GIT

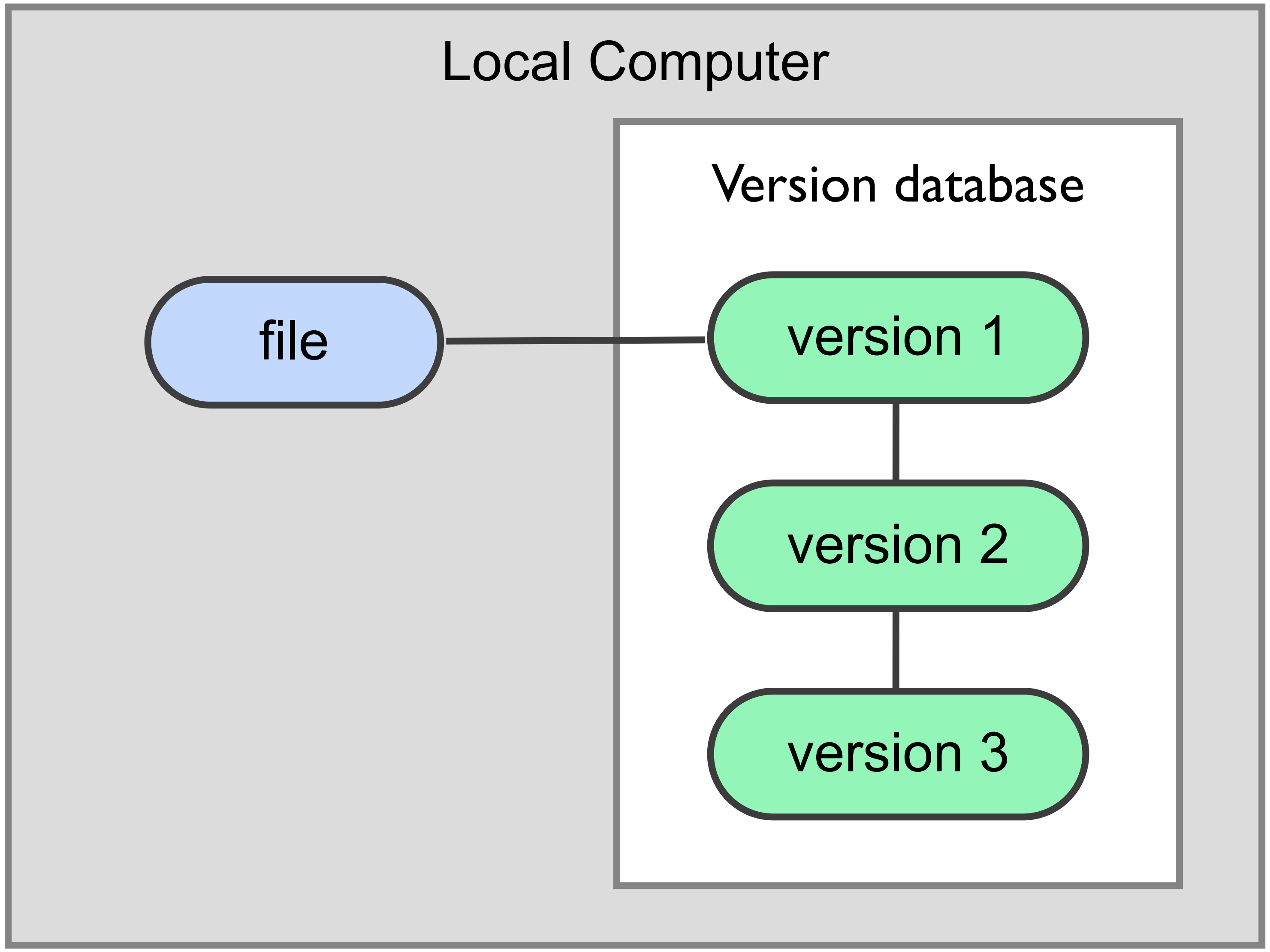
Merging into GRVM workflow

Intro

- GIT internals
- GIT basics
- GIT and GRVM

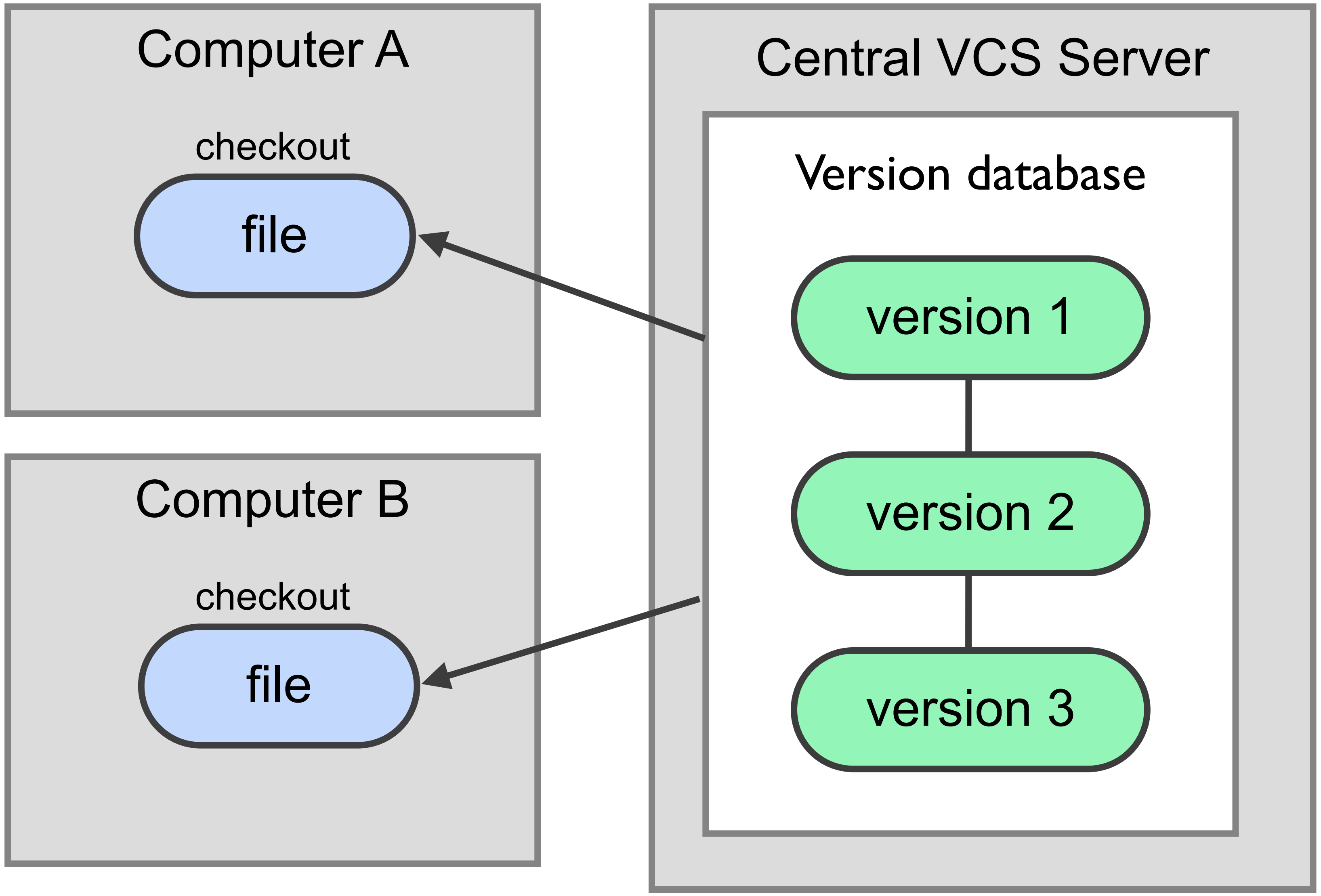
Source Control Taxonomy

- Local (rcs, time machine)



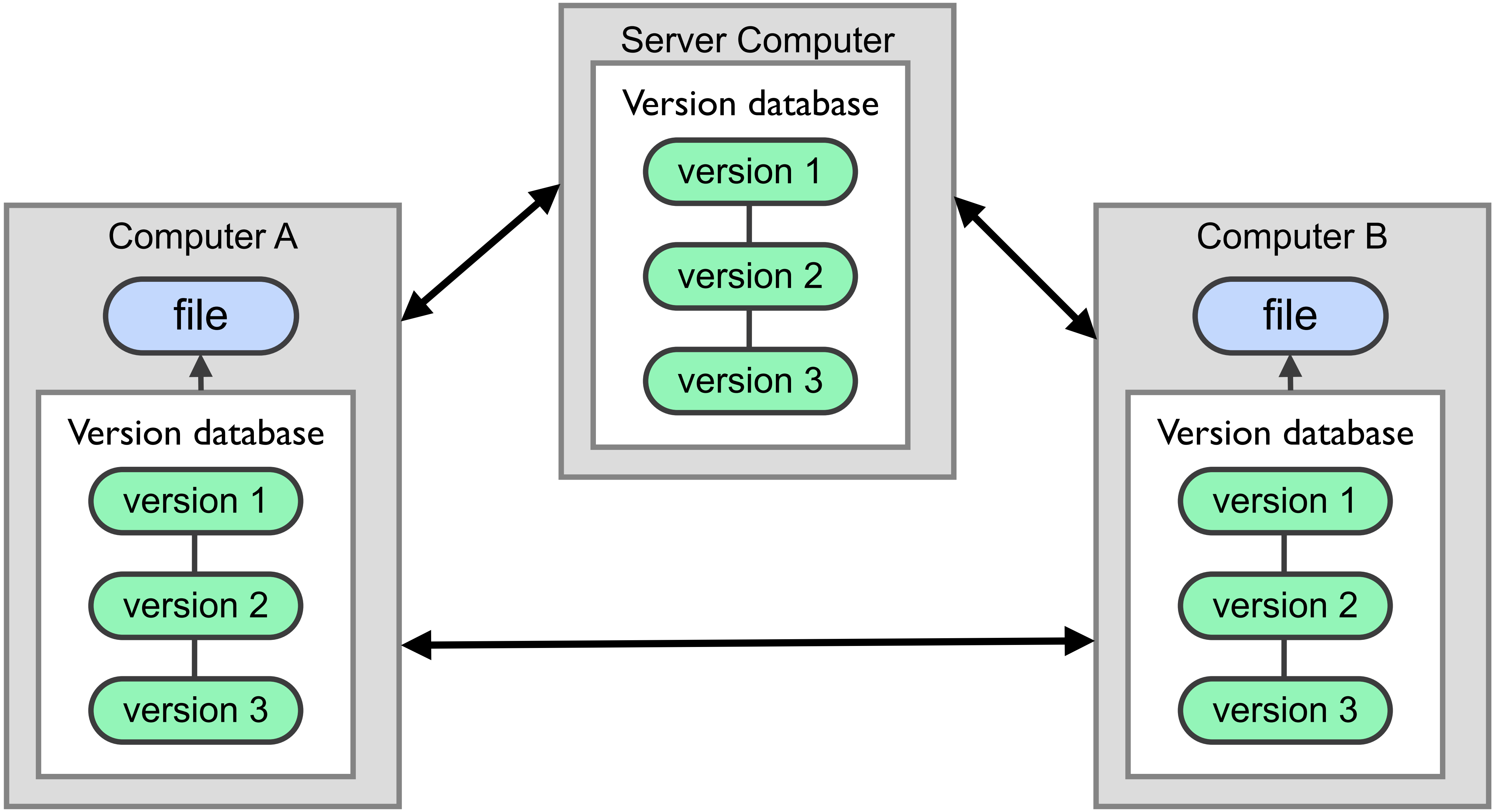
Source Control Taxonomy

- Centralized (CVS, Subversion, Perforce)



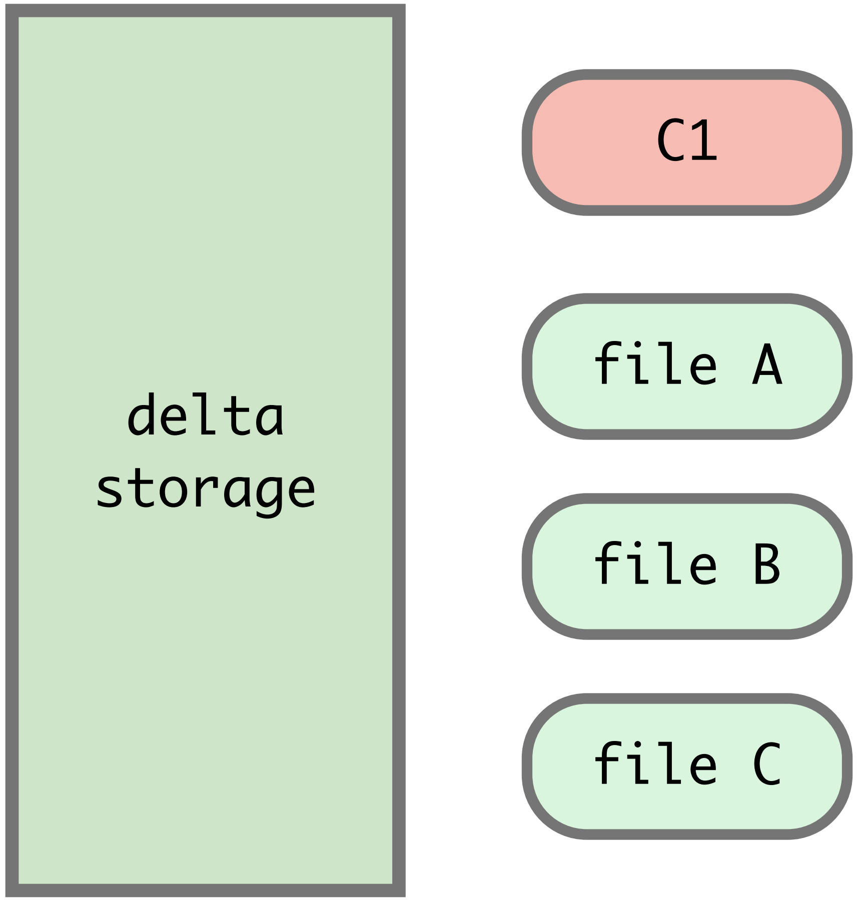
Source Control Taxonomy

- Distributed (Git, Mercurial, Bazaar or Darcs)



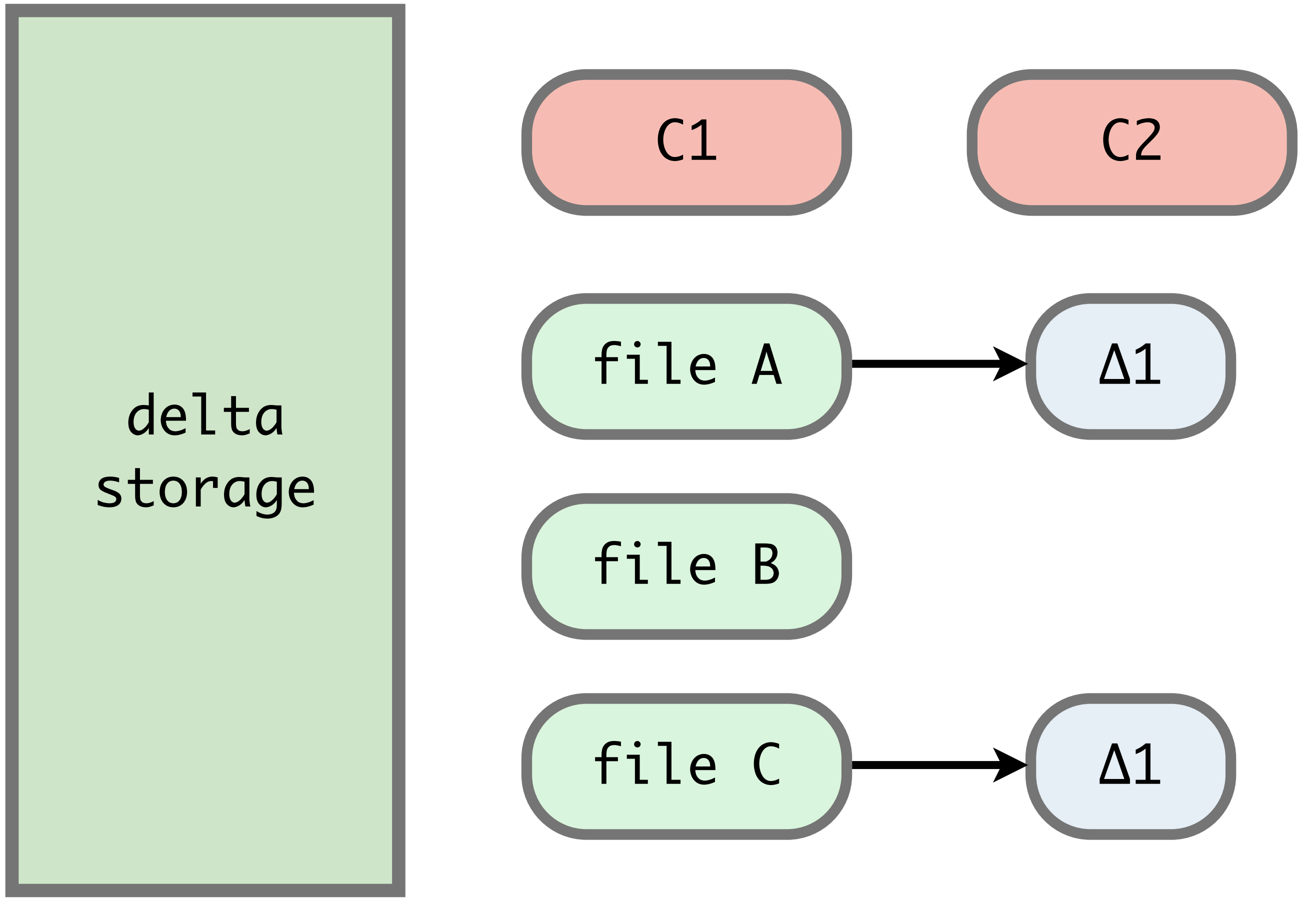
Source Control Taxonomy

- Delta based



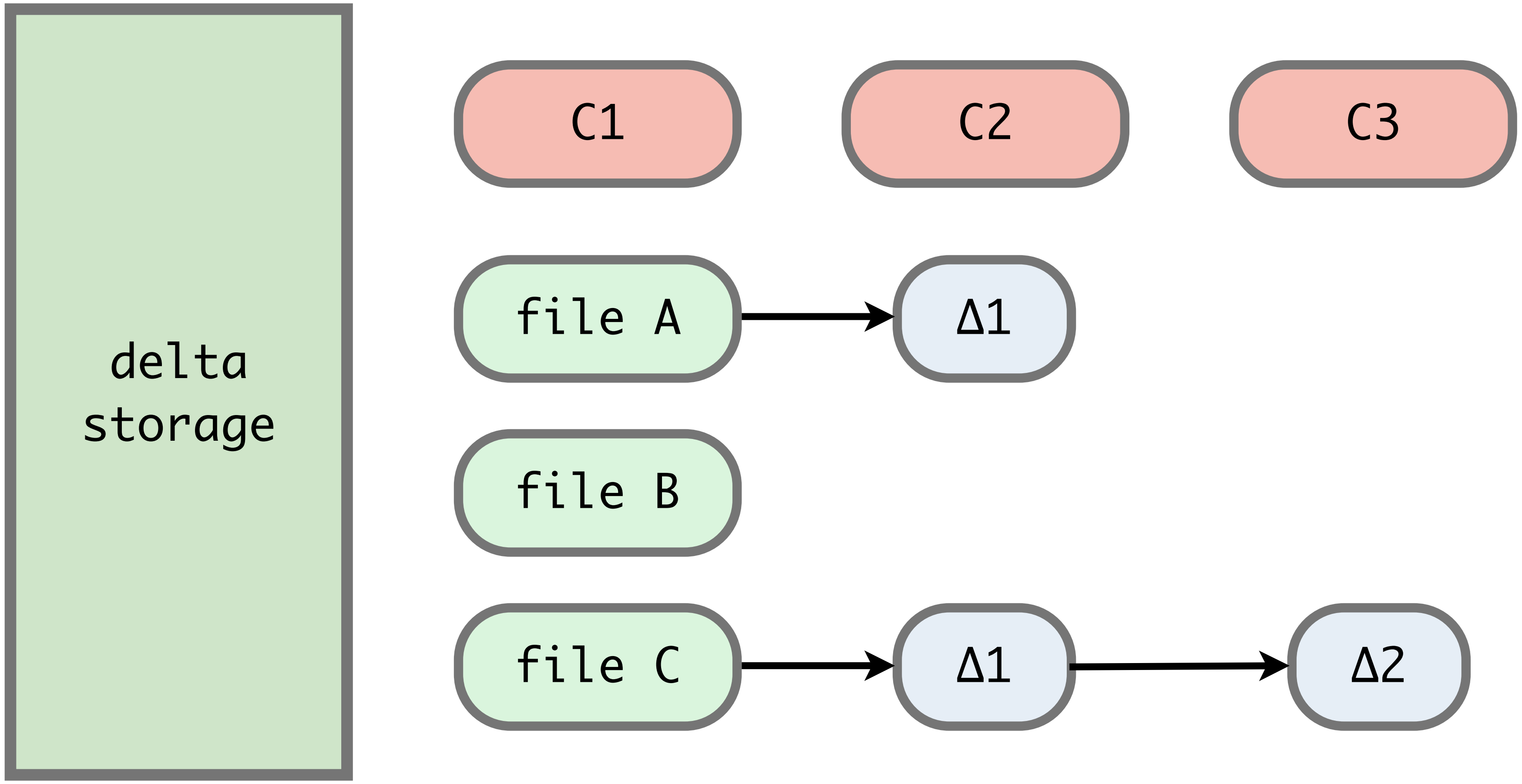
Source Control Taxonomy

- Delta based



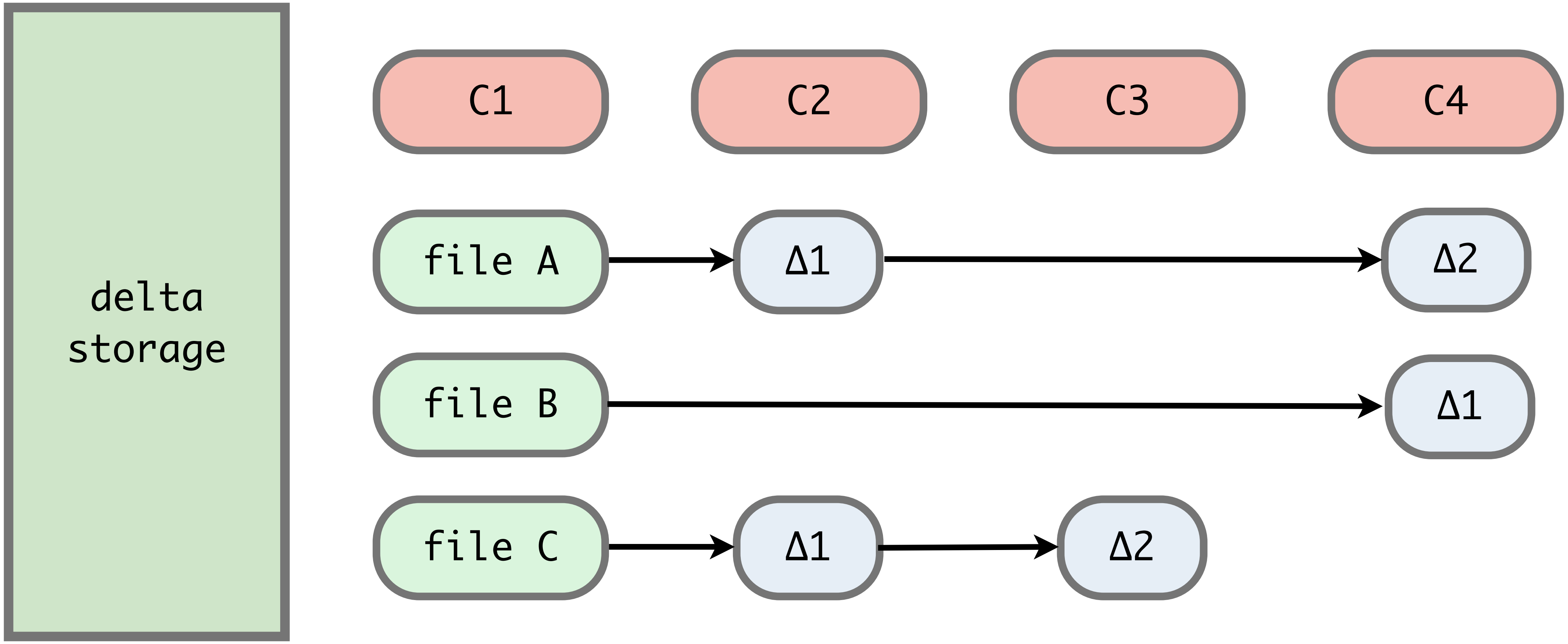
Source Control Taxonomy

- Delta based



Source Control Taxonomy

- Delta based



delta storage

C1

C2

C3

C4

file A

Δ1

Δ2

file B

Δ1

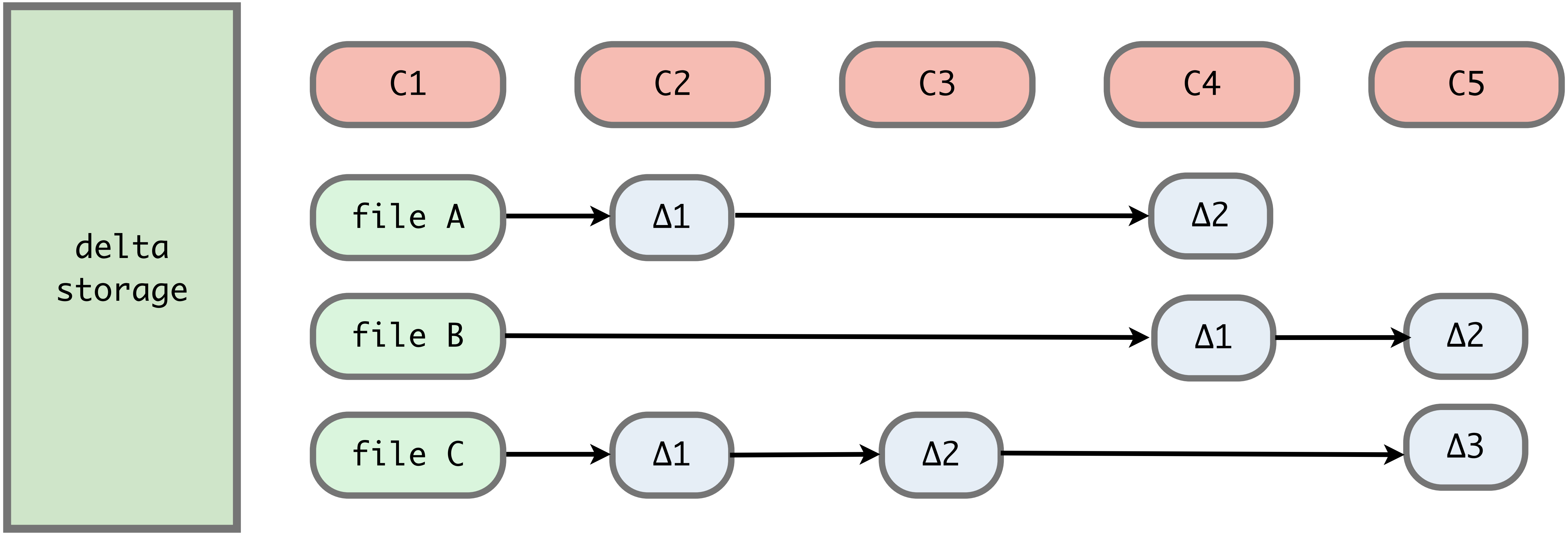
file C

Δ1

Δ2

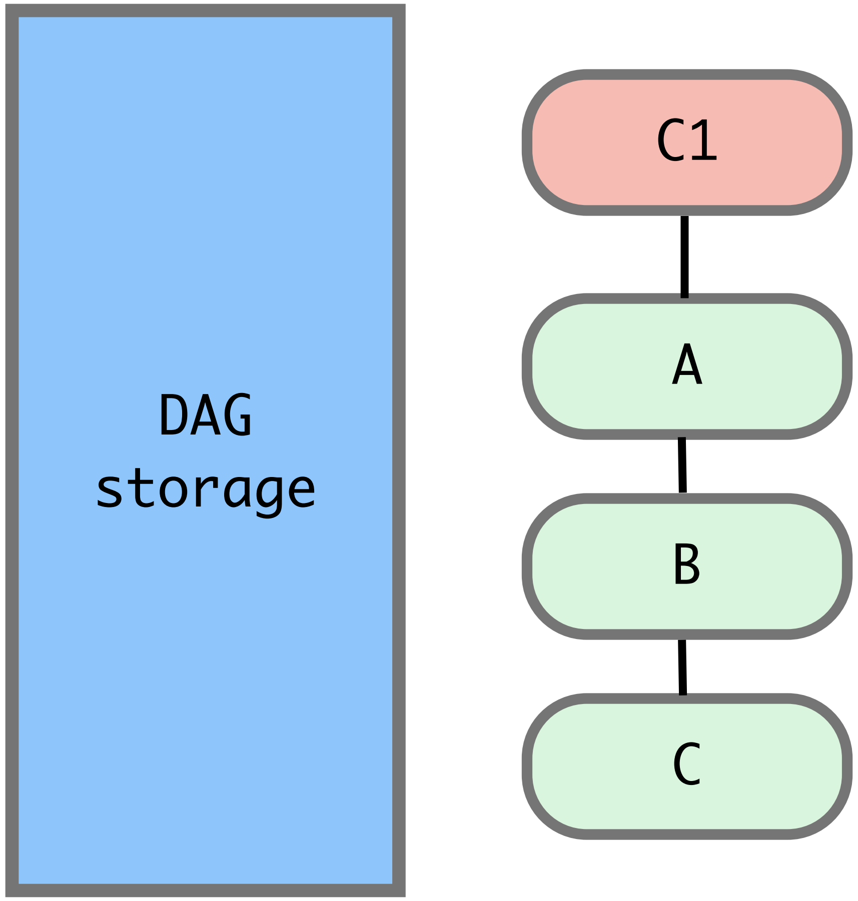
Source Control Taxonomy

- Delta based

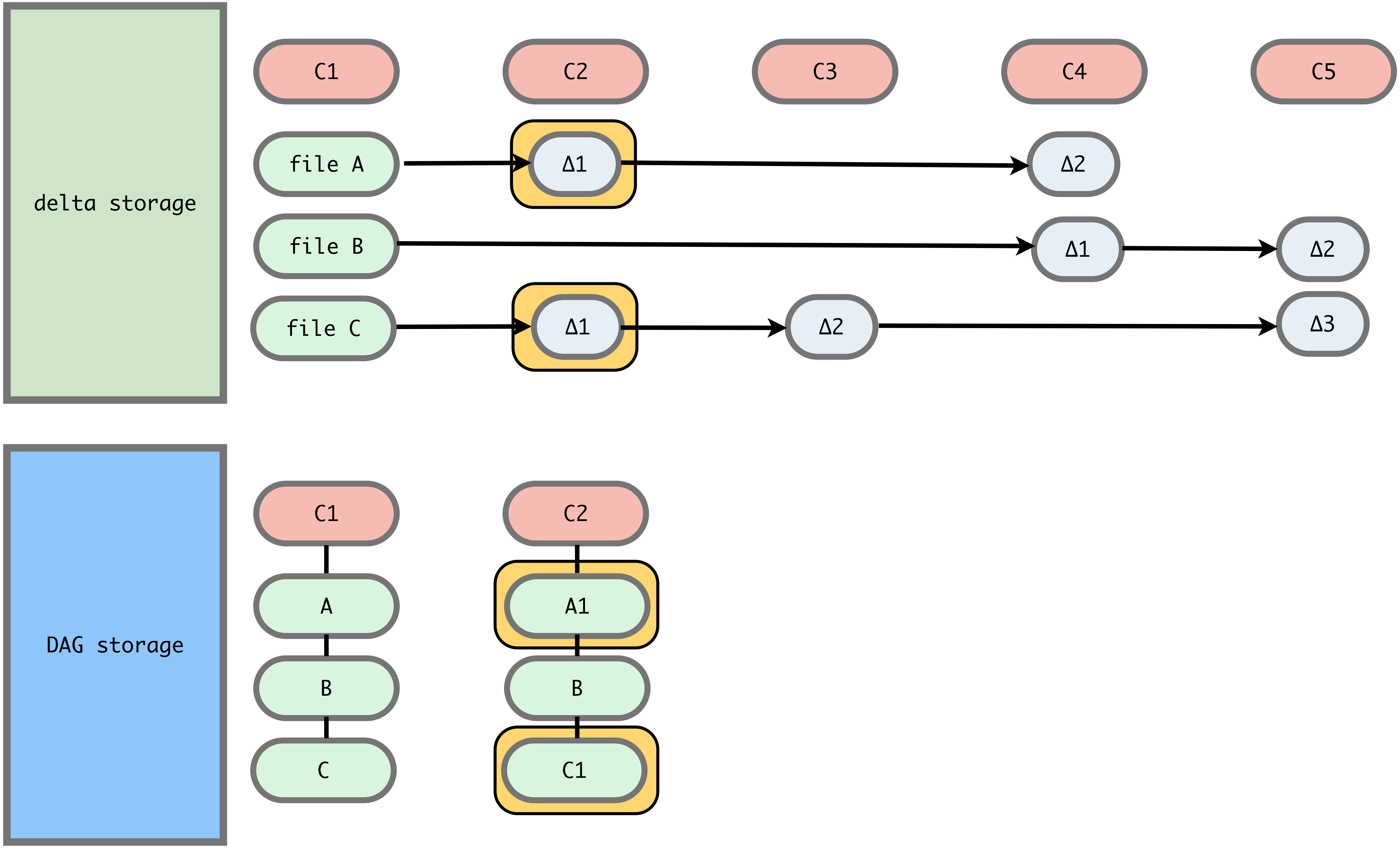


Source Control Taxonomy

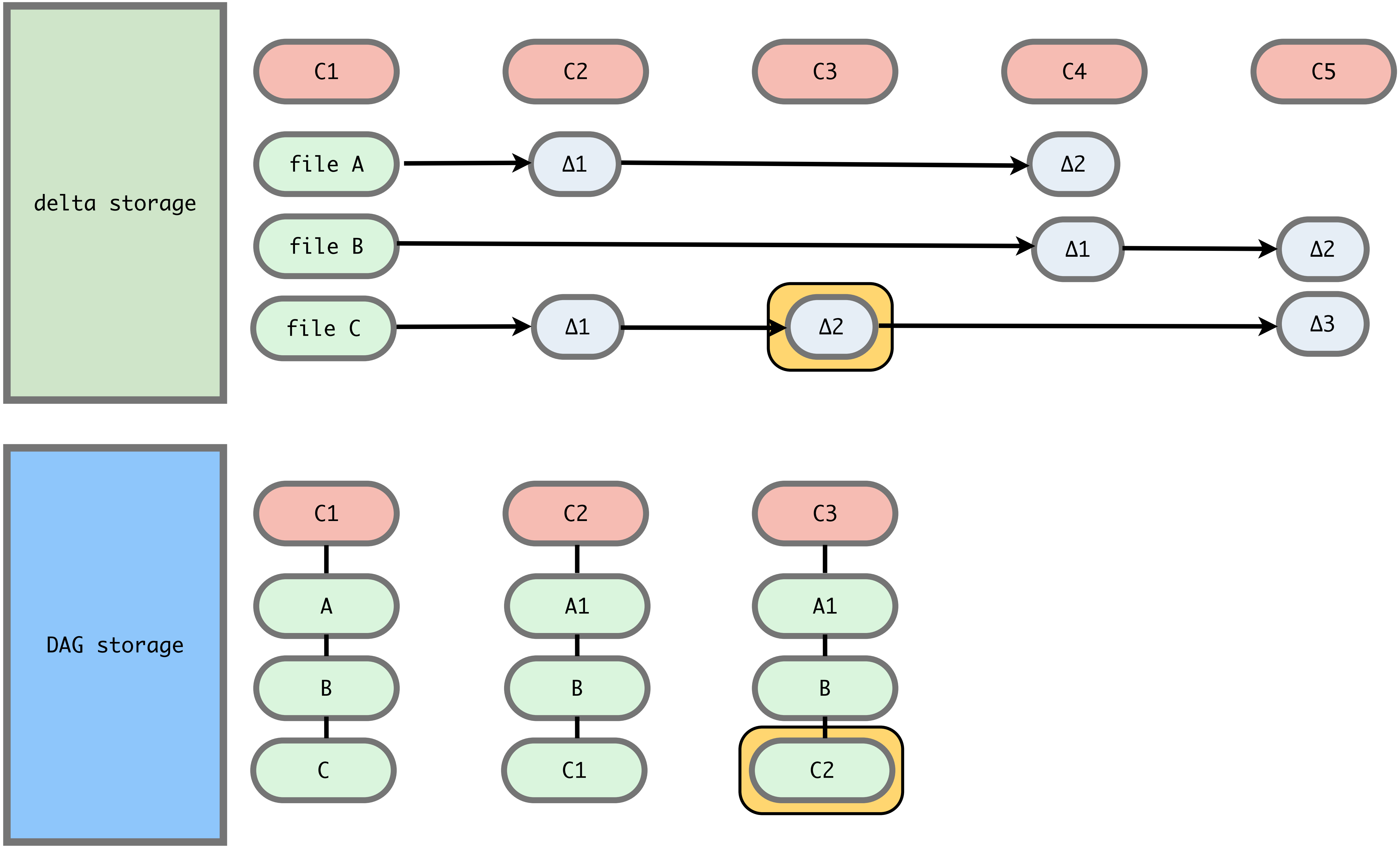
- Directed Acyclic Graph (DAG)



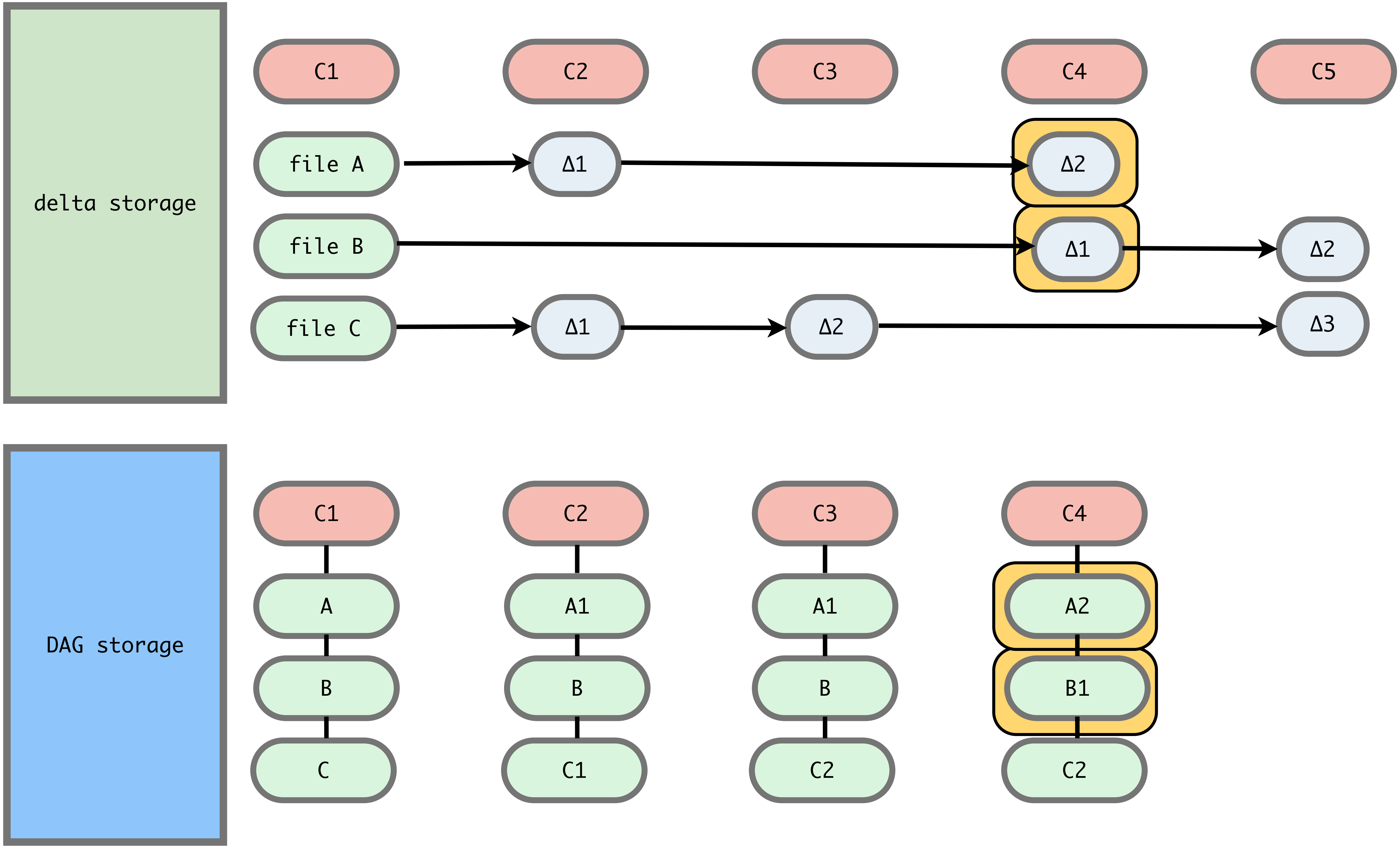
Delta vs DAG


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
DAG storage
C1
A
B
C
C2
A1
B
C1

Delta vs DAG


delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
DAG storage
C1
A
B
C
C2
A1
B
C1
C3
A1
B
C2

Delta vs DAG



delta storage

C1 C2 C3 C4 C5

file A → Δ1 → Δ2

file B → Δ1 → Δ2

file C → Δ1 → Δ2 → Δ3

DAG storage

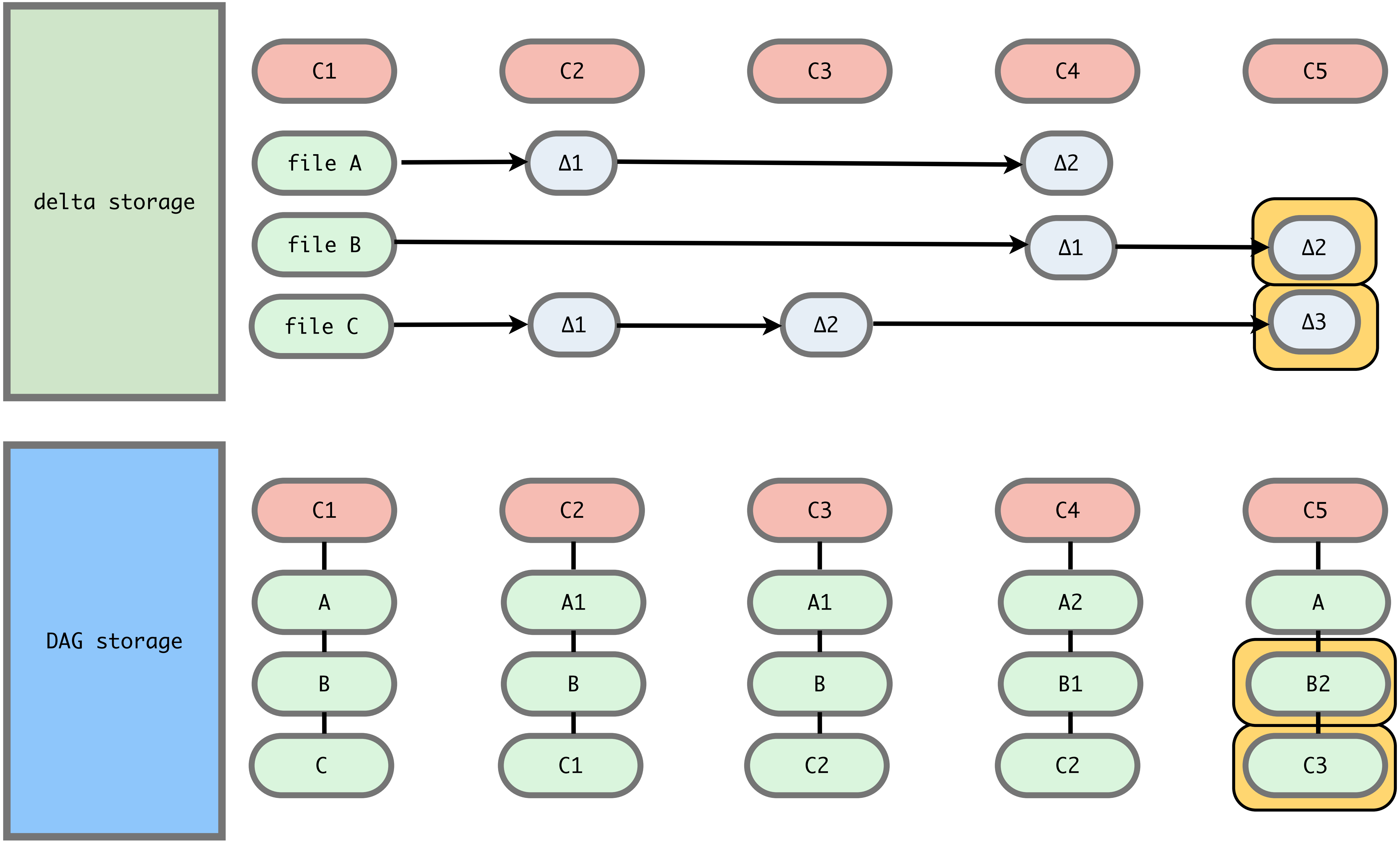
C1 C2 C3 C4

A A1 A1 A2

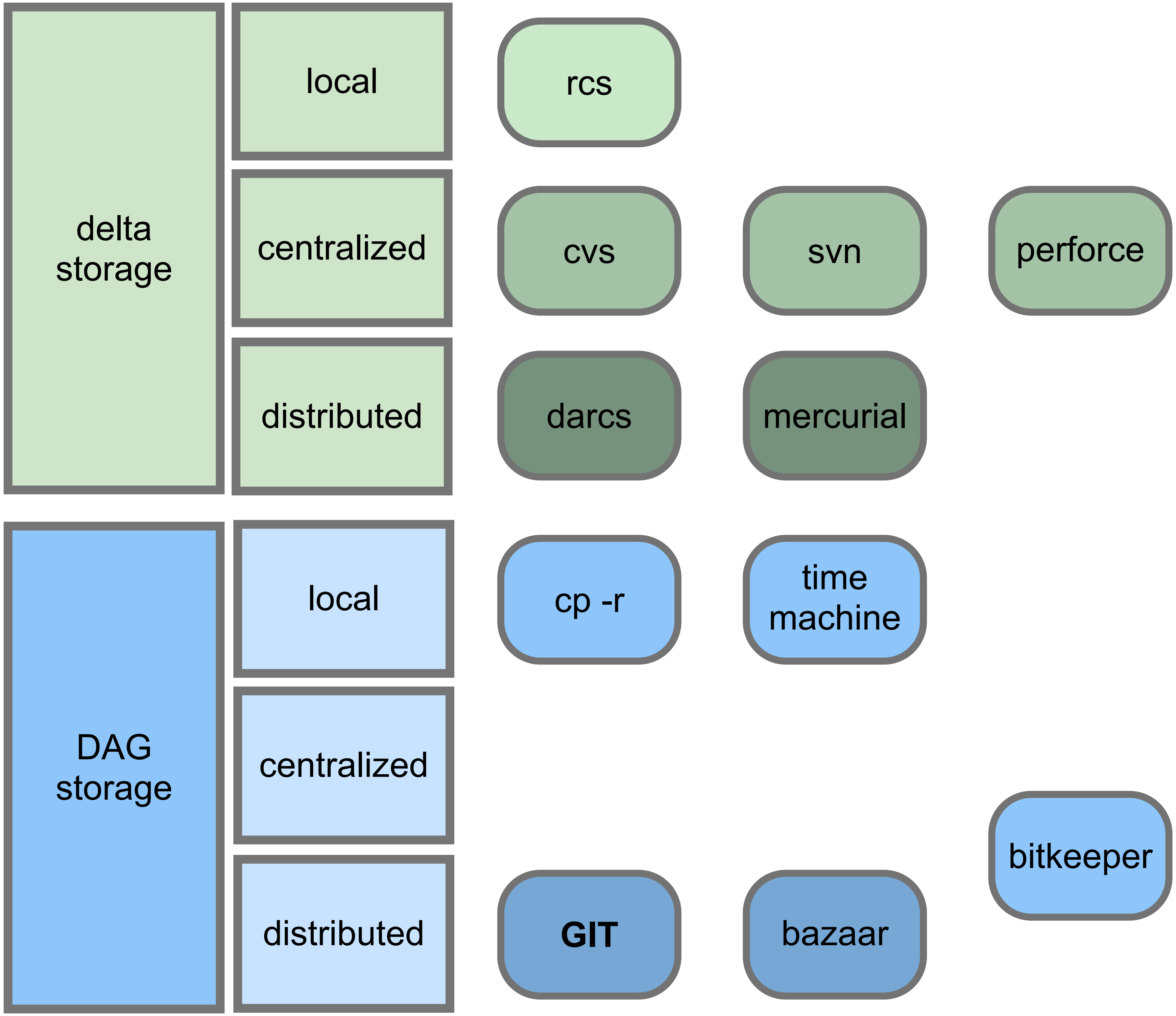
B B B B1

C C1 C2 C2

Delta vs DAG

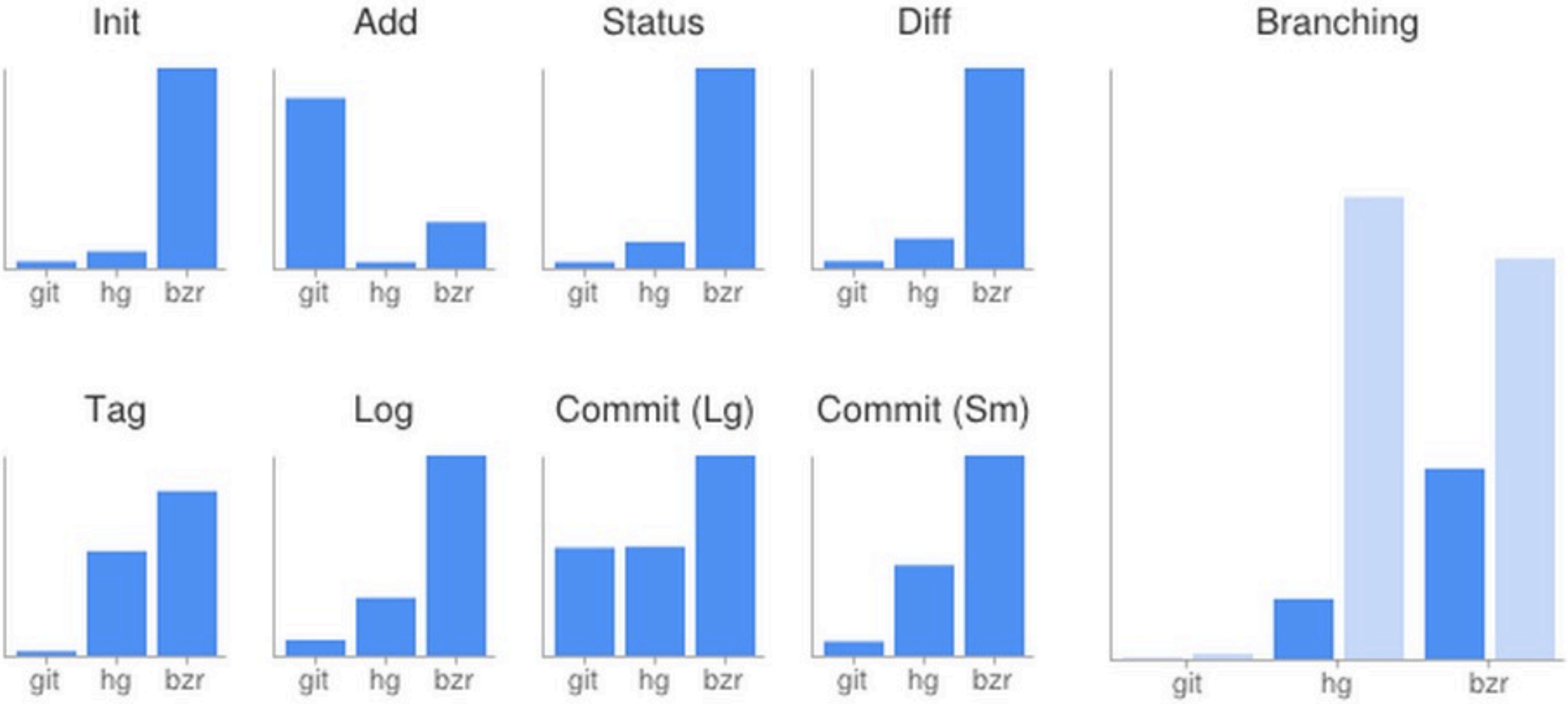

delta storage
C1
C2
C3
C4
C5
file A
Δ1
Δ2
file B
Δ1
Δ2
file C
Δ1
Δ2
Δ3
DAG storage
C1
A
B
C
C2
A1
B
C1
C3
A1
B
C2
C4
A2
B1
C2
C5
A
B2
C3

Source Control Taxonomy



delta storage	local	rcs		
	centralized	cvs	svn	perforce
	distributed	darcs	mercurial	
DAG storage	local	cp -r	time machine	
	centralized			bitkeeper
	distributed	**GIT**	bazaar	

Why GIT?

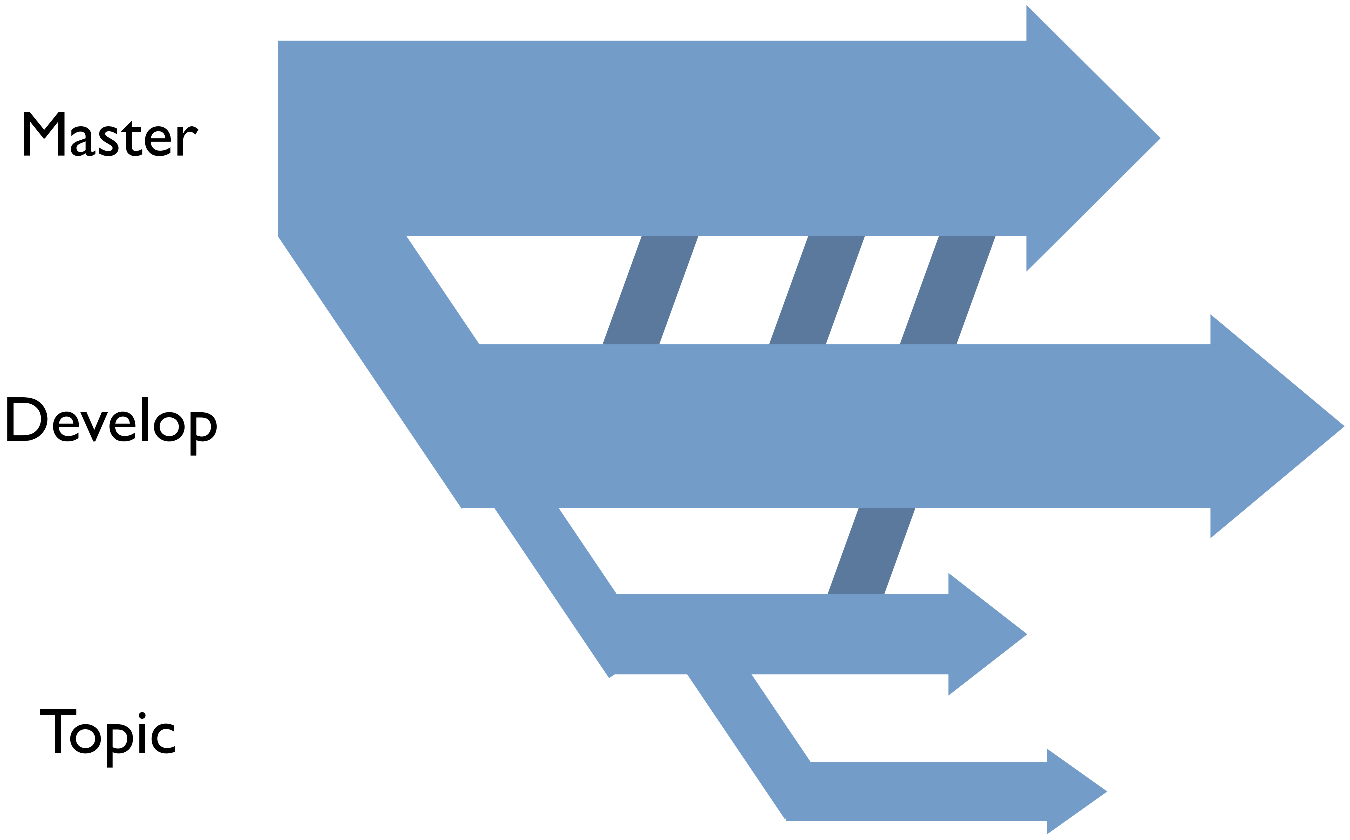

Init
git
hg
bzr
Add
git
hg
bzr
Status
git
hg
bzr
Diff
git
hg
bzr
Branching
Tag
git
hg
bzr
Log
git
hg
bzr
Commit (Lg)
git
hg
bzr
Commit (Sm)
git
hg
bzr
git
hg
bzr

Why GIT?

	Git	Hg	Bzr
Init	0.024s	0,059s	0,600s
Add	8,535s	0,368s	2,381s
Status	0,451s	1,946s	14,744s
Diff	0,543s	2,189s	14,248s
Tag	0,056s	1,201s	1,892s
Log	0,711s	2,650s	9,055s
Commit (large)	12,480s	12,500s	23,002s
Commit (small)	0,086s	0,517s	1,139s
Branch (Cold)	1,161s	94,681s	82,249s
Branch (Hot)	0,070s	12,300s	39,411s

Why GIT?

- Cheap local branch



Master

Develop

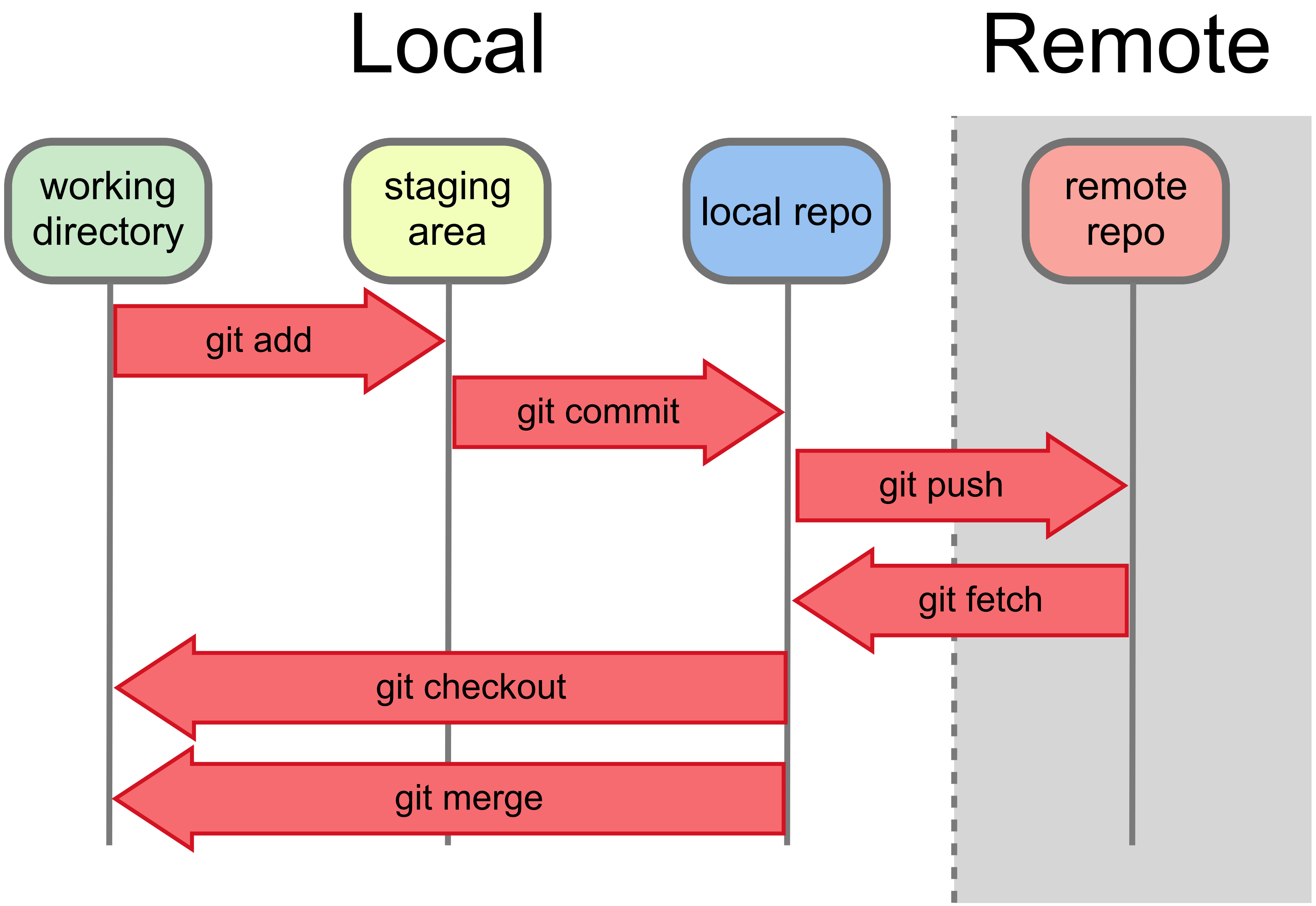
Topic

Why GIT?

- Cheap local branch
  - You can create branch for every new feature you are working on
  - You can commit your changes to your local base
  - You can push them to the server

Why GIT?

- Everything is local



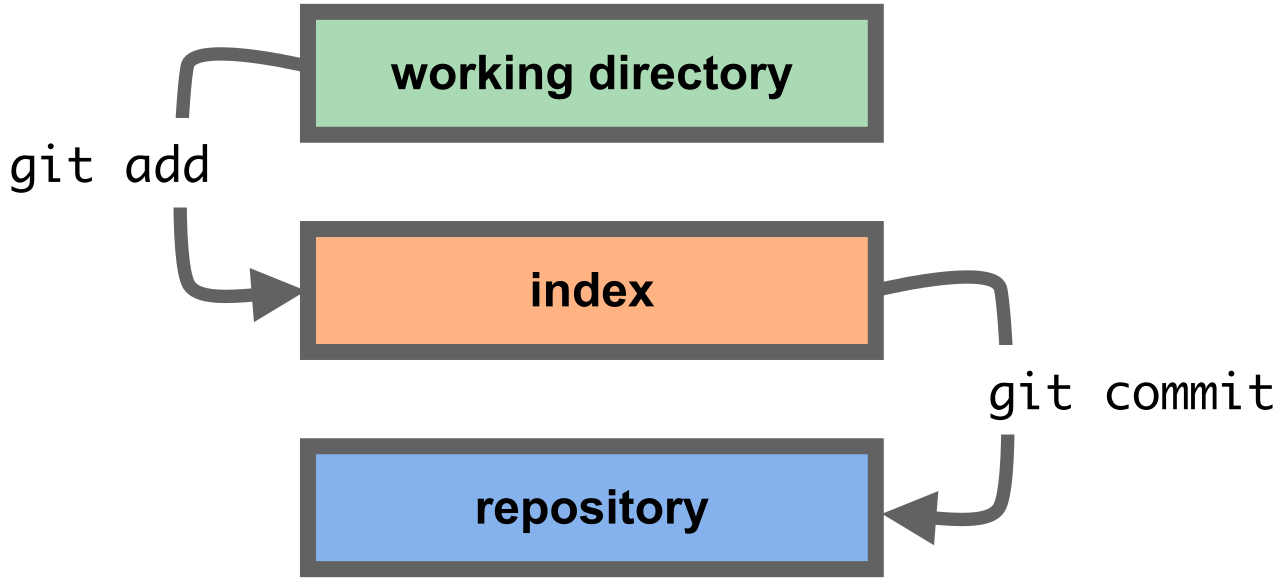
Why GIT?

- Git is small
    - Django project in the same point in history

Git	Hg	Bzr	SVN
24M	34M	45M	
43M	53M	64M	61M

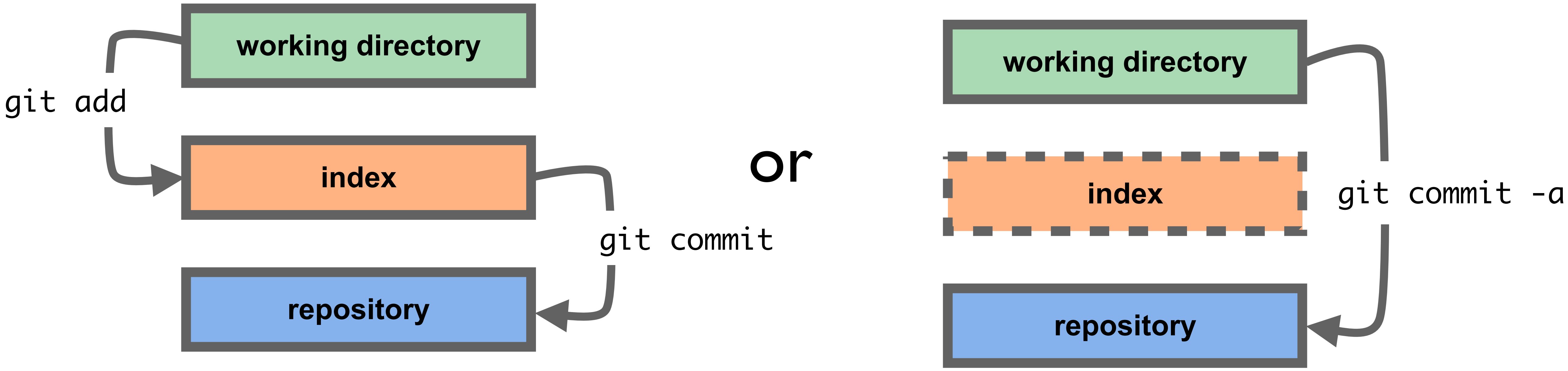
Why GIT?

- The staging area



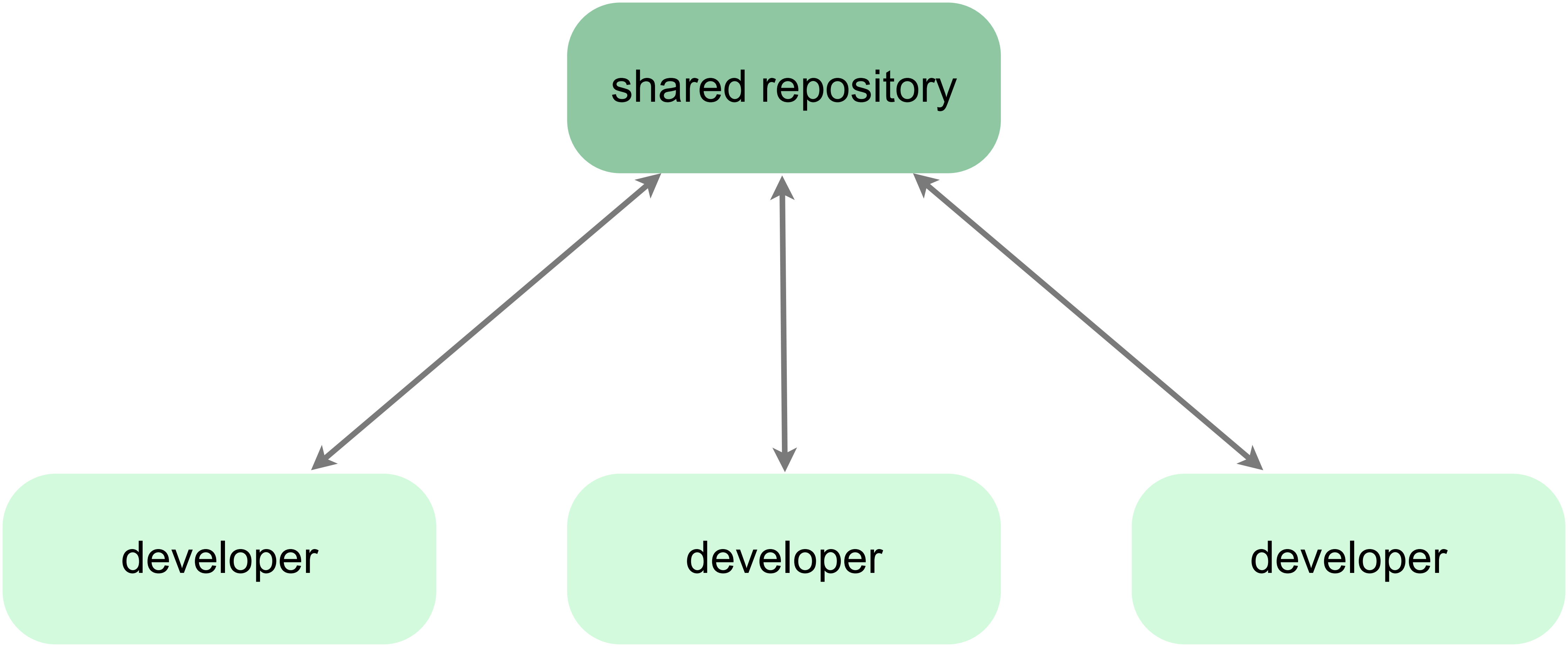
Why GIT?

- The staging area



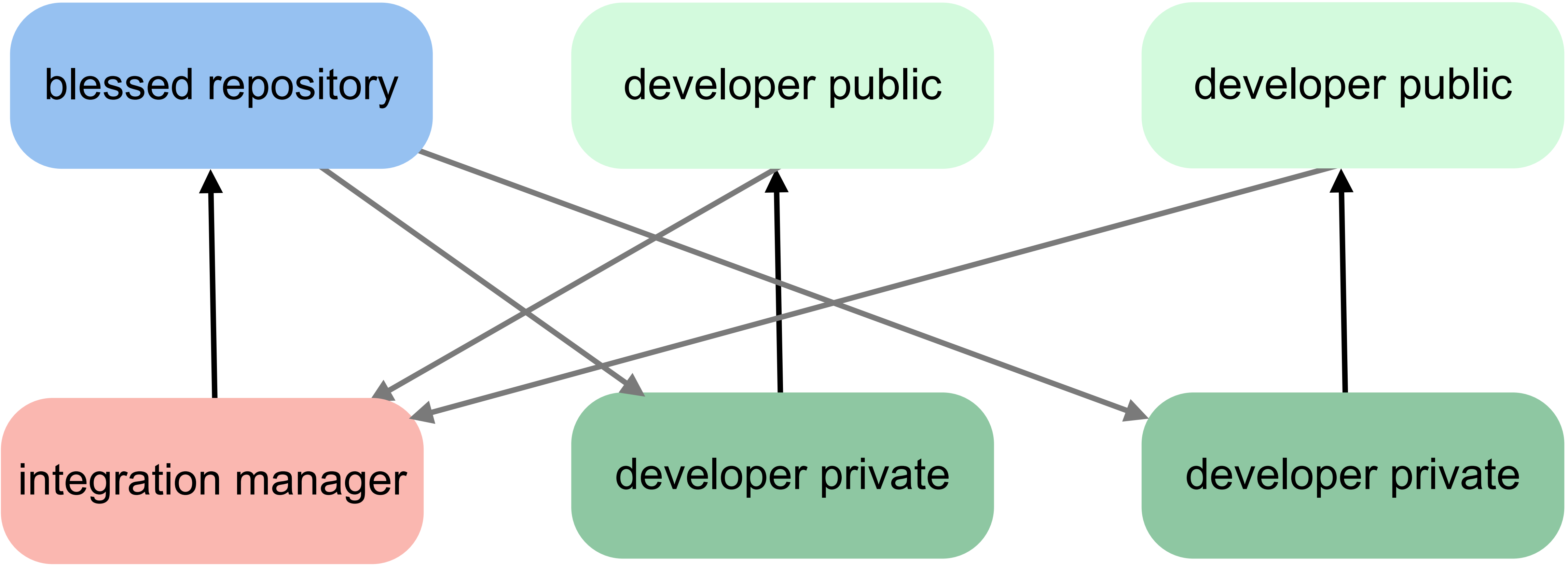
Why GIT?

- Any workflow



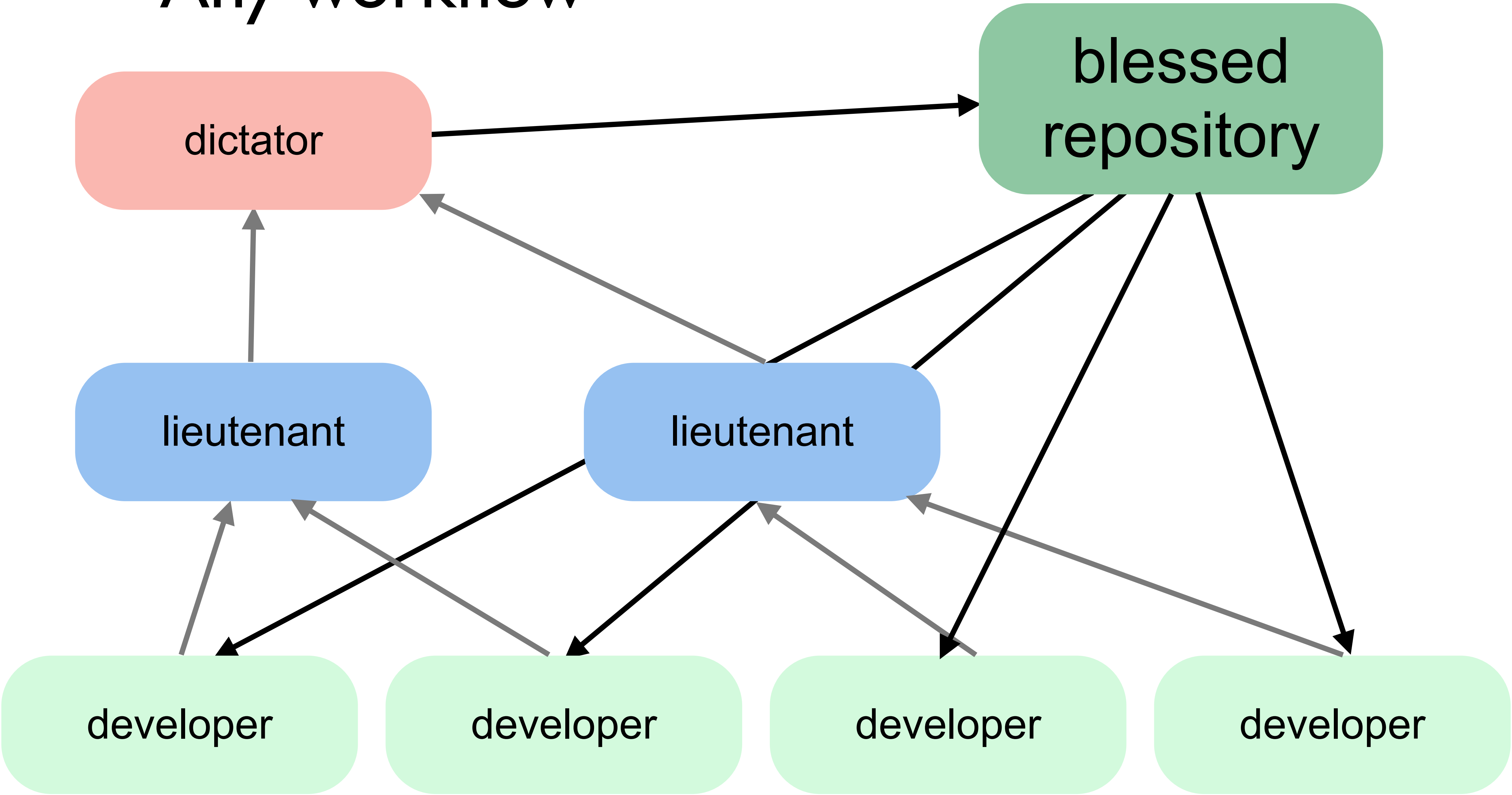
Why GIT?

- Any workflow



Why GIT?

- Any workflow



Why GIT?

- Easy tagging
- Easy version recovery
- Fast switch between branches, tags and commits

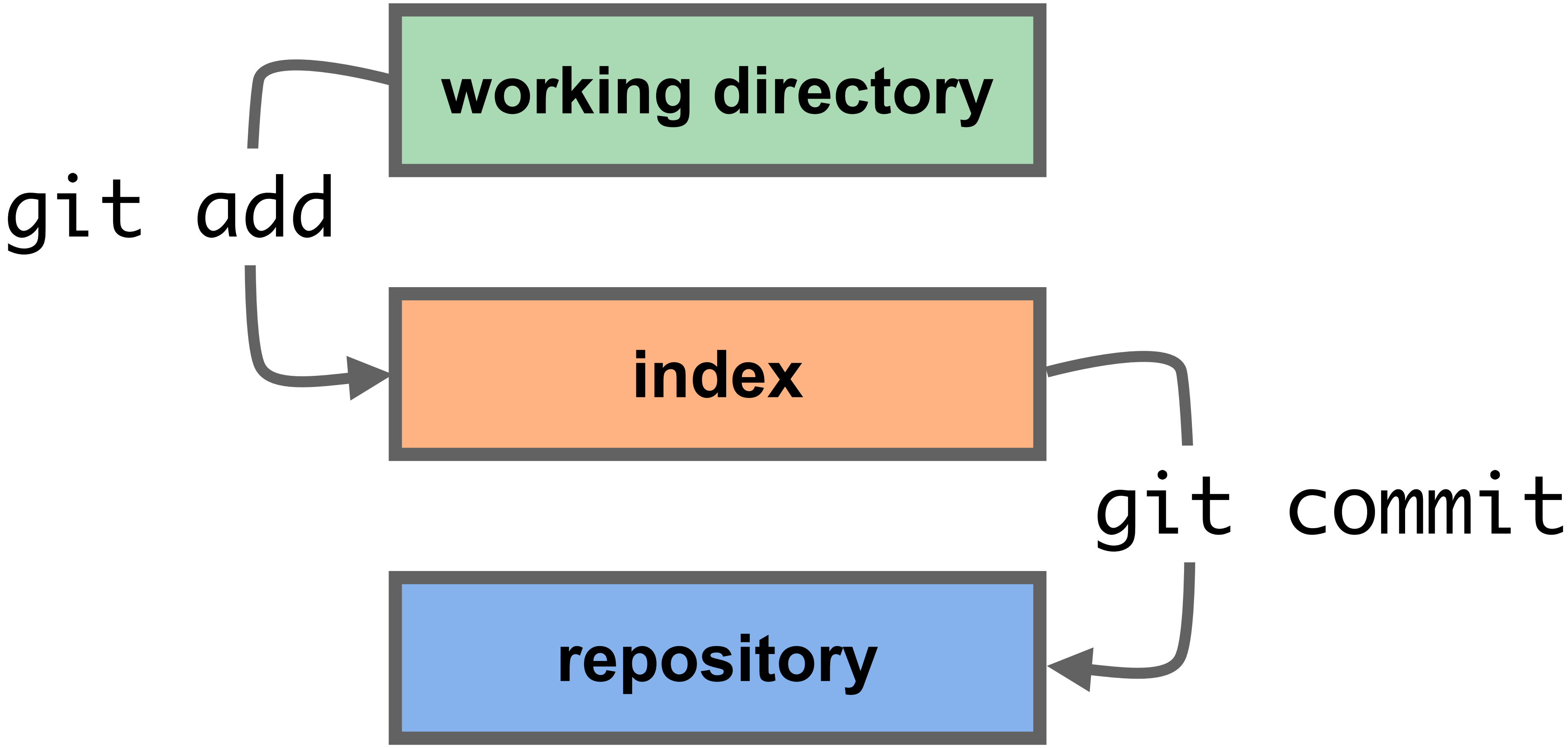
GIT basics

- git init
  - Create an empty git repo
- git clone ssh://server.com/repo
  - Clones an entire remote repo

GIT basics

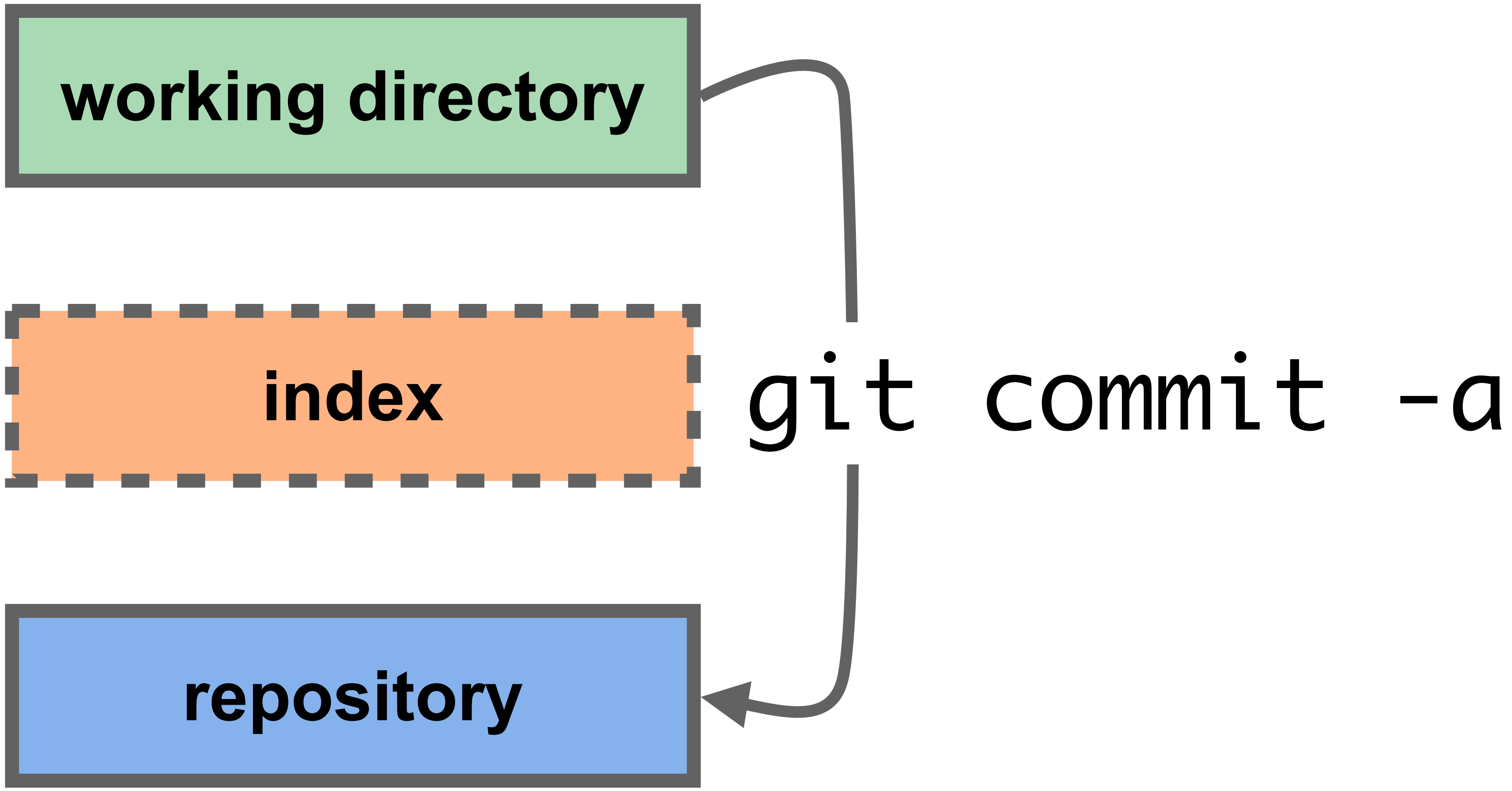
- git add [file | directory]
- git commit [-a] -m “commit message”
- git status

GIT basics



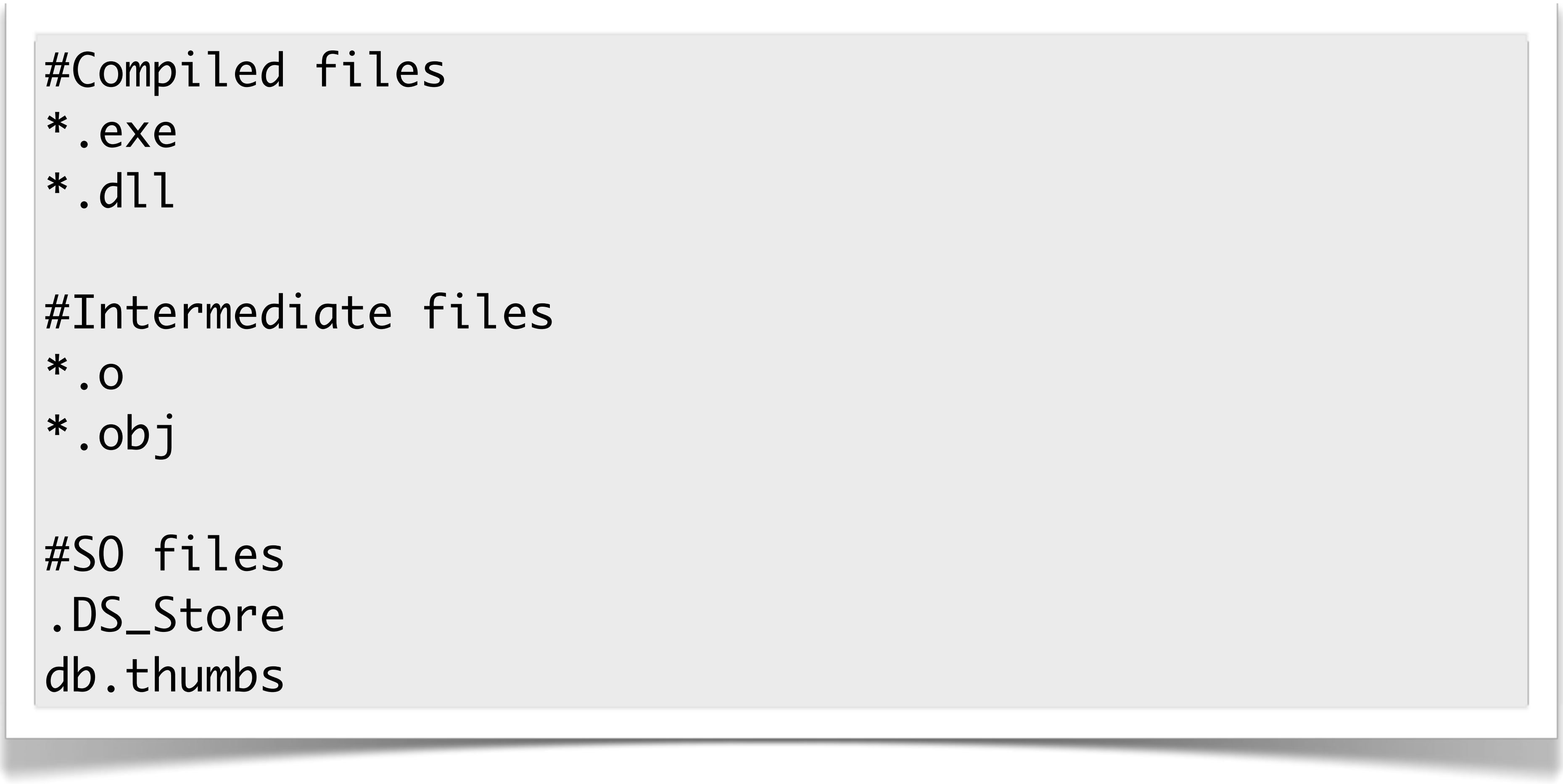
GIT basics

- git commit -a -m “commit message”



GIT basics

- .gitignore



```
#Compiled files
*.exe
*.dll

#Intermediate files
*.o
*.obj

#SO files
.DS_Store
db.thumbs
```

GIT basics

- git push [origin branch_name]
- git fetch
    - saves modified files into origin/ branch_name
- git merge origin/branch_name
    - merges origin/branch_name into current branch

GIT basics

- git push [origin branch_name]
- git pull
    - fetch + merge

GIT internals

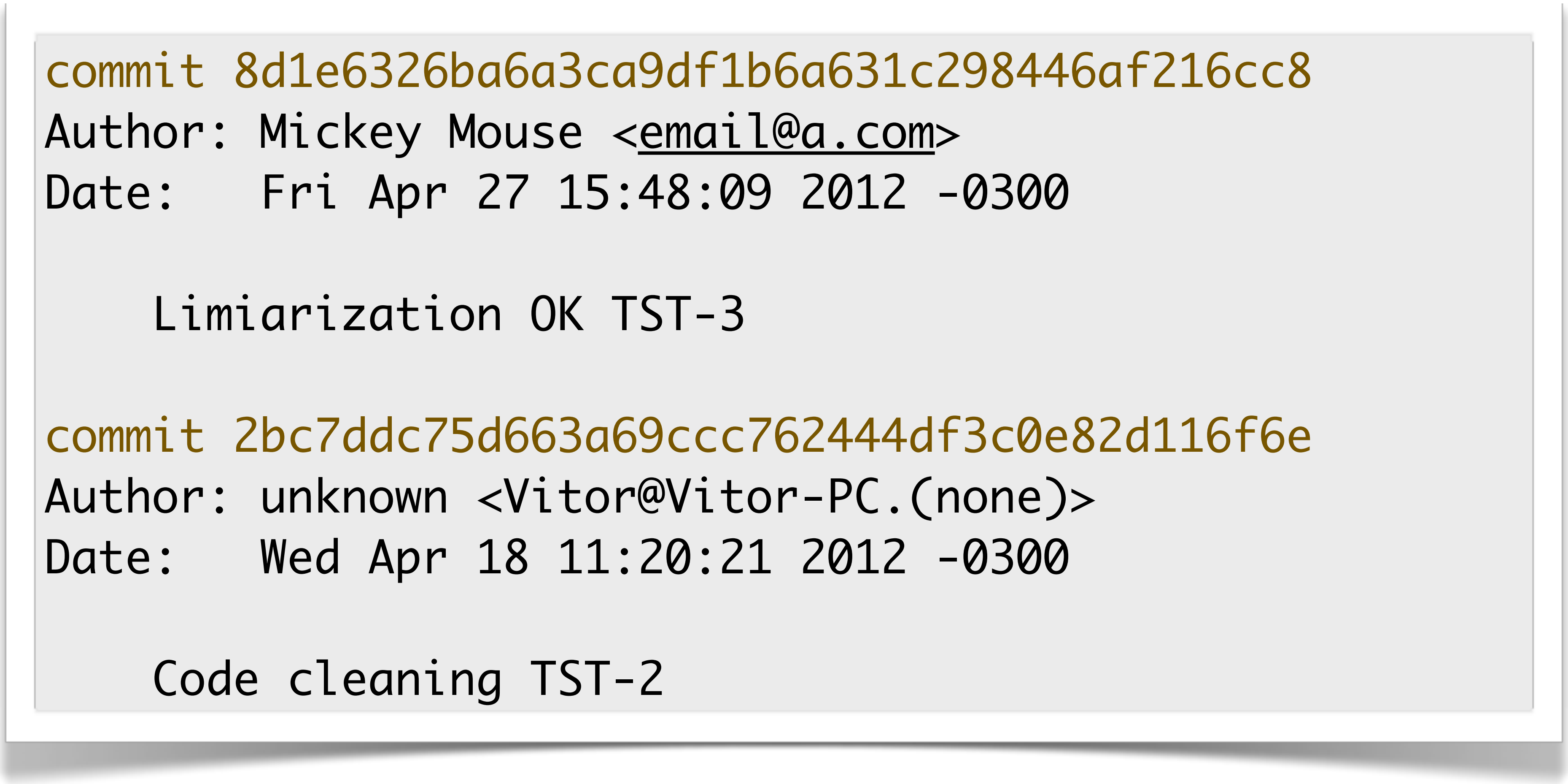
- Every versioned object has a unique 128bits hash code
- files, commits, tags and branches
- GIT stores compressed objects and their hash codes
- You seek any file, commit, tag or branch based on their hash code

GIT commands

- git checkout *
  - commit hash code
  - commit hash code + file name
  - branch name
  - tag name

GIT commands

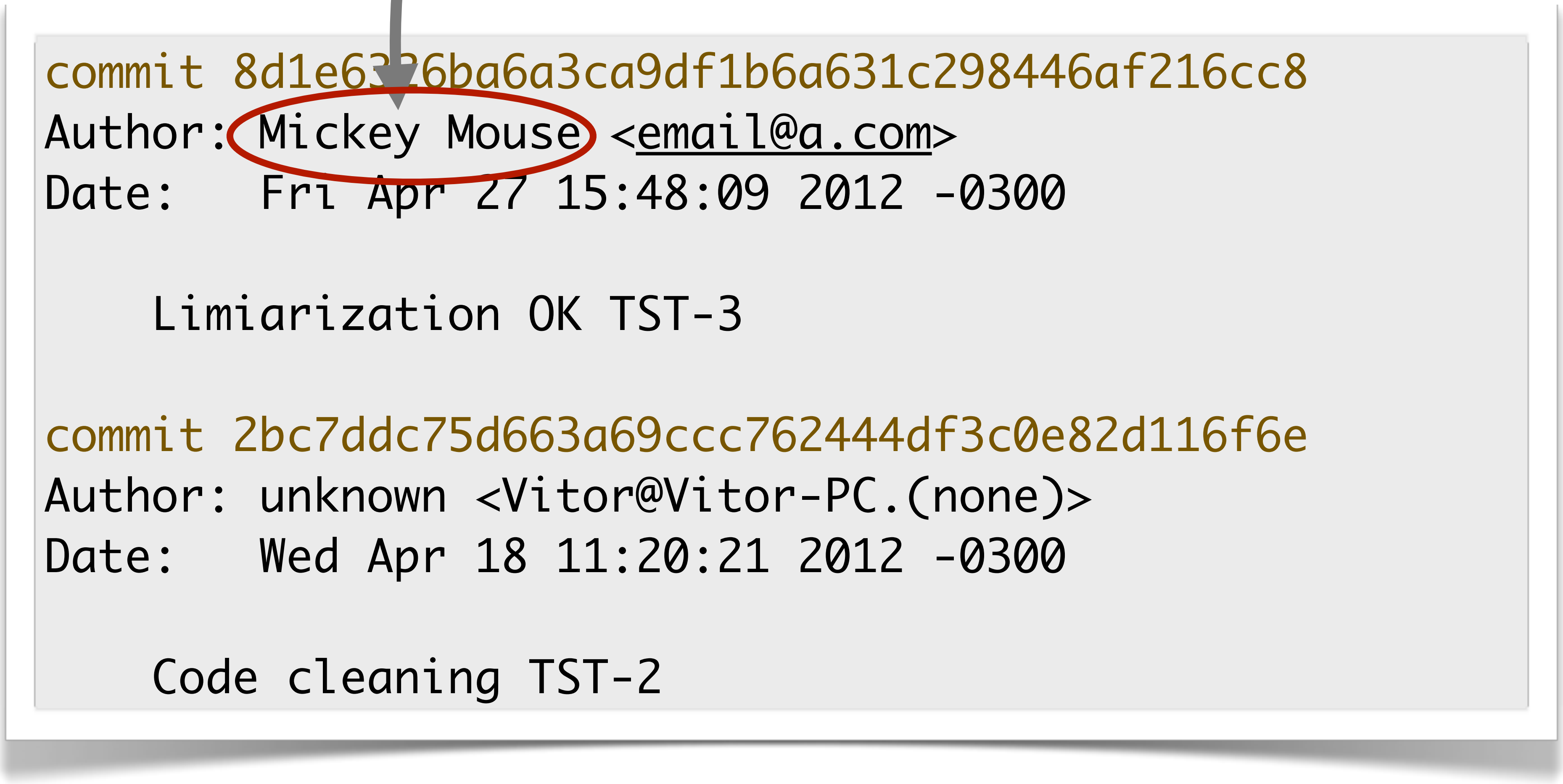
- git log



```
commit 8d1e6326ba6a3ca9df1b6a631c298446af216cc8
Author: Mickey Mouse <email@a.com>
Date:   Fri Apr 27 15:48:09 2012 -0300

    Limiarization OK TST-3

commit 2bc7ddc75d663a69ccc762444df3c0e82d116f6e
Author: unknown <Vitor@Vitor-PC.(none)>
Date:   Wed Apr 18 11:20:21 2012 -0300

    Code cleaning TST-2
```

GIT commands

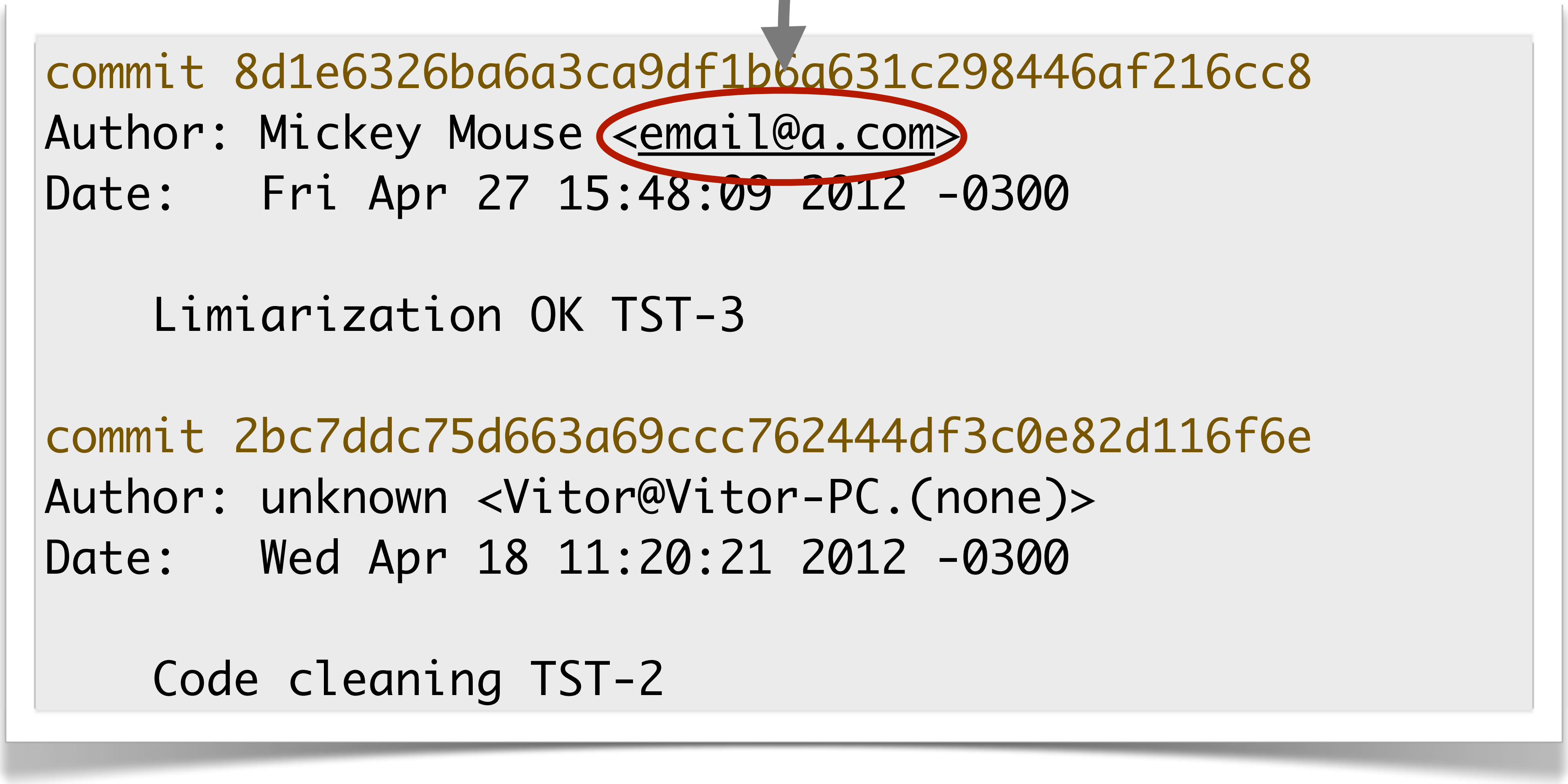
- git config --global user.name "Mickey Mouse"
- git config --global user.email "email@a.com"



```
commit 8d1e6326ba6a3ca9df1b6a631c298446af216cc8
Author: Mickey Mouse <email@a.com>
Date:   Fri Apr 27 15:48:09 2012 -0300

    Limiarization OK TST-3

commit 2bc7ddc75d663a69ccc762444df3c0e82d116f6e
Author: unknown <Vitor@Vitor-PC.(none)>
Date:   Wed Apr 18 11:20:21 2012 -0300

    Code cleaning TST-2
```

GIT commands

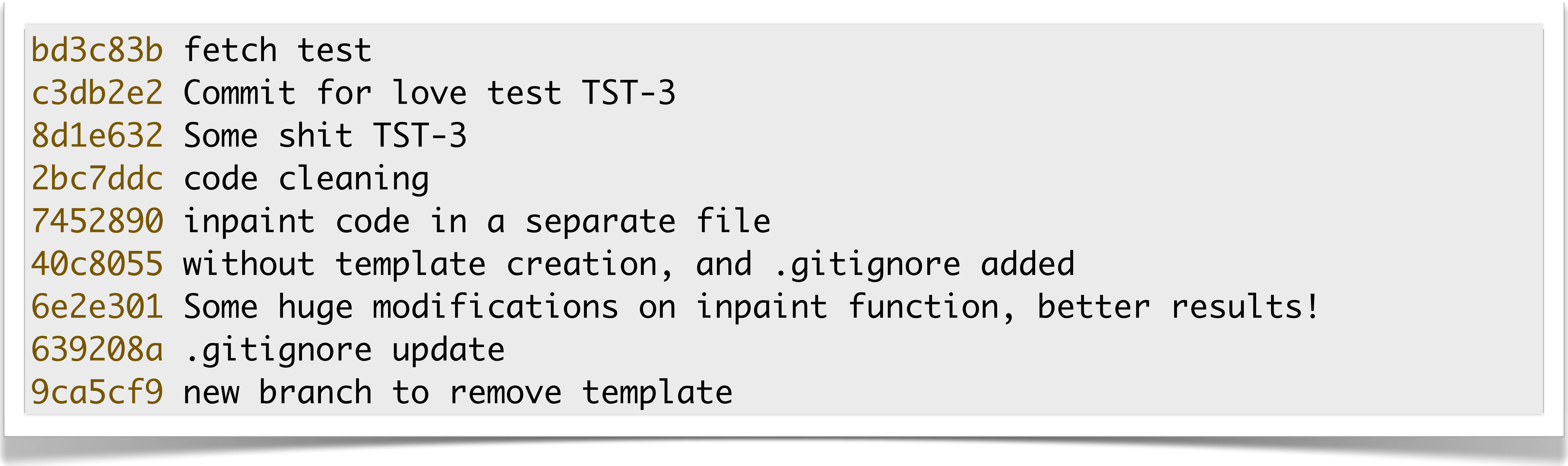
- git config --global user.name “Mickey Mouse”
- git config --global user.email “email@a.com”



```
commit 8d1e6326ba6a3ca9df1b6a631c298446af216cc8
Author: Mickey Mouse <email@a.com>
Date:   Fri Apr 27 15:48:09 2012 -0300

    Limiarization OK TST-3

commit 2bc7ddc75d663a69ccc762444df3c0e82d116f6e
Author: unknown <Vitor@Vitor-PC.(none)>
Date:   Wed Apr 18 11:20:21 2012 -0300

    Code cleaning TST-2
```

GIT commands

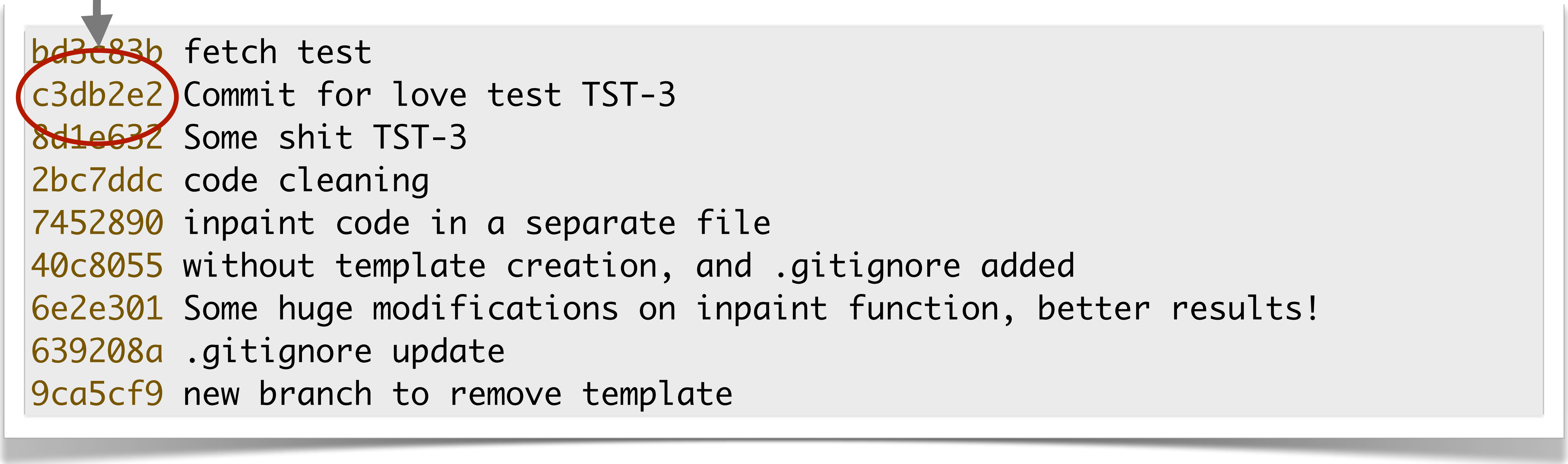
- git log --pretty-oneline --abbrev-commit



```
bd3c83b fetch test
c3db2e2 Commit for love test TST-3
8d1e632 Some shit TST-3
2bc7ddc code cleaning
7452890 inpaint code in a separate file
40c8055 without template creation, and .gitignore added
6e2e301 Some huge modifications on inpaint function, better results!
639208a .gitignore update
9ca5cf9 new branch to remove template
```

GIT commands

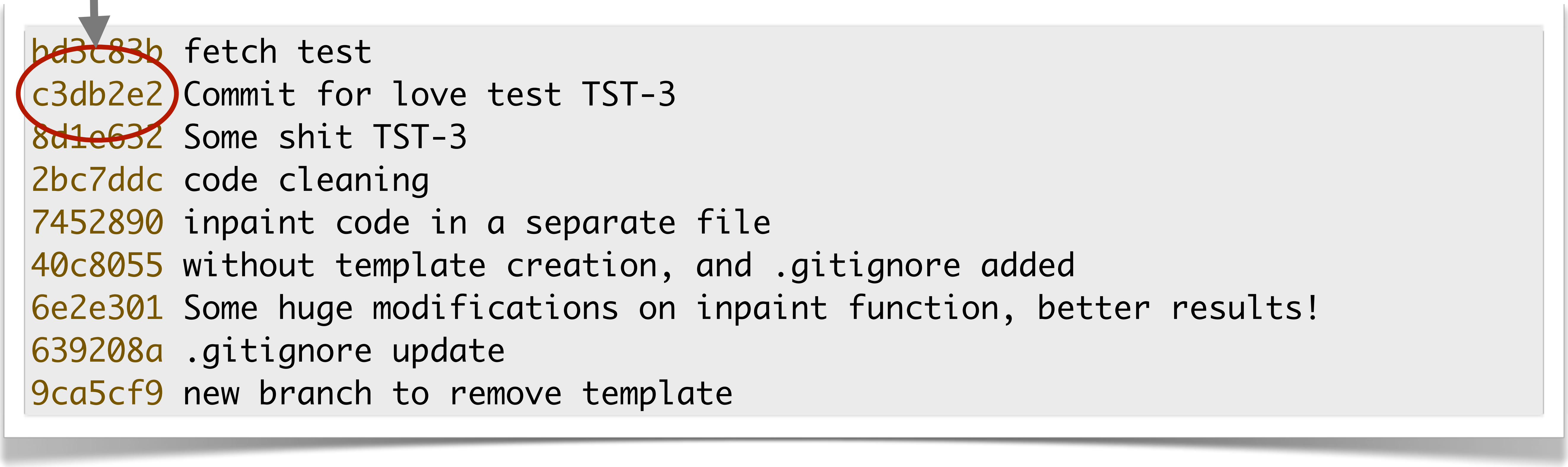
- git checkout c3db
  - at least 4 characters from hash code



```
bd3c83b fetch test
c3db2e2 Commit for love test TST-3
8d1e632 Some shit TST-3
2bc7ddc code cleaning
7452890 inpaint code in a separate file
40c8055 without template creation, and .gitignore added
6e2e301 Some huge modifications on inpaint function, better results!
639208a .gitignore update
9ca5cf9 new branch to remove template
```

GIT commands

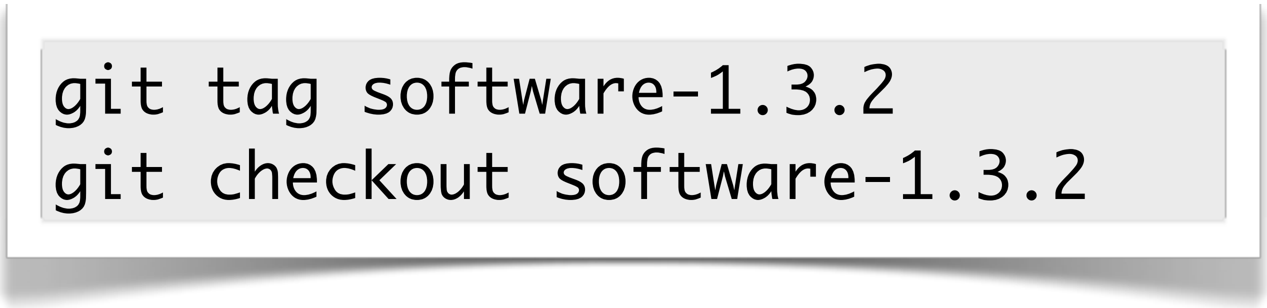
- git checkout c3db README.txt
  - at least 4 characters from hash code
  - Optional filename to restore



```
bd3c83b fetch test
c3db2e2 Commit for love test TST-3
8d1e632 Some shit TST-3
2bc7ddc code cleaning
7452890 inpaint code in a separate file
40c8055 without template creation, and .gitignore added
6e2e301 Some huge modifications on inpaint function, better results!
639208a .gitignore update
9ca5cf9 new branch to remove template
```

GIT commands

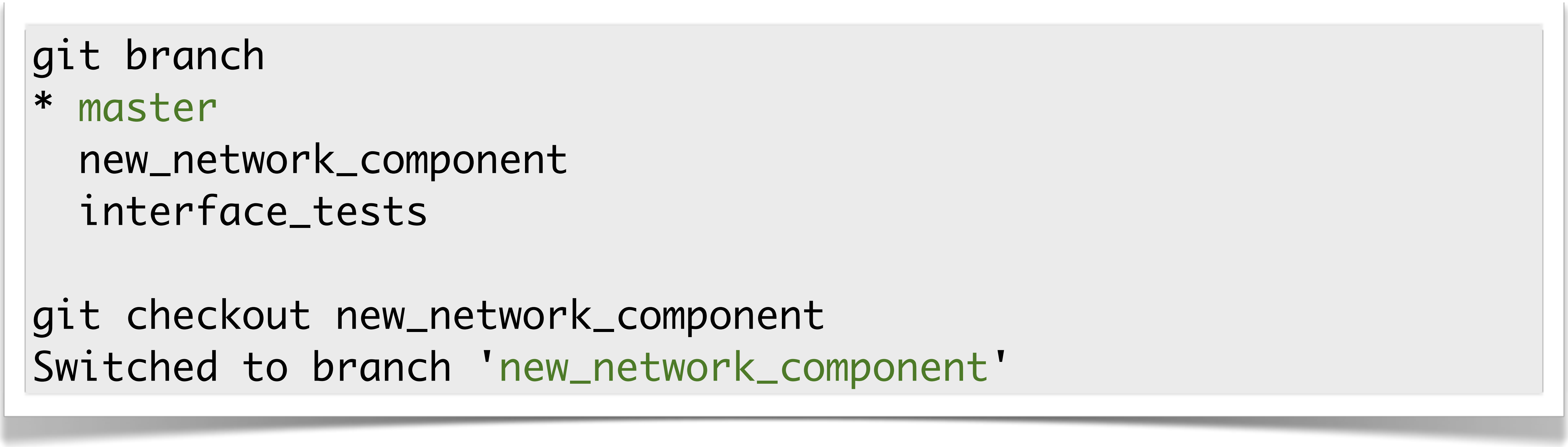
- git tag
- git tag tag_name
- git tag -d tag_name
- git push origin tag_name



```
git tag software-1.3.2
git checkout software-1.3.2
```

GIT commands

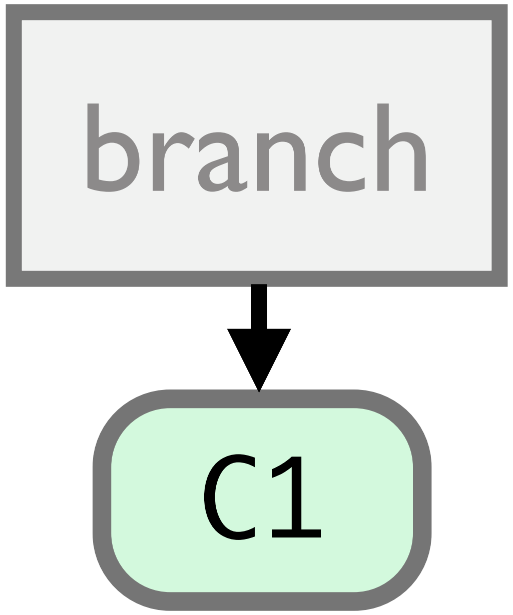
- git branch
- git branch branch_name
- git branch -d branch_name



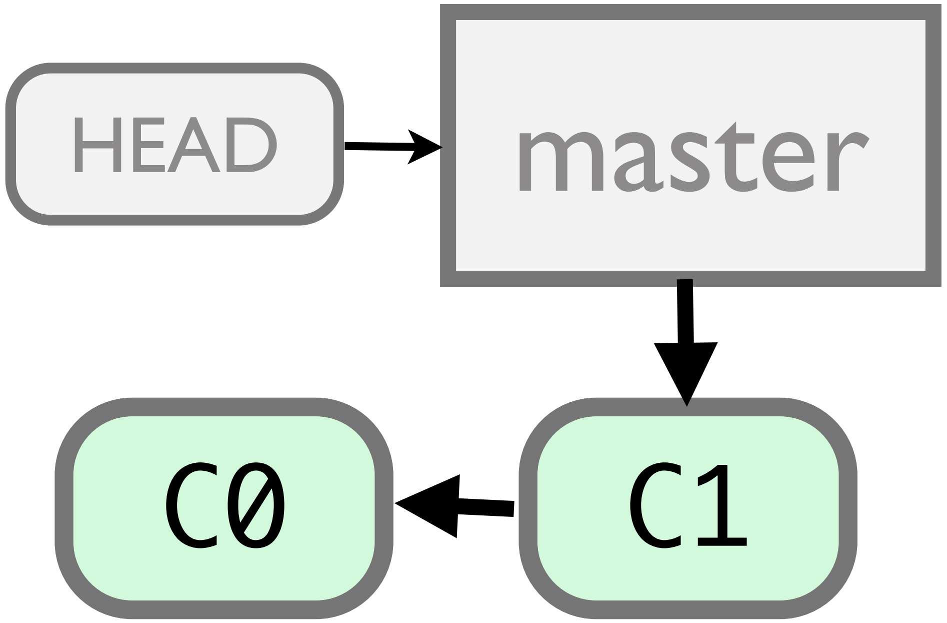
```
git branch
* master
  new_network_component
  interface_tests

git checkout new_network_component
Switched to branch 'new_network_component'
```

GIT internals

- Branch is a lightweight movable pointer to a commit

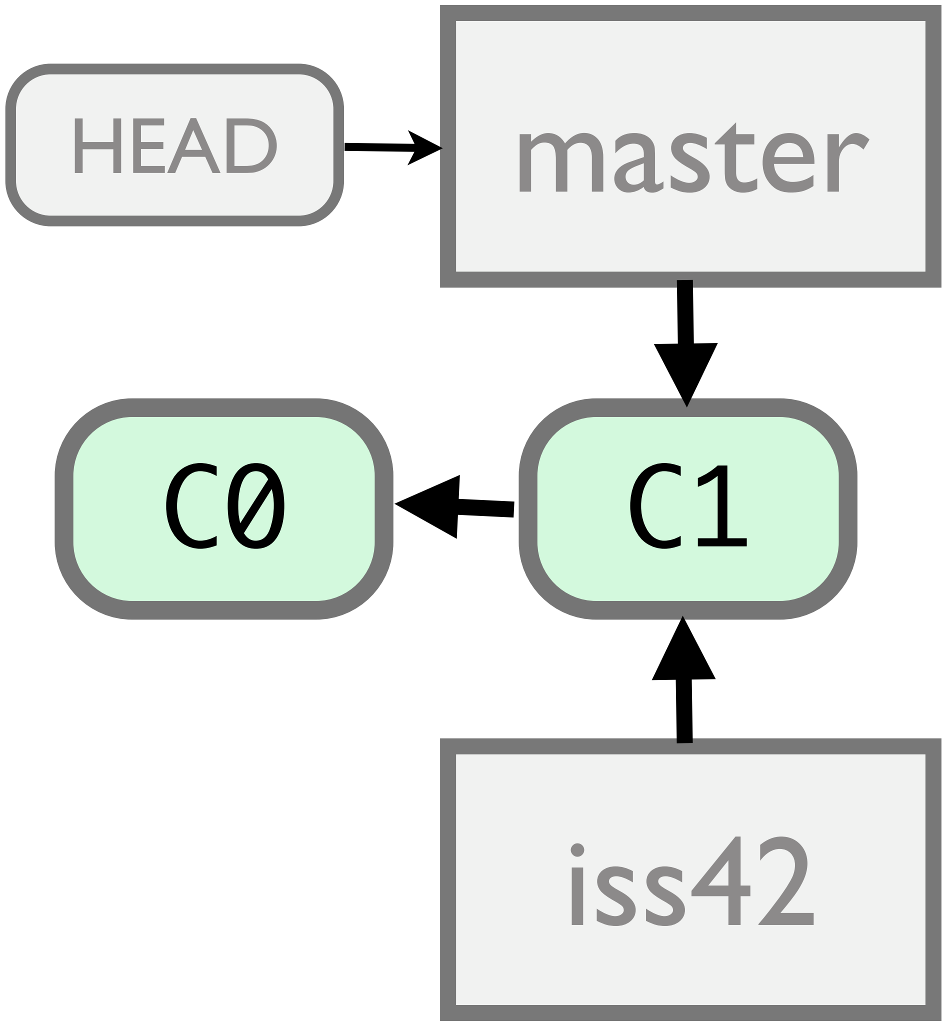


GIT internals



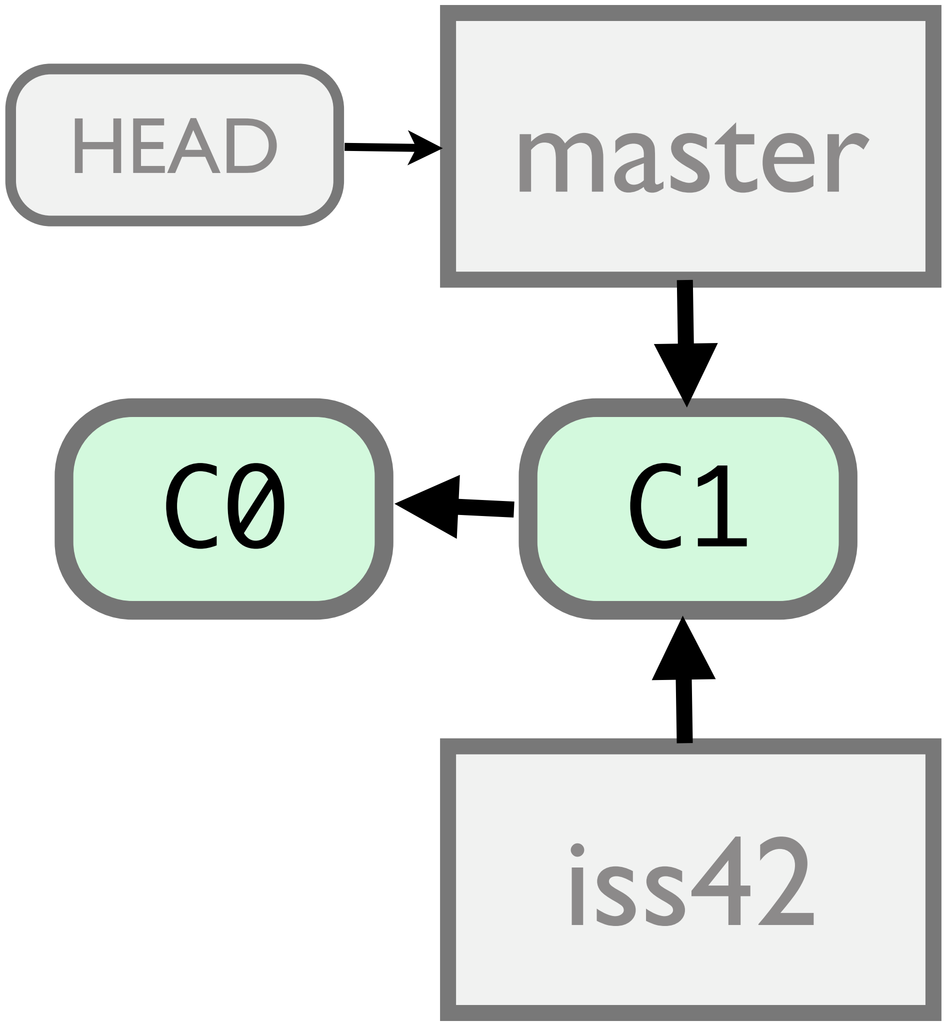
```
git commit
```

GIT internals



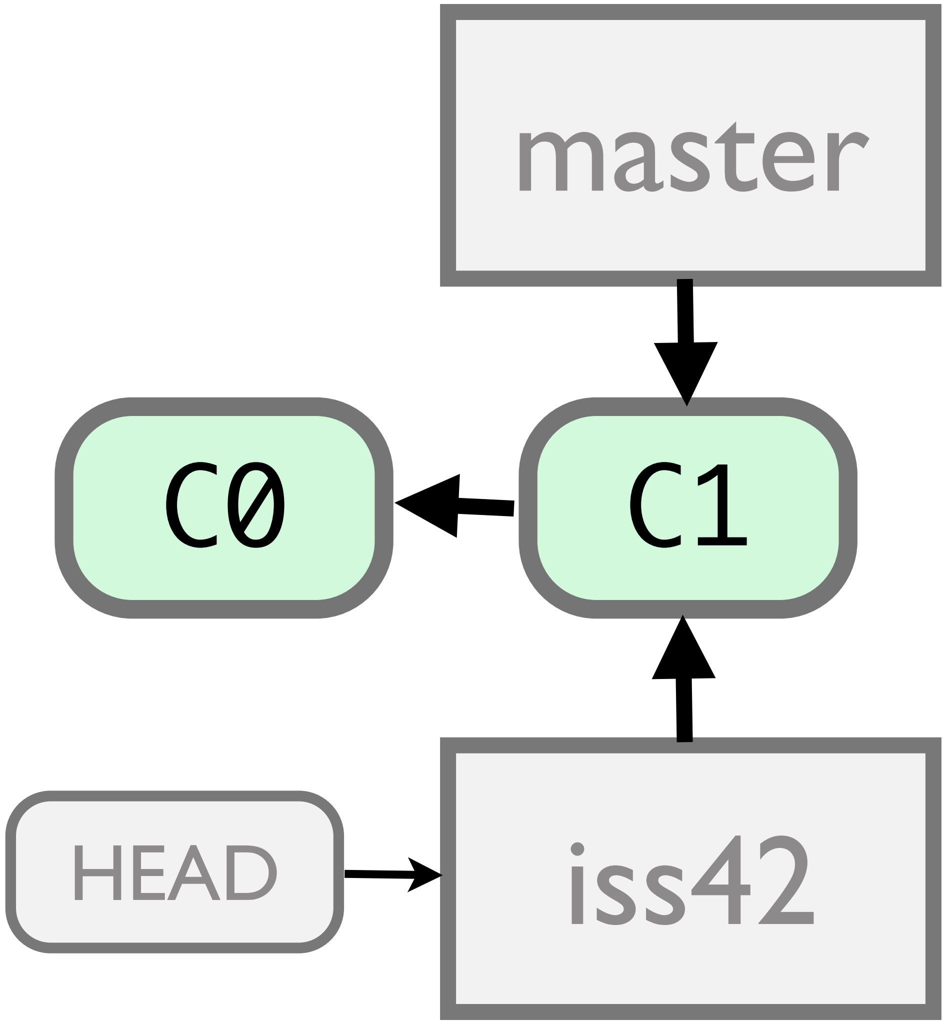
```
git branch iss42
```

GIT internals

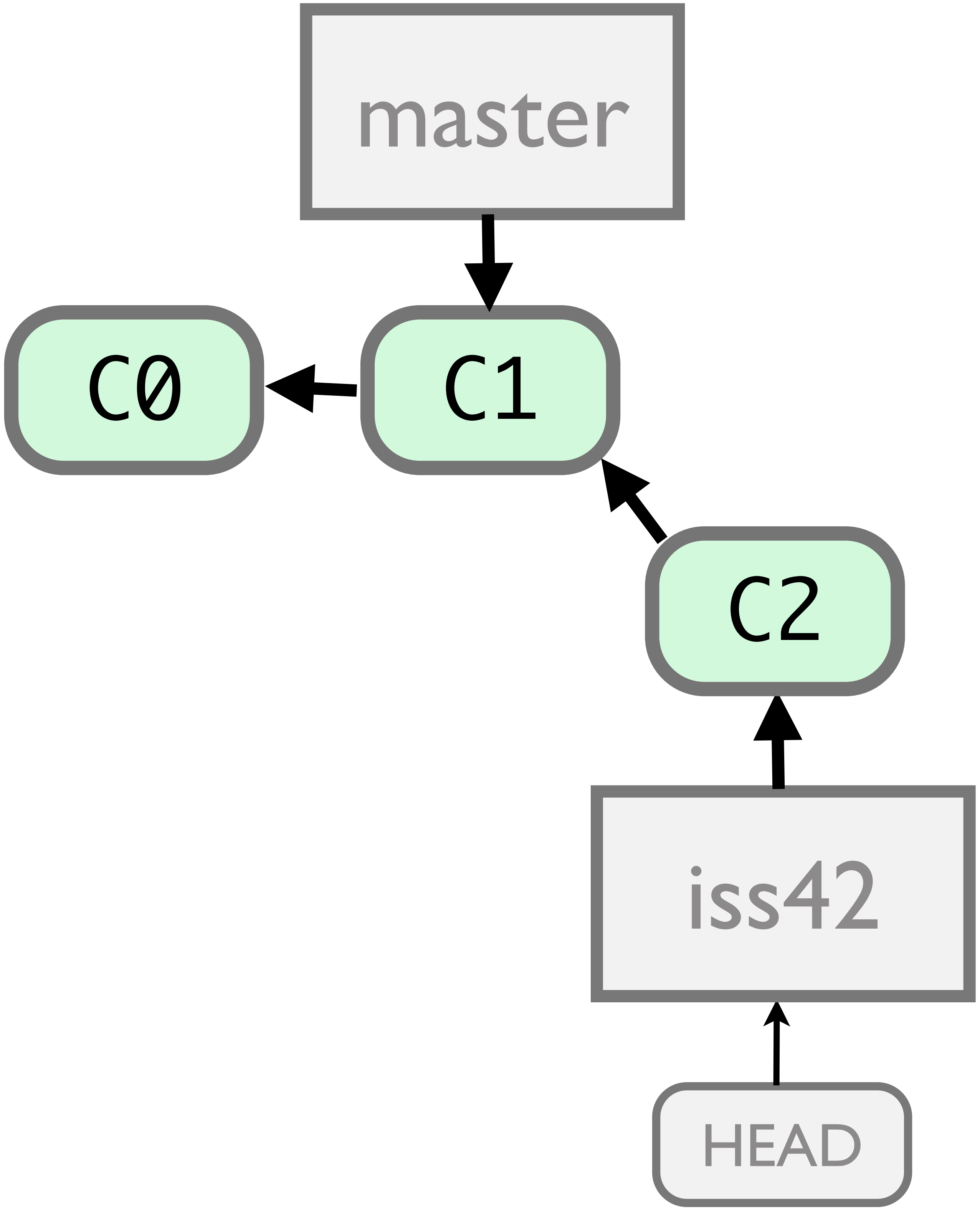


```
git branch
* master
  iss42
```

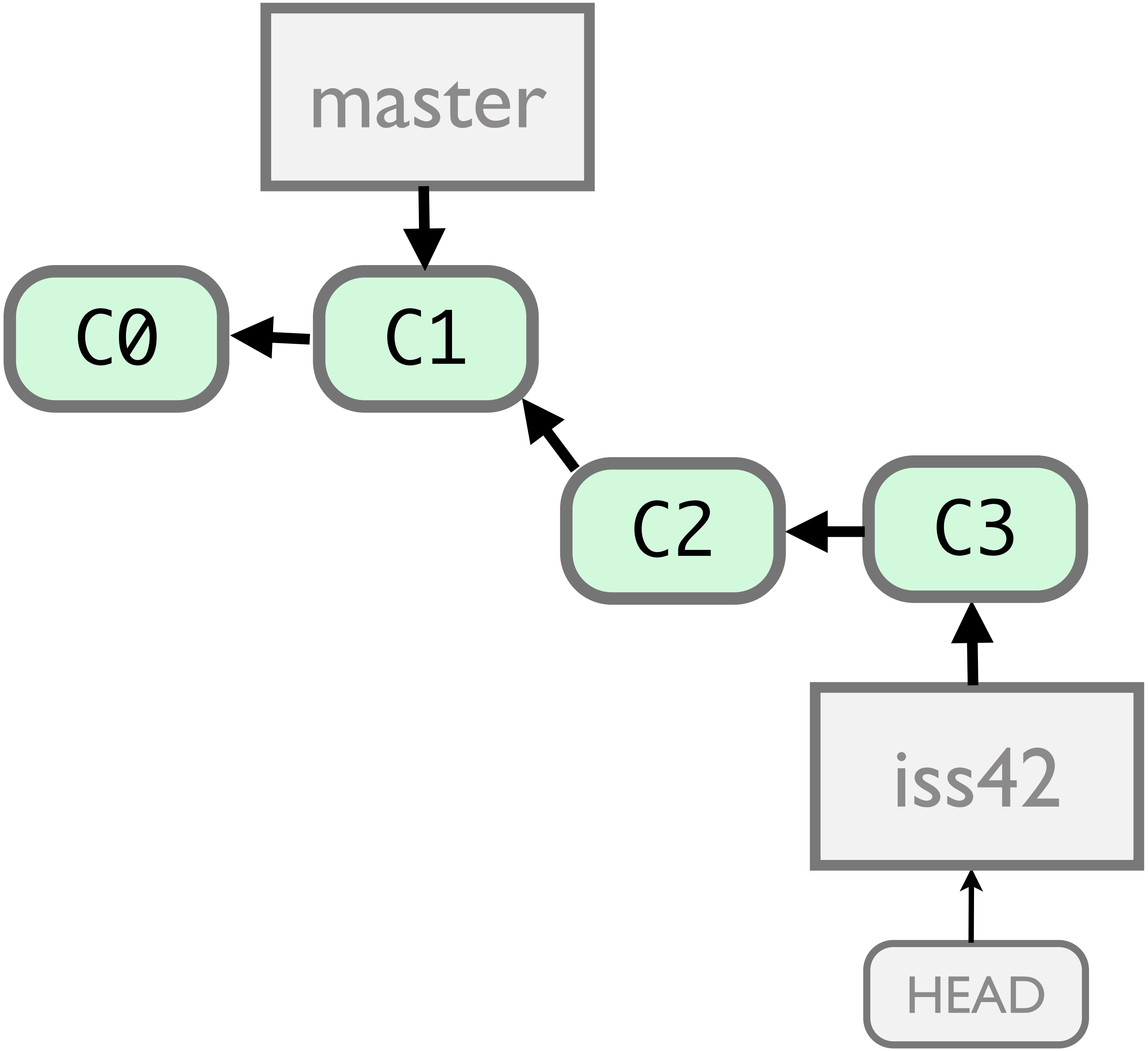
GIT internals



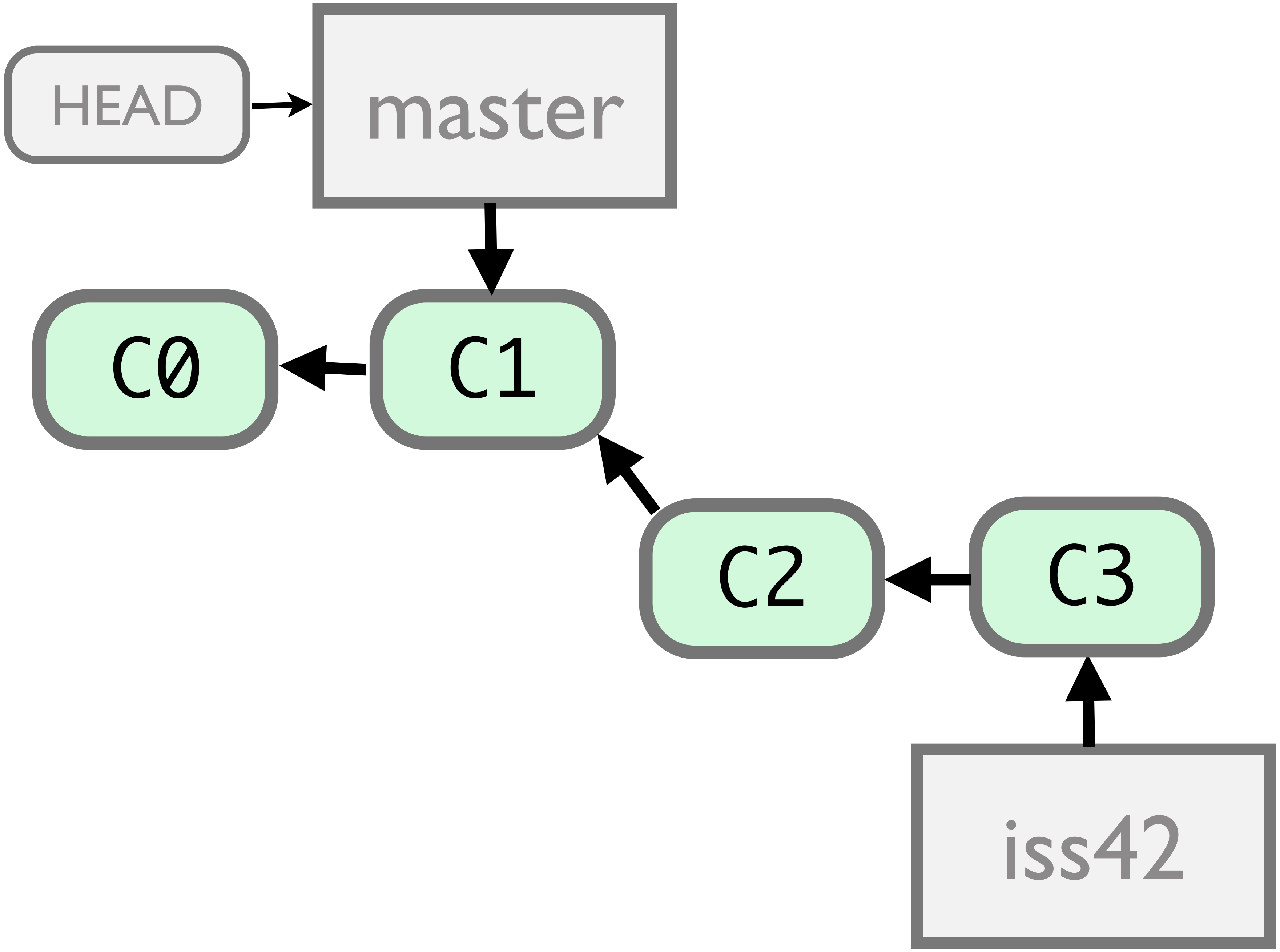
```
git checkout iss42
```



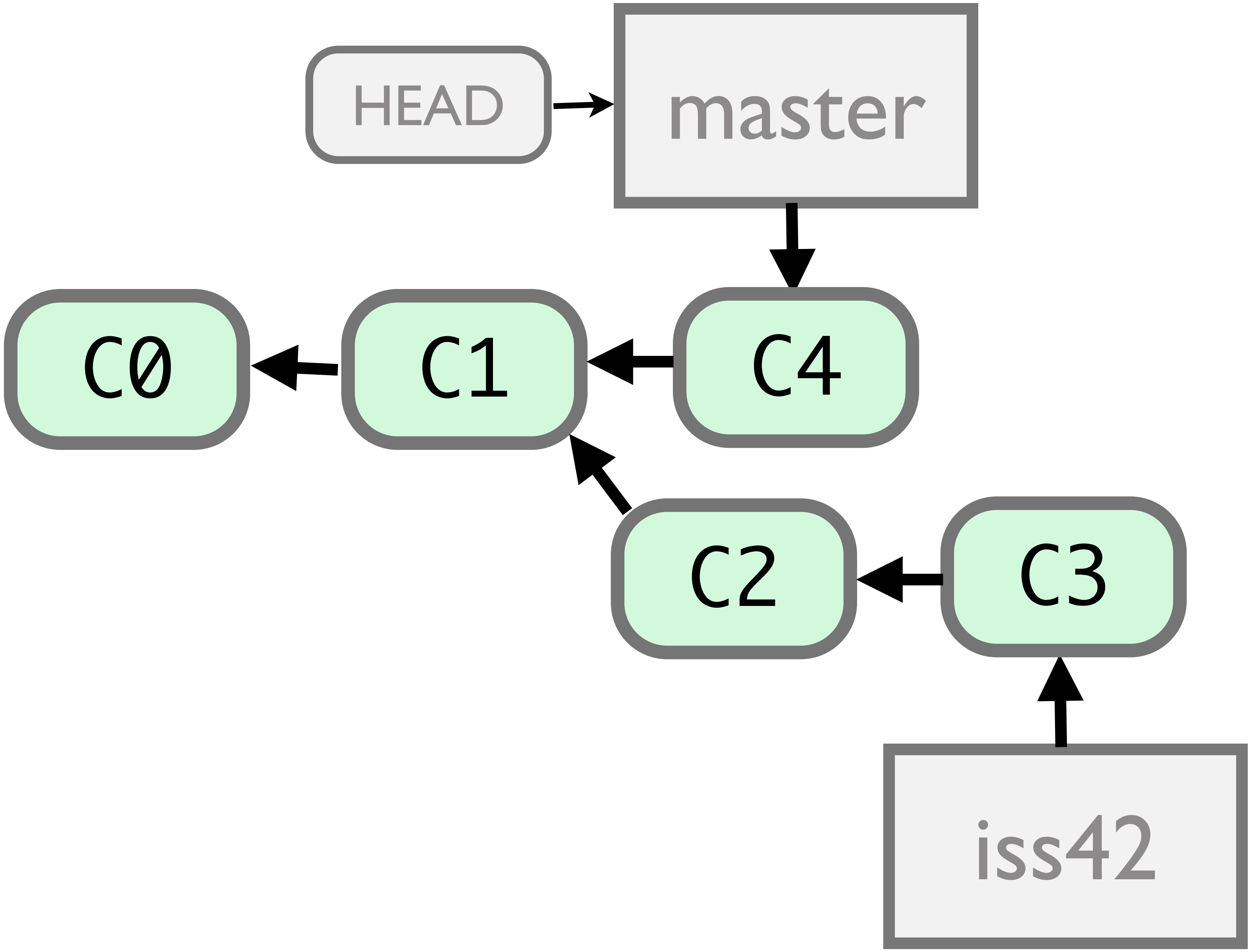
```
git commit
```



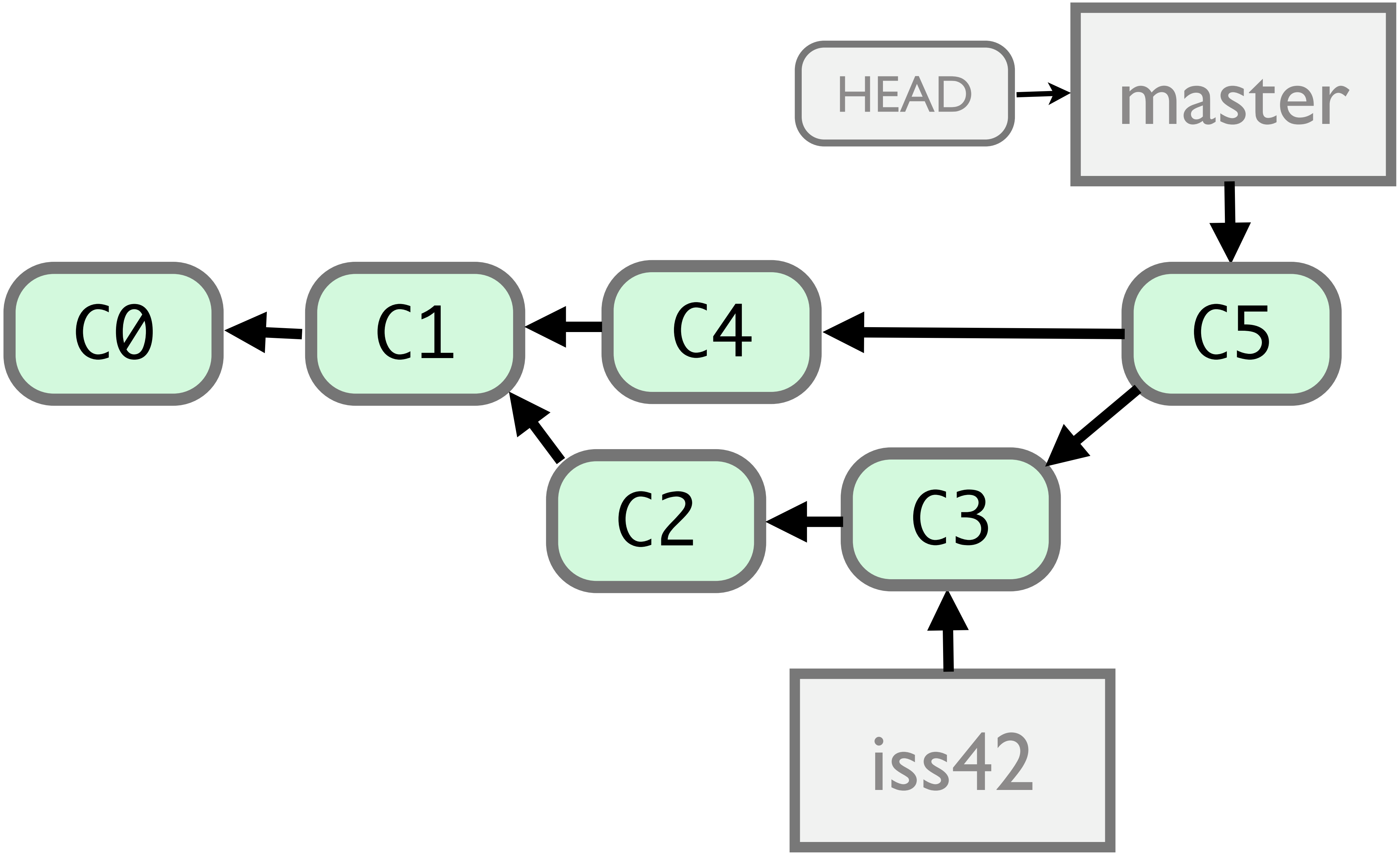
```
git commit
```



```
git checkout master
```



```
git commit
```



```
git merge iss42
```

GIT example workflow

GIT example workflow

- clone the code that is in production

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)

GIT example workflow

- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)
- fix the issue

GIT example workflow

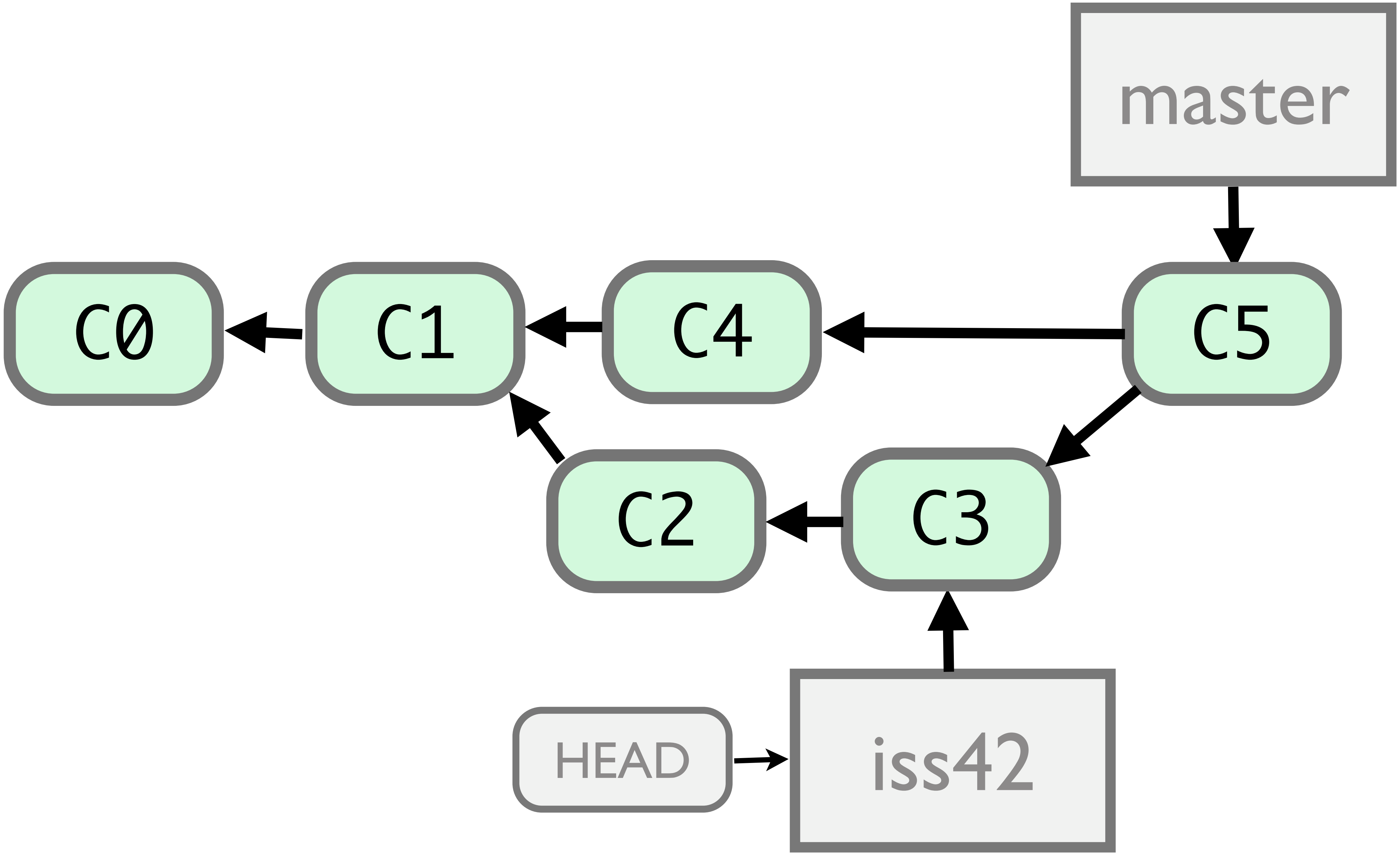
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)
- fix the issue
- checkout ‘production’, merge ‘iss102’

GIT example workflow

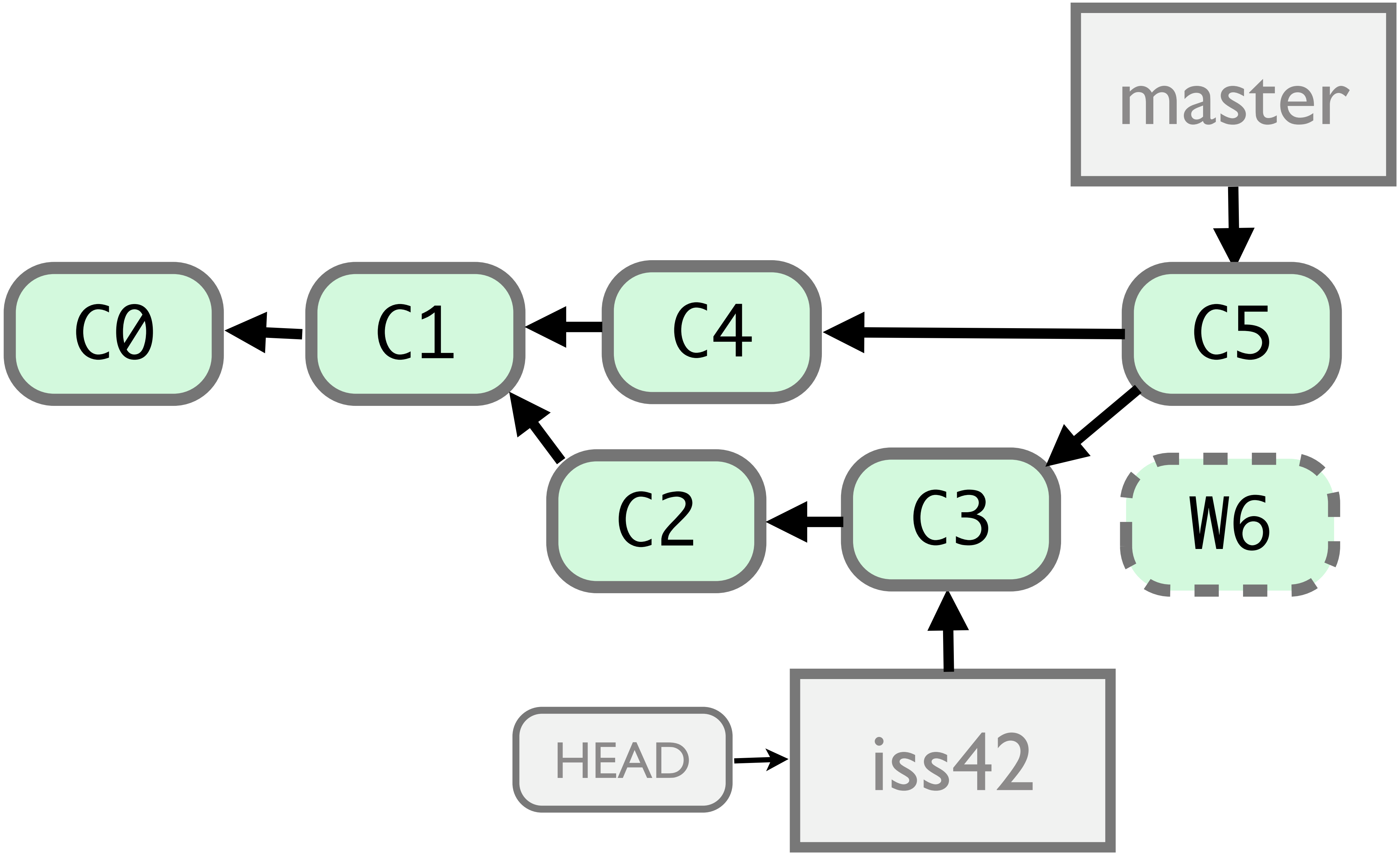
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout ‘production’
- create a branch (iss102)
- fix the issue
- checkout ‘production’, merge ‘iss102’
- push ‘production’

GIT example workflow

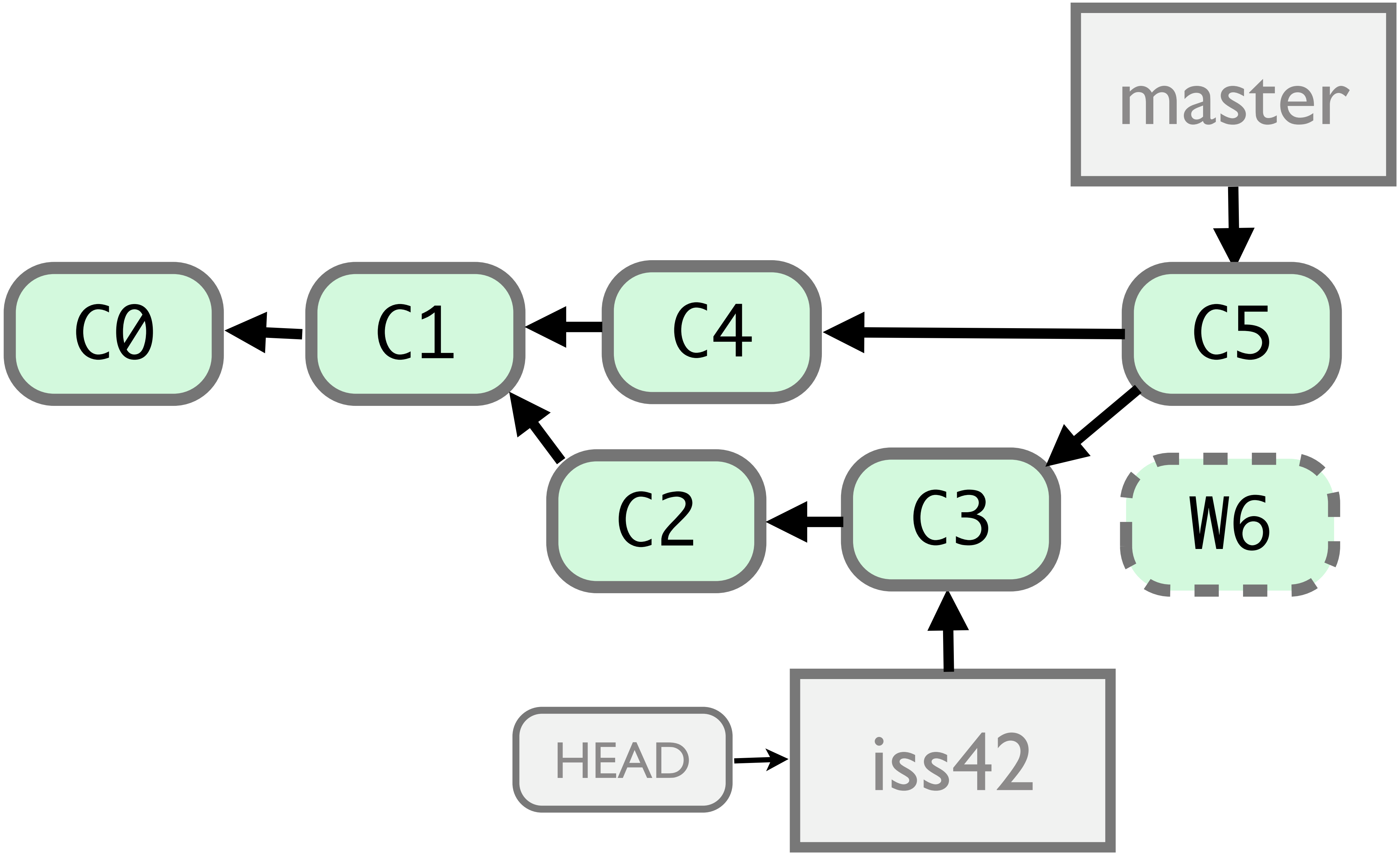
- clone the code that is in production
- create a branch for issue #53 (iss53)
- work for 10 minutes
- someone asks for a hotfix for issue #102
- checkout 'production'
- create a branch (iss102)
- fix the issue
- checkout 'production', merge 'iss102'
- push 'production'
- checkout 'iss53' and keep working



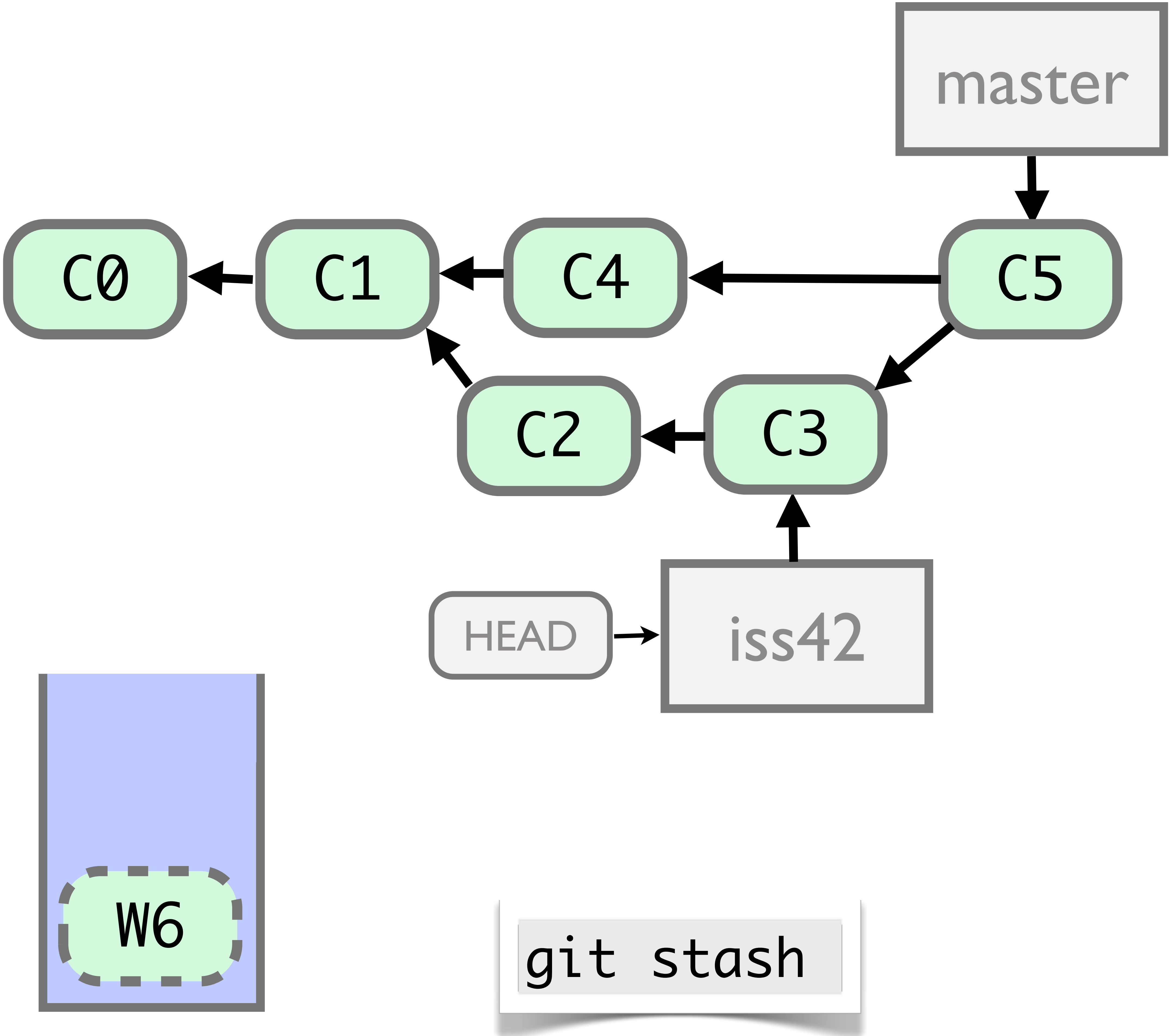
```
git checkout iss42
```



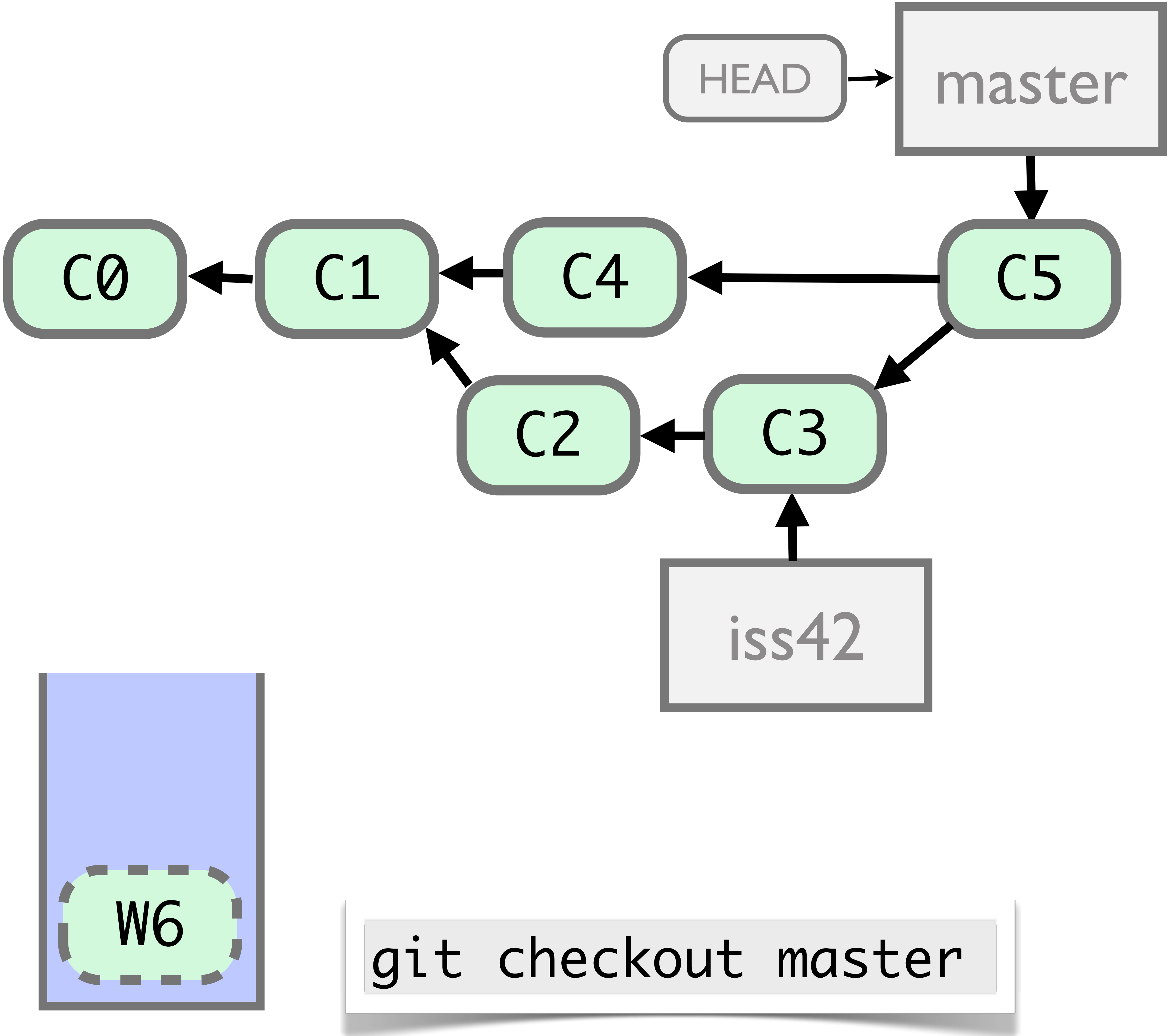
Starts coding



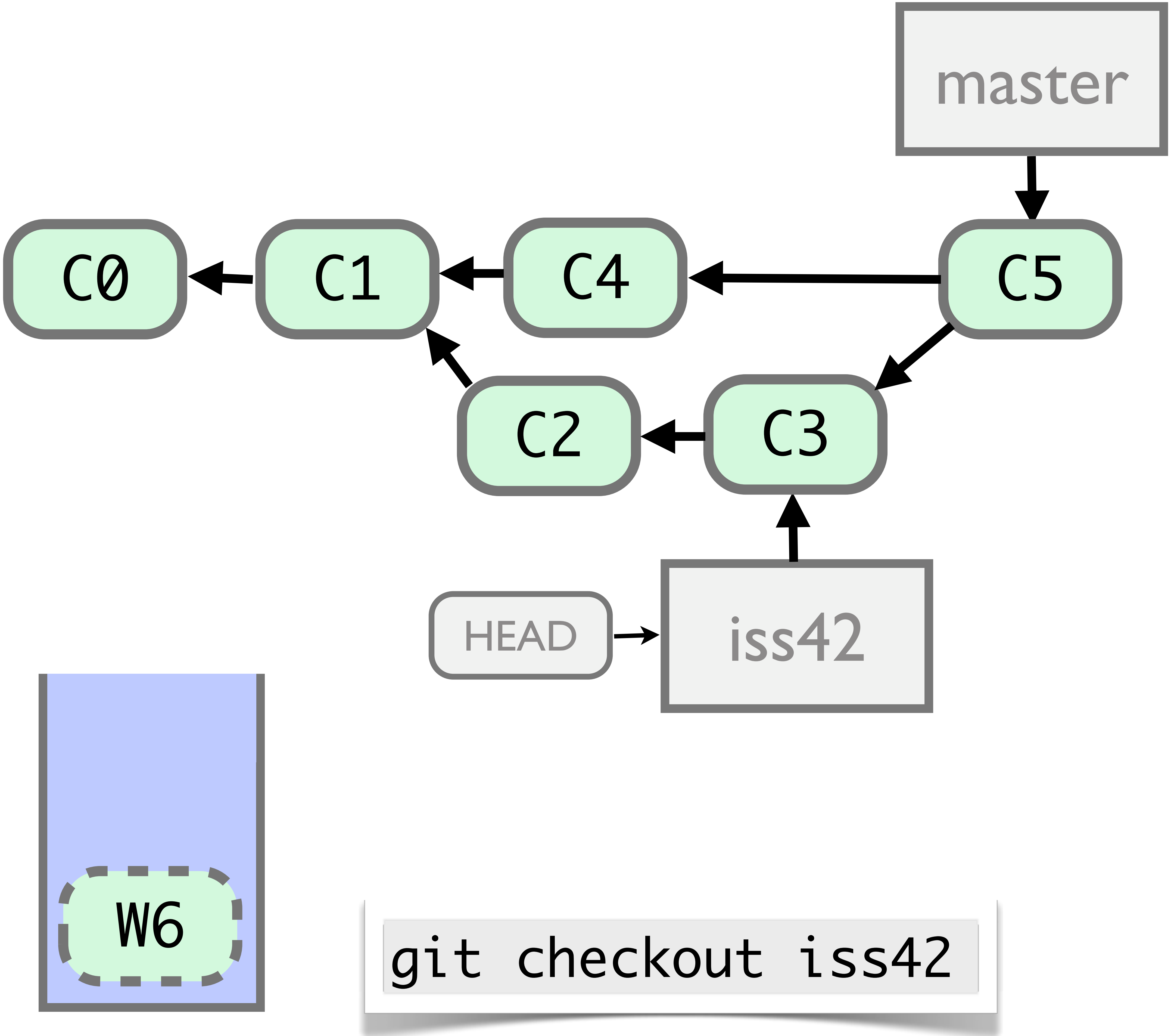
Your boss
come in!



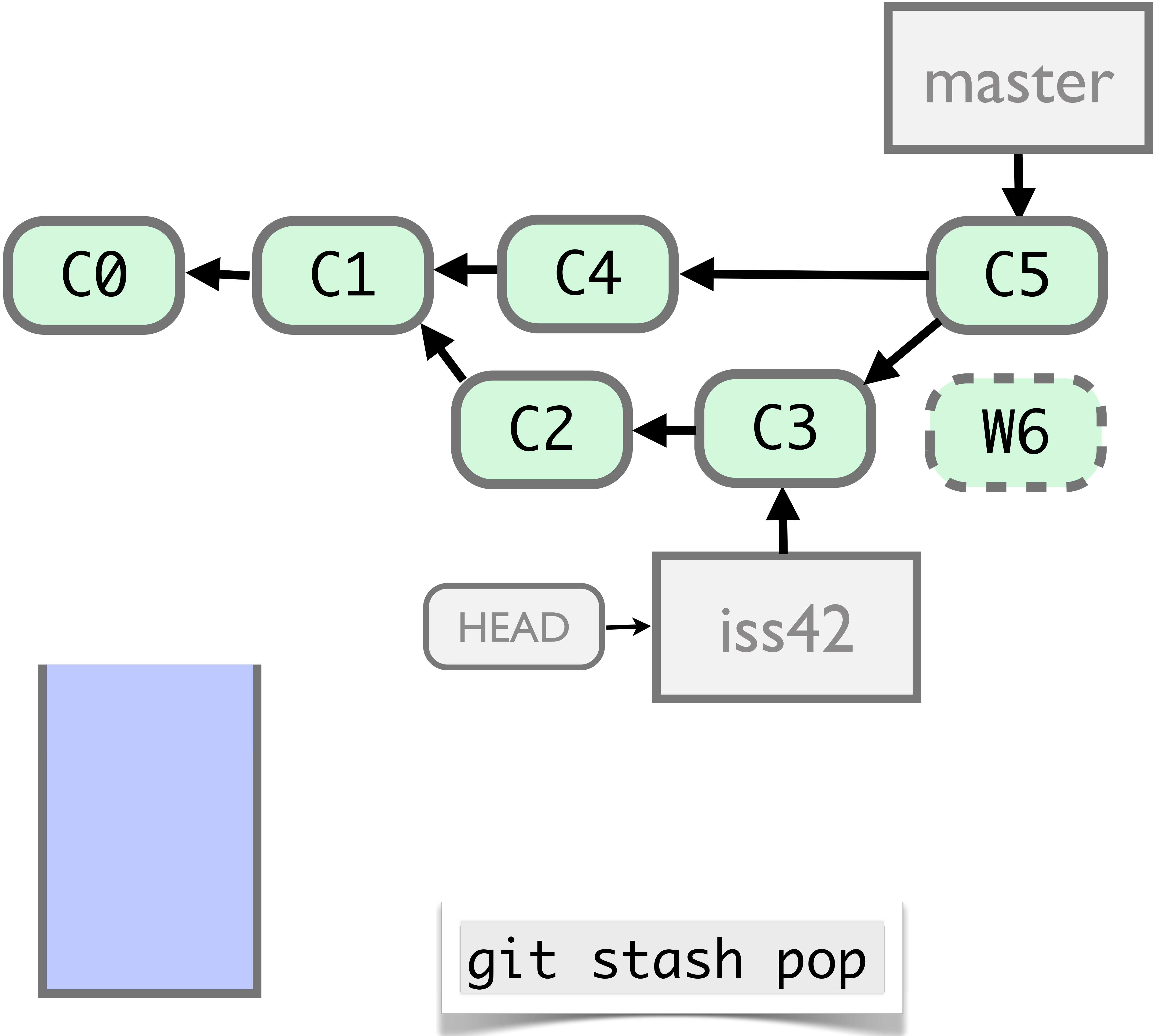
```
git stash
```



```
git checkout master
```



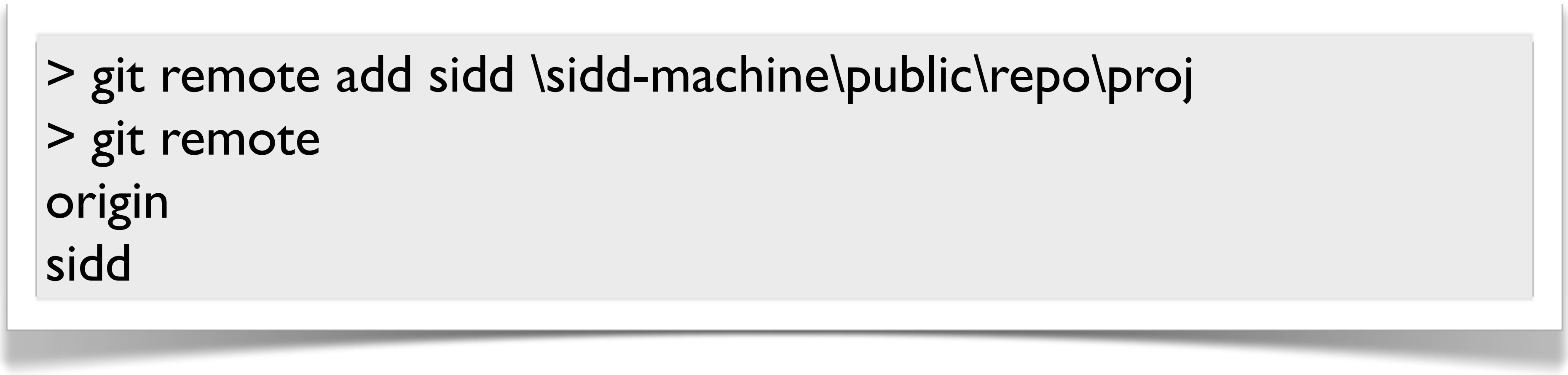
```
git checkout iss42
```



```
git stash pop
```

GIT commands

- git remote
- git remote add remote_name server_url



```
> git remote add sidd \sidd-machine\public\repo\proj
> git remote
origin
sidd
```

- git remote rm remote_name

GIT and GRVM

GIT and GRVM

- Every project must have a Versioning Control System (VCS)

GIT and GRVM

- Every project must have a Versioning Control System (VCS)
- From now on, GRVM will start using GIT as the default VCS

GIT and GRVM

- Every project must have a Versioning Control System (VCS)
- From now on, GRVM will start using GIT as the default VCS
- To maximize the efficiency of any VCS, some rules must be followed by every team member

GIT and GRVM

GIT and GRVM

- Rule #1 - The master branch is your blessed branch, never put dirty code in it

GIT and GRVM

- Rule #1 - The master branch is your blessed branch, never put dirty code in it
- Rule #2 - The master branch is your blessed branch, never put dirty code in it

GIT and GRVM

GIT and GRVM

- If possible, create a new branch for every new issue/feature

GIT and GRVM

- If possible, create a new branch for every new issue/feature
- When you arrive at GRVM, do a fetch

GIT and GRVM

- If possible, create a new branch for every new issue/feature
- When you arrive at GRVM, do a fetch
- At least, make a push of your working branch before you leave GRVM, EVERYDAY

GIT and GRVM

- If possible, create a new branch for every new issue/feature
- When you arrive at GRVM, do a fetch
- At least, make a push of your working branch before you leave GRVM, EVERYDAY
- At least once a week, merge master into your working branch

GIT and GRVM

GIT and GRVM

- For every stable release, make a TAG using a revision number (repo_name-X.Y)

GIT and GRVM

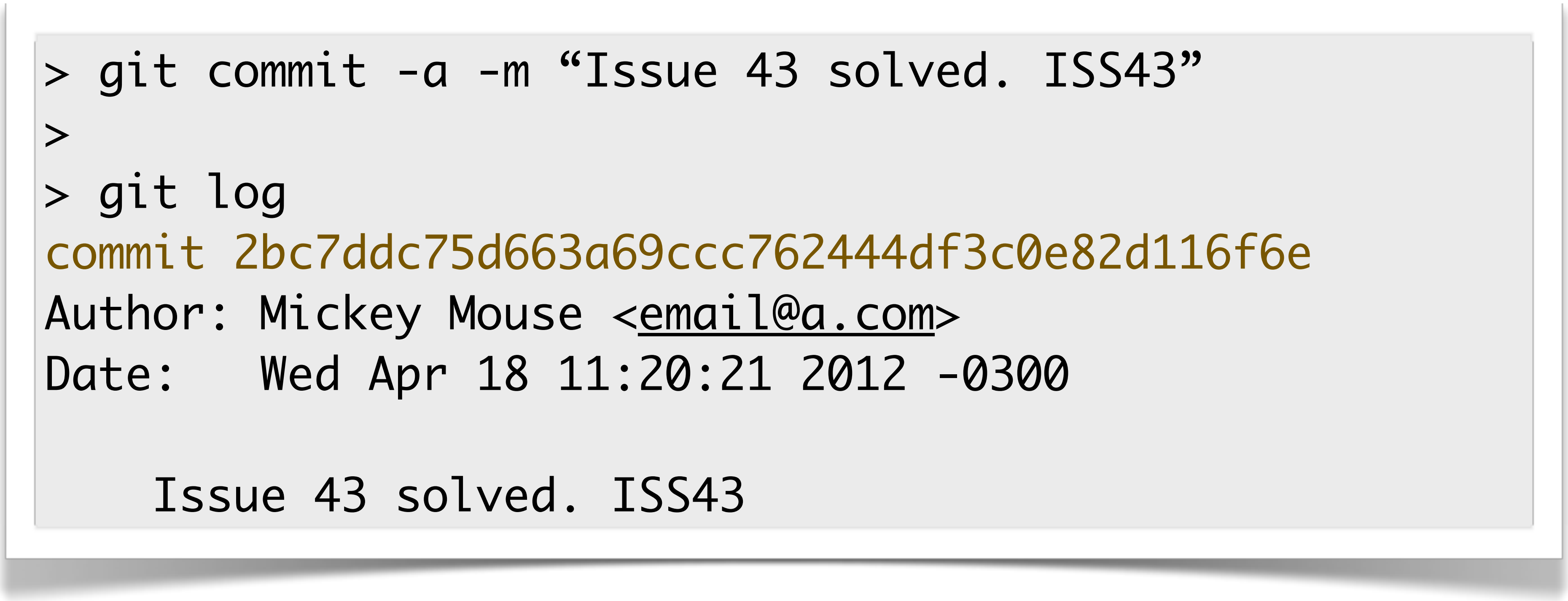
- For every stable release, make a TAG using a revision number (repo_name-X.Y)
- Commit your branch whenever you want, but if your code is not compiling, put a “[DIRTY]” at the beginning of your commit message

GIT and GRVM

- For every stable release, make a TAG using a revision number (repo_name-X.Y)
- Commit your branch whenever you want, but if your code is not compiling, put a “[DIRTY]” at the beginning of your commit message
- When your issue is OK, merge it into the master branch

GIT and GRVM

- In every commit message, put the issue code from JIRA (if your project is being managed through it)



```
> git commit -a -m “Issue 43 solved. ISS43”
>
> git log
commit 2bc7ddc75d663a69ccc762444df3c0e82d116f6e
Author: Mickey Mouse <email@a.com>
Date:   Wed Apr 18 11:20:21 2012 -0300

    Issue 43 solved. ISS43
```

GIT and GRVM

GIT and GRVM

- When your issue/feature is done:

GIT and GRVM

- When your issue/feature is done:
  - merge your features into master

GIT and GRVM

- When your issue/feature is done:
  - merge your features into master
  - remove your branch

GIT and GRVM

- When your issue/feature is done:
  - merge your features into master
  - remove your branch
  - remove your branch from remote

GIT and GRVM

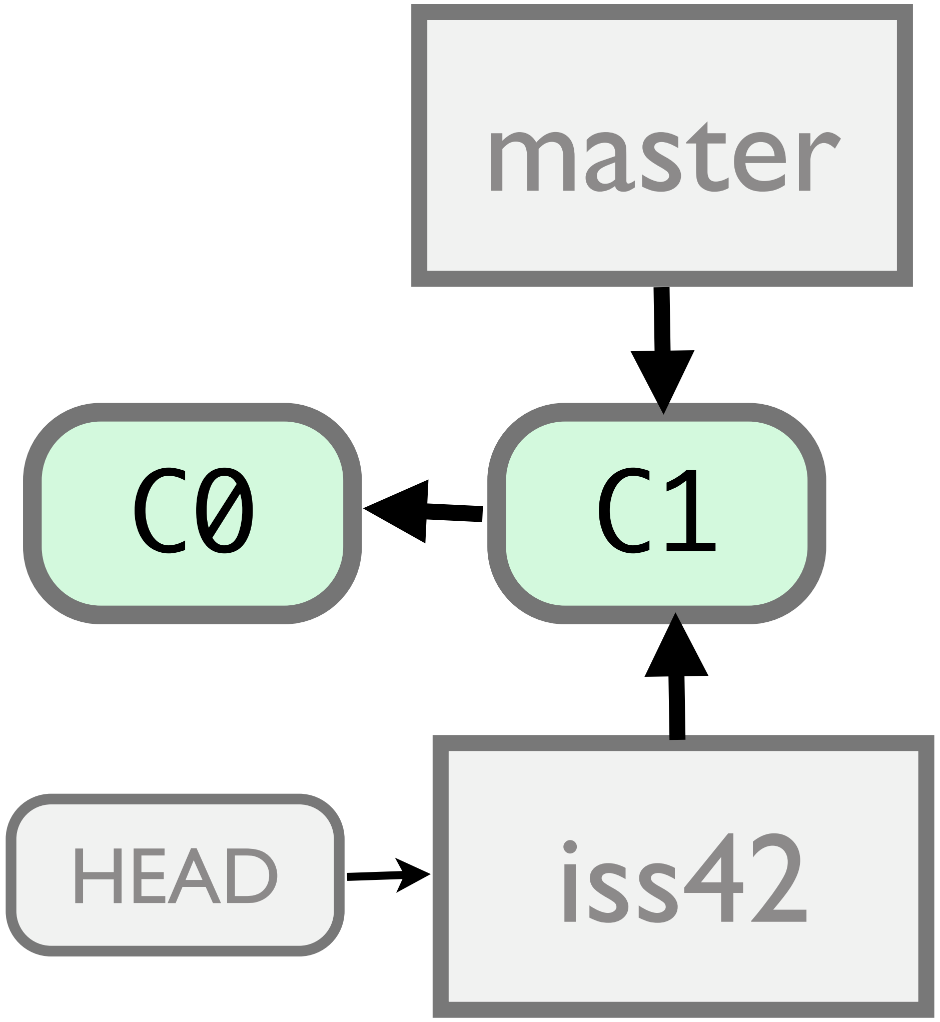
- When your issue/feature is done:
  - merge your features into master
  - remove your branch
  - remove your branch from remote
  - push master to remote

GIT and GRVM

- When your issue/feature is done:
  - merge your features into master
  - remove your branch
  - remove your branch from remote
  - push master to remote

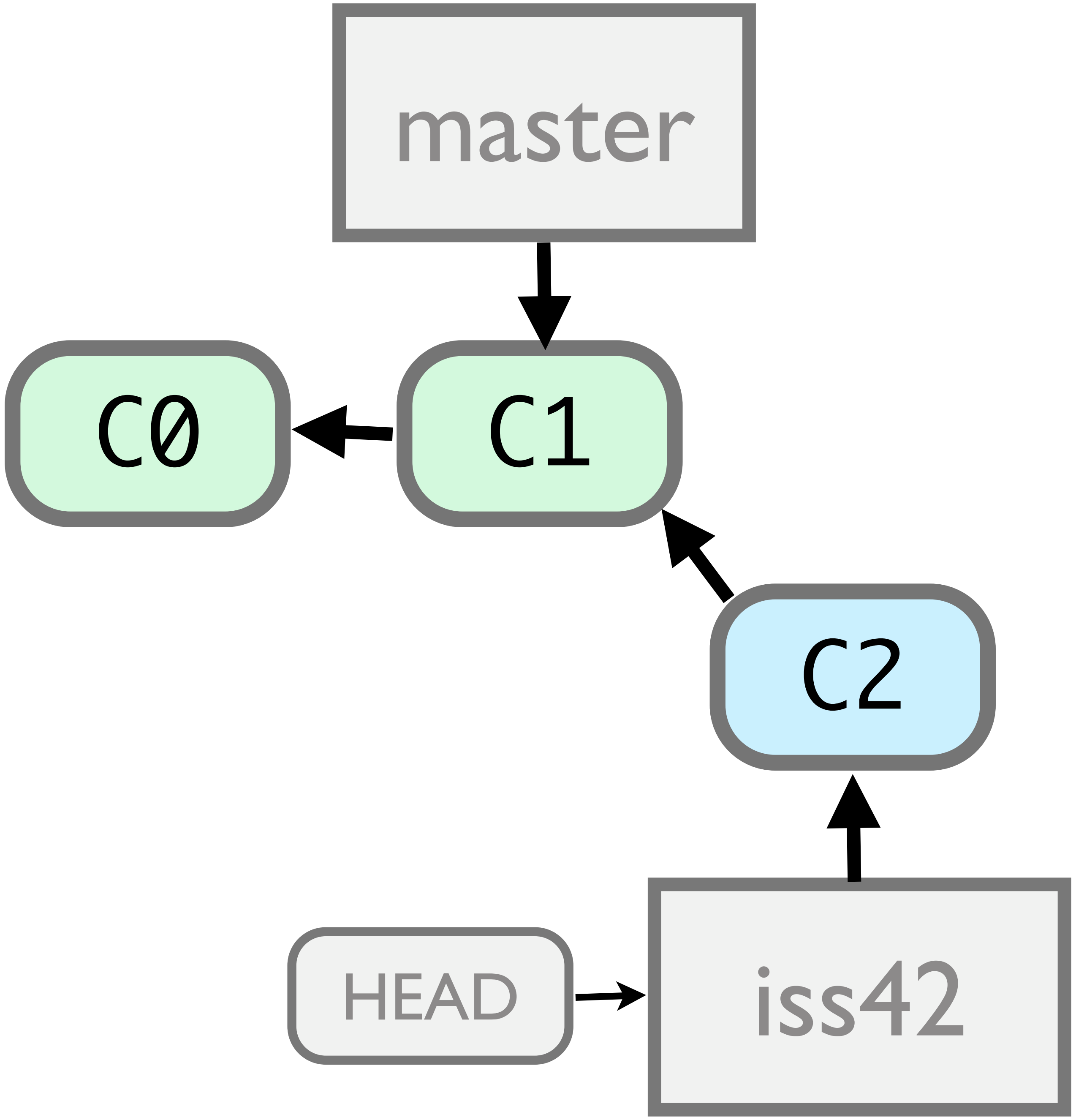
```
> git commit -a -m “hotfix 35 done.  ISS35”
> git checkout master
> git merge --no-ff iss35
> git branch -d iss35
> git push origin :iss35
```

GIT and GRVM



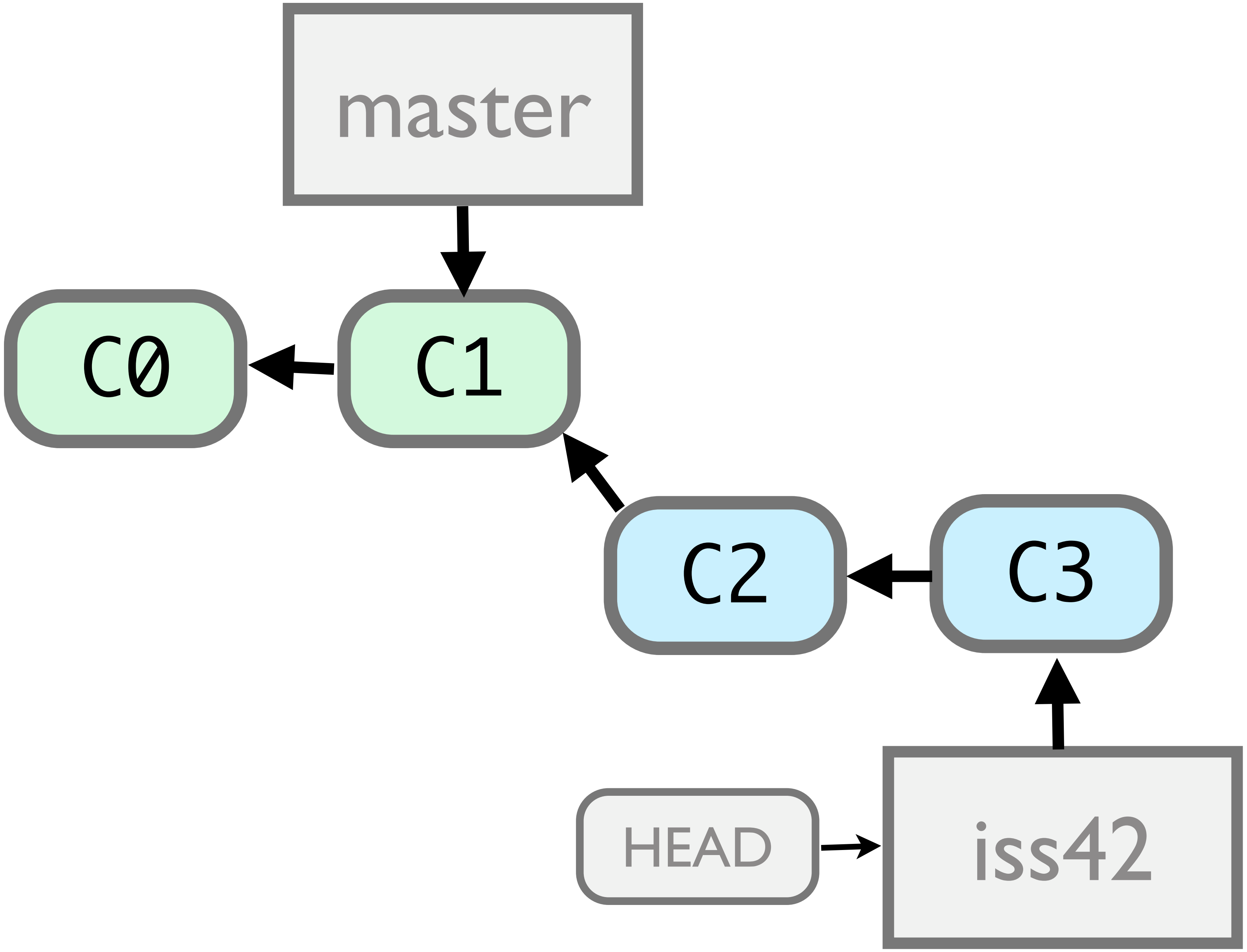
```
> git branch iss42
```

GIT and GRVM



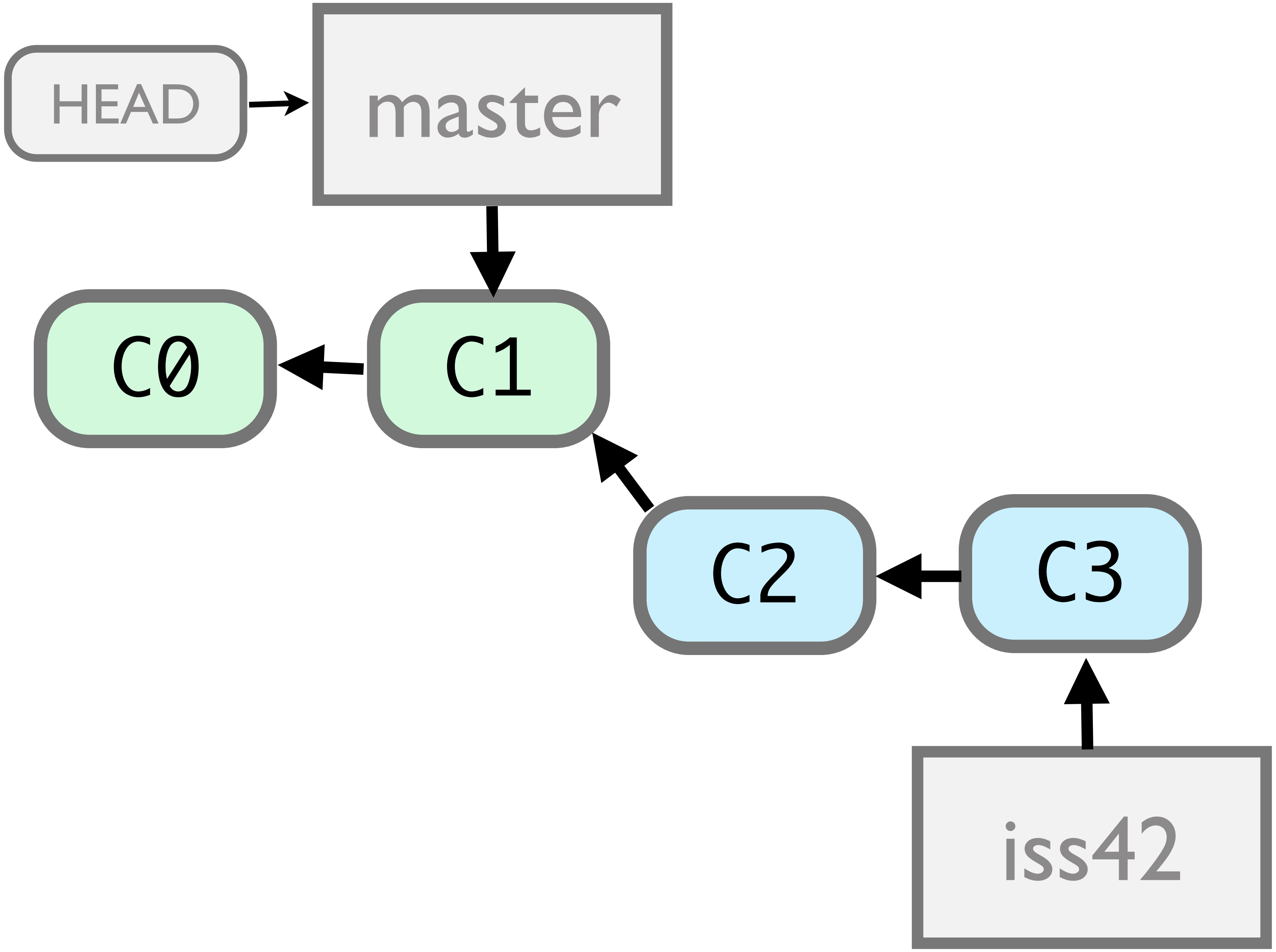
```
> git commit
```

GIT and GRVM



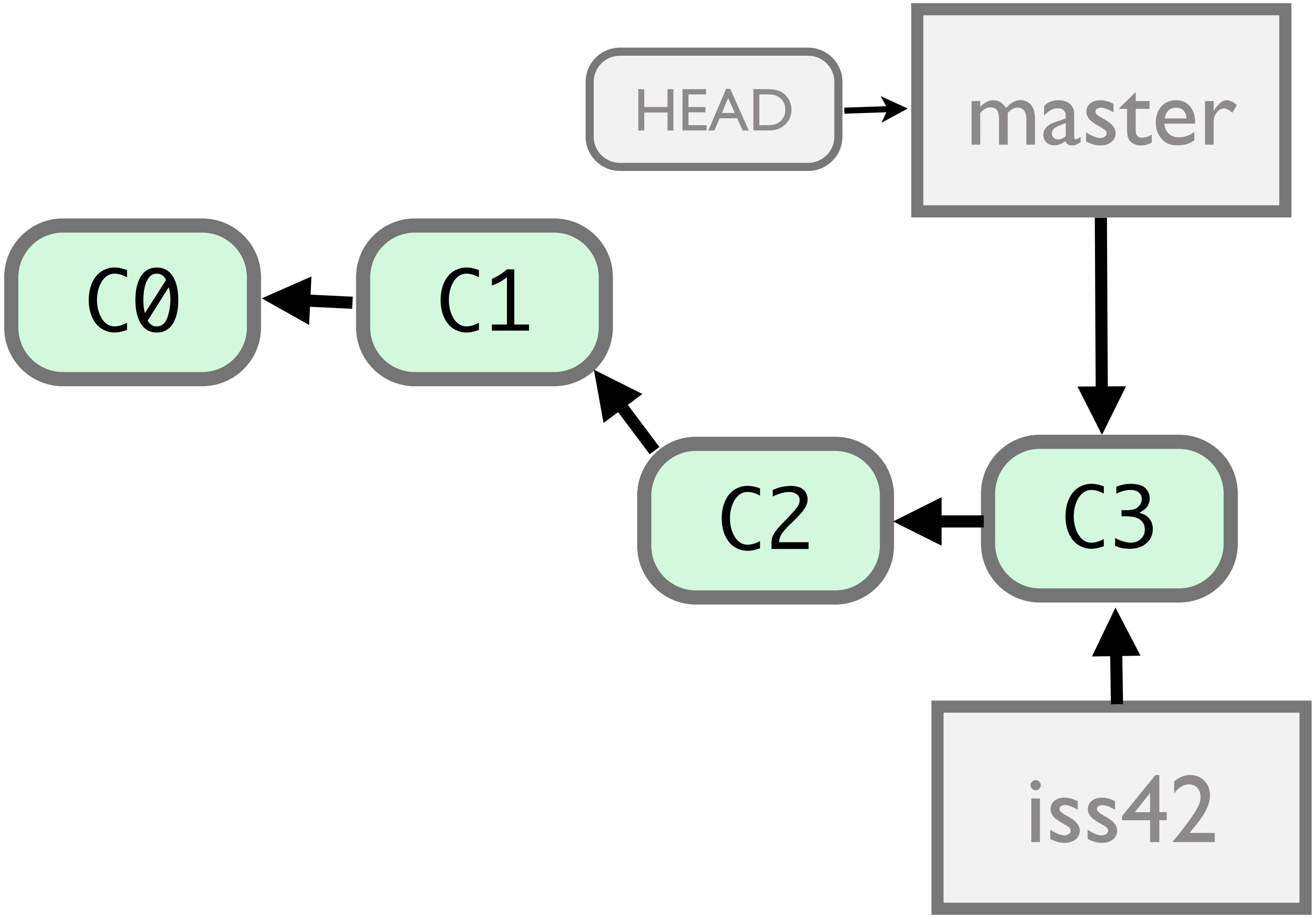
```
> git commit
```

GIT and GRVM



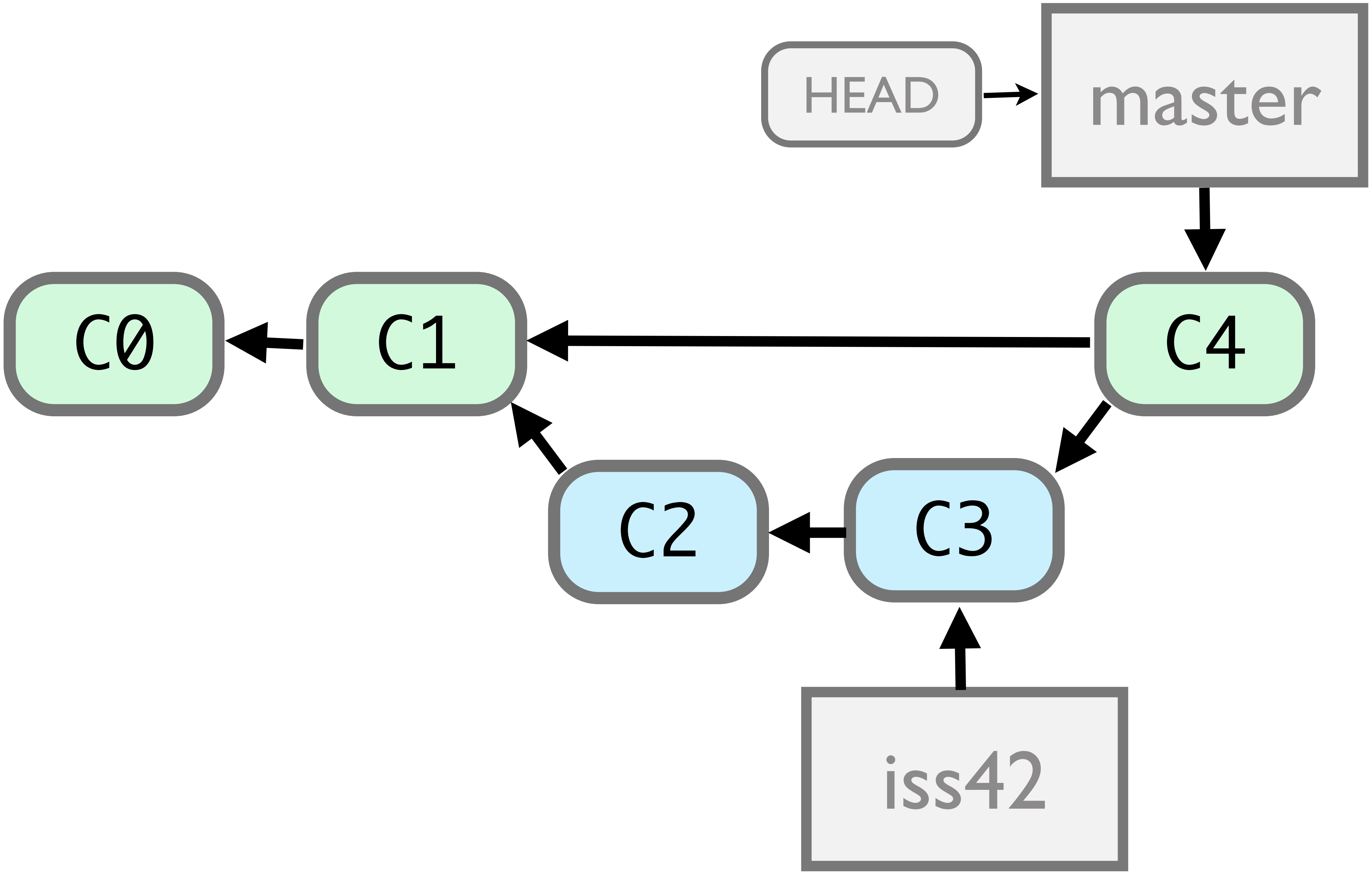
```
> git checkout master
```

GIT and GRVM



```
> git merge iss42
```

GIT and GRVM



```
> git merge --no-ff iss42
```

GIT is huge...

- git daemon
- git diff
- git blame
- git bisect
- git push -u / git branch --track
- git mergetool
- git prune
- git rebase
- git reset
- git show
- gitk
- git instaweb
- git archive
- git gc

Resources

- http://whygitisbetterthanx.com/
- http://git-scm.com/
- http://progit.org/
- http://book.git-scm.com/
- http://help.github.com/git-cheat-sheets/