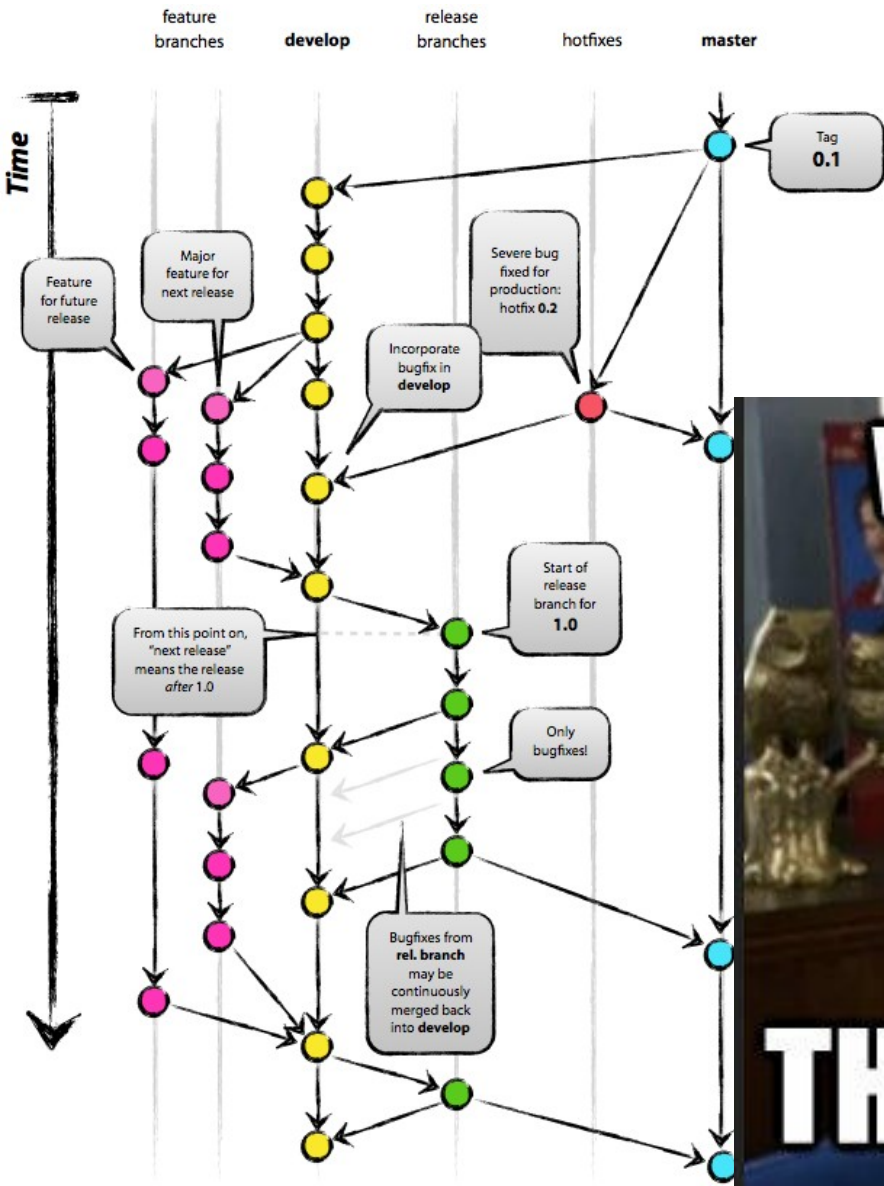
- **branch**, head
- locking / shared checkout
- merge
- merge conflict
- diff (change, delta)
- merge conflict resolve





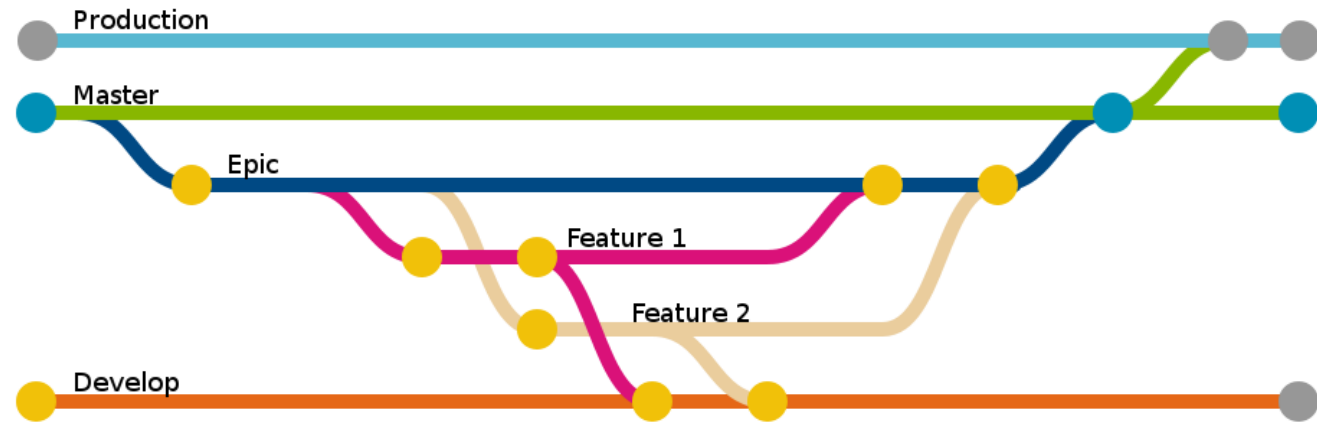
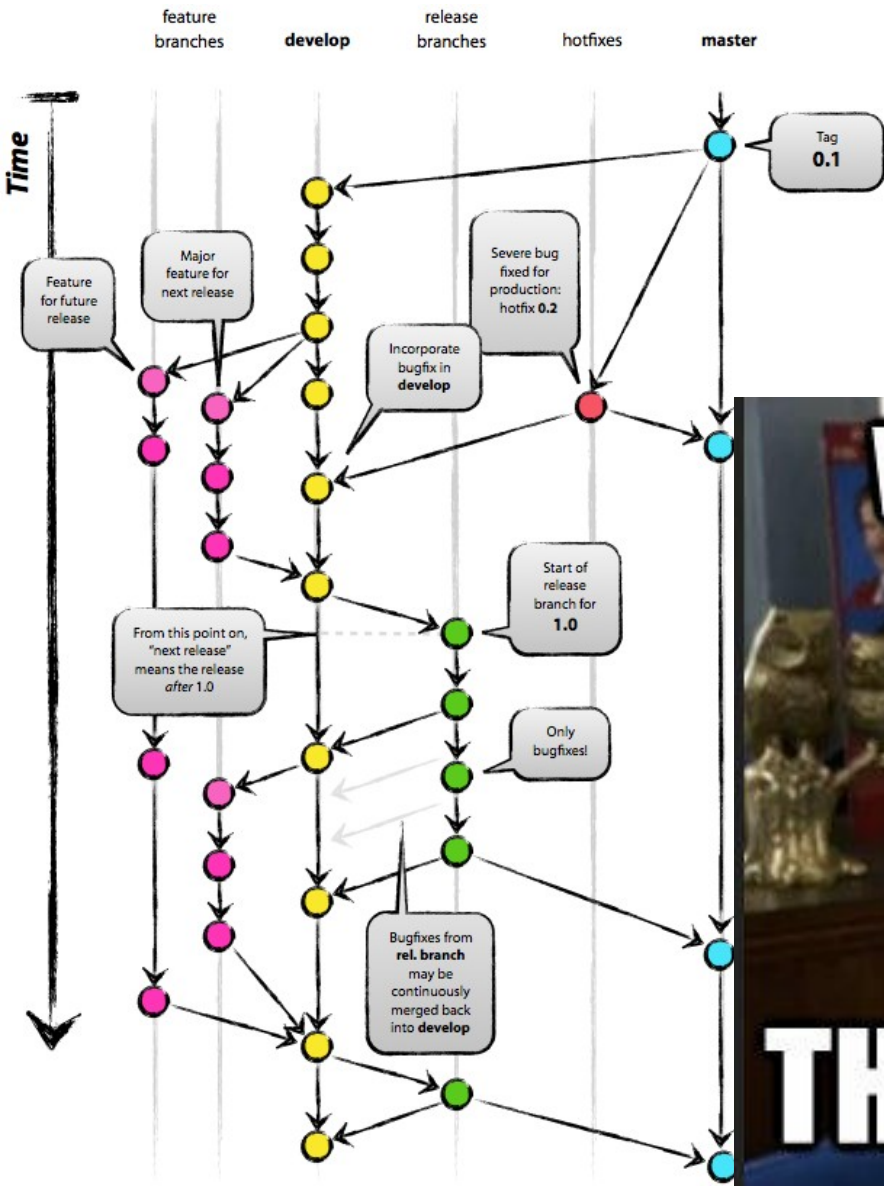
*GIT is not a VCS. That's why GIT is a great VCS*

# VCS is about branching



*GIT is not a VCS. That's why GIT is a great VCS*

# Development is about branching



*GIT is not a VCS. That's why GIT is a great VCS*

---

**GIT**

# GIT

**In case of fire**



1. `git commit`



2. `git push`



3. `leave building`

# Three states

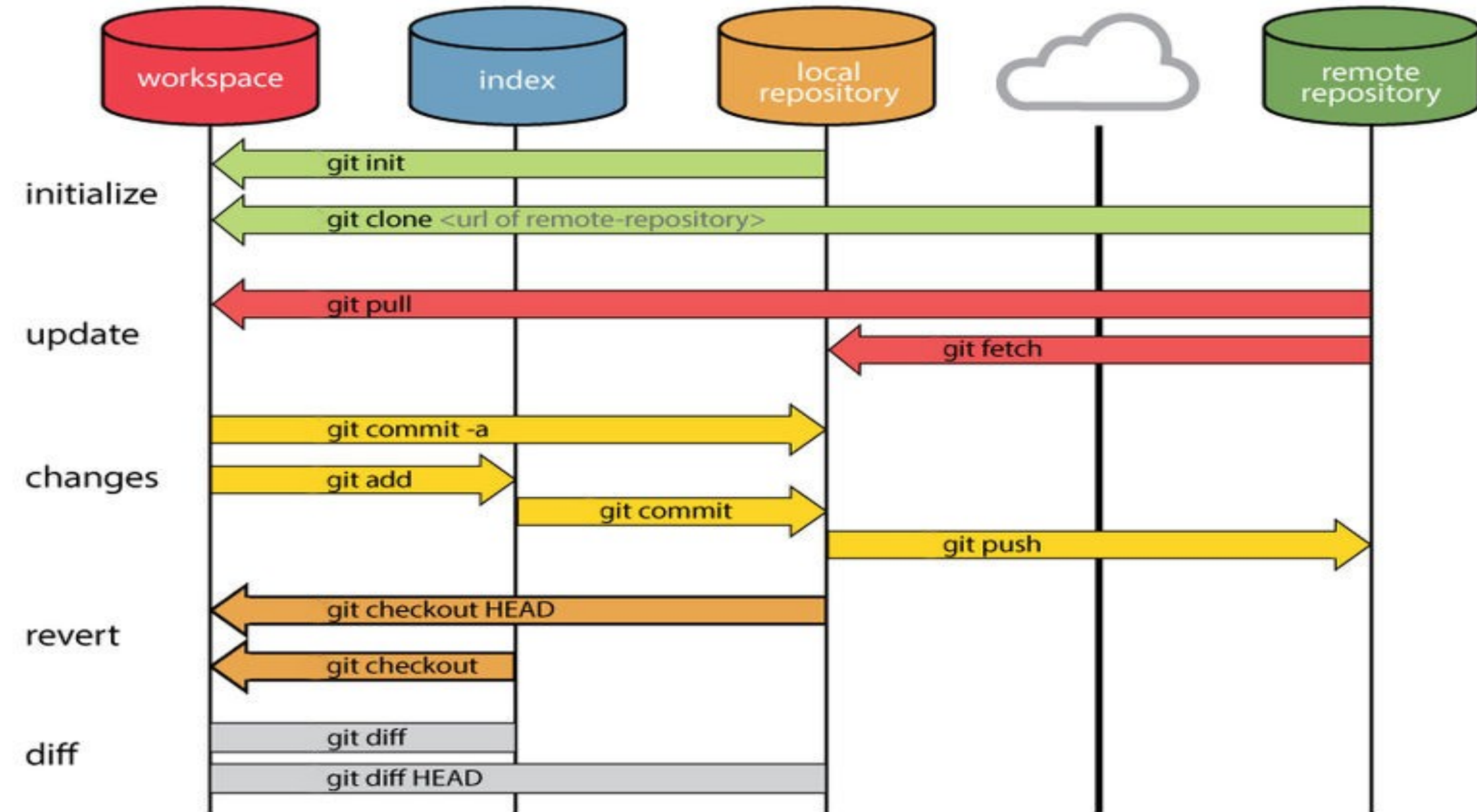
- Working Directory
  - Staging Area (a.k.a. Index)
  - Local Repository `.git/`
- 
- Remote Repository

# Three states

- Working Directory
  - Staging Area (a.k.a. Index)
  - Local Repository `.git/`
- 
- Remote Repository  
(GitHub, GitLab etc.)



# *GIT is not a VCS. That's why GIT is a great VCS*



# diff

```
One two three four
Can I have a little cake
Five six seven eight nine ten
I love you
A B C D
Can I bring my friend to coffee
```



```
One two three four
Can I have a little more
Five six seven eight nine ten
I love you
A B C D
Can I bring my friend to tea
E F G H I J
I love you
```

```
--- a 2018-04-23
21:35:37.0000000000 +0200
+++ b 2018-04-23
21:35:10.0000000000 +0200
@@ -1,6 +1,8 @@
    One two three four
-Can I have a little cake
+Can I have a little more
    Five six seven eight nine ten
    I love you
    A B C D
-Can I bring my friend to coffee
+Can I bring my friend to tea
+E F G H I J
+I love you
```



# colordiff

```
One two three four
Can I have a little cake
Five six seven eight nine ten
I love you
A B C D
Can I bring my friend to coffee
```



```
One two three four
Can I have a little more
Five six seven eight nine ten
I love you
A B C D
Can I bring my friend to tea
E F G H I J
I love you
```

```
--- a 2018-04-23
21:35:37.000000000 +0200
+++ b 2018-04-23
21:35:10.000000000 +0200
@@ -1,6 +1,8 @@
One two three four
-Can I have a little cake
+Can I have a little more
Five six seven eight nine ten
I love you
A B C D
-Can I bring my friend to coffee
+Can I bring my friend to tea
+E F G H I J
+I love you
```

# The repository

- series of snapshots
- diff to previous
- commit: new node based on selected one



*(stay tuned)*

# hash (md1)

```
$ echo "We Build Robots" | sha1sum
0b19fb7e8ad55d5b0fc5bf27411c63a4c0b182fe -

$ echo "We Build Robots." | sha1sum
99d7c75c3cf51f3712058a3eba56837314b35156 -

$ echo "We build robots" | sha1sum
fc3504b54ff836e66d956b1056c6549b991d01fa -

$ echo "We Build Robots" | sha1sum
0b19fb7e8ad55d5b0fc5bf27411c63a4c0b182fe -
```

# hash (md1)

```
commit 029978ed2767be597f159b5e0eb29f754b410379
Author: ern0 <ern0@linkbroker.hu>
Date:   Fri Apr 20 00:00:24 2018 +0200

    fmt stub added

commit f705e402ceb518011bea3d8a813e45233101ed27
Author: ern0 <ern0@linkbroker.hu>
Date:   Thu Apr 19 22:46:03 2018 +0200

    added universal hash check function
```

git log

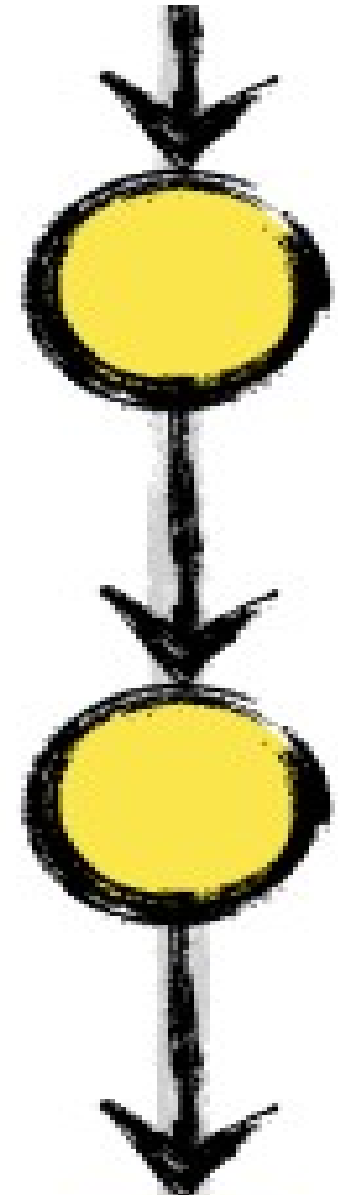


tig

```
029978e 2018-04-20 00:00 ern0  o fmt stub added
f705e40 2018-04-19 22:46 ern0  o added universal hash check function
```

# The repository element (commit)

- commit ID (hash)
- previous commit ID
- snapshot (diff...)
- merged commit ID (opt.)
- timestamp
- author
- commit message




# Working with commits

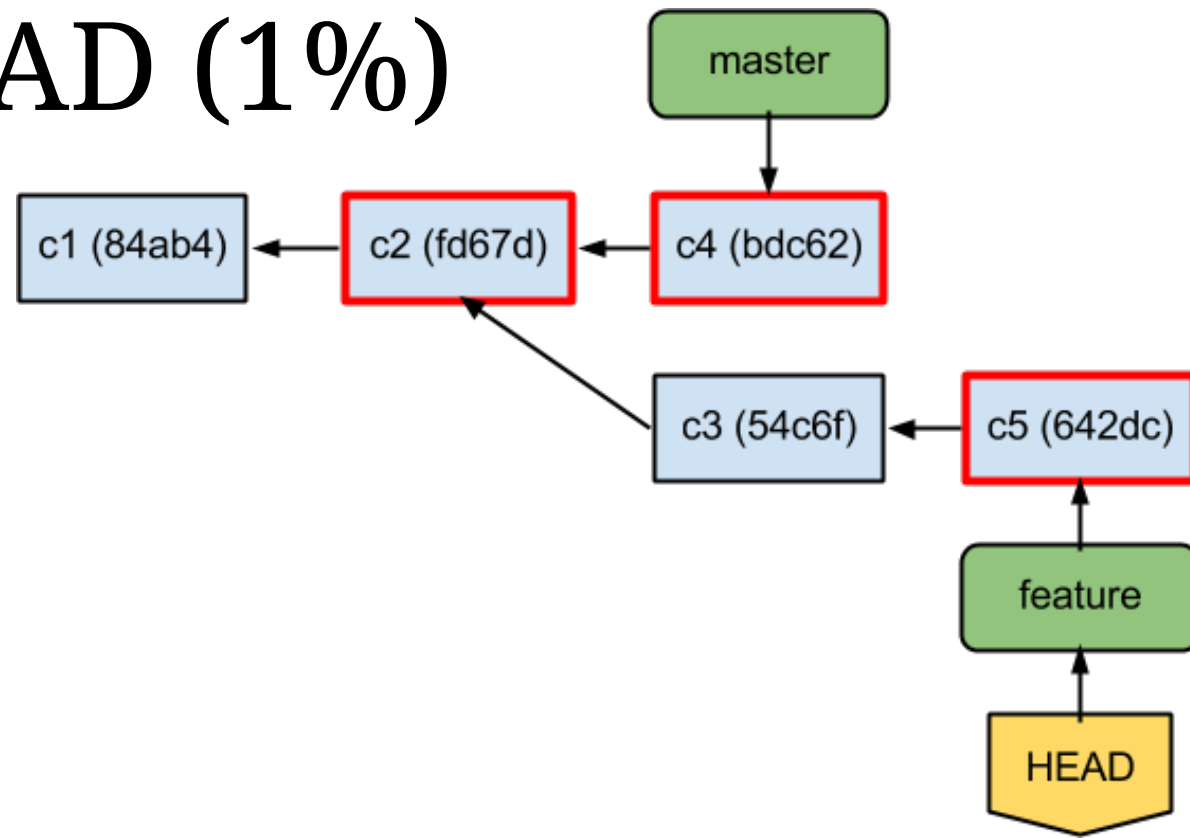
- pick a base commit
- apply changes
- add new commit  
to the repository



*GIT logo*

# Referring to commits

- commit ID (0.00%) 
- tag (0.01%)
- relative to HEAD (1%)
- **branch (99%)**



# Referring by commit ID



```
$ git checkout fb154891753699fa38beb4525412ed7b1995ed4e
HEAD is now at fb15489 html file created
```

```
$ echo "</html>" >> index.html      # workspace
$ git add index.html                # staging
```

```
$ git commit -m "closing tag added to html file"
[detached HEAD bbbd55a] closing tag added to html file
1 file changed, 1 insertion(+)
```

```
$ git log
commit bbbd55a86a6a54a360bc830700c720fb6b65edde (HEAD)
Author: ern0 <ern0@linkbroker.hu>
    closing tag added to html file
```

```
commit fb154891753699fa38beb4525412ed7b1995ed4e
Author: ern0 <ern0@linkbroker.hu>
    html file created
```



# Referring by commit ID

```
$ git checkout fb699fa38b222ed7b199  
$ git show --format=%B fb699fa38b222ed7b199
```

```
$ echo "</html>" >> index.html # workspace  
$ git add index.html           # staging
```

```
$ git commit -m "closing tag added to html file"  
[detached HEAD b1548917] closing tag added to html file  
1 file changed, 1 insertion(+)
```

```
$ git log  
commit b1548917700c720fb6b65ed4e (HEAD)  
Author: ern0 <ern0@broker.hu>  
Date:   2015-04-15 15:48:17 +0200  
    closing tag added to html file  
  
commit b1548917700c720fb6b65ed4e  
Author: ern0 <ern0@broker.hu>  
Date:   2015-04-15 15:48:17 +0200  
    html file created
```

# Referring by tag

- named reference to a node
- once set, never changes
- `git checkout v1.8.1`

git checkout

❖ v0.8.5.53 Tag at 0a8d1bd0

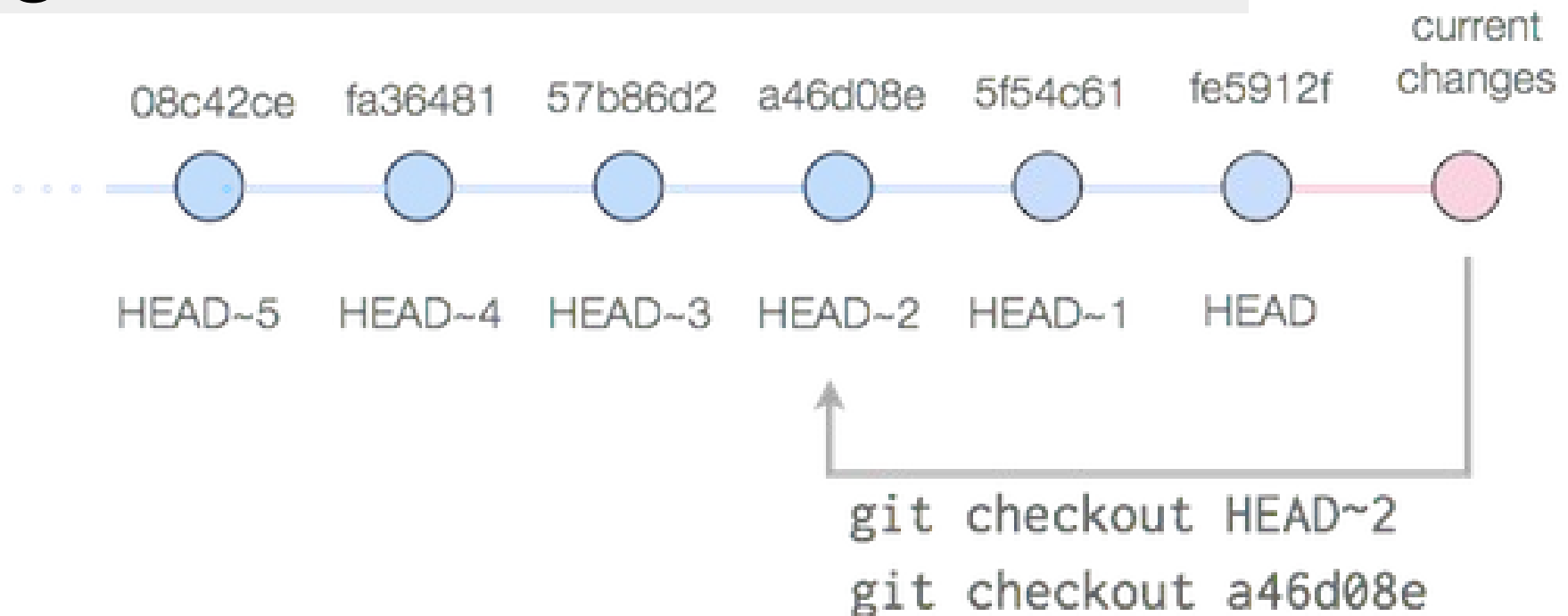
❖ v0.8.5.56 Tag at 17053acb

❖ v0.8.61 Tag at c1c1c4fd

❖ v0.8.65 Tag at da0902c7

# Referring relative to HEAD

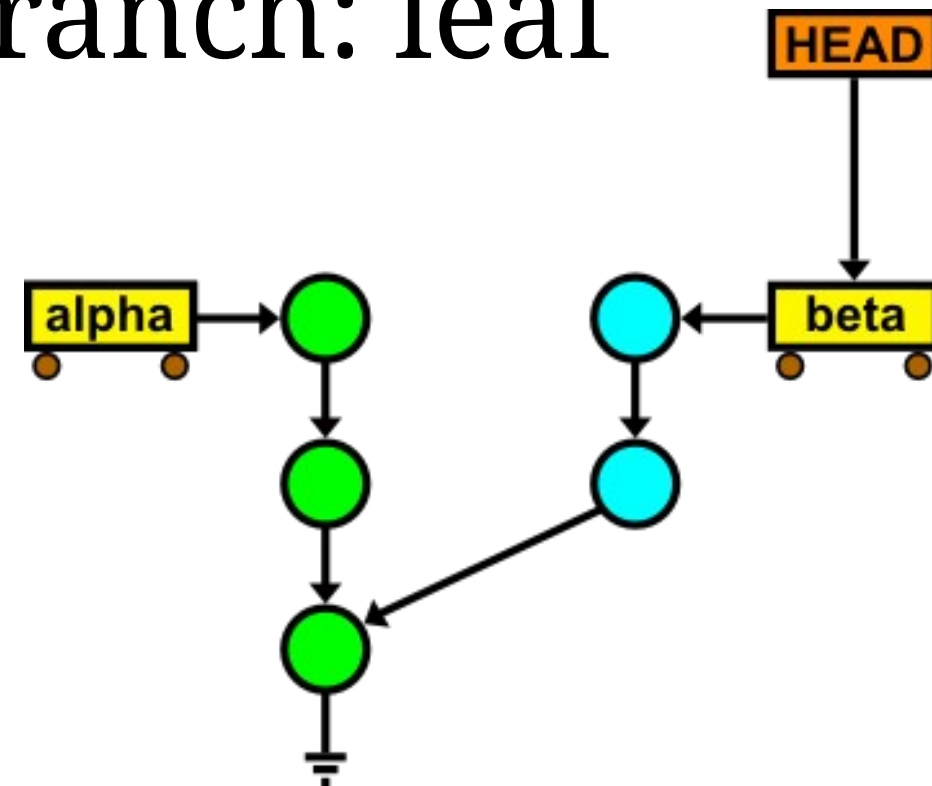
- using *previous commit ID* refs
- `git diff HEAD^`
- `git checkout HEAD~2`



## Referring by **branch** (99%)

- named reference to a node
- actual branch follows HEAD
- repository: tree, branch: leaf
- detached HEAD:  
no branch,

GC will delete it

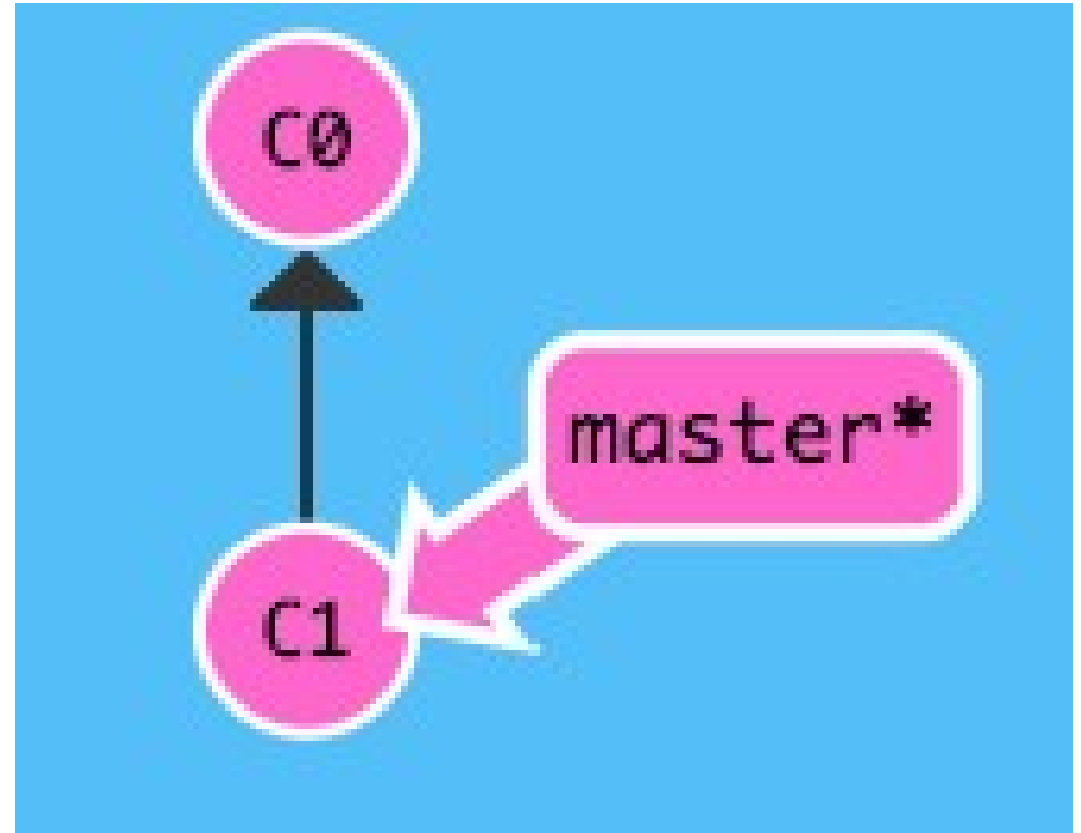


# Branch



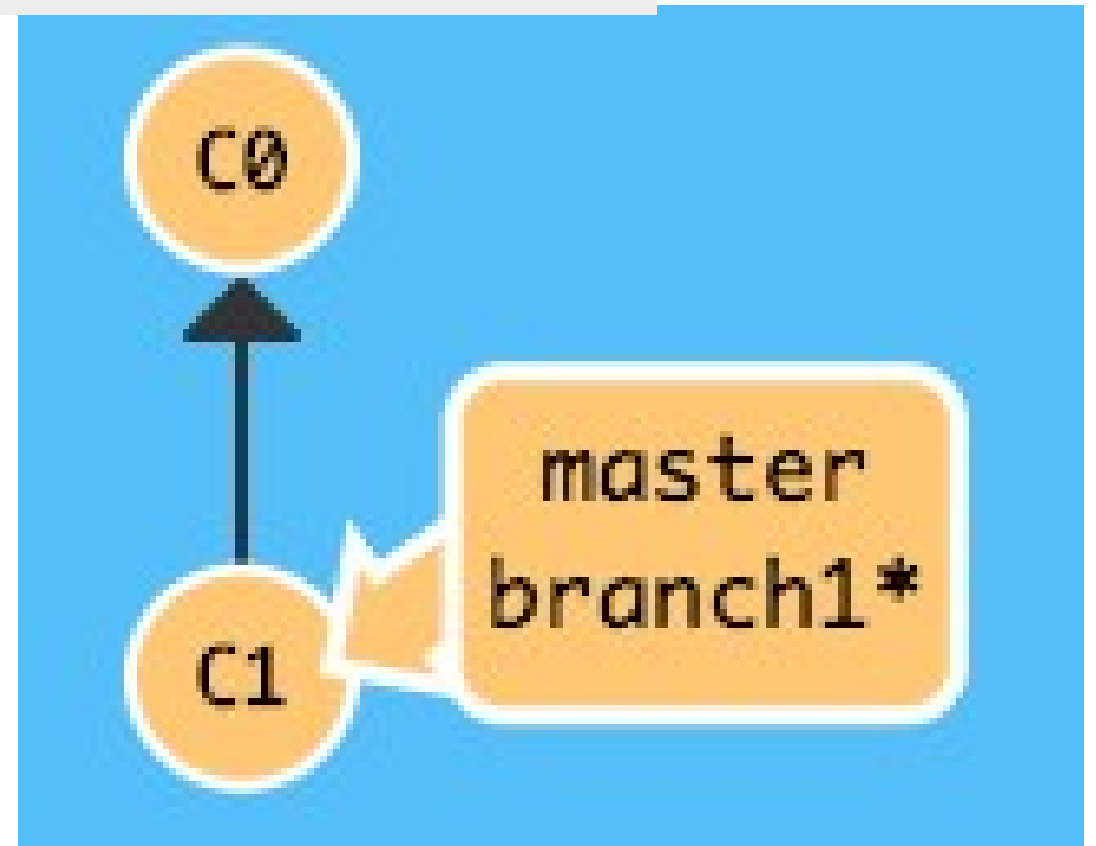
# Creating a commit

```
git commit
```



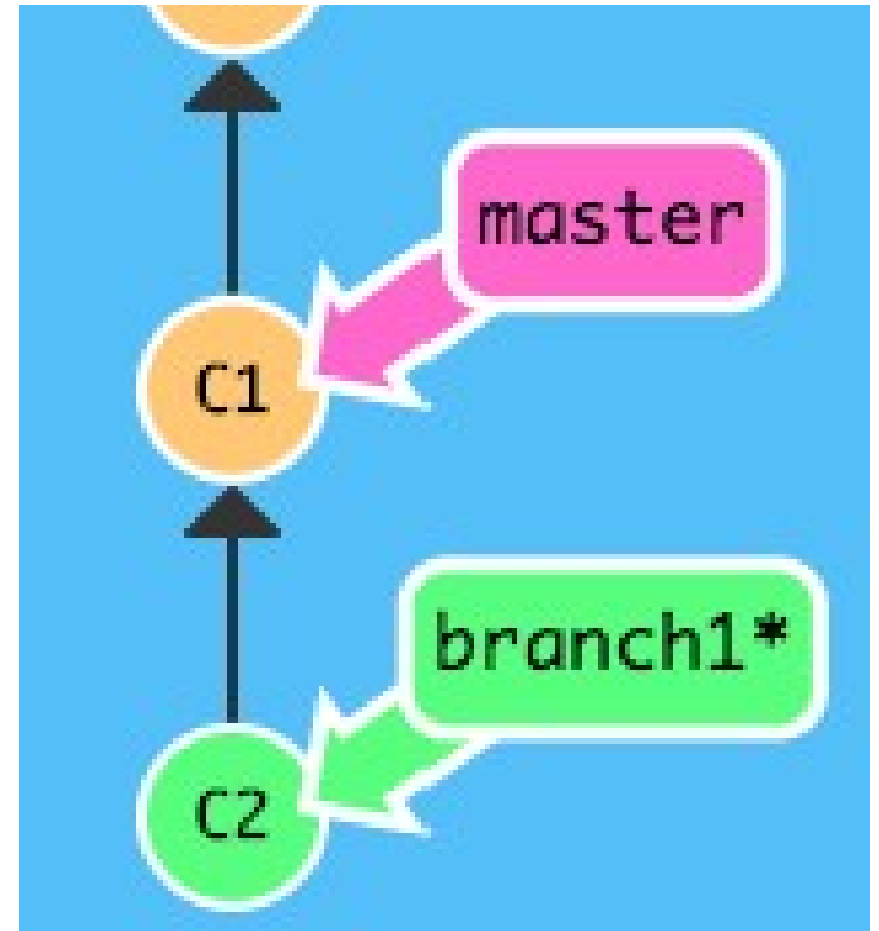
# Creating a branch

```
git checkout -b branch1
```



# Commit on the new branch

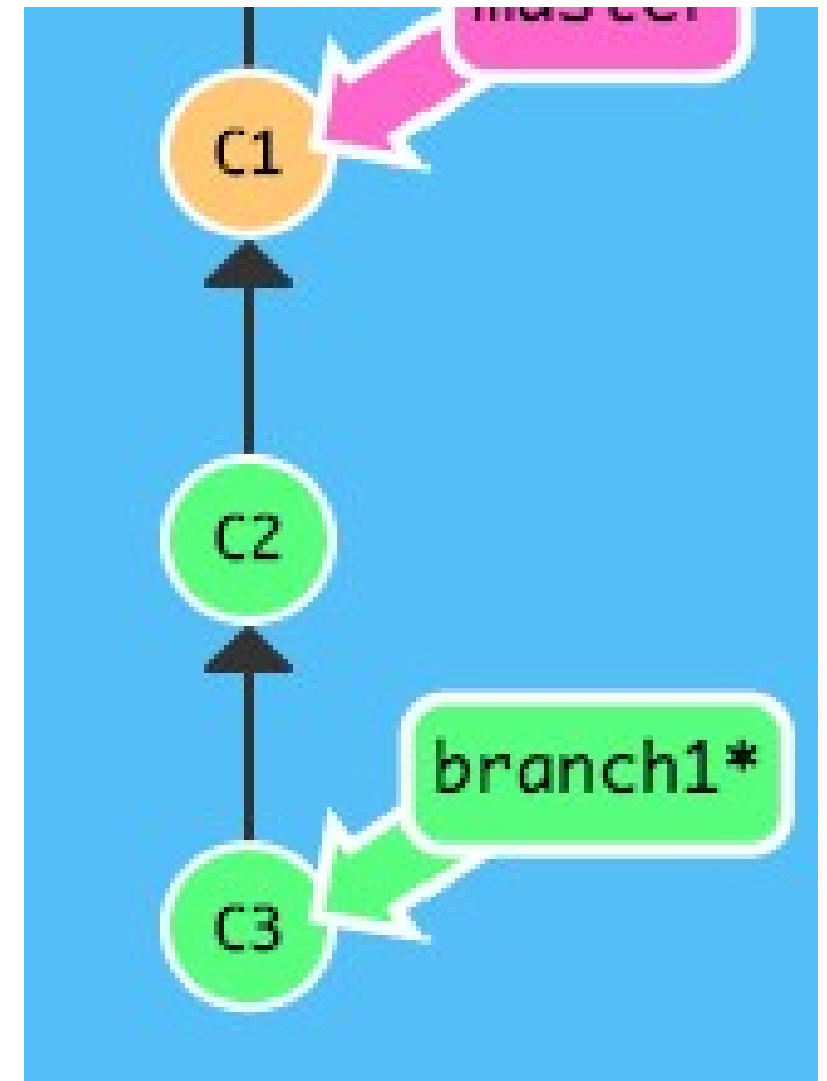
```
git commit ...
```





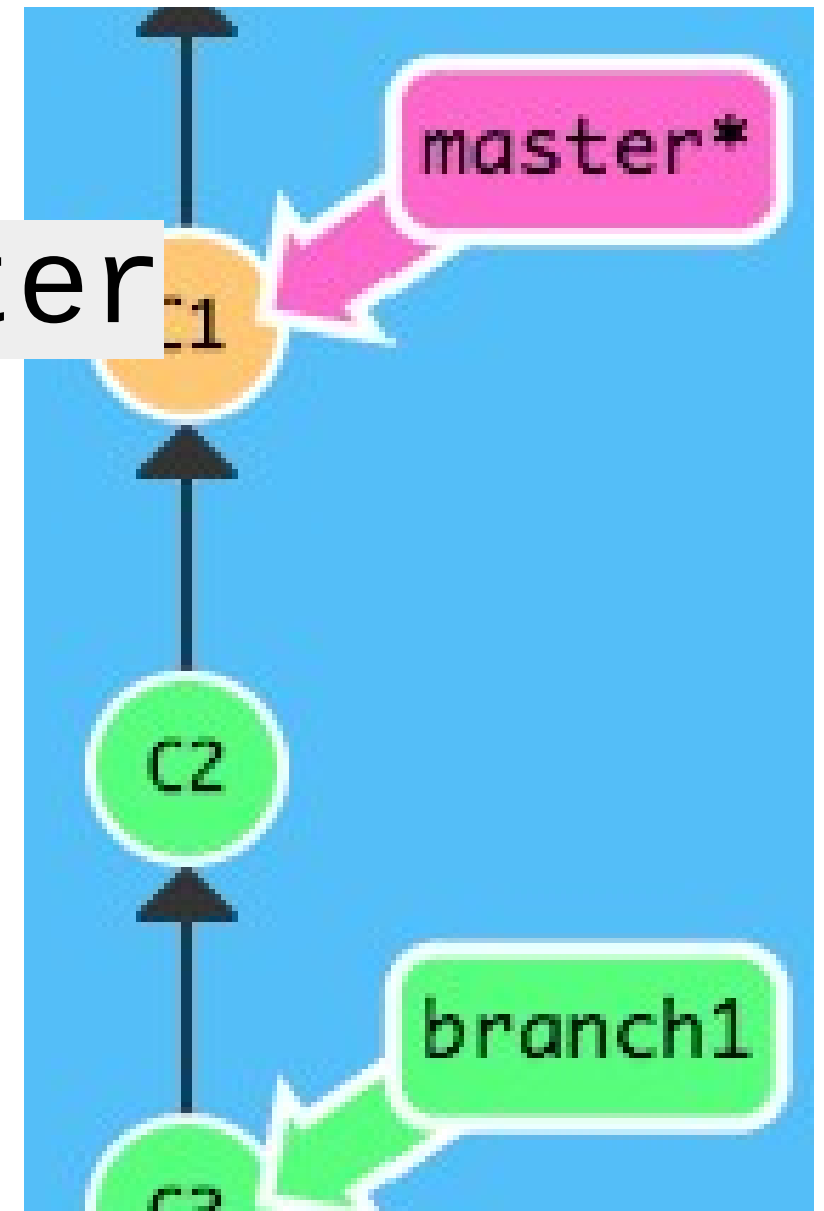
# Another commit

```
git commit ...
```



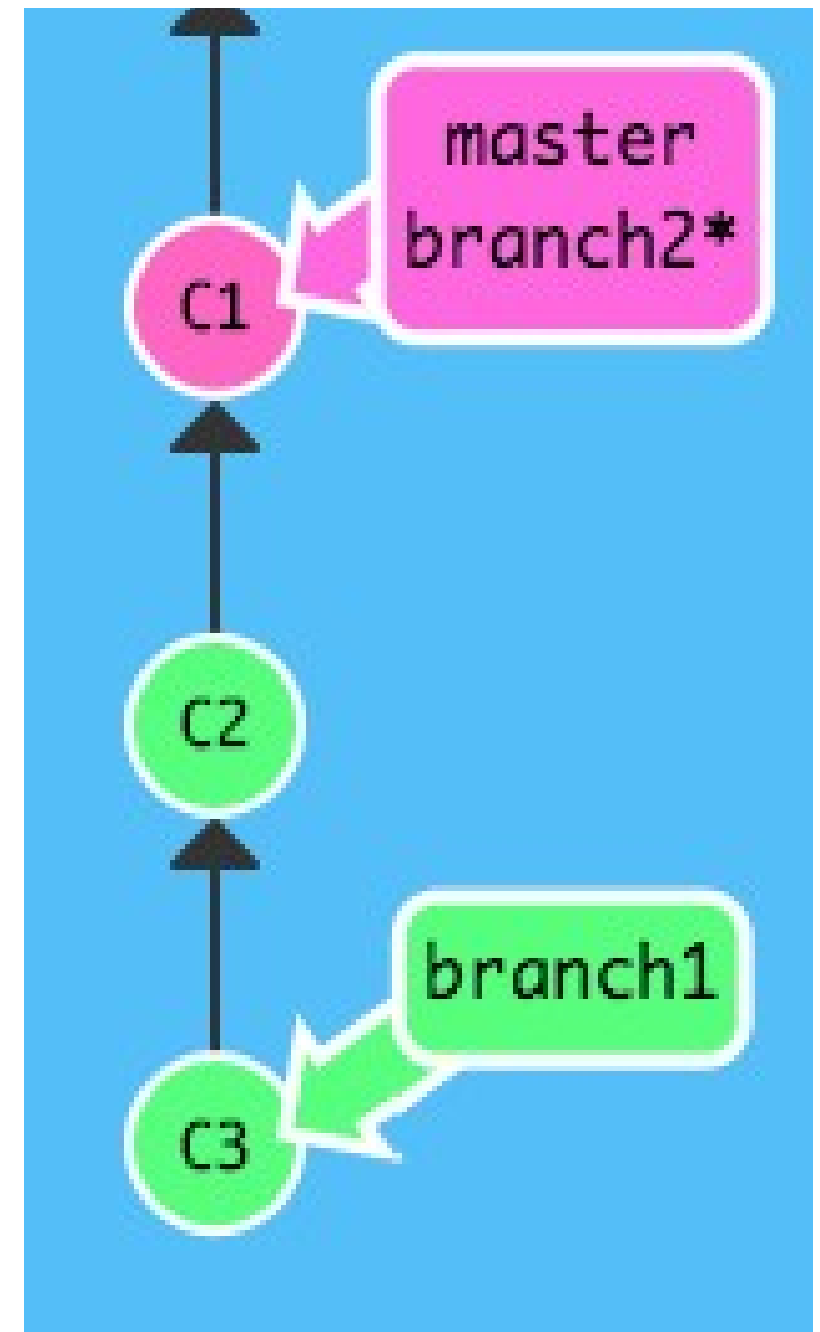
# Back to master

```
git checkout master
```



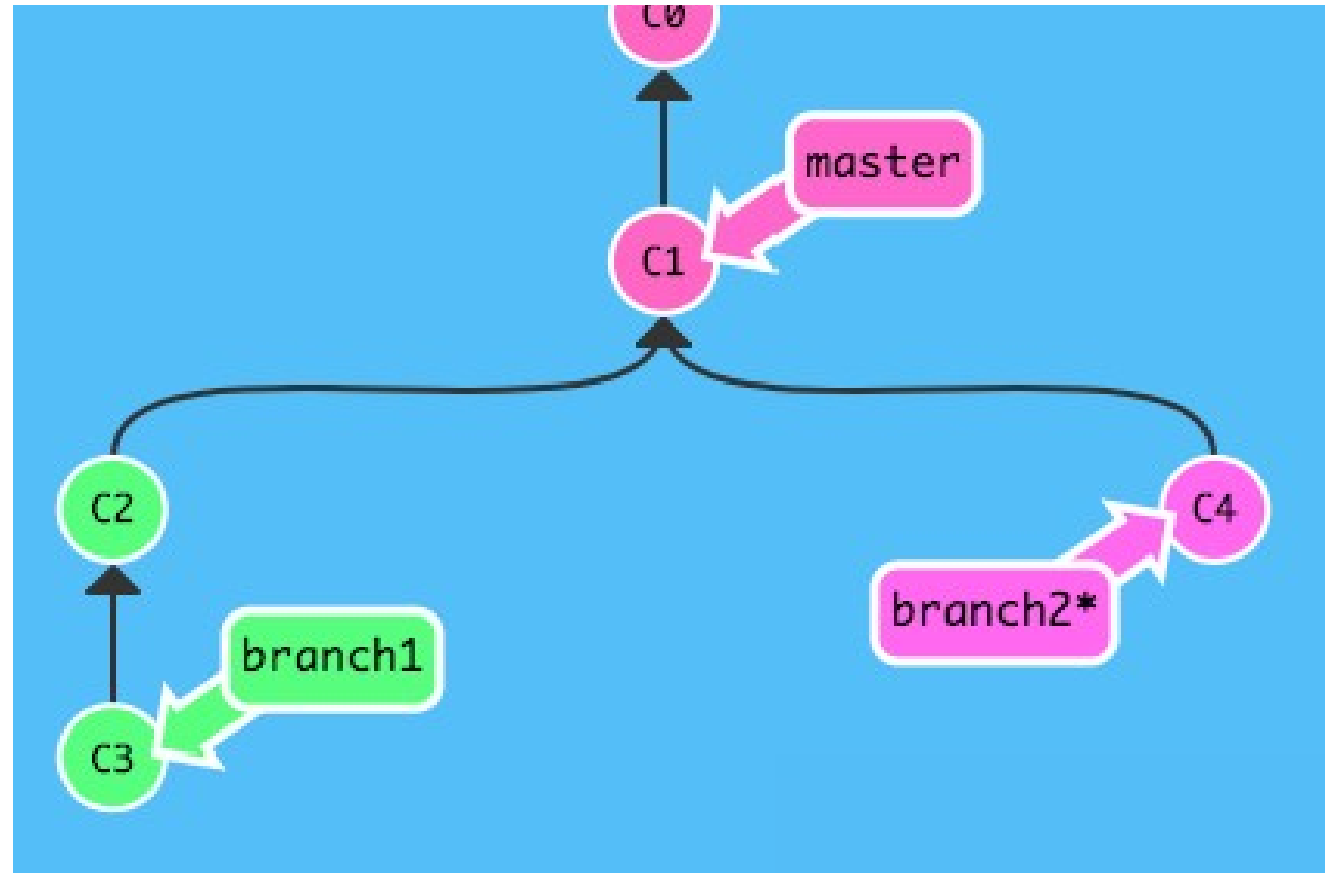
## Another new branch

```
git checkout \  
-b branch2
```



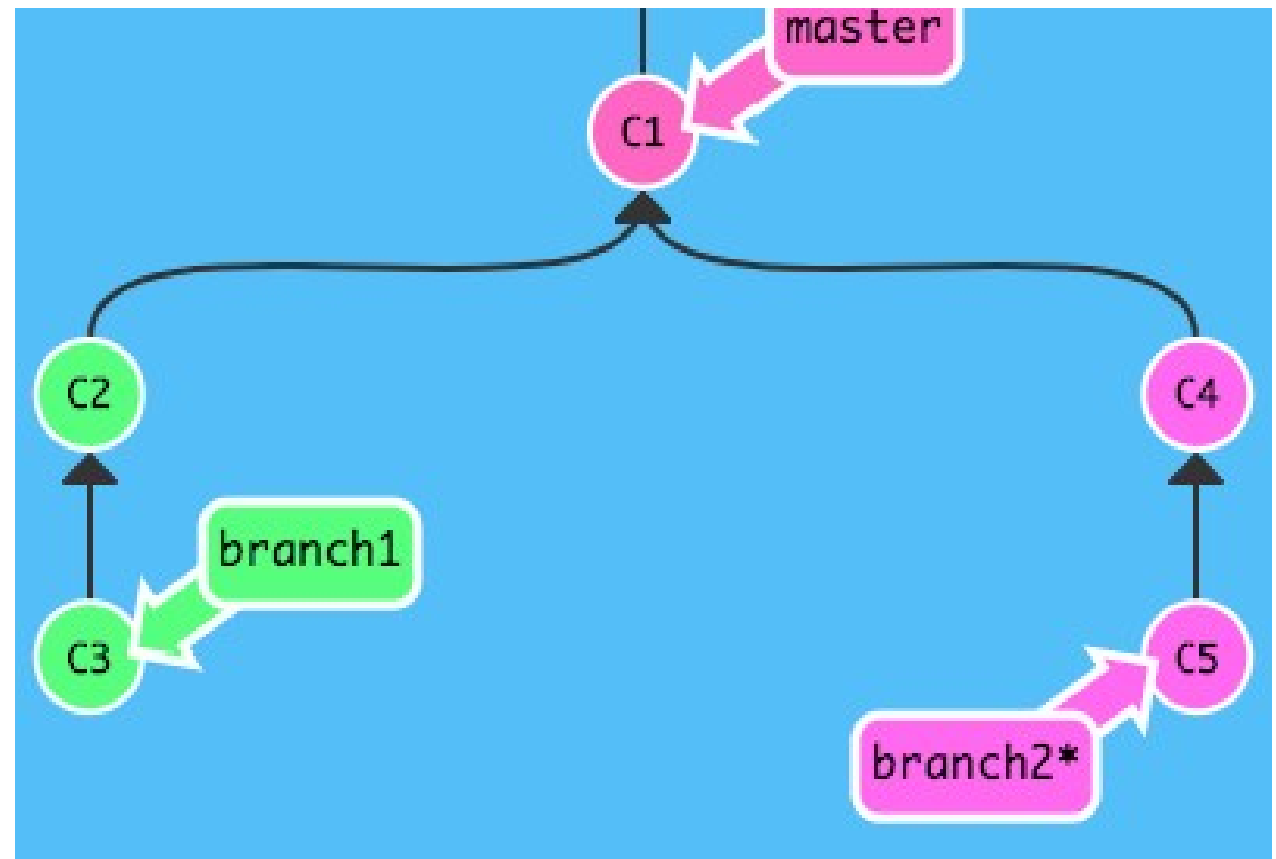
# Commit on new branch

```
git commit
```



# Another commit on new branch

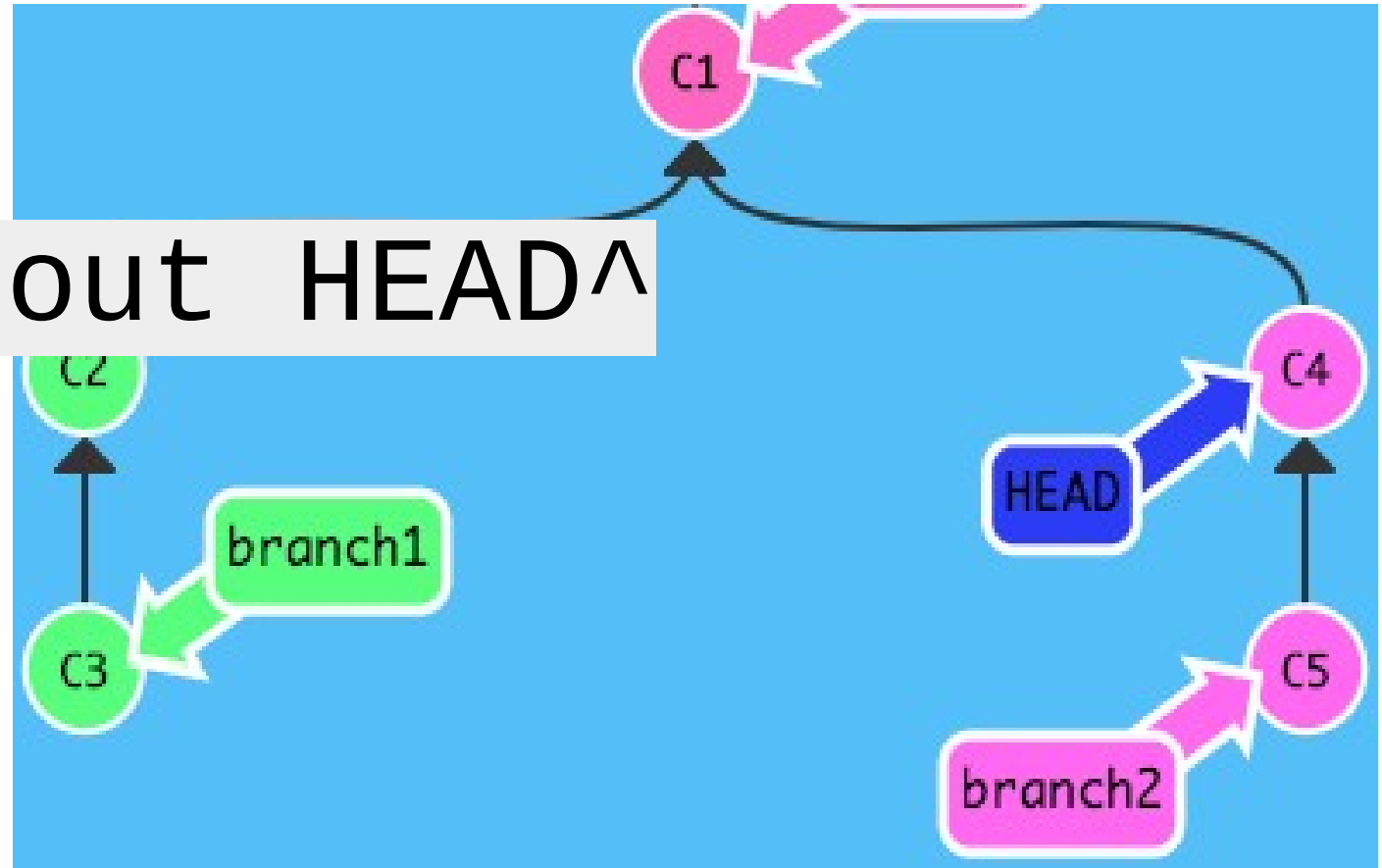
```
git commit
```



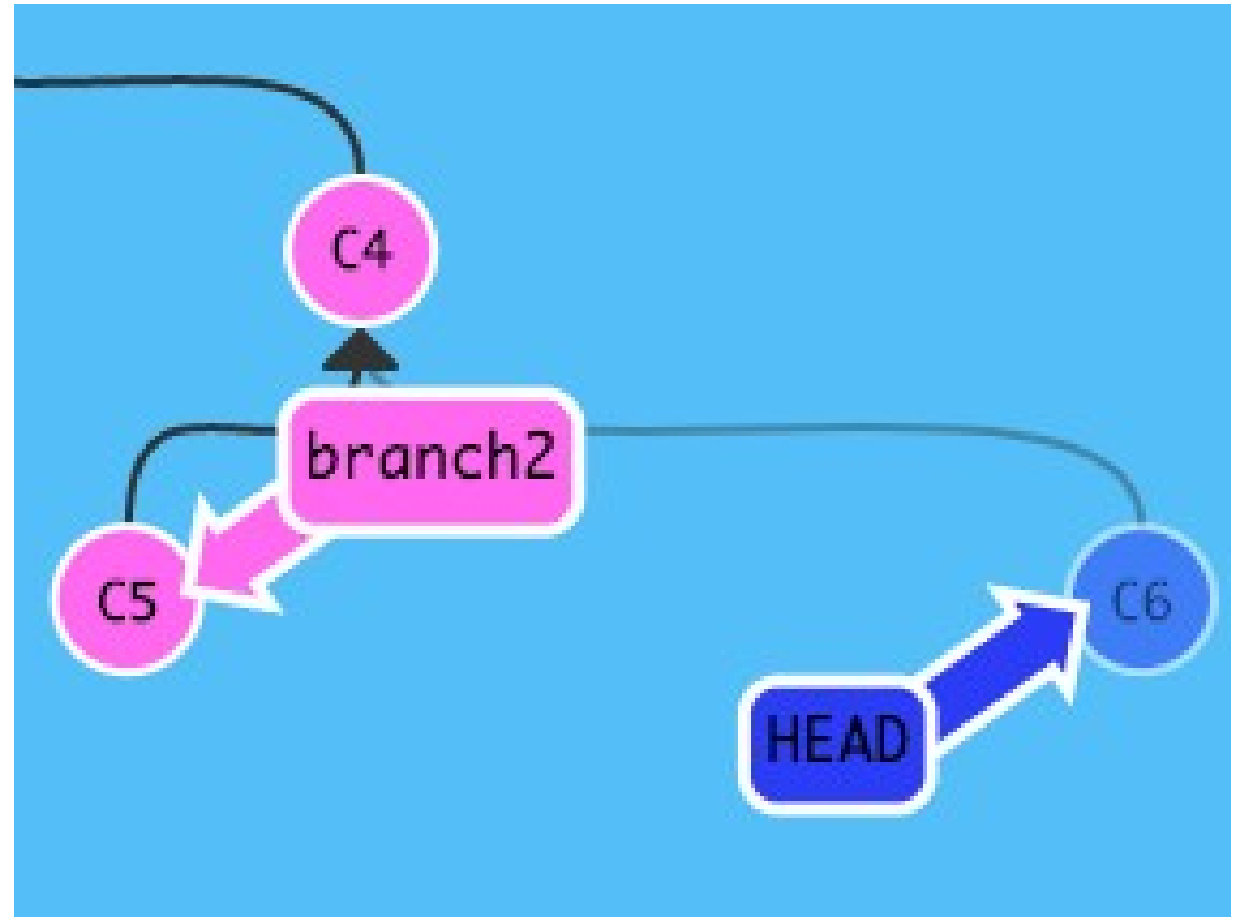
# Step back

`git checkout HEAD^`

detached  
HEAD

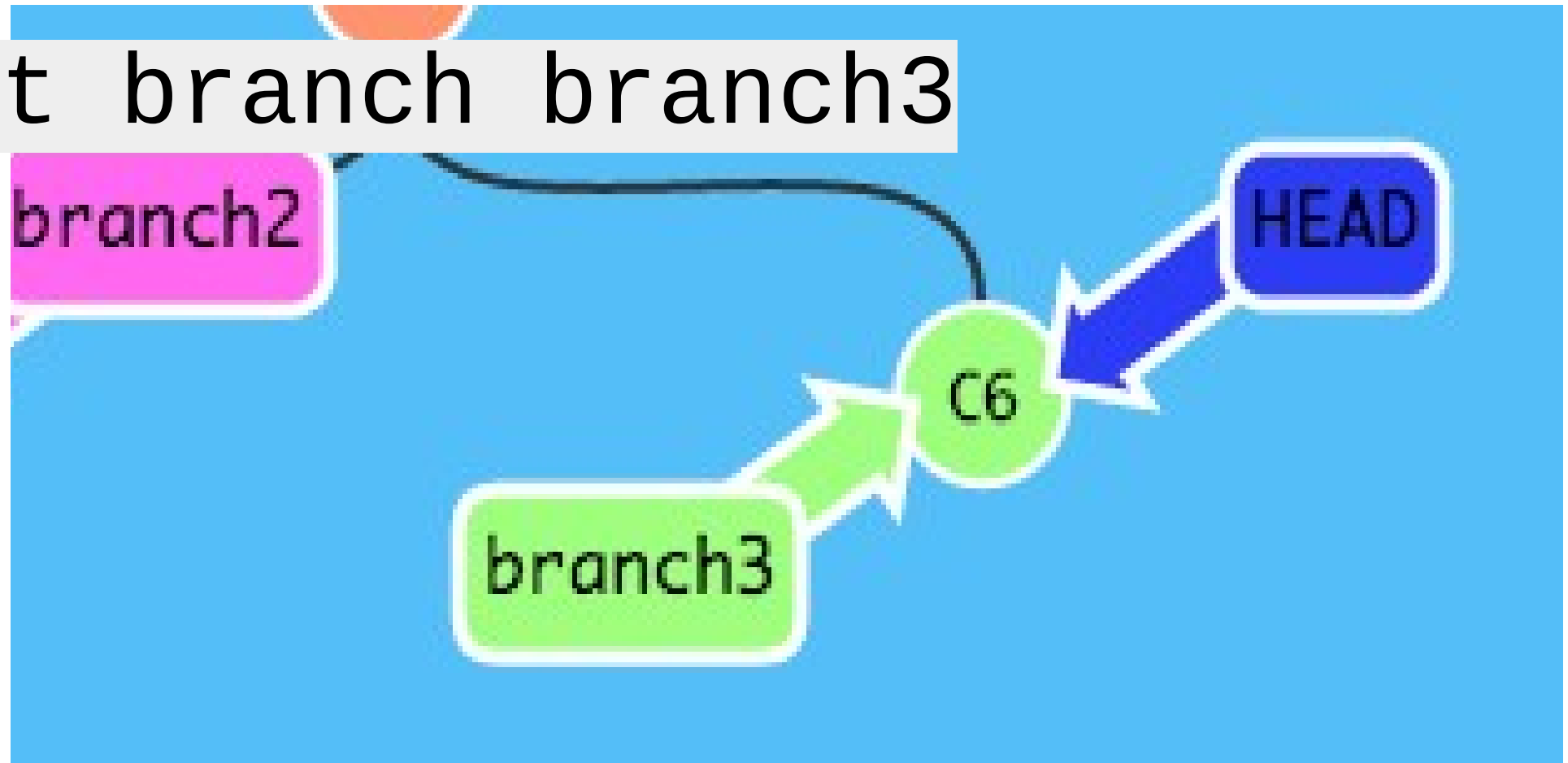


# Commit on detached HEAD



# Give name to child

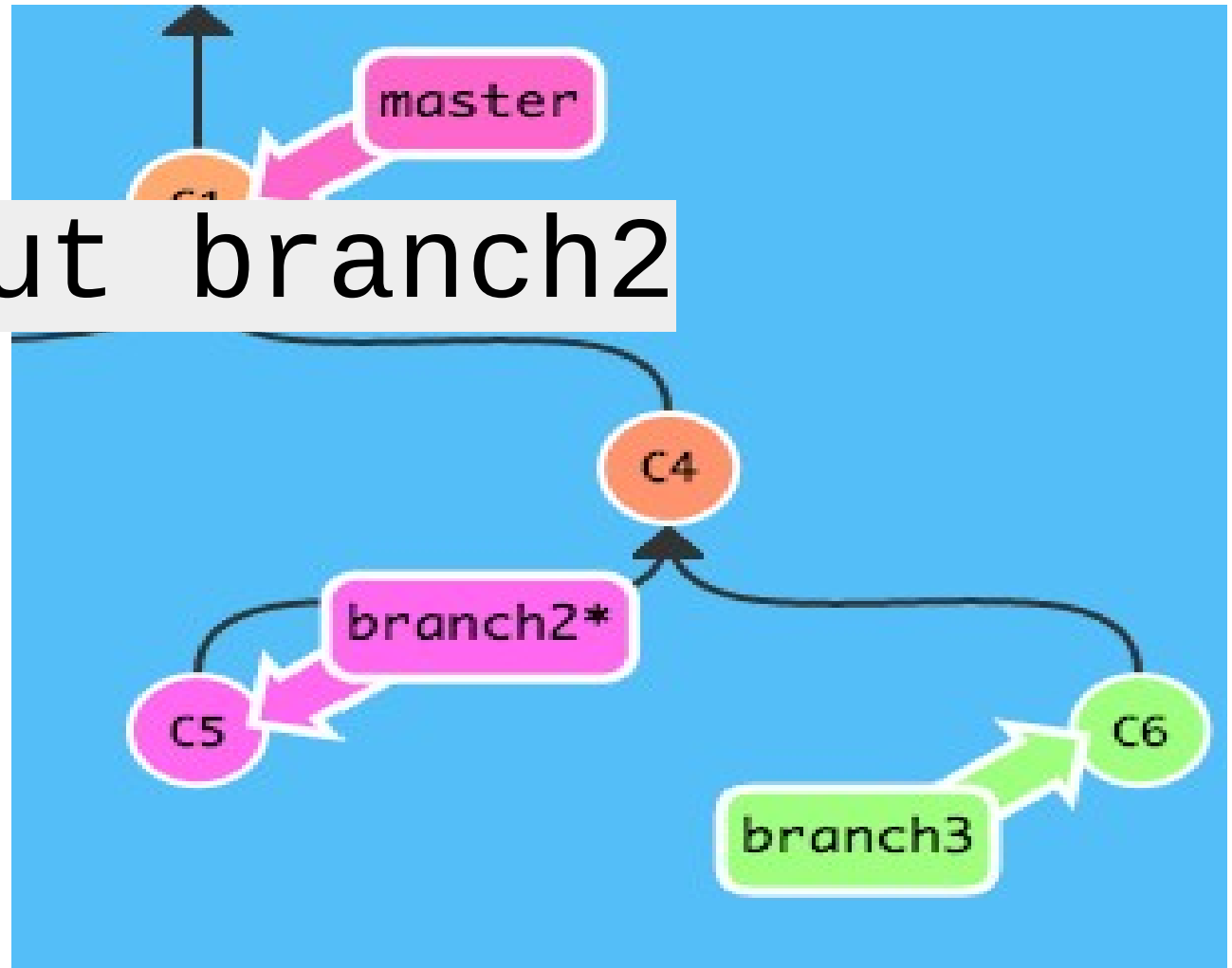
```
git branch branch3
```





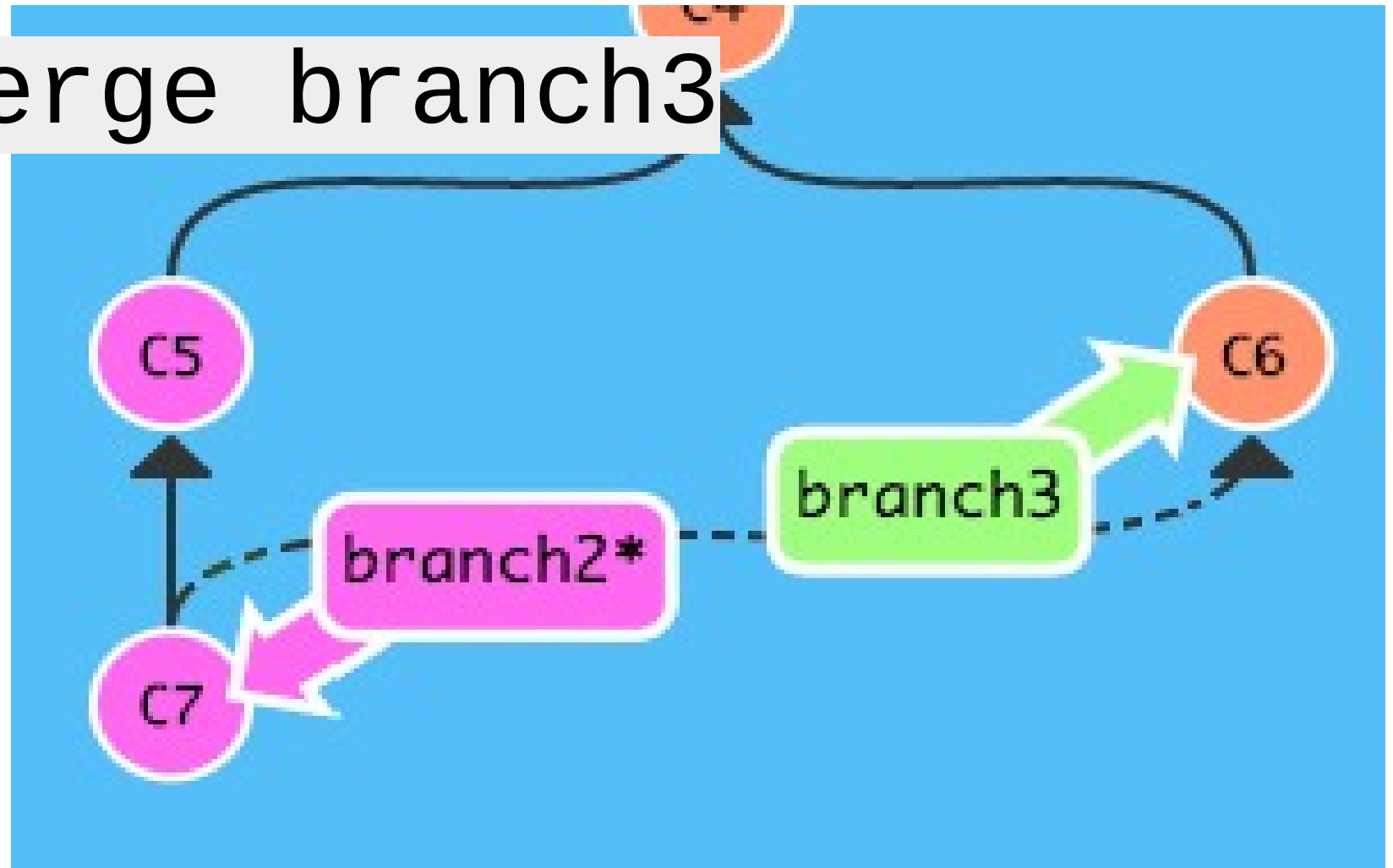
# Merge 1/2.

```
git checkout branch2
```



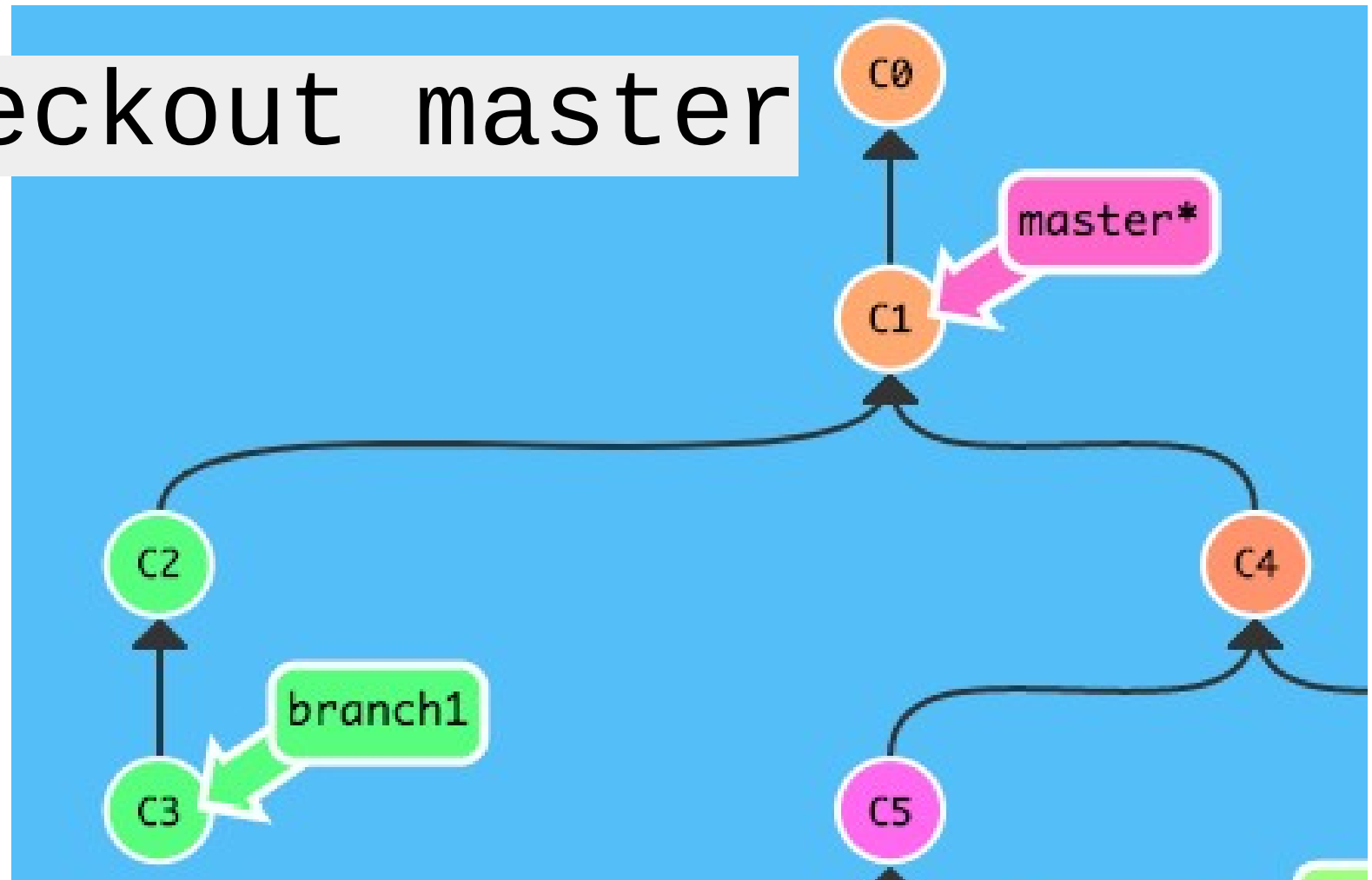
# Merge 2/2.

```
git merge branch3
```



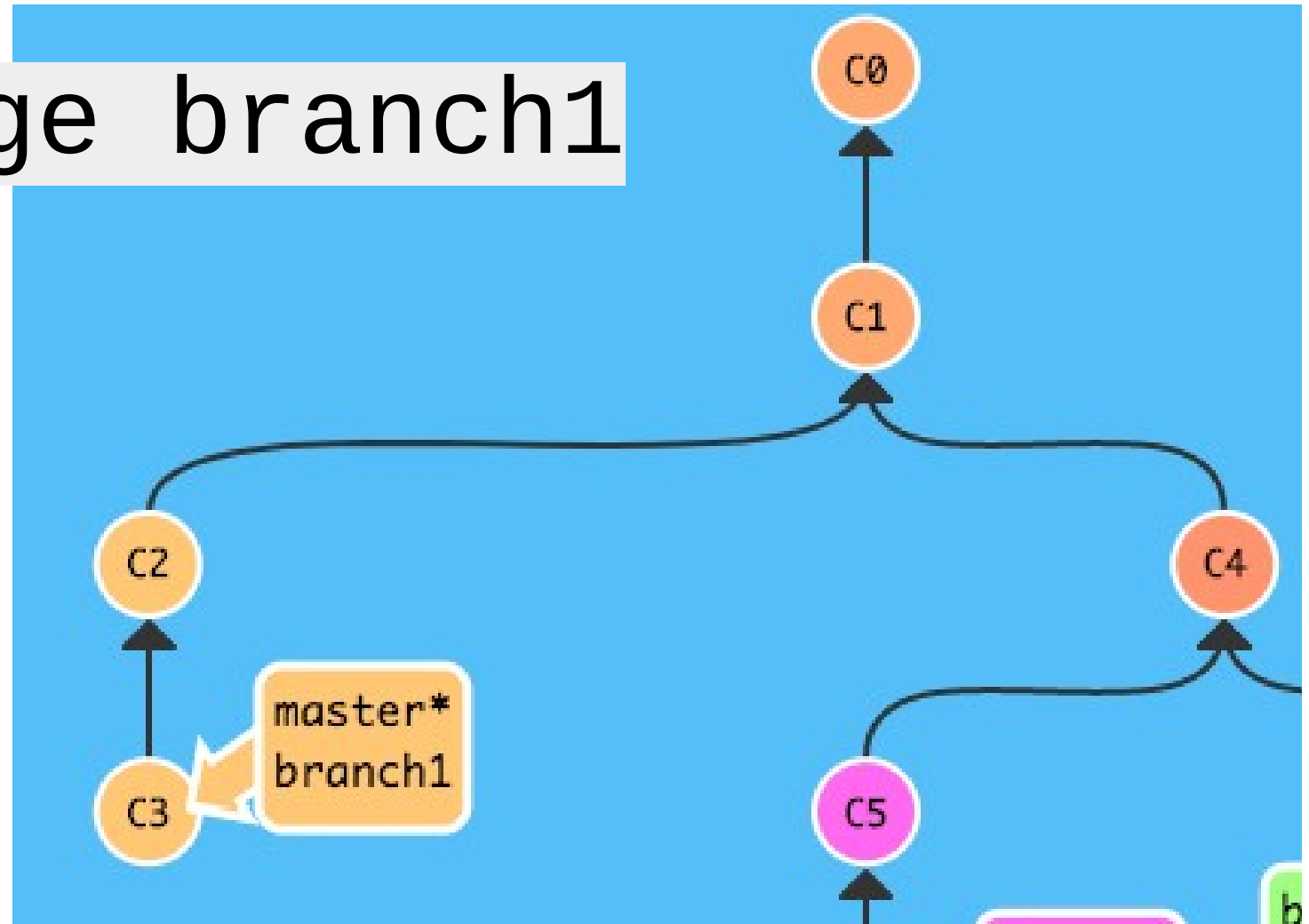
# Merge fwd 1/2.

```
git checkout master
```

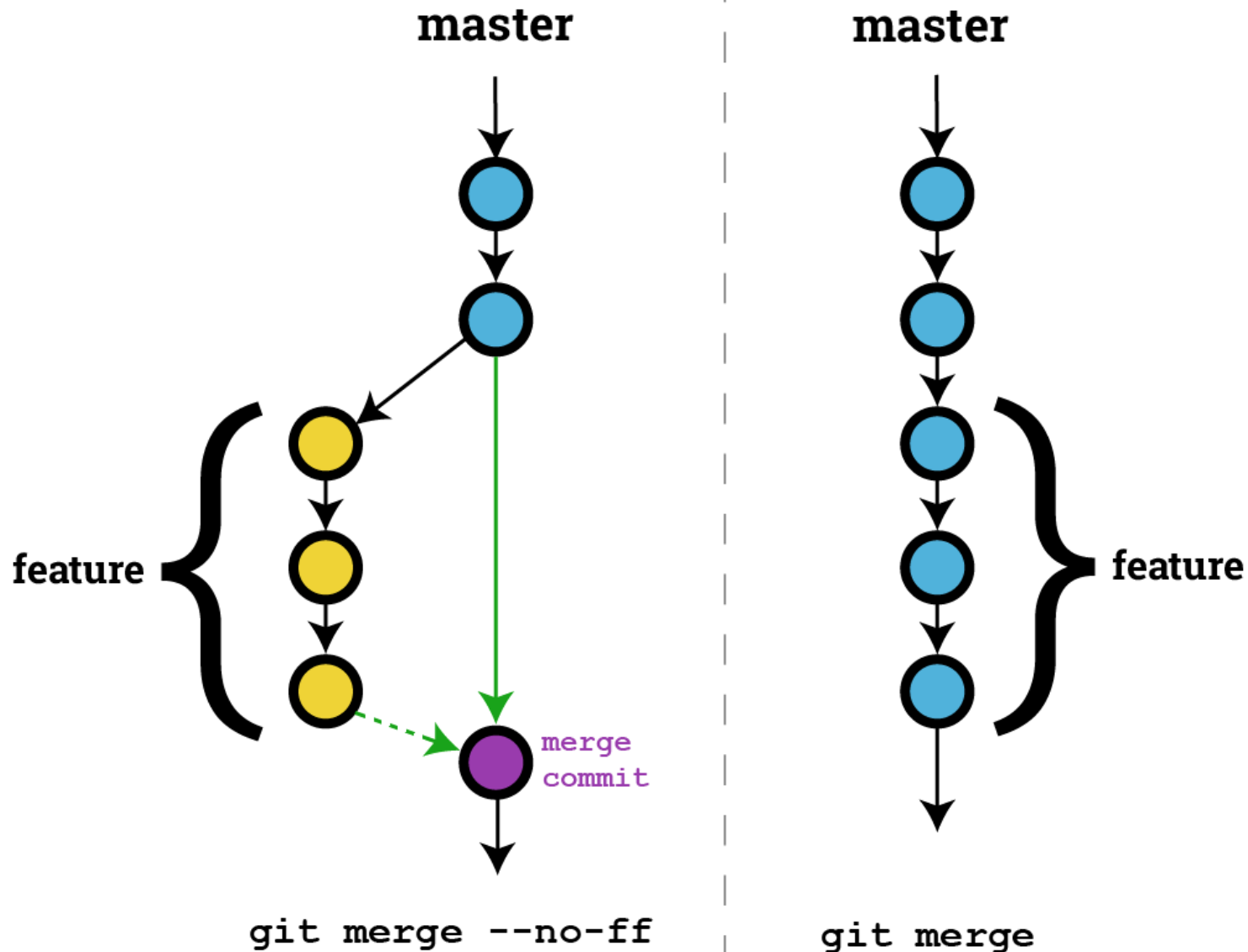


# Merge fwd 2/2.

```
git merge branch1
```



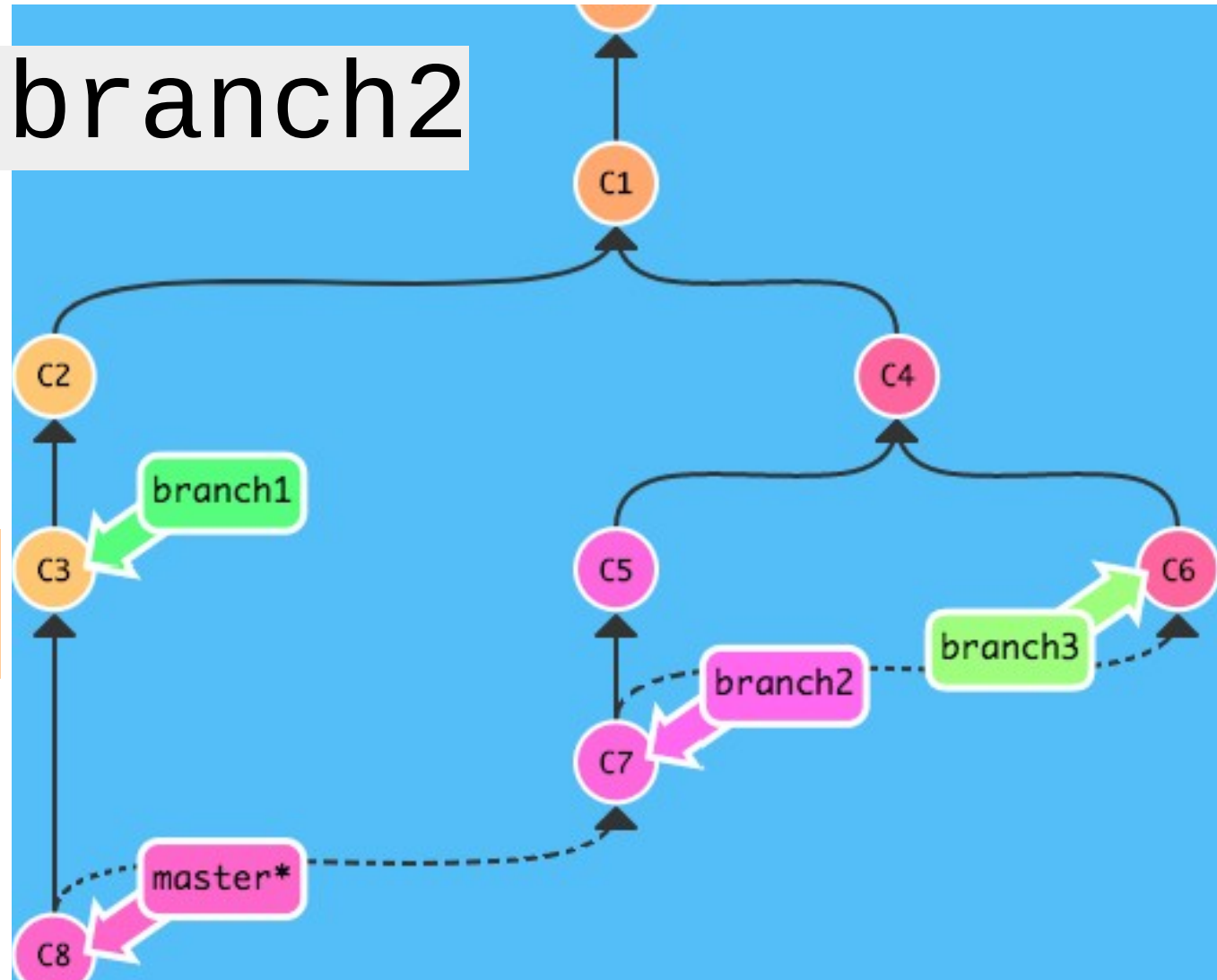
# Merge fast forward



# Basic branching strategy

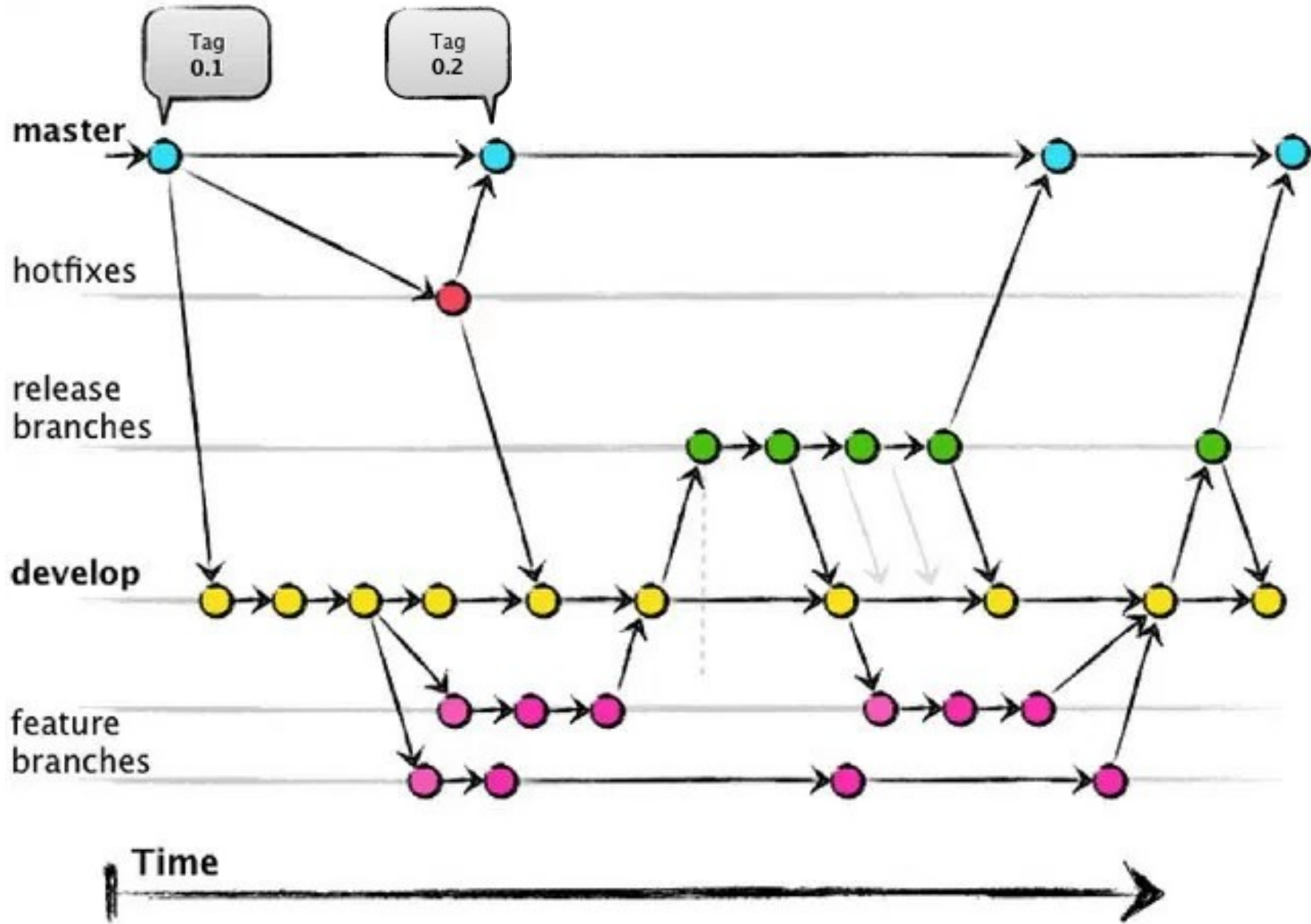
```
git merge branch2
```

- merge to master
- pls delete merged branches



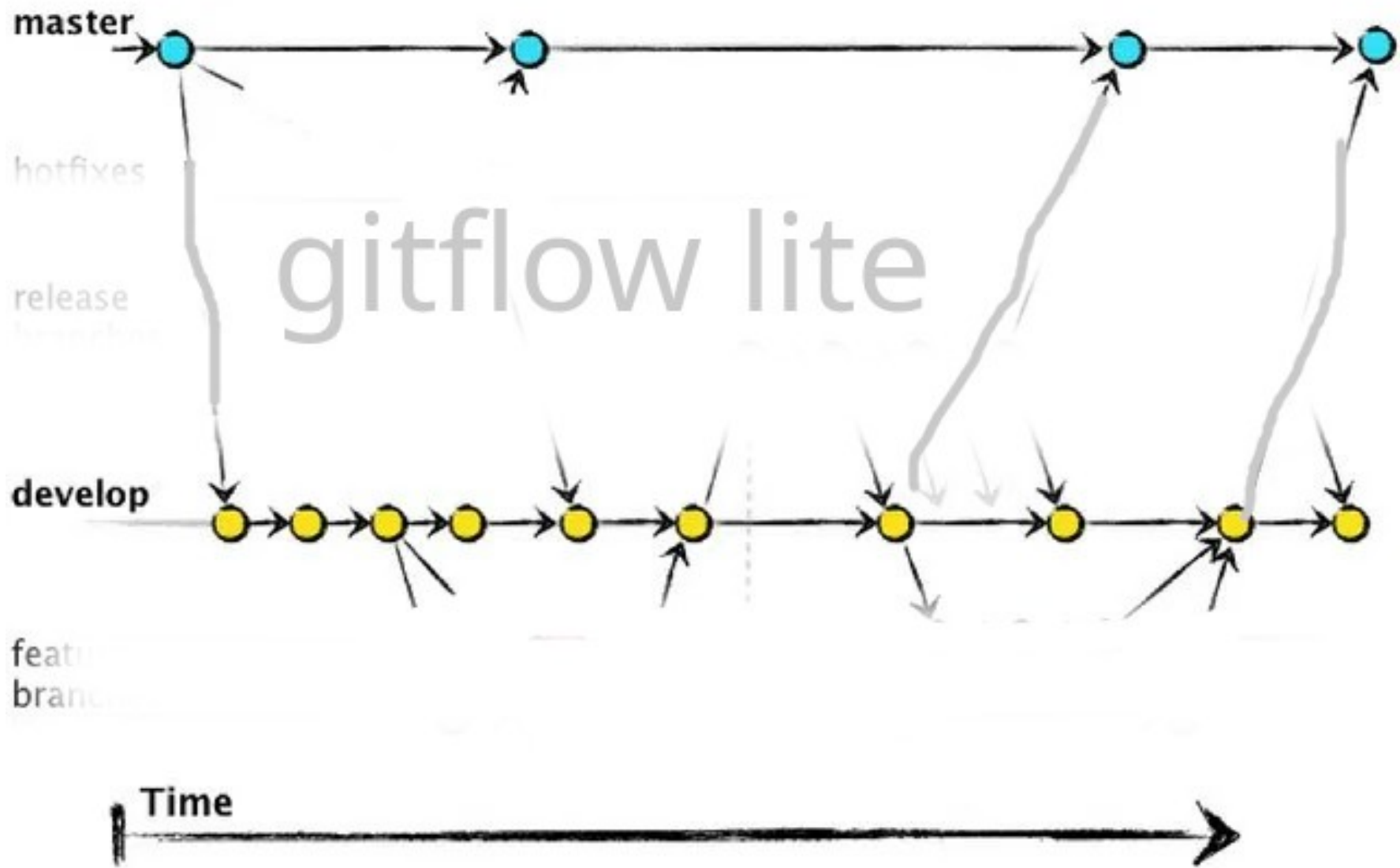
*Git is not a VCS. That's why Git is a great VCS*

# Gitflow



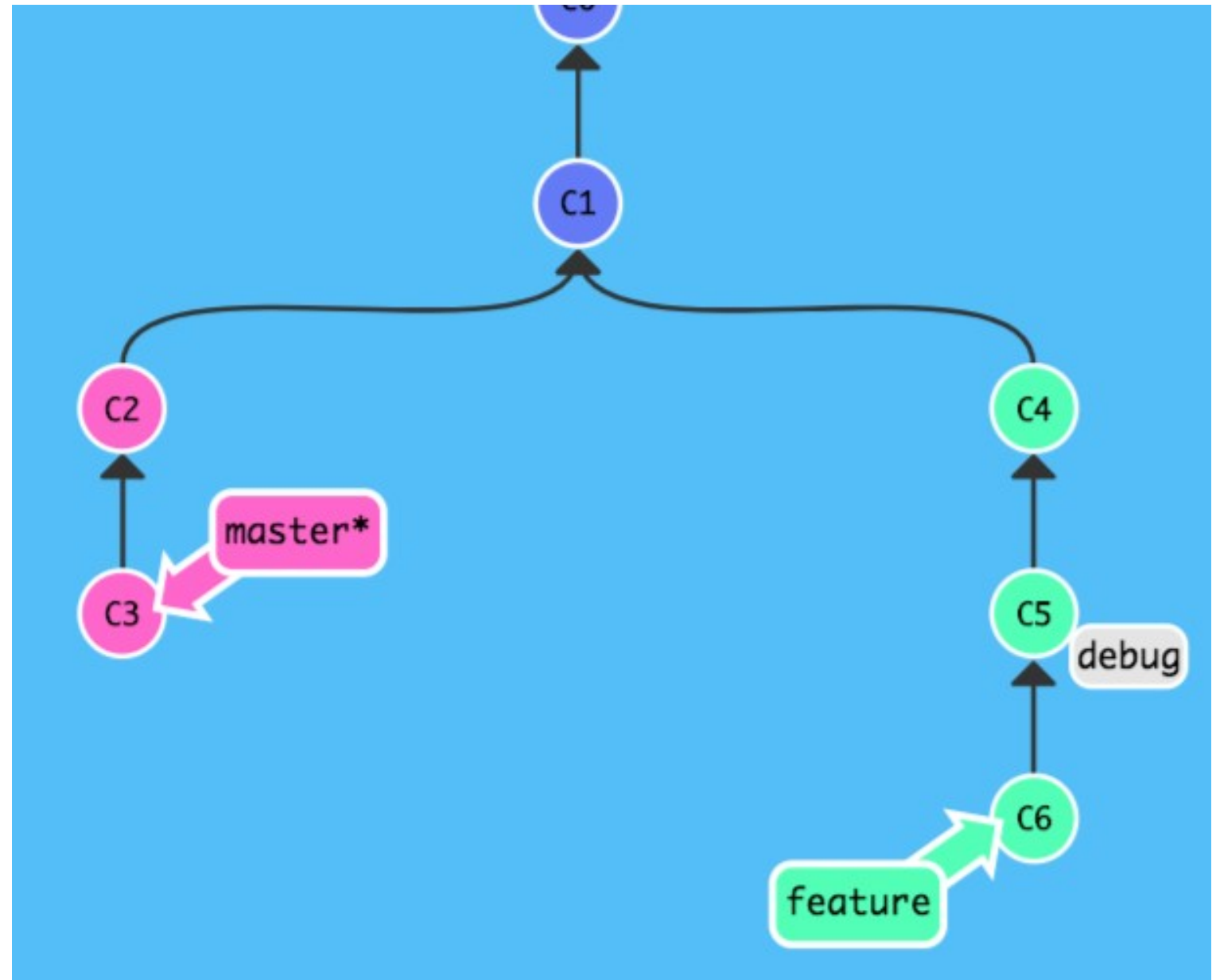
*GIT is not a VCS. That's why GIT is a great VCS*

# Gitflow



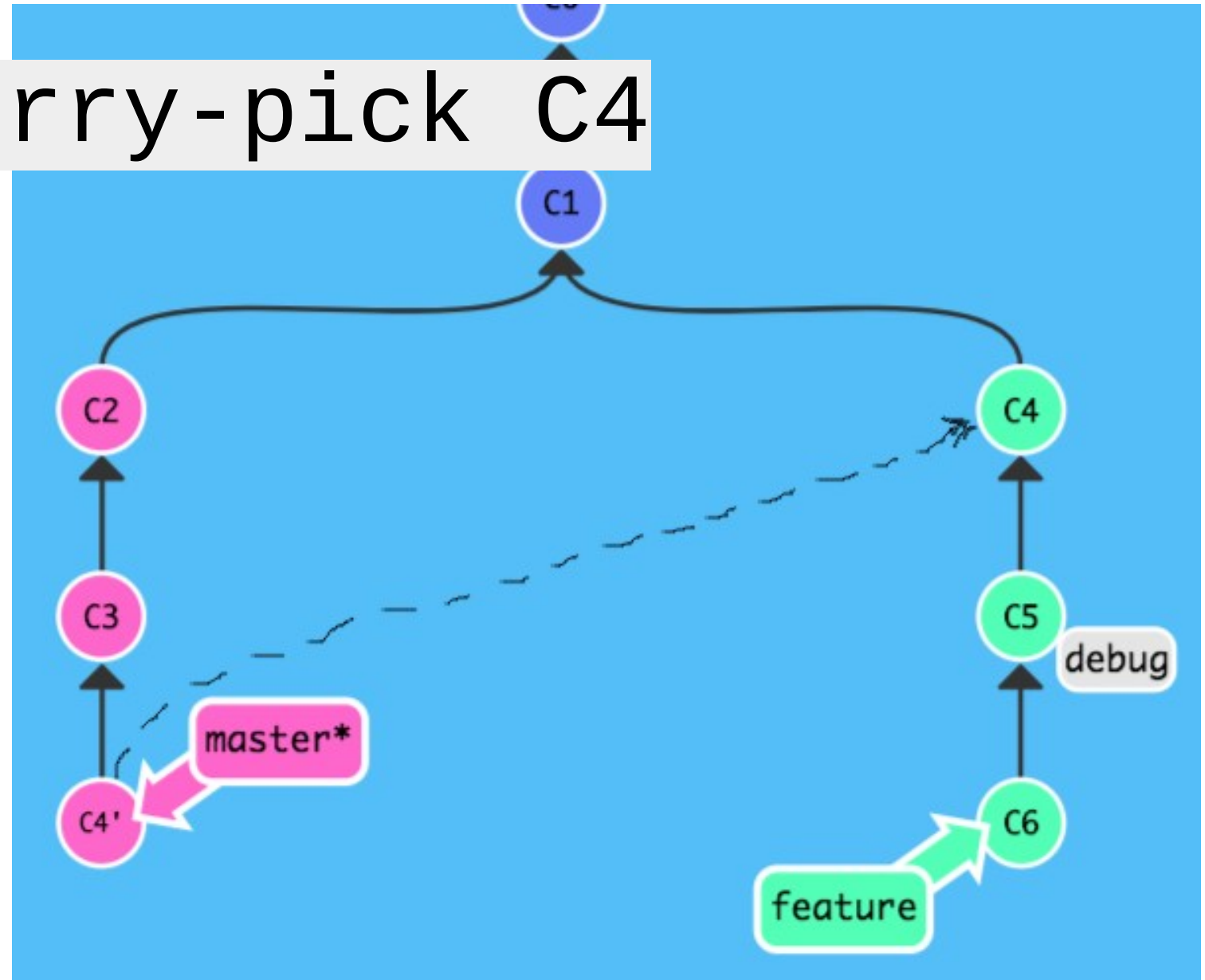


# Cherry-pick 1/3.



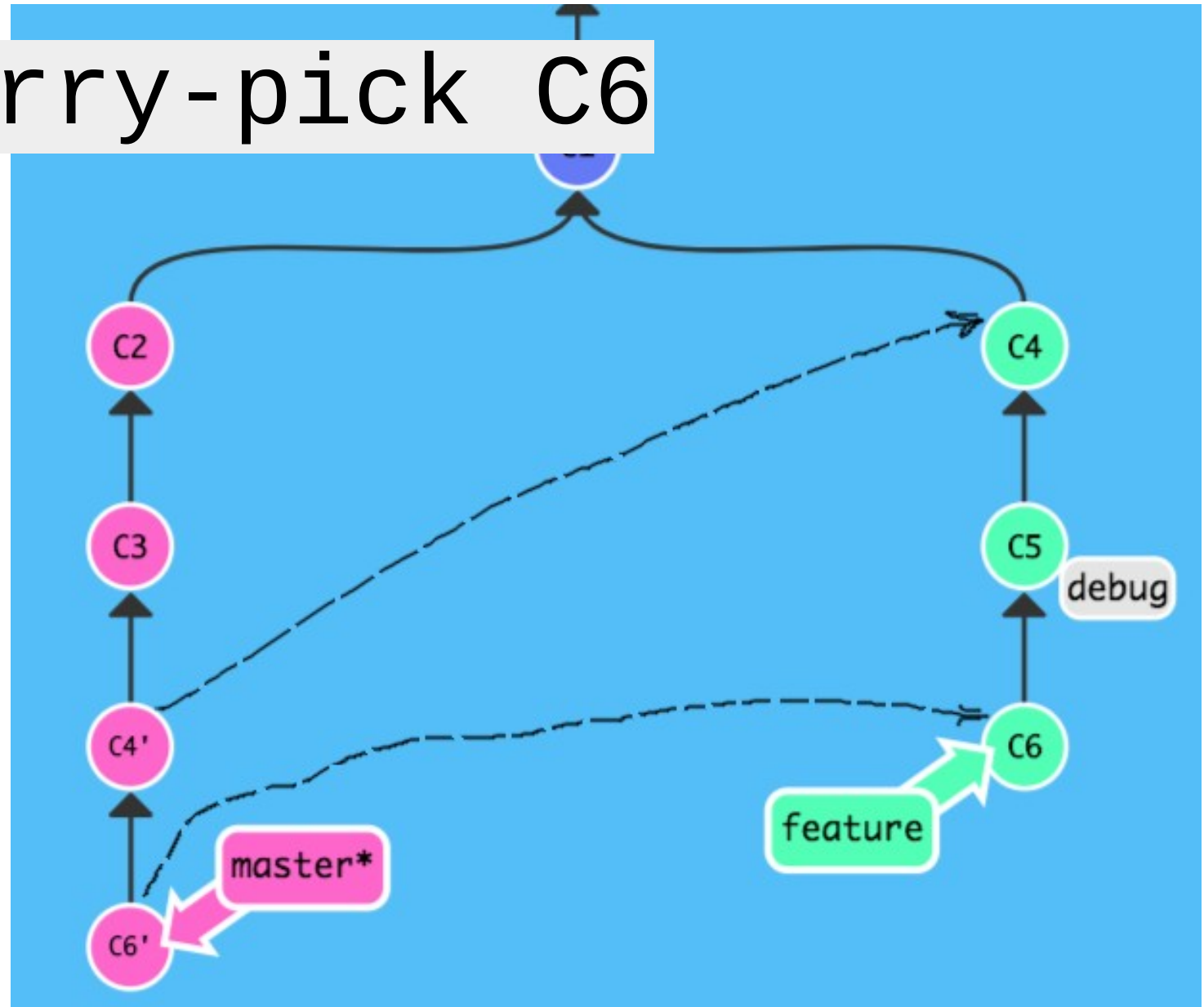
# Cherry-pick 2/3.

```
git cherry-pick C4
```

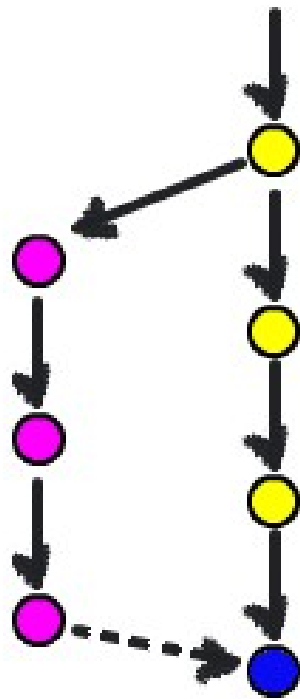


# Cherry-pick 3/3.

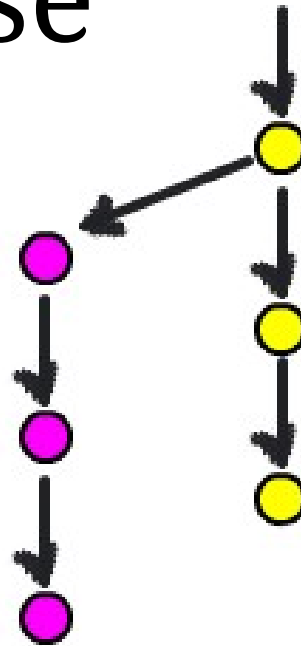
```
git cherry-pick C6
```



# Merge vs Rebase

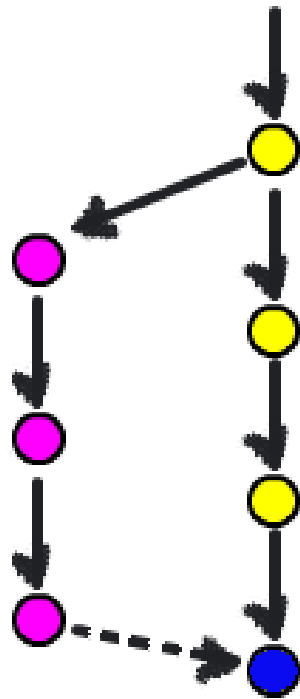


merge

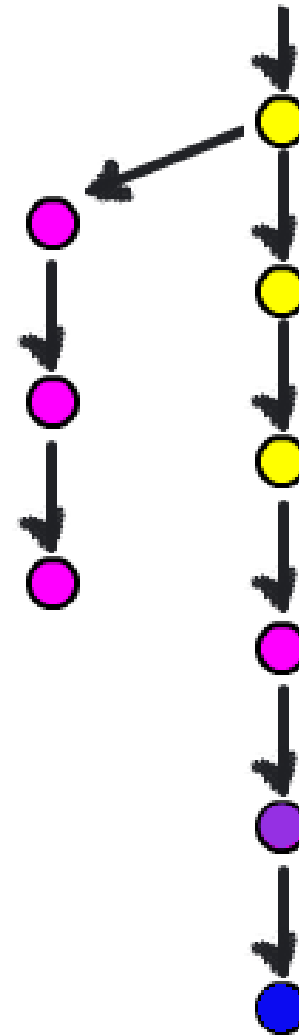
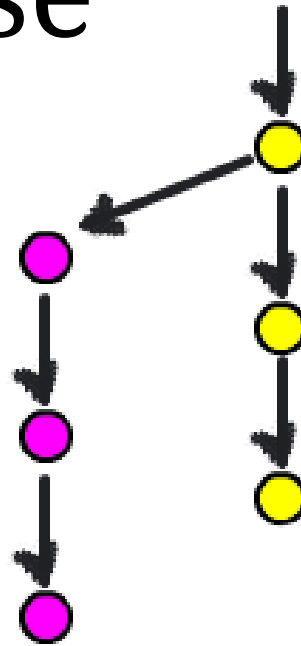


rebase

# Merge vs Rebase



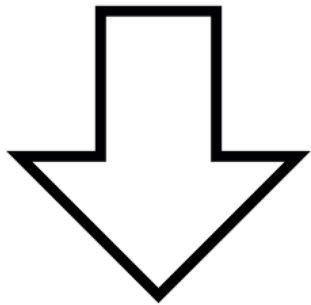
merge



rebase

*GIT is not a VCS. That's why GIT is a great VCS*

# For more information: see internet



# Survival pack

`git status` use often (it's free)

[learngitbranching.js.org](https://learngitbranching.js.org)

```
sudo apt install git \
  tig gitk gitg ... meld
```



**GitHub**



GitLab

# Start now

```
cd myprj ; git init
```





# Start now

```
cd myprj ; git init
```

```
git status
```



