การบริหารจัดการ Source Code ด้วย

สถาบัน ไอเอ็มซี

วันจันทร์ที่ 25 – วันอังคารที่ 26 ตุลาคม พ.ศ. 2564

สมเกียรติ ปุ๋ยสูงเนิน (ปุ๋ย)

Developer + Agile Coach + Technical Coach

Agile Practitioner

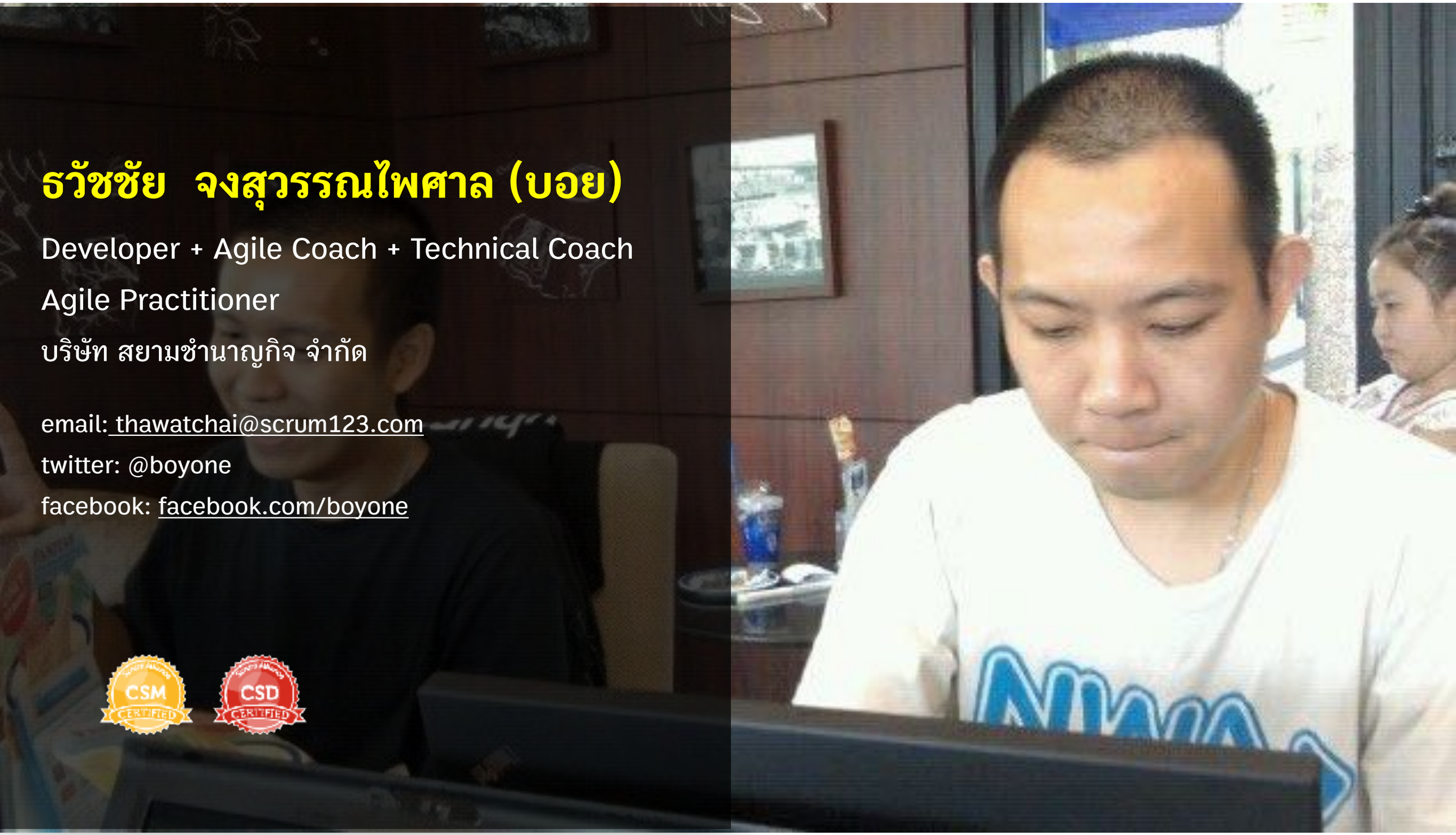
บริษัท สยามชำนาญกิจ จำกัด

Owner + Blogger @ somkiat.cc

email: somkiat@scrum123.com

twitter: @somkiat

facebook: facebook.com/somkiatspns



ธวัชชัย จงสุวรรณไพศาล (บอย)

Developer + Agile Coach + Technical Coach

Agile Practitioner

บริษัท สยามชำนาญกิจ จำกัด

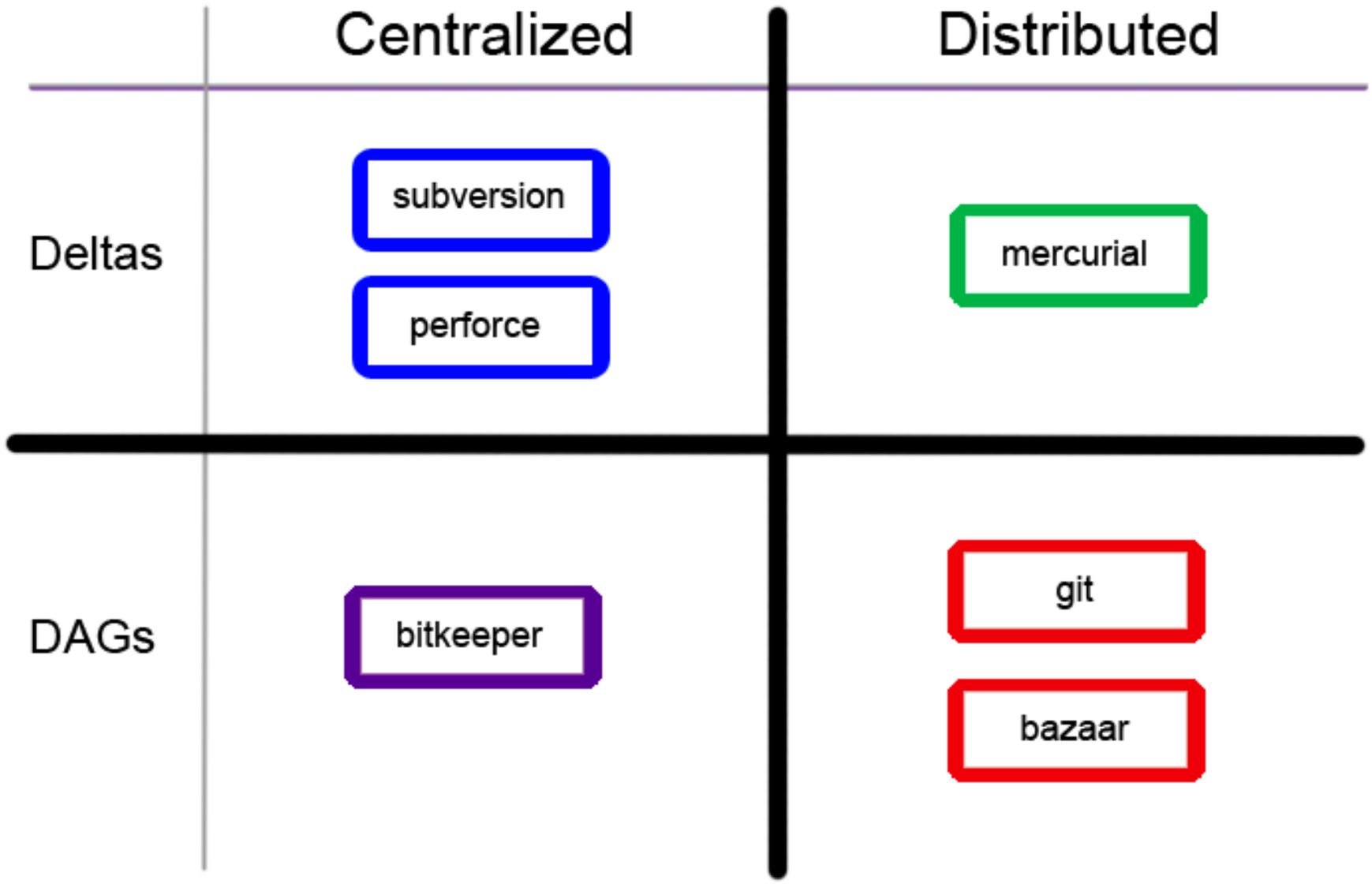
email: thawatchai@scrum123.com

twitter: @boyone

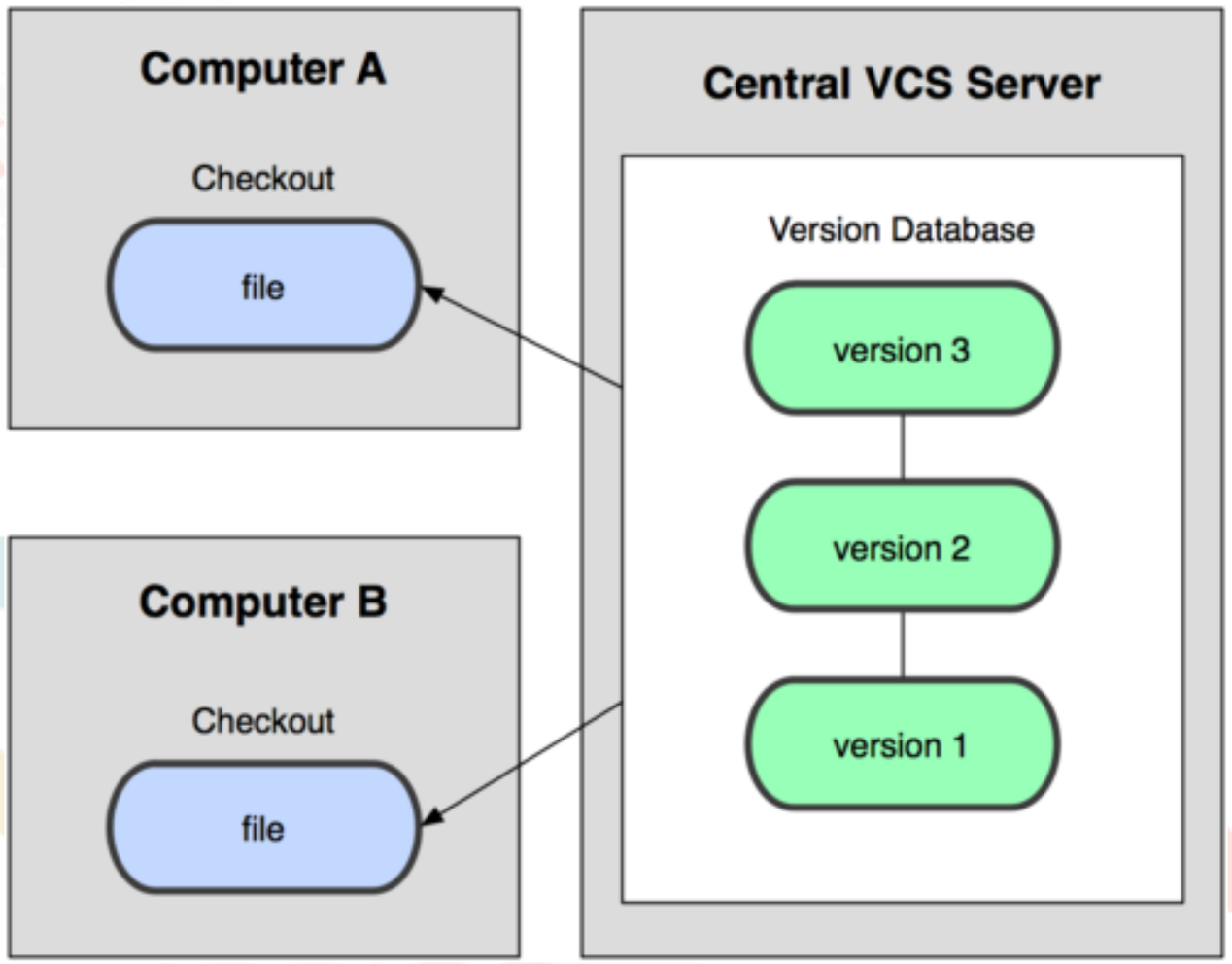
facebook: facebook.com/boyone


KEEP
CALM
AND
GIT PUSH
ORIGIN MAIN

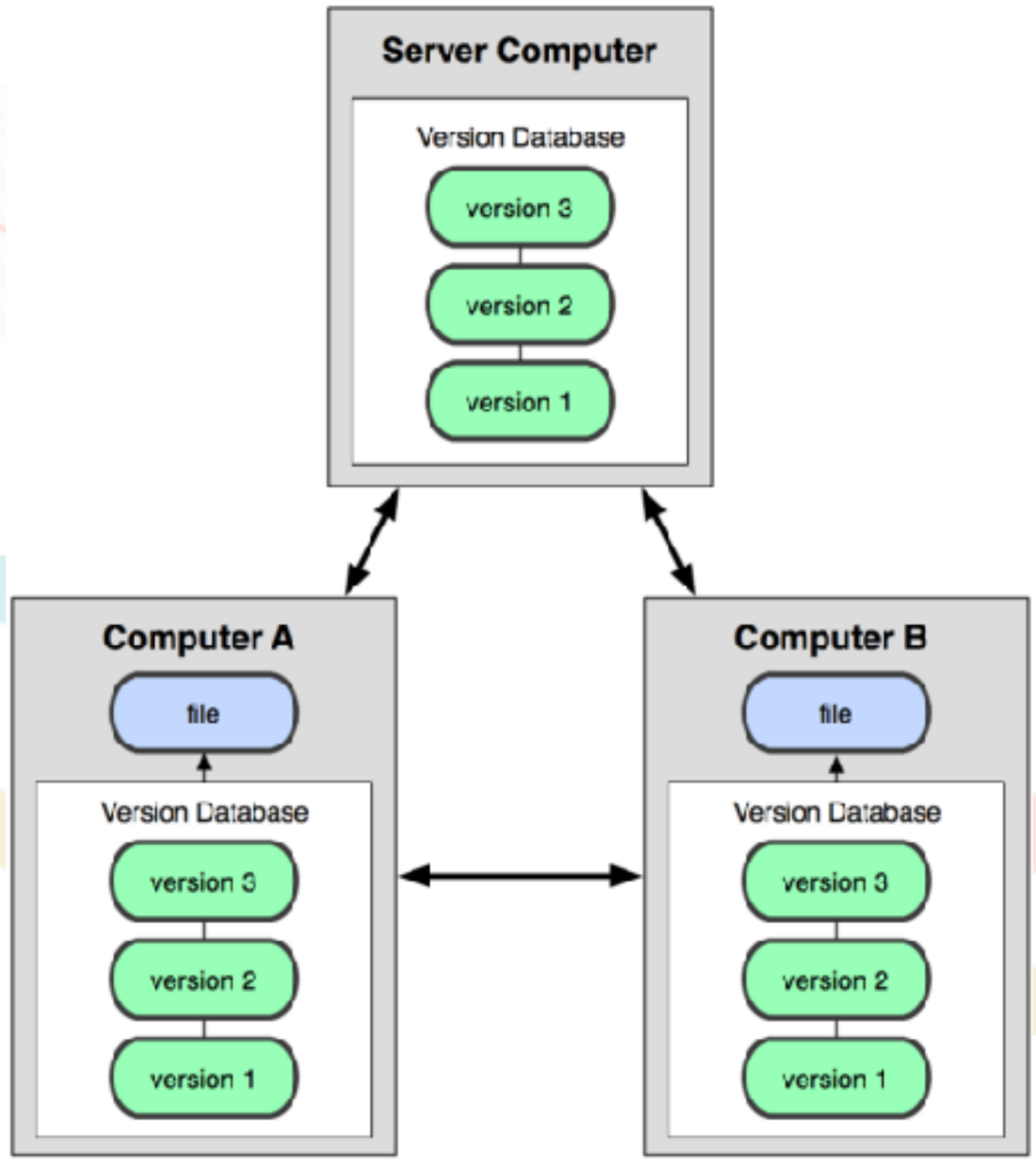
Version Control



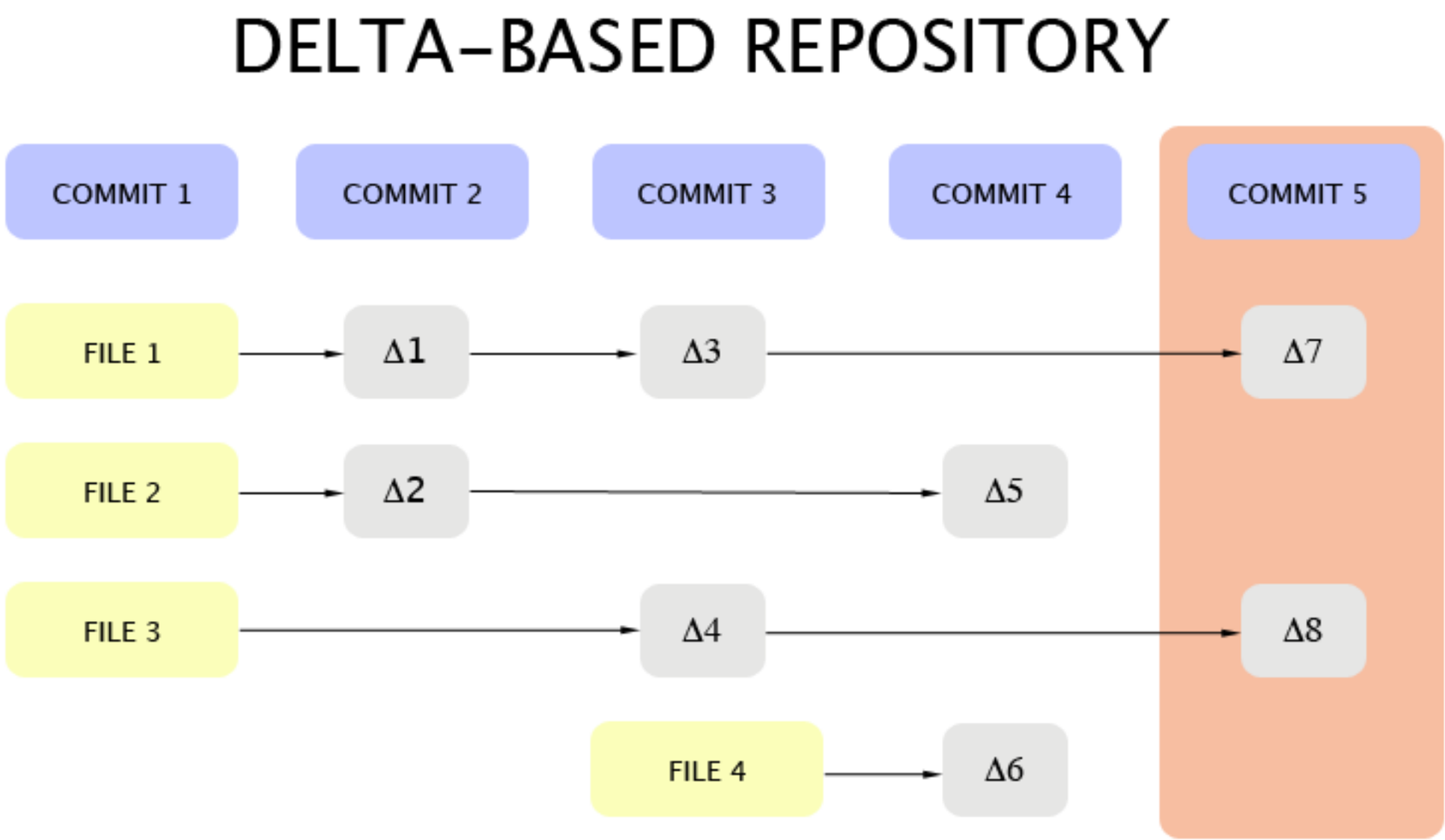
Centralized Version Control



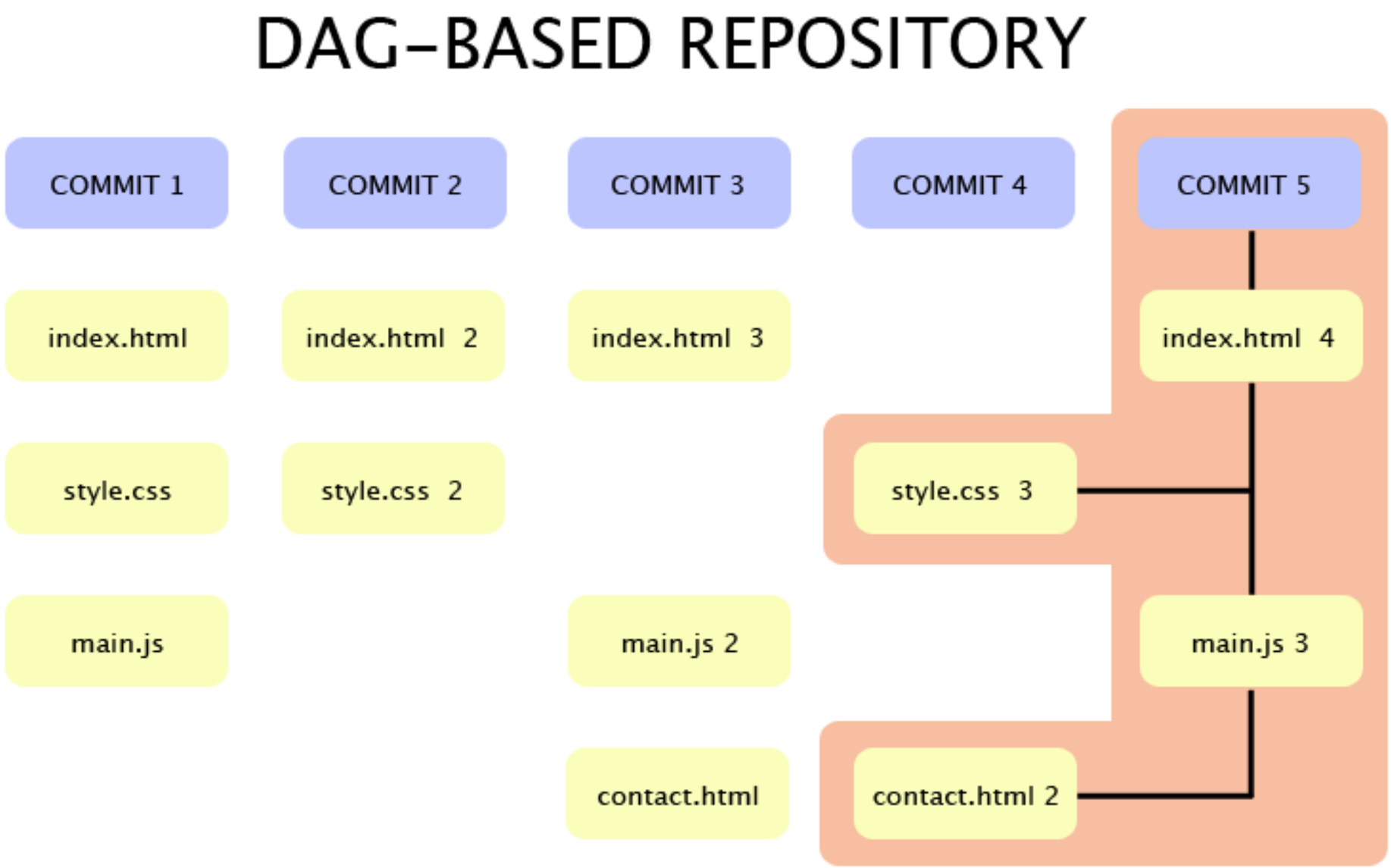
Distributed Version Control



Delta Repository



DAG Repository



เป้าหมายของการออกแบบ Git

Speed

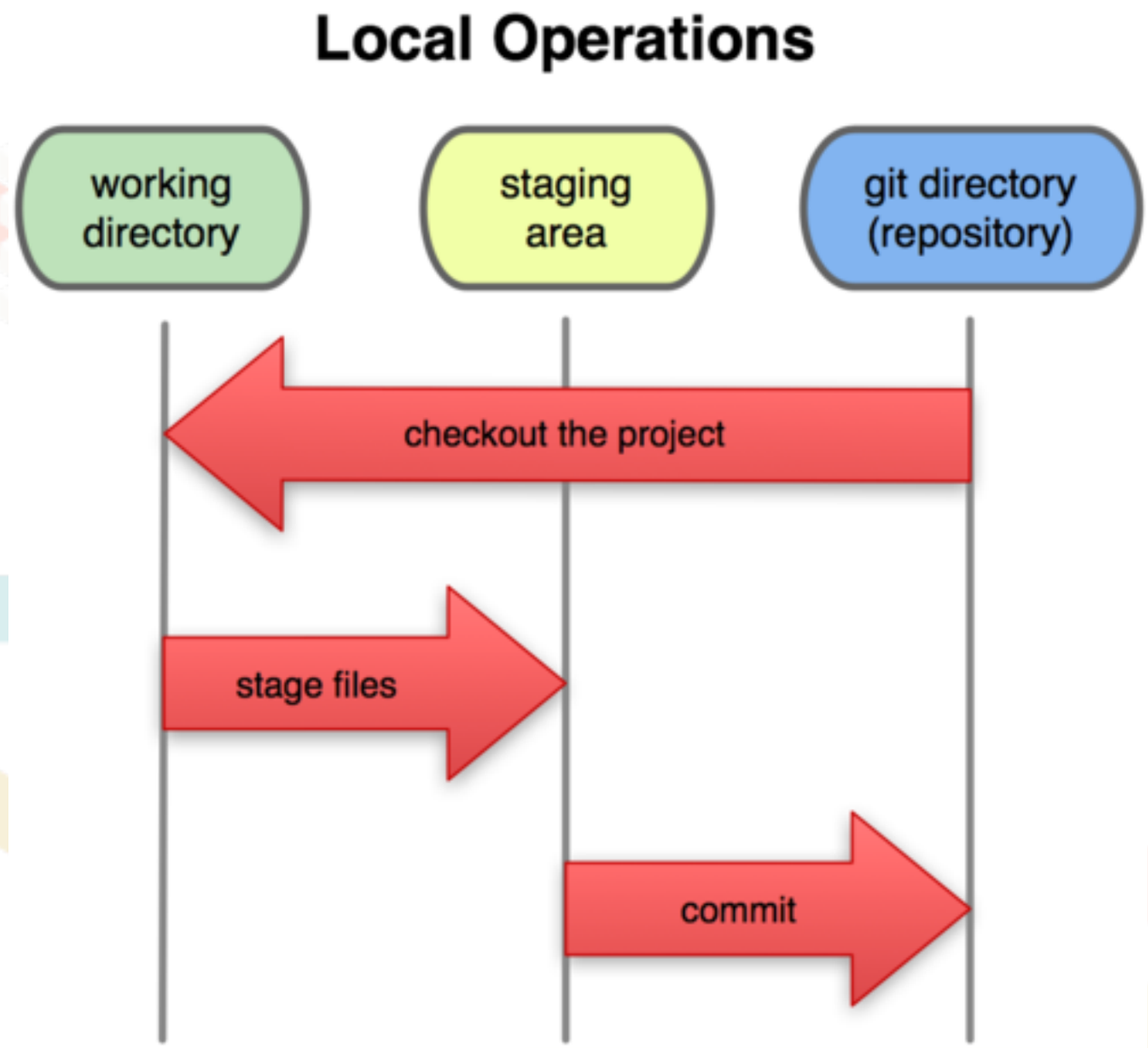
Simple design

Support for many parallel branches

Fully distributed

To handle large project like Linux kernel

Git Workflow



Who use Git

Companies & Projects Using Git

Google facebook Microsoft twitter LinkedIn NETFLIX PostgreSQL

android RAILS Qt GNOME eclipse

Install Git

http://git-scm.com/

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	Mac or Linux
Select git bash	Open Terminal

$mkdir *git-workspace*

$cd *git-workspace*

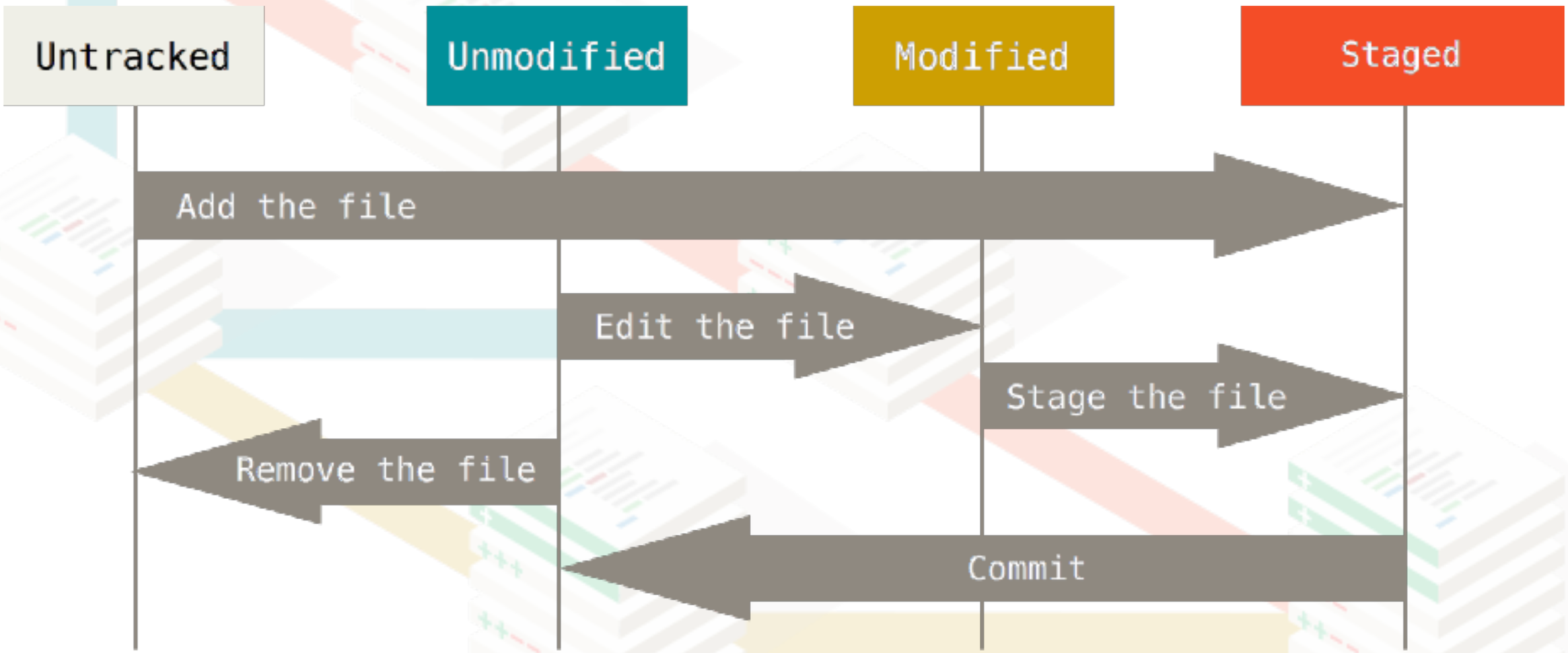
$git **init**

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
```

Life Cycle of File Status



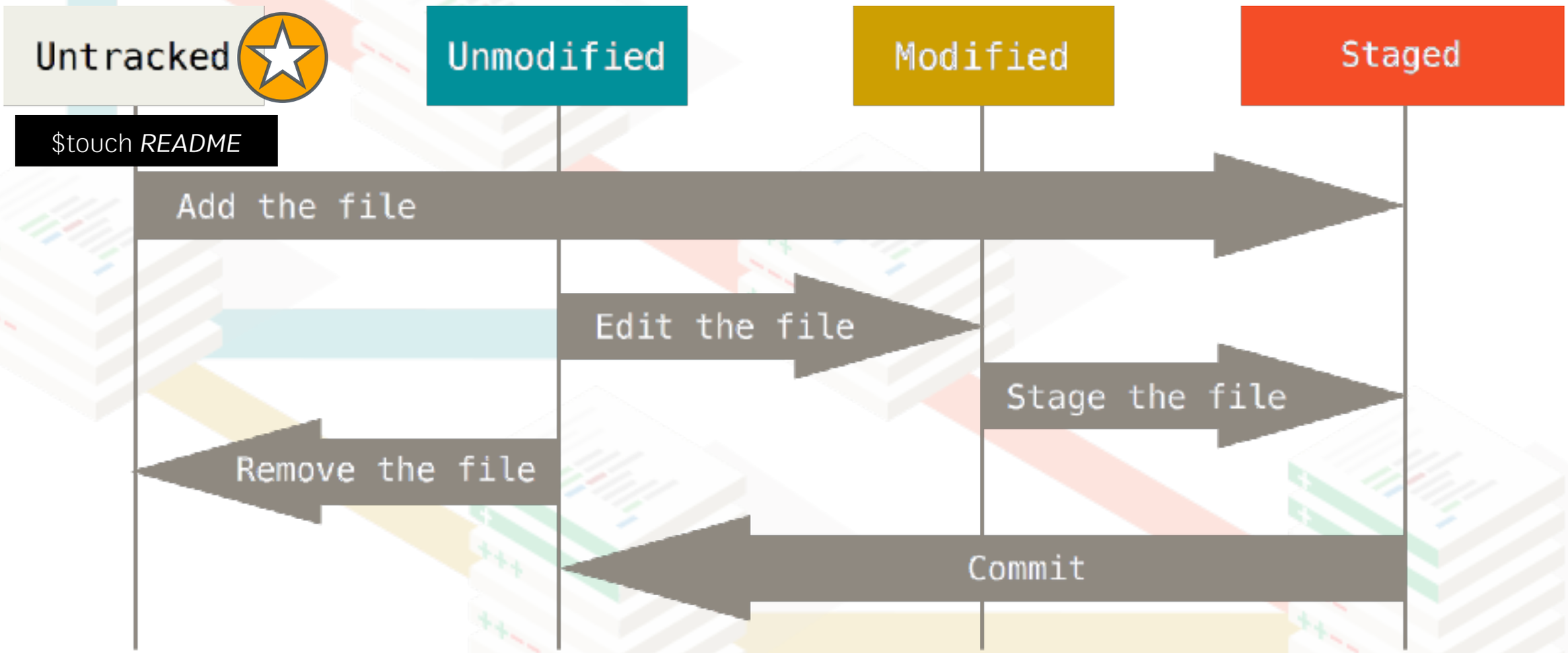
Create a file and check the status

$touch *README*

$git **status**

```
On branch main

No commits yet

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)
```

Life Cycle of File Status



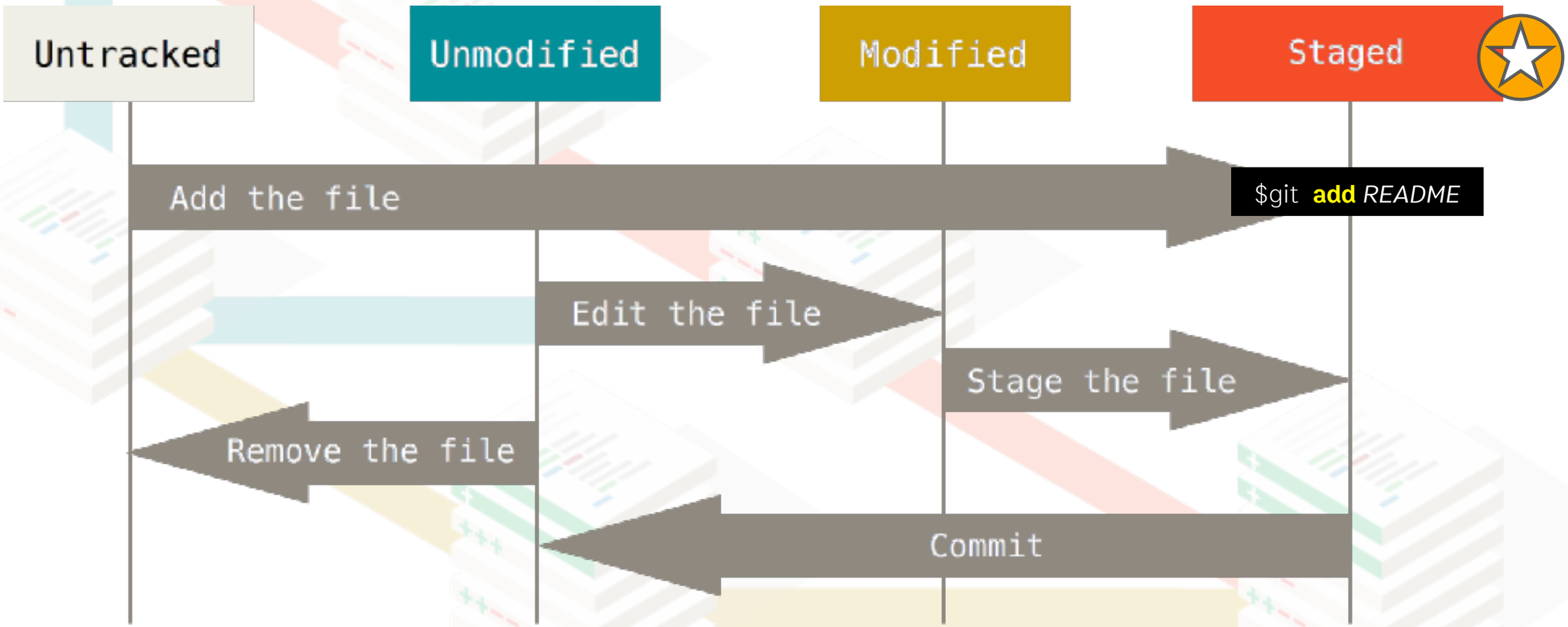
Add the file and check the status

$git **add** *README*

$git **status**

```
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README
```

Life Cycle of File Status



Commit your changes

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

Tips

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

$git **commit** -m “*Add README*”

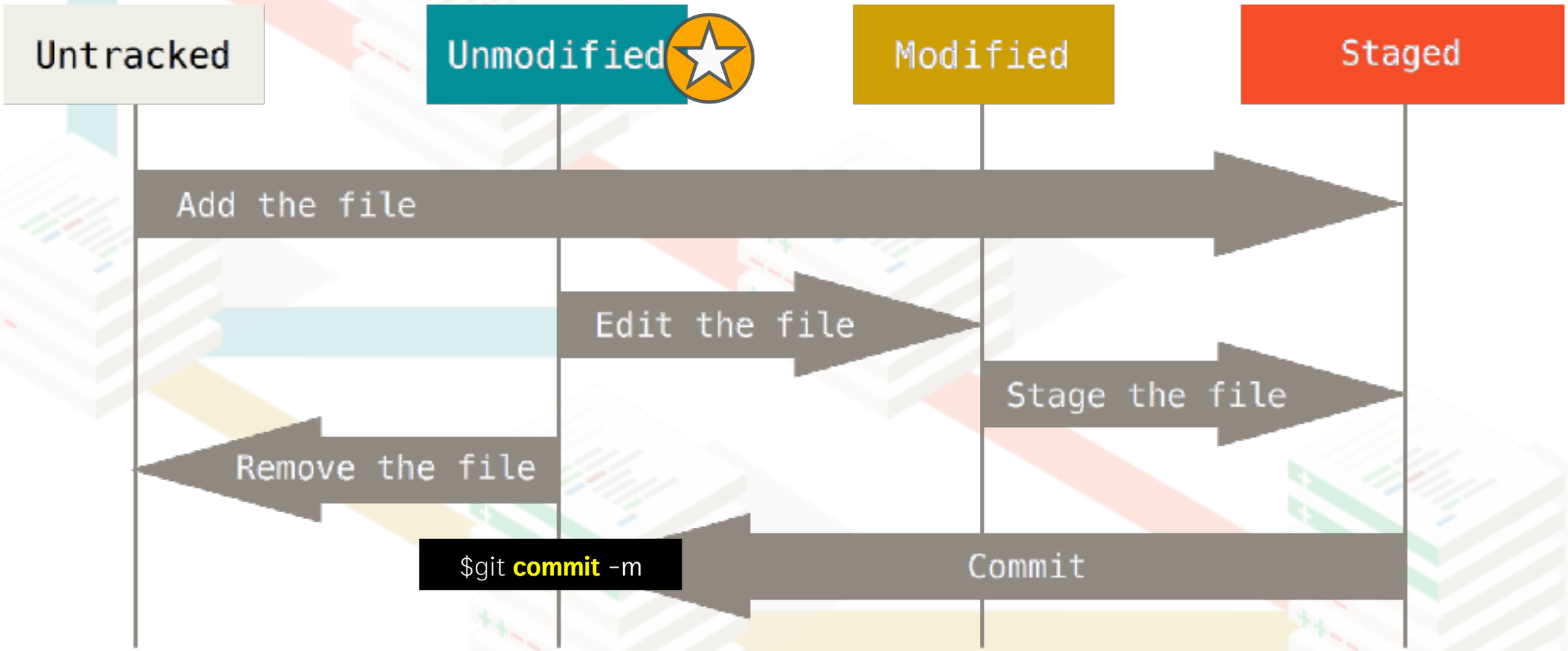
```
[main (root-commit) 779b411] Add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README
```

Check the status

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Life Cycle of File Status



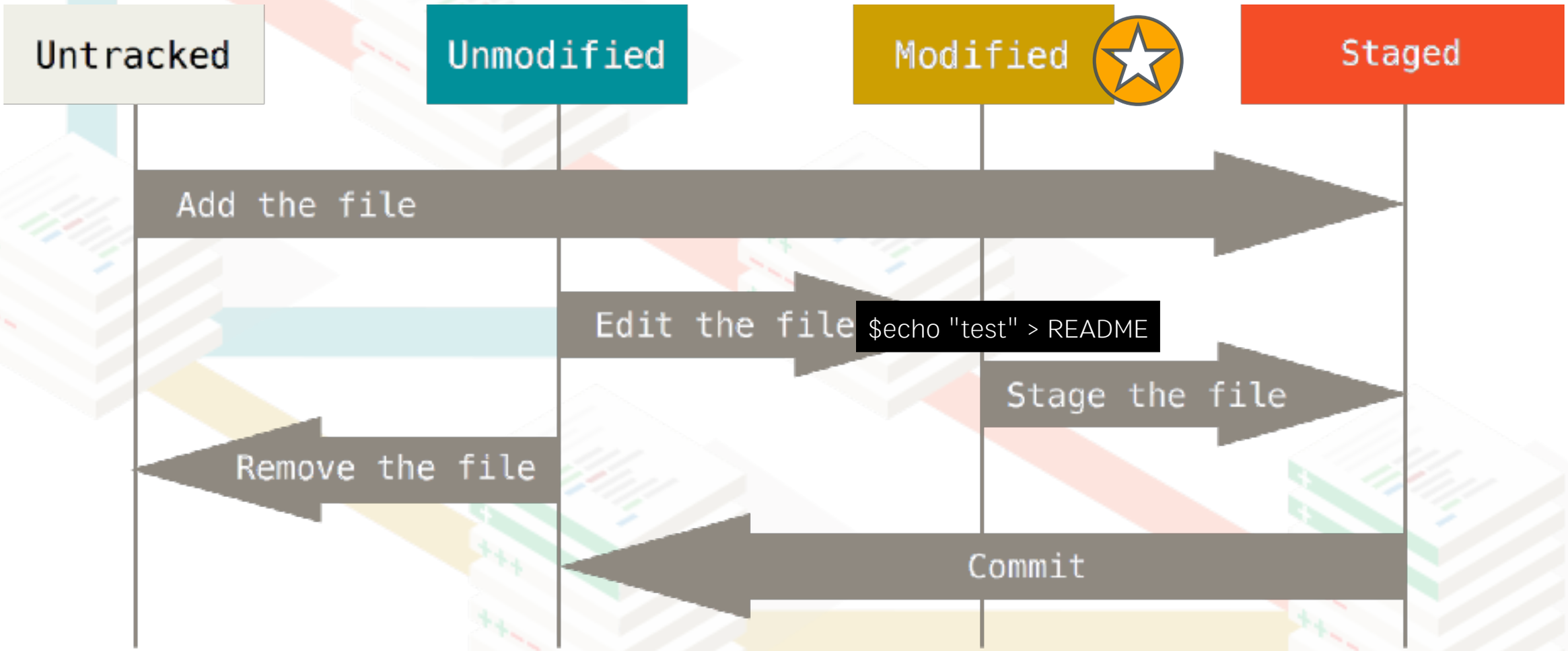
View staged change

$echo “test” > README

$git **status**

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README

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

Life Cycle of File Status



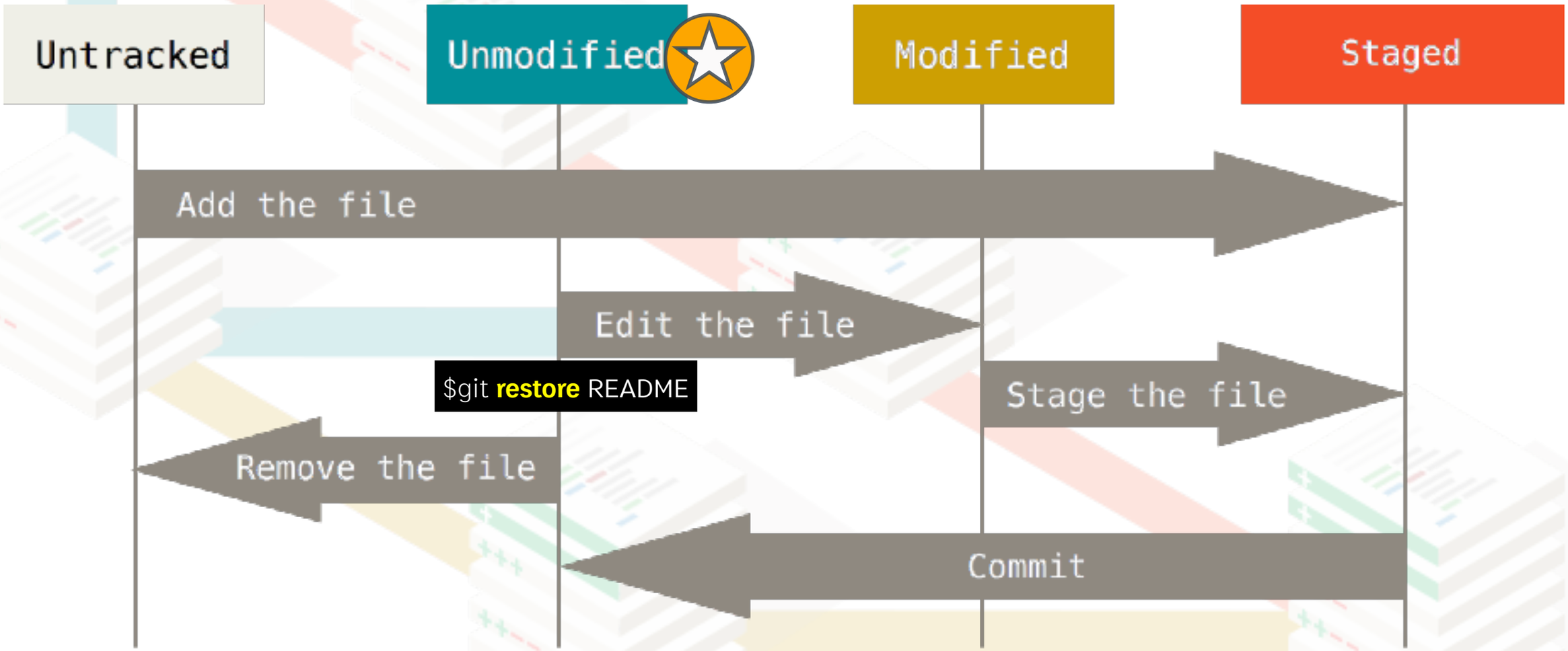
Restore staged

$git **restore** README

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Life Cycle of File Status



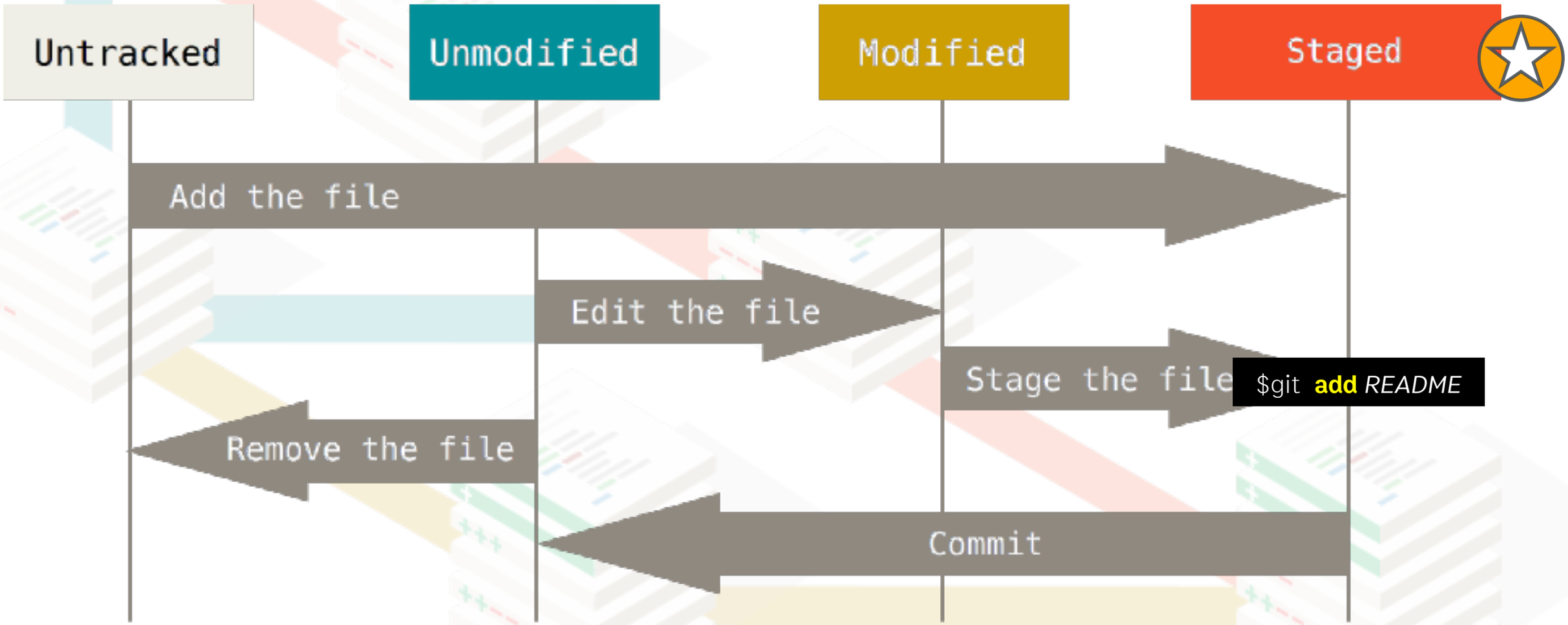
Add the file and check the status

$git **add** *README*

$git **status**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README
```

Life Cycle of File Status



Commit your changes

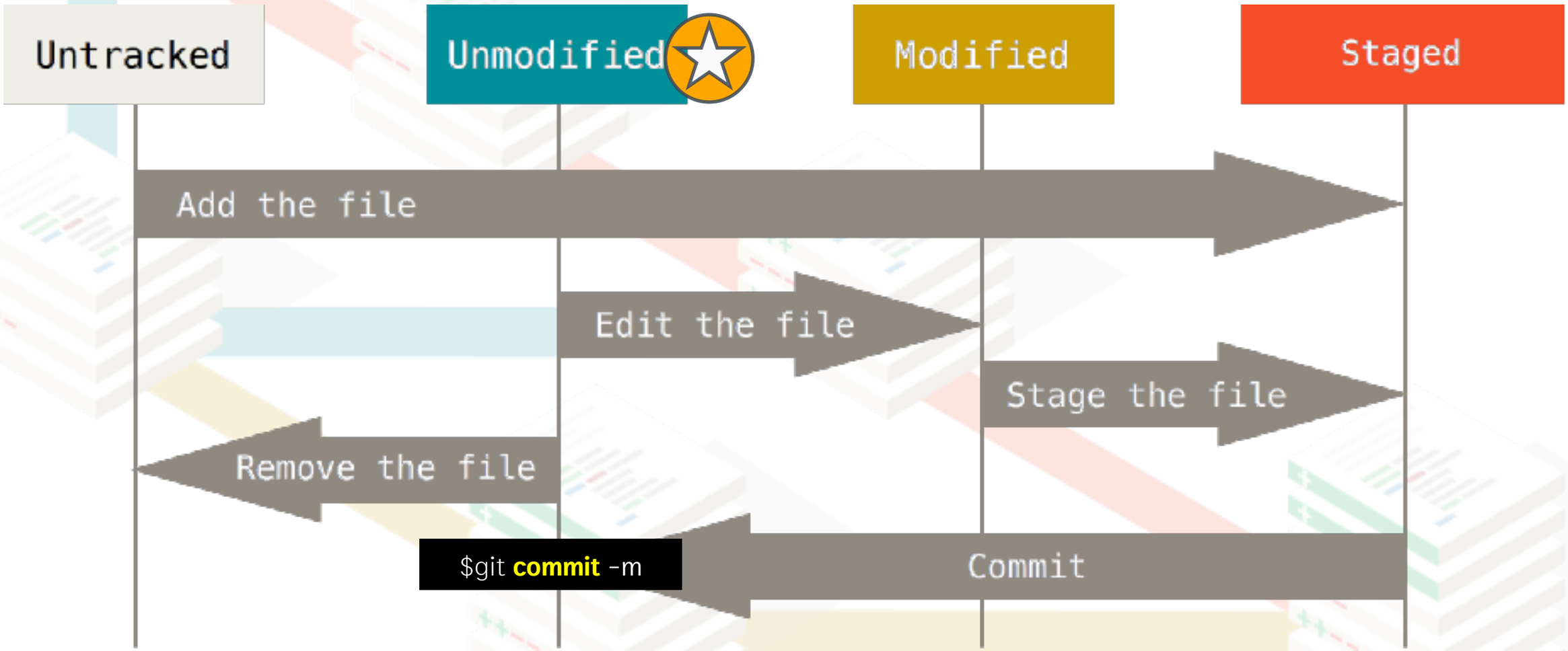
$git **commit** -m “Edit *README*”

```
[main 656bbdd] Edit README
 1 file changed, 1 insertion(+)
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Life Cycle of File Status



Ignore files and folders

$touch *note*

$mkdir *tmp*

$touch *tmp/file*

$git **status**

$touch *.gitignore*

$echo "note" > *.gitignore*

$echo "tmp/" >> *.gitignore*

Ignore files and folders

$git **status**

$git **add** *.gitignore*

$git **commit** -m "Add .gitignore"

```
[main e4c9eb3] Add .gitignore
 1 file changed, 2 insertions(+)
 create mode 100644 .gitignore
```

```
On branch main
nothing to commit, working tree clean
```

Move/Rename files and folders

$mv *README README.md*

$git **status**

```
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    README

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

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

Move/Rename files and folders

$git add *README.md*

$git **status**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    README
```

Move/Rename files and folders

$git add *README*

$git **status**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    README -> README.md
```

Move/Rename files and folders

$git commit -m "Rename README to README.md"

```
[main 6d17431] Rename README to README.md
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename README => README.md (100%)
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Move/Rename files and folders

$git mv *README.md README*

$git **status**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    README.md -> README
```

Move/Rename files and folders

$git commit -m "Rename README.md back to README"

```
[main bf96888] Rename README.md back to README
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename README.md => README (100%)
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Move/Rename files and folders

try to mv *untracked files* with *git mv*

$git mv *note mynote*

```
fatal: not under version control, source=note, destination=mynote
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Git restore

$echo "# hello" > README

$cat README

```
# hello
```

Modified

$git **status**

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README

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

Git restore

$git ***restore*** README

$cat README

```
test
```

Unmodified

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Git restore --staged

$echo "# hello" > README

$cat README

```
# hello
```

Modified

$git **add** README

Stage

$git **status**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README
```

Git restore --staged

$git **restore --staged** README

$cat README

Modified

```
# hello
```

$git **status**

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README

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

Git reset

$git **add** README

$git **commit -m "Edit README content"**

```
[main 3bb9951] Edit README content
 1 file changed, 1 insertion(+), 1 deletion(-)
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

Git reset

$git **reset <mode> <commit>**

$git **reset --soft <commit>**

$git **reset --mixed <commit>**

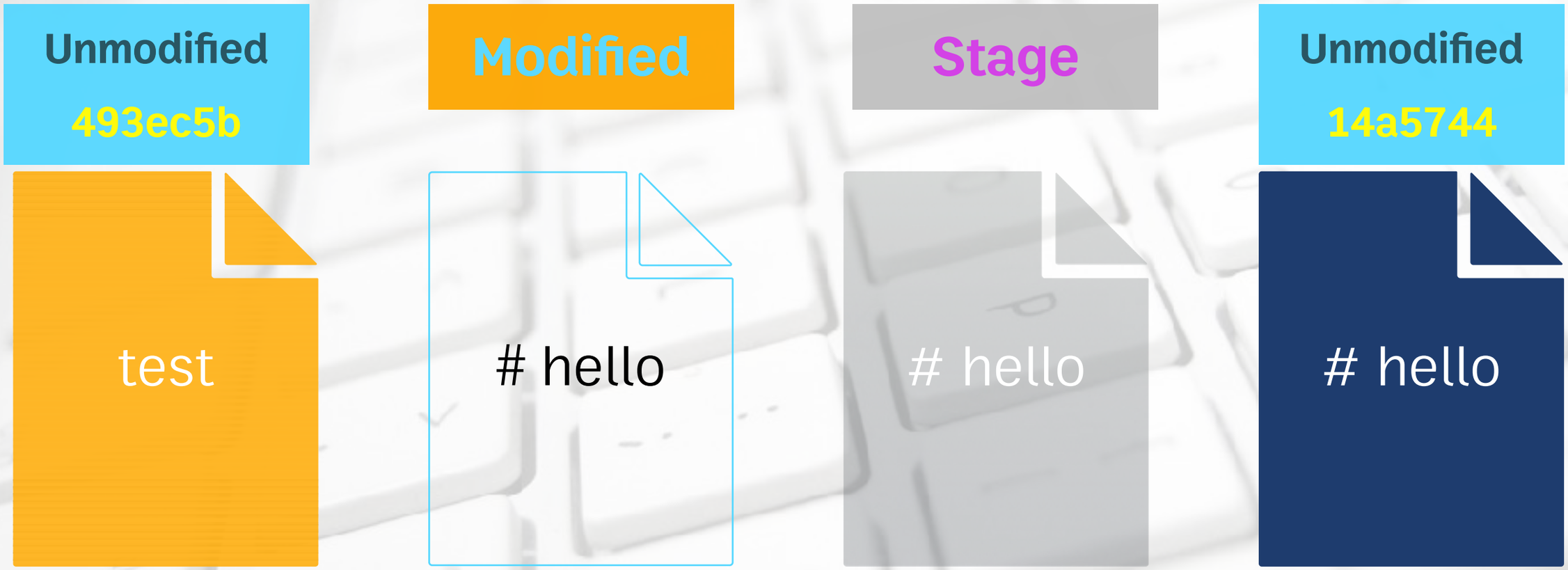
$git **reset --hard <commit>**

Git log

$git **log --oneline**

```
3bb9951 (HEAD -> main) Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

Git reset


Unmodified
493ec5b
test
Modified
hello
Stage
hello
Unmodified
14a5744
hello

git reset --mixed [default]

$git **reset 493ec5b**

$git **reset --mixed 493ec5b**

```
Unstaged changes after reset:
M       README
```

git reset --mixed [default]

$git **status**

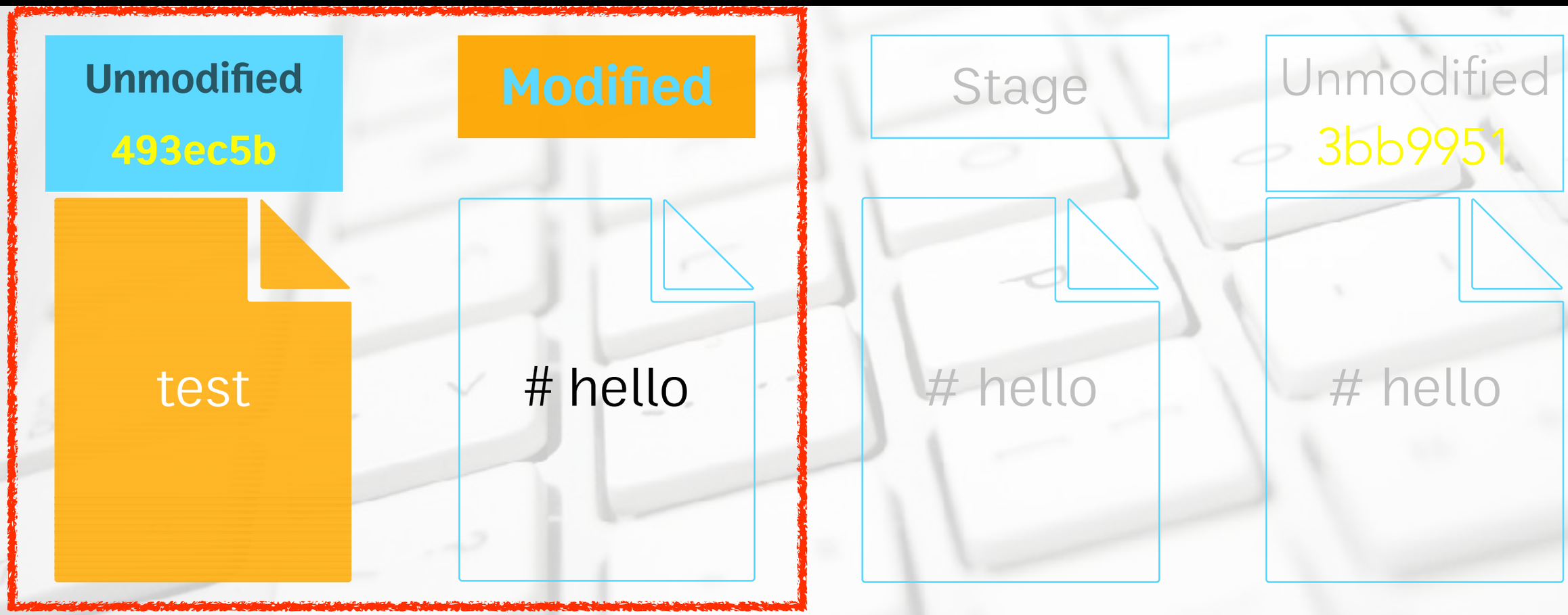
```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README

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

$git **log --oneline**

```
bf96888 (HEAD -> main) Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

git reset --mixed [default]

$git **reset 493ec5b**



git reset --hard

$git **reset --hard 3bb9951**

```
HEAD is now at 3bb9951 Edit README content
```

$git **reset --hard bf96888**

```
HEAD is now at bf96888 Rename README.md back to README
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

$git **log --oneline**

```
bf96888 (HEAD -> main) Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

git reset --hard

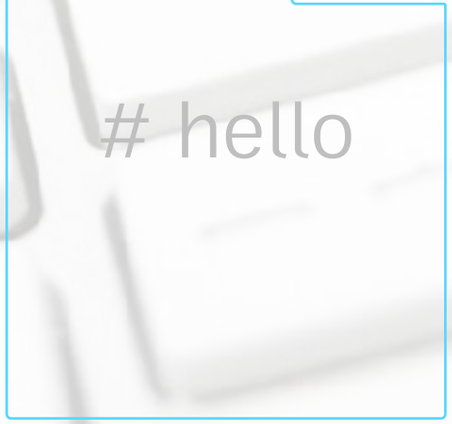
$git **reset bf96888**

Unmodified

bf96888

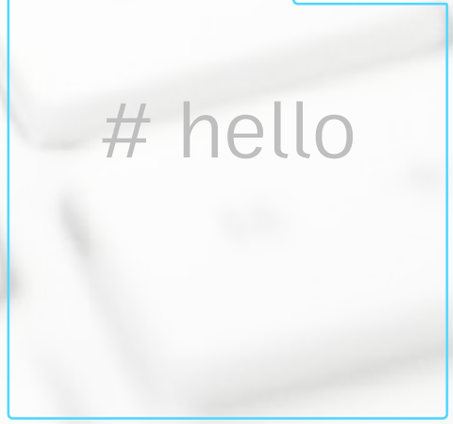
MODIFIED

Stage



Unmodified

3bb9951



git reset --soft

$git reset --hard 3bb9951

```
HEAD is now at 3bb9951 Edit README content
```

$git **reset --soft bf96888**

$git **status**

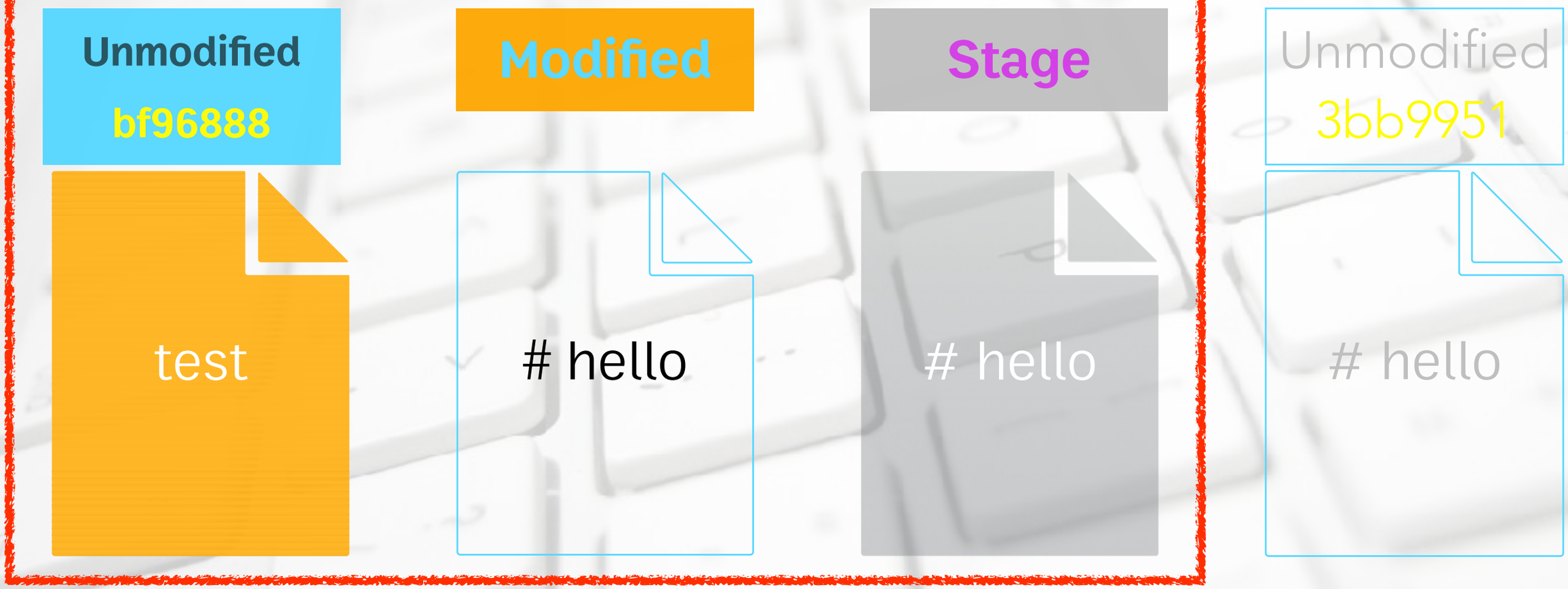
```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README
```

$git **log --oneline**

```
bf96888 (HEAD -> main) Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

git reset --soft

$git **reset --soft bf96888**



Remove files and folders

$rm **README**

$git **status**

```
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    README

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

Remove files and folders

$git **add README**

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    README
```

$git **commit -m "Delete README"**

```
[main 5882b53] Delete README
 1 file changed, 1 deletion(-)
 delete mode 100644 README
```

Remove files and folders

$git **status**

```
On branch main
nothing to commit, working tree clean
```

$git **log --oneline**

```
5882b53 (HEAD -> main) Delete README
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

Remove files and folders

$ls

```
note  tmp
```

$git reset --hard 3bb9951

$ls

```
README  note  tmp
```

git rm

$git rm **README**

$git **status**

$git **commit**

Delete file from git history with git rm

$touch *password*

$echo *"top secret"* > *password*

$git add **password**

$git **commit –m "Add password"**

```
[main 5e9b8d2] Add password
 1 file changed, 1 insertion(+)
 create mode 100644 password
```

Delete file from git history with git rm

$echo *"Hello, World" >> README*

$git add **README**

$git **commit -m "Add new content to README"**

```
[main 96b943a] Add new content to README
 1 file changed, 1 insertion(+)
```

Delete file from git history with git rm

$ls

```
README    note      password  tmp
```

$git **log --oneline**

```
96b943a (HEAD -> main) Add new content to README
5e9b8d2 Add password
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

Delete file from git history with git rm

$git **filter-branch --index-filter "git rm -rf --cached --ignore-unmatch password"**

```
WARNING: git-filter-branch has a glut of gotchas generating mangled history
         rewrites.  Hit Ctrl-C before proceeding to abort, then use an
         alternative filtering tool such as 'git filter-repo'
         (https://github.com/newren/git-filter-repo/) instead.  See the
         filter-branch manual page for more details; to squelch this warning,
         set FILTER_BRANCH_SQUELCH_WARNING=1.
Proceeding with filter-branch...

Rewrite 779b411f276c8706b46bad5b8be1636030e4b351 (1/8) (0 seconds passed, rem
aining 0 Rewrite 656bbdd892f5c59a9a197bbac209539138f2a8ca (2/8) (0 seconds pa
ssed, remaining 0 Rewrite e4c9eb3325a6c2907154131b7f0709c0c7fb835b (3/8) (0 s
econds passed, remaining 0 Rewrite 6d174313a4df18c1254ba164722e55b7458cc2b8 (
4/8) (0 seconds passed, remaining 0 Rewrite bf9688891ca6bfff62580d204ec2a3bb4
904a7e2 (5/8) (0 seconds passed, remaining 0 Rewrite 3bb99510126c972501121d32
ef2ea2264c0189ca (6/8) (0 seconds passed, remaining 0 Rewrite 5e9b8d22499ee57
2f4db0d60d2de4f84f9b552d6 (7/8) (0 seconds passed, remaining 0 predicted)
rm 'password'
Rewrite 96b943adeb7c4d07f3817cd5e75a08e7c4cd4384 (8/8) (0 seconds passed, rem
aining 0 predicted)    rm 'password'

Ref 'refs/heads/main' was rewritten
```

Delete file from git history with git rm

$git **log --oneline**

```
de35152 (HEAD -> main) Add new content to README
0a0b6de Add password
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

Delete file from git history with git rm

Compare git history

Before

```
96b943a (HEAD -> main) Add new content to README
5e9b8d2 Add password
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

After

```
de35152 (HEAD -> main) Add new content to README
0a0b6de Add password
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

Remove files and folders

$git reset --hard 0a0b6de

```
HEAD is now at 0a0b6de Add password
```

$ls

```
README note    tmp
```

.gitkeep

$mkdir resources

$ls

```
README      note        resources tmp
```

$git **status**

```
On branch main
nothing to commit, working tree clean
```

.gitkeep

$touch resources/.gitkeep

$git **status**

```
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        resources/

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

See difference of file

$git **diff**

$git **diff** --cached

$git **diff <commit>**

$git **diff <commit> <commit>**

git diff

$echo "สวัสดี, ชาวโลก" >> README

$git **diff**

```
diff --git a/README b/README
index 9b4922a..ab76275 100644
--- a/README
+++ b/README
@@ -1,2 +1,3 @@
 # hello
 Hello, World
+สวัสดี, ชาวโลก
```

git diff --cached

$git add README

$git **diff --cached**

--STAGED IS A SYNONYM OF --CACHED.

$git **diff** --staged

```
diff --git a/README b/README
index 9b4922a..ab76275 100644
--- a/README
+++ b/README
@@ -1,2 +1,3 @@
 # hello
 Hello, World
+สวัสดี, ชาวโลก
```

git diff <commit>

$git commit -m "Add new content to README"

$git log --oneline

```
55d43f4 (HEAD -> main) Add new content to README
7fc9998 Add resources directory
de35152 Add new content to README
0a0b6de Add password
3bb9951 Edit README content
bf96888 Rename README.md back to README
6d17431 Rename README to README.md
e4c9eb3 Add .gitignore
656bbdd Edit README
779b411 Add README
```

git diff <commit>

$git **diff 7fc9998**

$git **diff HEAD^**

```
diff --git a/README b/README
index 9b4922a..ab76275 100644
--- a/README
+++ b/README
@@ -1,2 +1,3 @@
 # hello
 Hello, World
+สวัสดี, ชาวโลก
```

git diff <commit>

$git **diff de35152**

$git **diff HEAD^^**

```
diff --git a/README b/README
index 9b4922a..ab76275 100644
--- a/README
+++ b/README
@@ -1,2 +1,3 @@
 # hello
 Hello, World
+สวัสดี, ชาวโลก
diff --git a/resources/.gitkeep b/resources/.gitkeep
new file mode 100644
index 0000000..e69de29
```

git diff <commit> <commit>

$git **diff de35152 7fc9998**

$git **diff HEAD^^ HEAD^**

```
diff --git a/resources/.gitkeep b/resources/.gitkeep
new file mode 100644
index 0000000..e69de29
```

git diff

Delete "Hello, World" in README

$git **diff**

```
diff --git a/README b/README
index ab76275..70e58dc 100644
--- a/README
+++ b/README
@@ -1,3 +1,2 @@
 # hello
-Hello, World
 สวัสดี, ชาวโลก
```

git diff

echo "Hello, World" resources/.gitkeep

$git **diff**

```
diff --git a/README b/README
index ab76275..70e58dc 100644
--- a/README
+++ b/README
@@ -1,3 +1,2 @@
 # hello
-Hello, World
 สวัสดี, ชาวโลก
diff --git a/resources/.gitkeep b/resources/.gitkeep
index e69de29..3fa0d4b 100644
--- a/resources/.gitkeep
+++ b/resources/.gitkeep
@@ -0,0 +1 @@
+Hello, World
```

Working with Branch

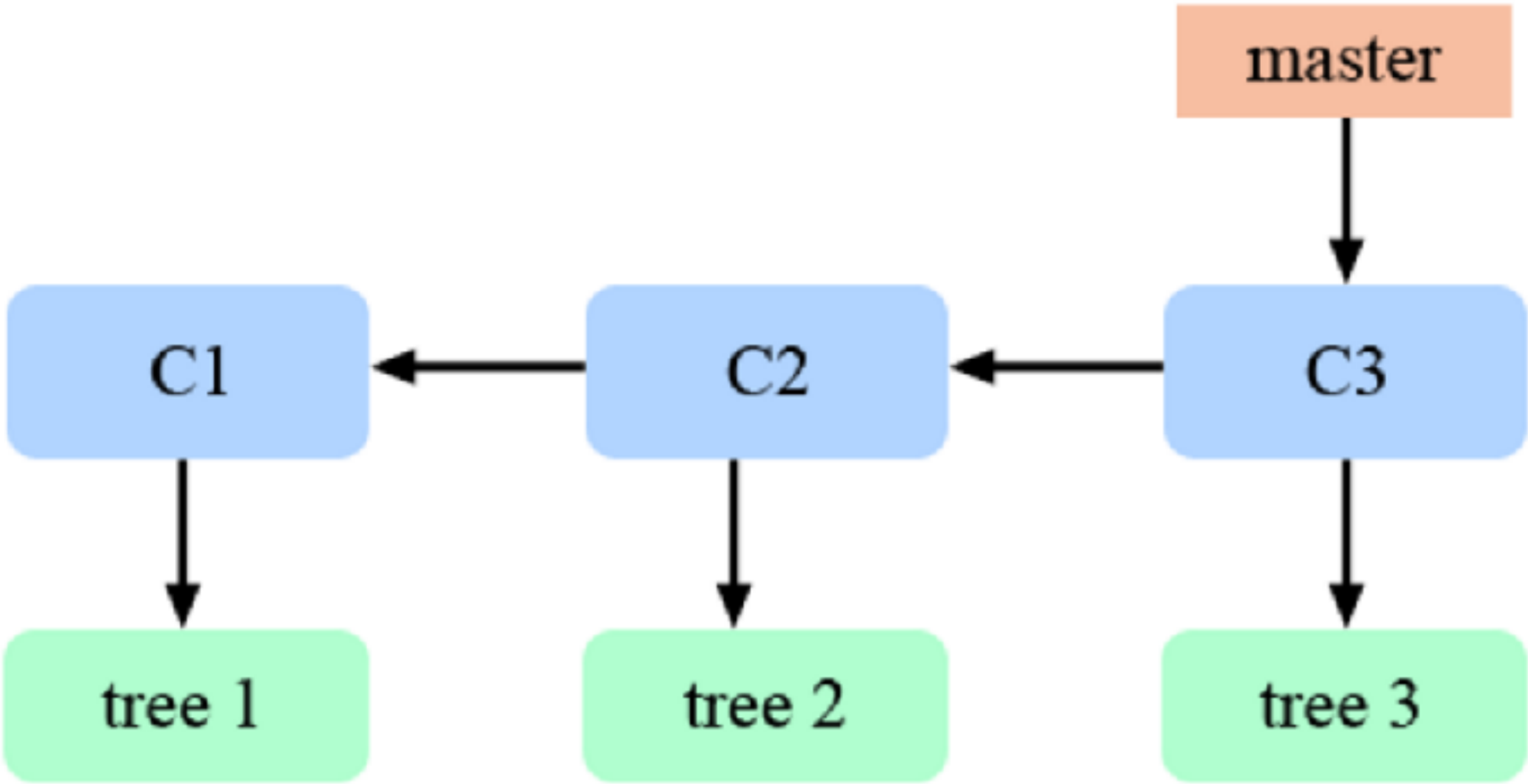
git branch

$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
```

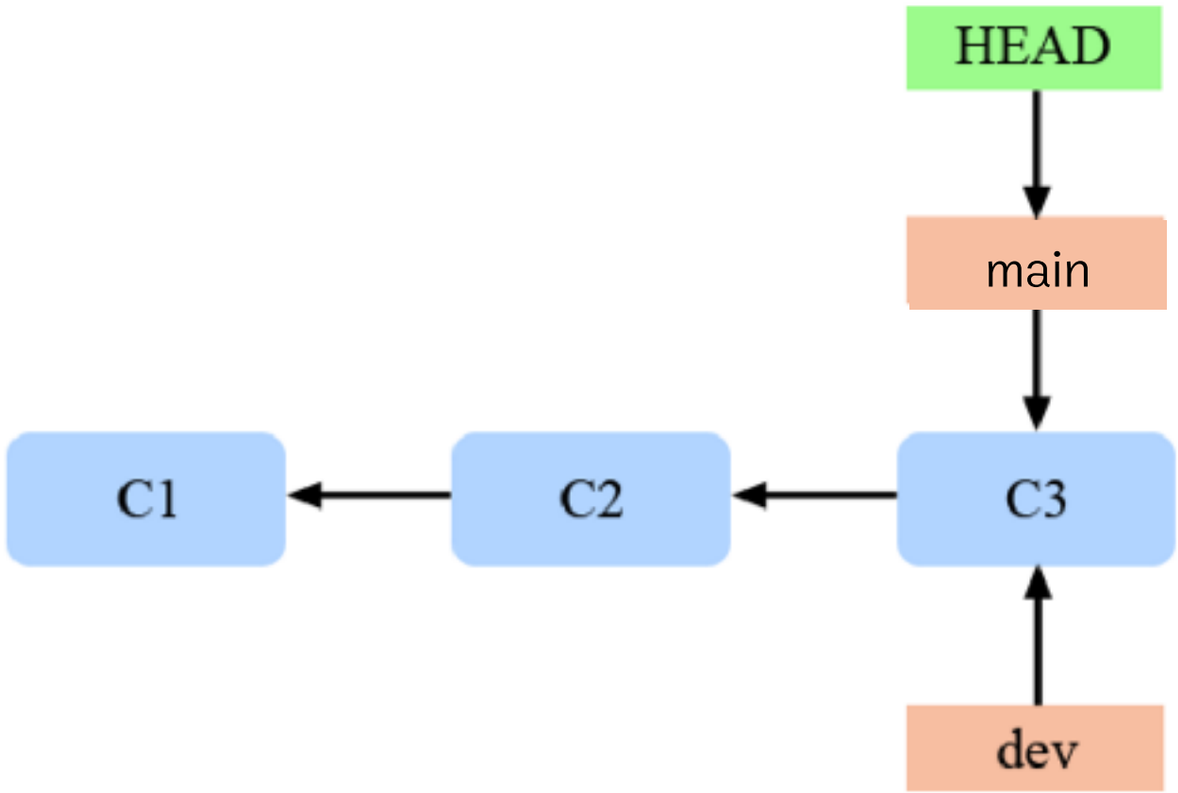
Each branch points to a commit



Create new branch

$git **branch** <*BRANCH NAME*>

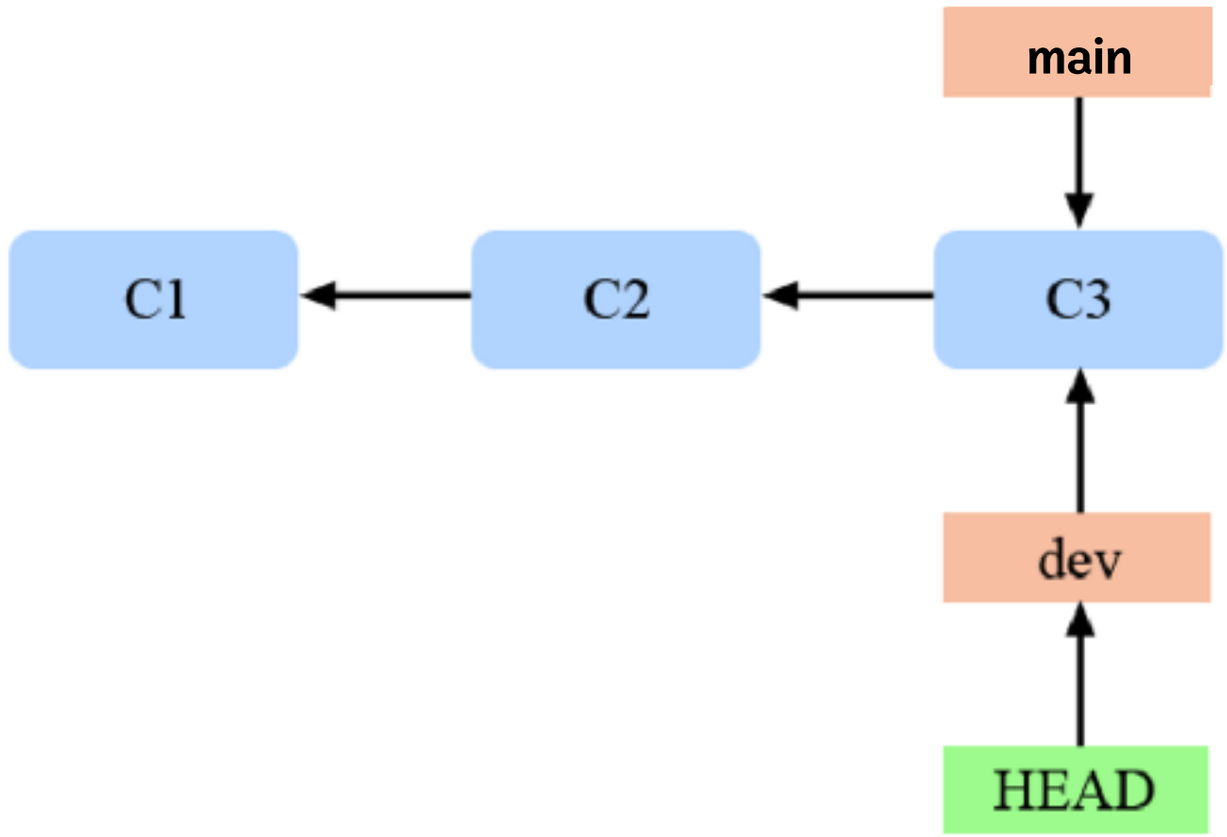
$git **branch** dev



Switch branch

$git **checkout** <*BRANCH NAME*>

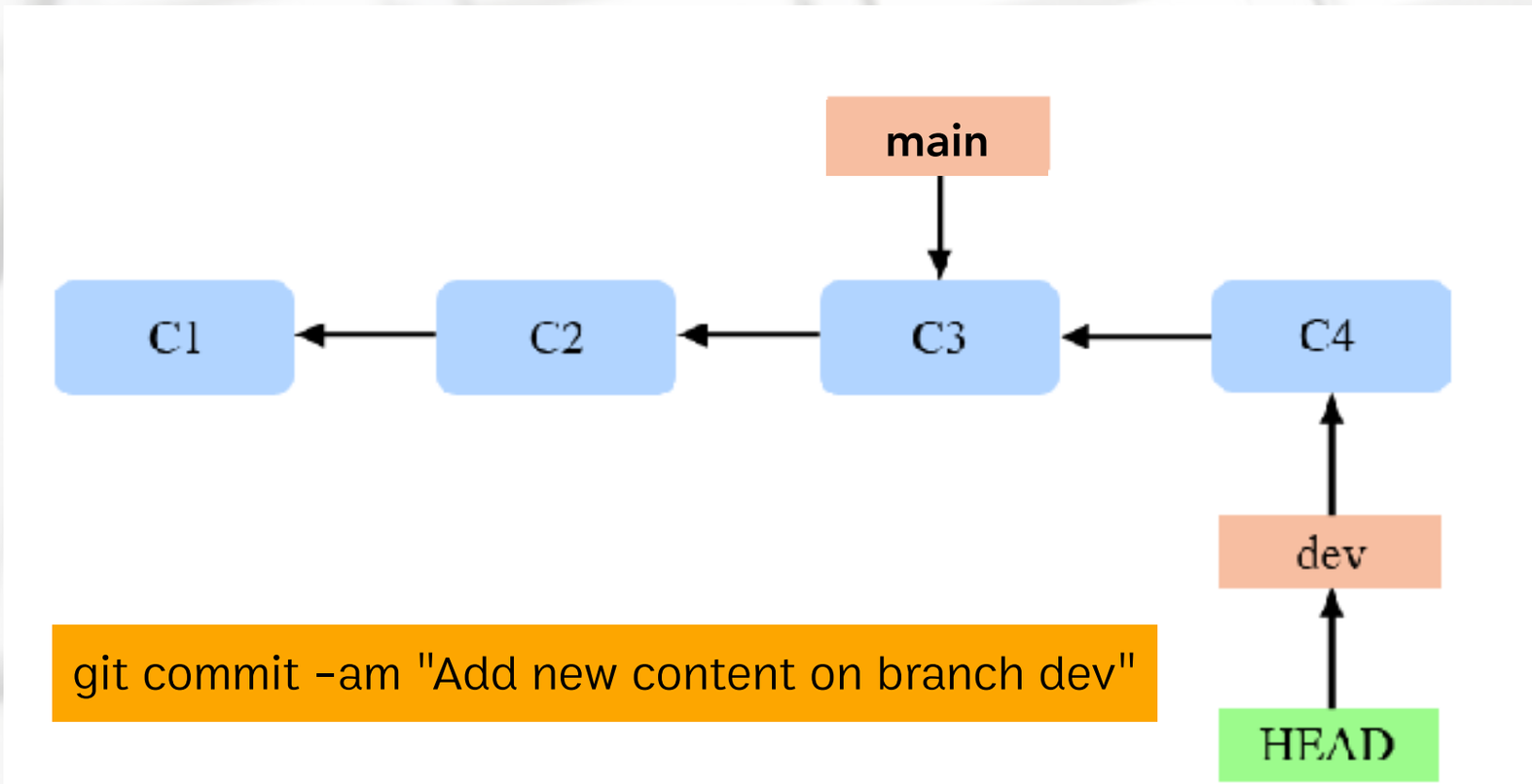
$git **checkout** dev



Modified and commit on branch

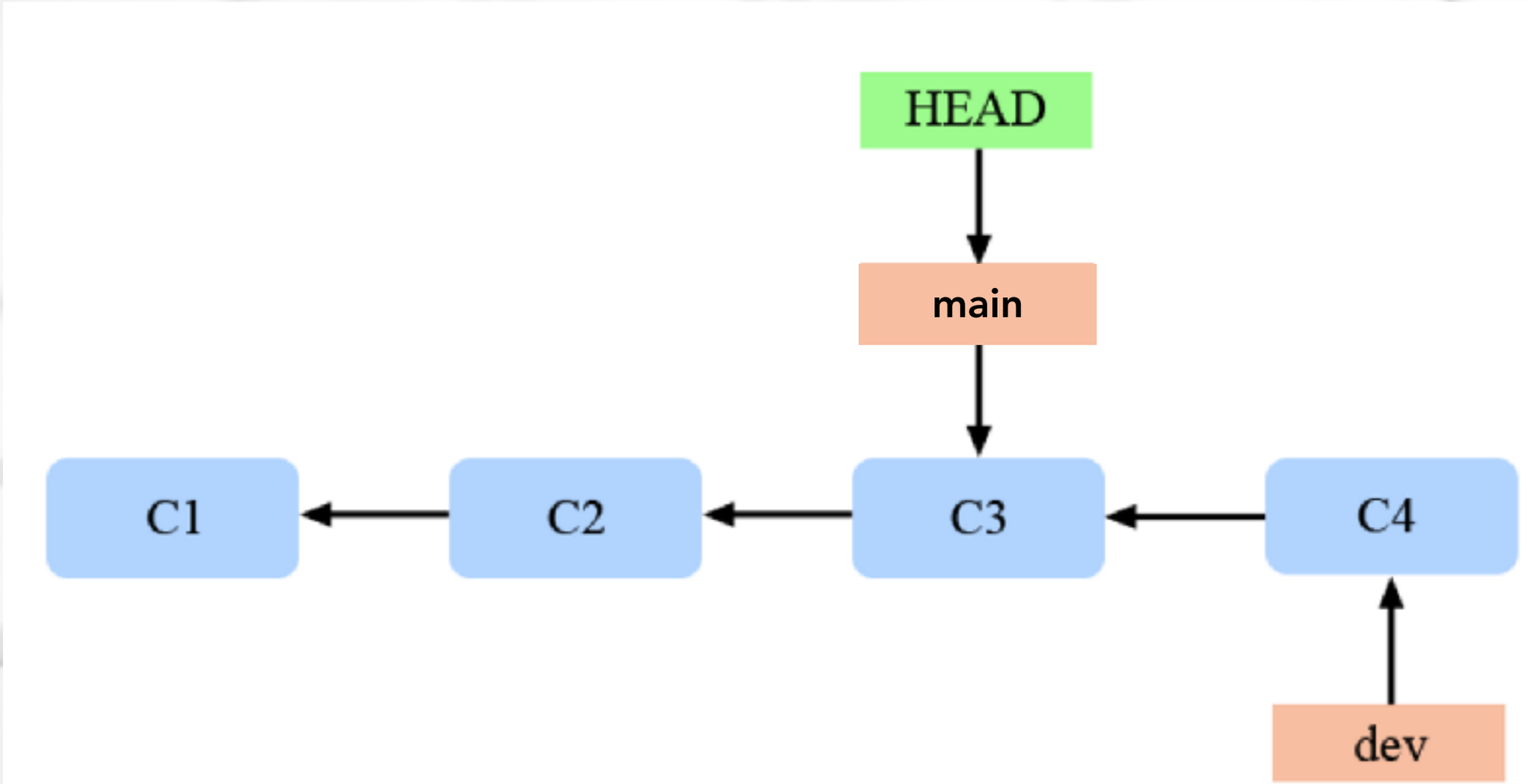
$echo “On branch dev” > *README*

$git **commit** -am “Add new content on branch dev”



Switch to master branch

$git **checkout** *main*



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**

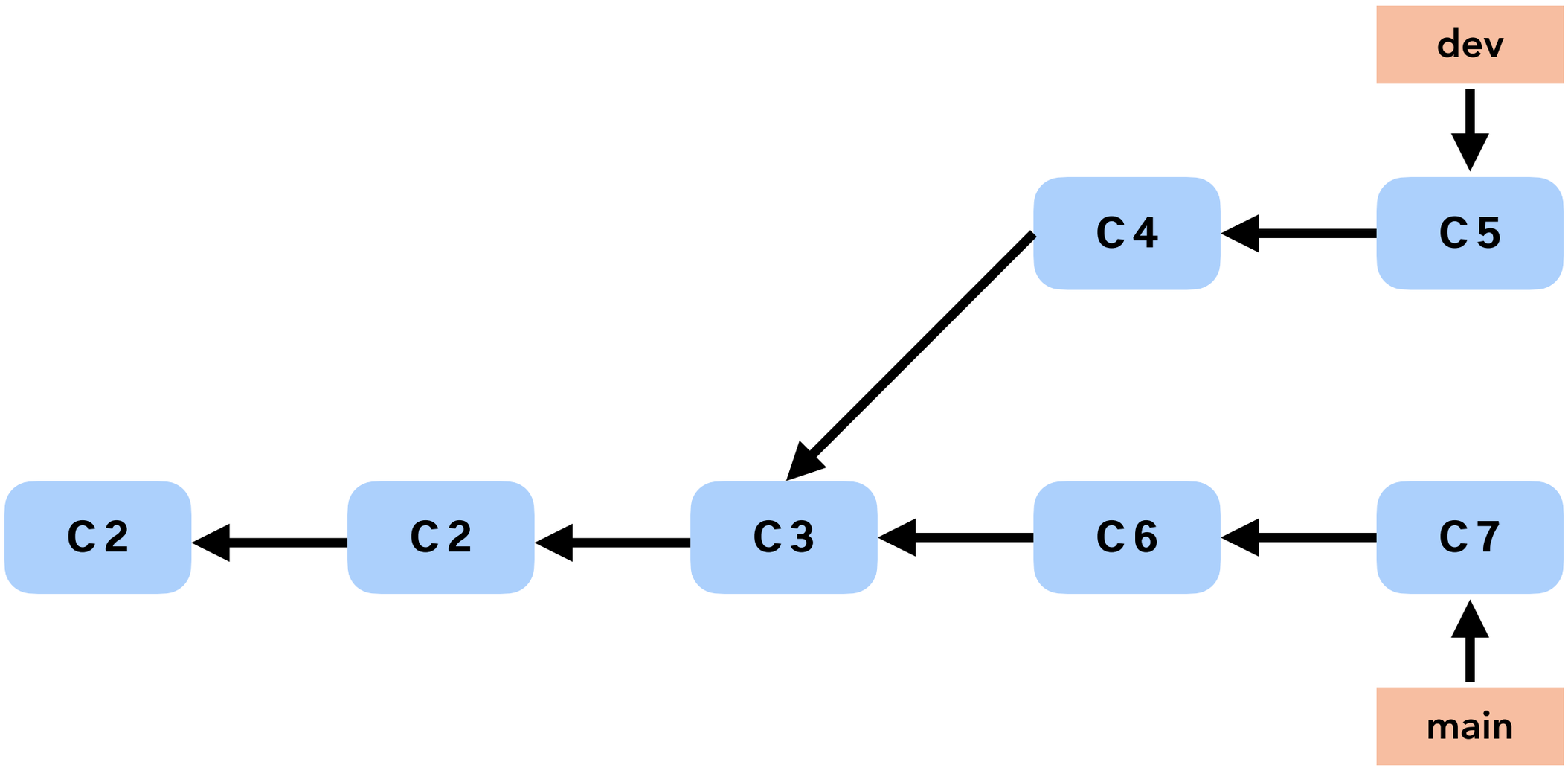
Exercise with git branch

create branch name **"uat"**

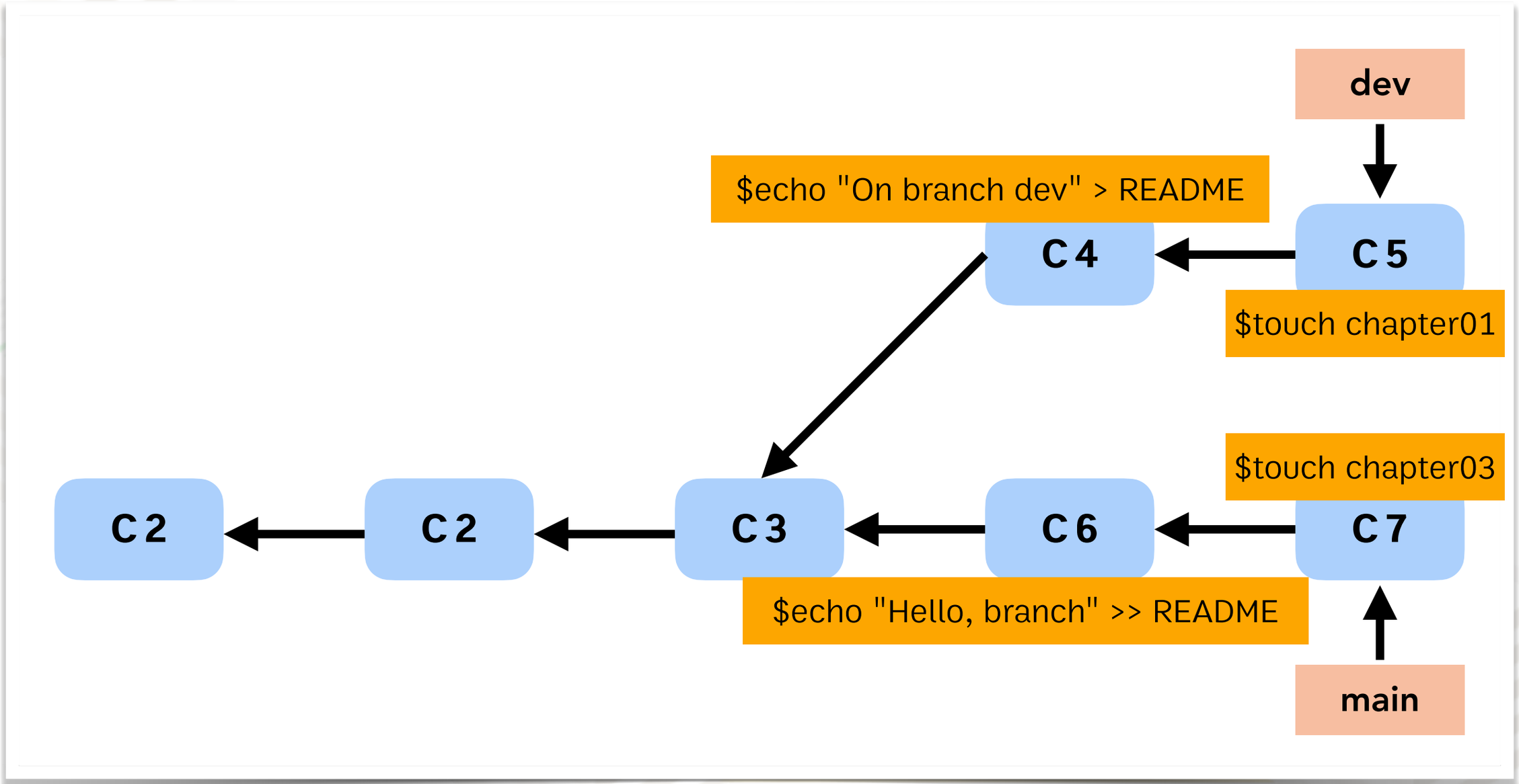
remove branch uat


BRANCHED FROM MASTER 3 WEEKS AGO
MERGED BACK WITHOUT ANY CONFLICTS

Start before **Merge**

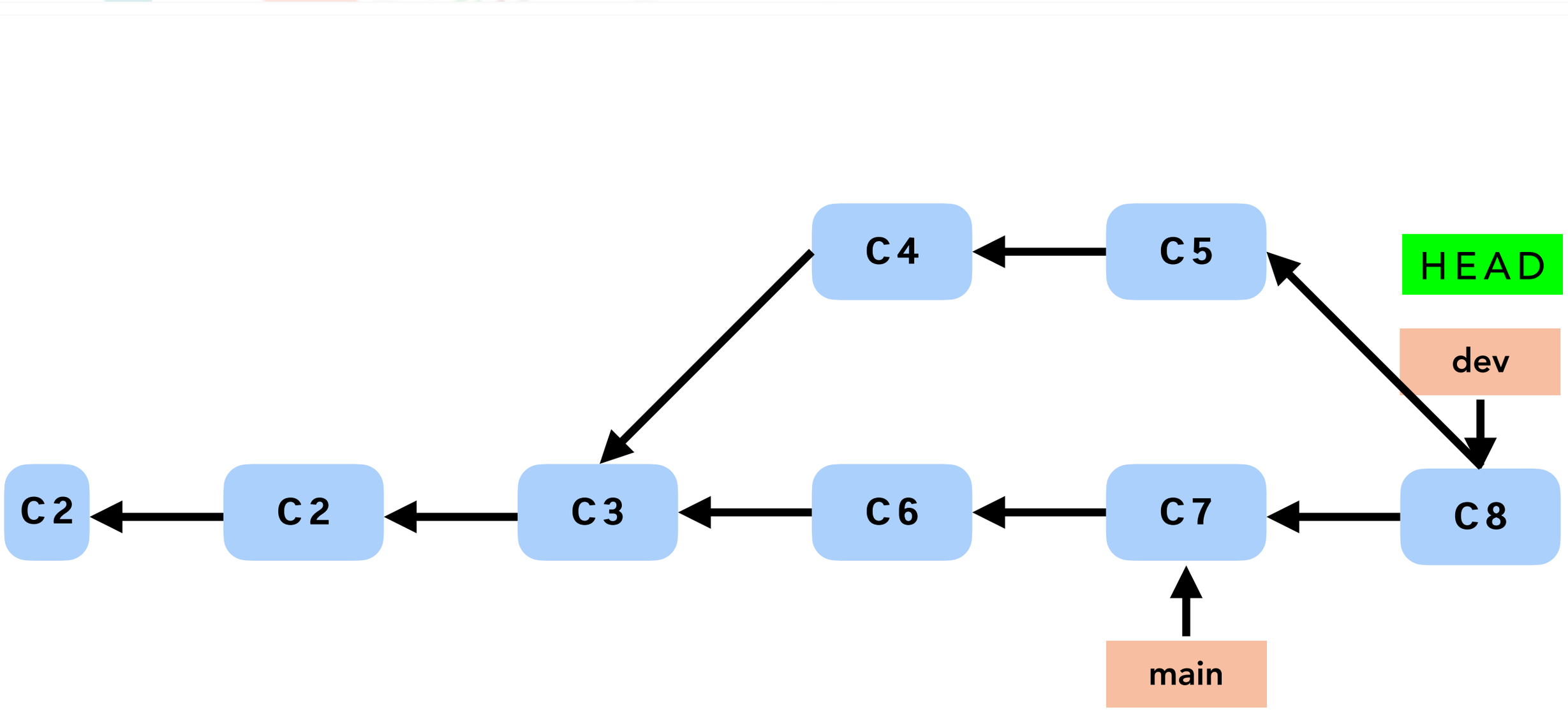


Start before **Merge**



Git Merge

$git **merge main**



Conflict!!!!!

$git **merge main**

```
Auto-merging README
CONFLICT (content): Merge conflict in README
Automatic merge failed; fix conflicts and then commit the result.
```

$git **status**

```
On branch dev
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

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

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

Conflict!!!!!

README

```
<<<<<<< HEAD (Current Change)
On branch dev

=======
# hello
Hello, World
สวัสดี, ชาวโลก
Hello, branch
>>>>>>> main (Incoming Change)
```

Exercise fix conflicts

git **status**

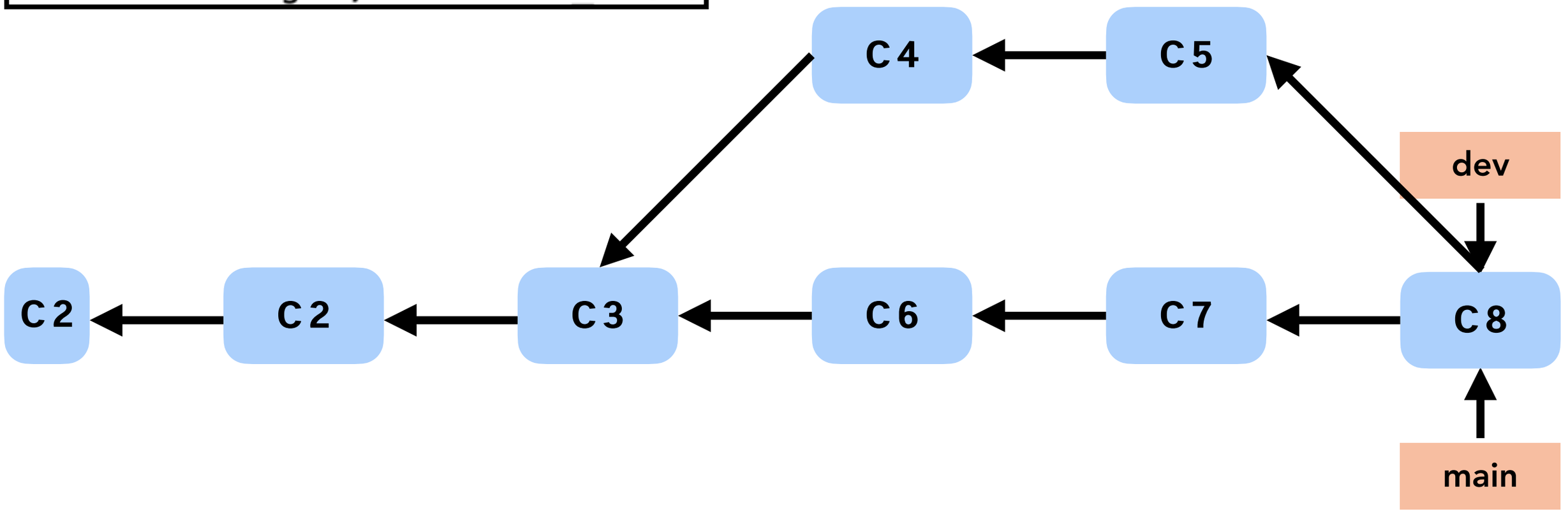
git **add README**

git **comit**

Git Merge

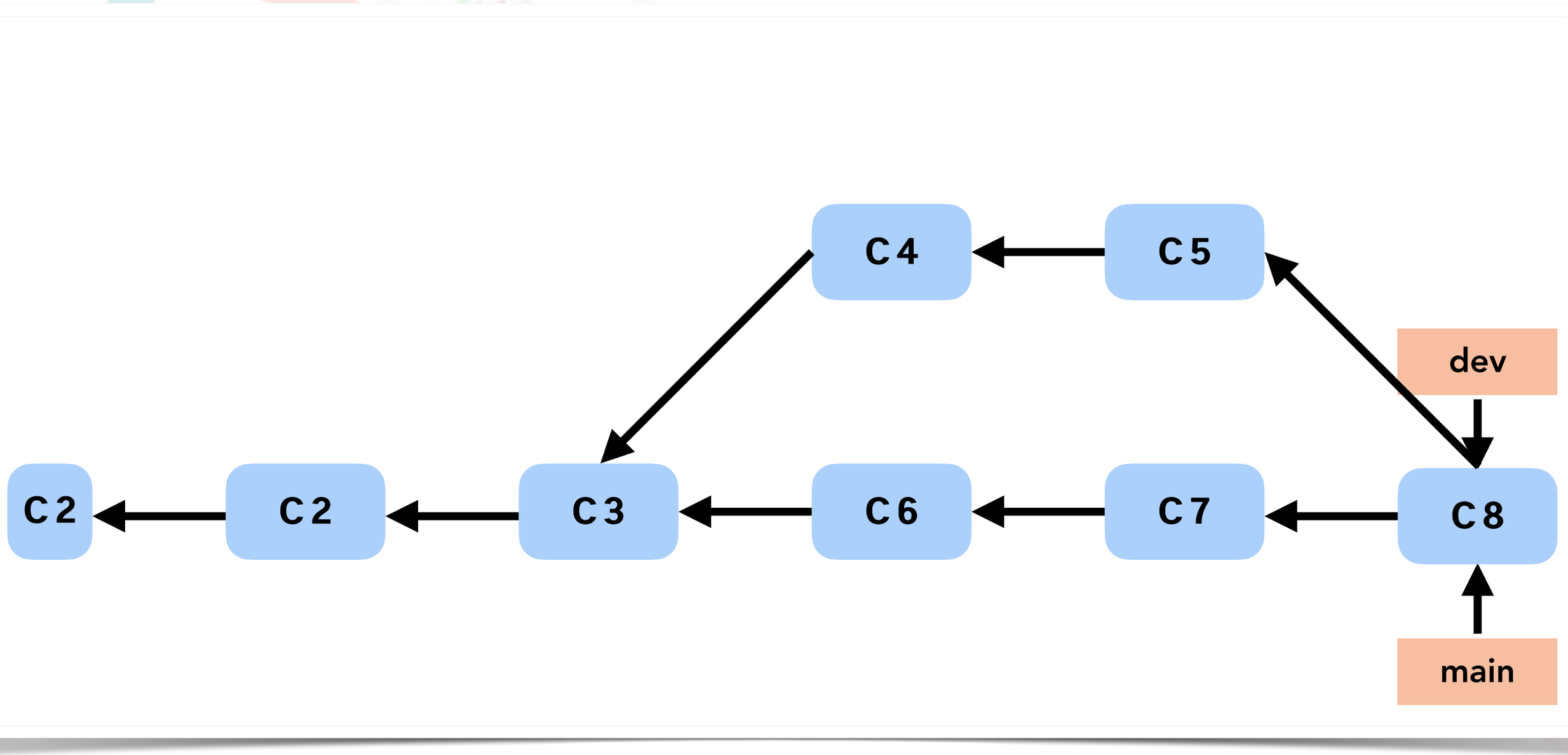
$git **merge dev**

```
Updating e9cb7f6..c4db201
Fast-forward
 README | 1 +
 1 file changed, 1 insertion(+)
```



After Merge

$git **log --oneline --graph**



Avoid Merge Conflict

Small change and commit

Early merge

Single Responsibility Principle

Communication is a Key

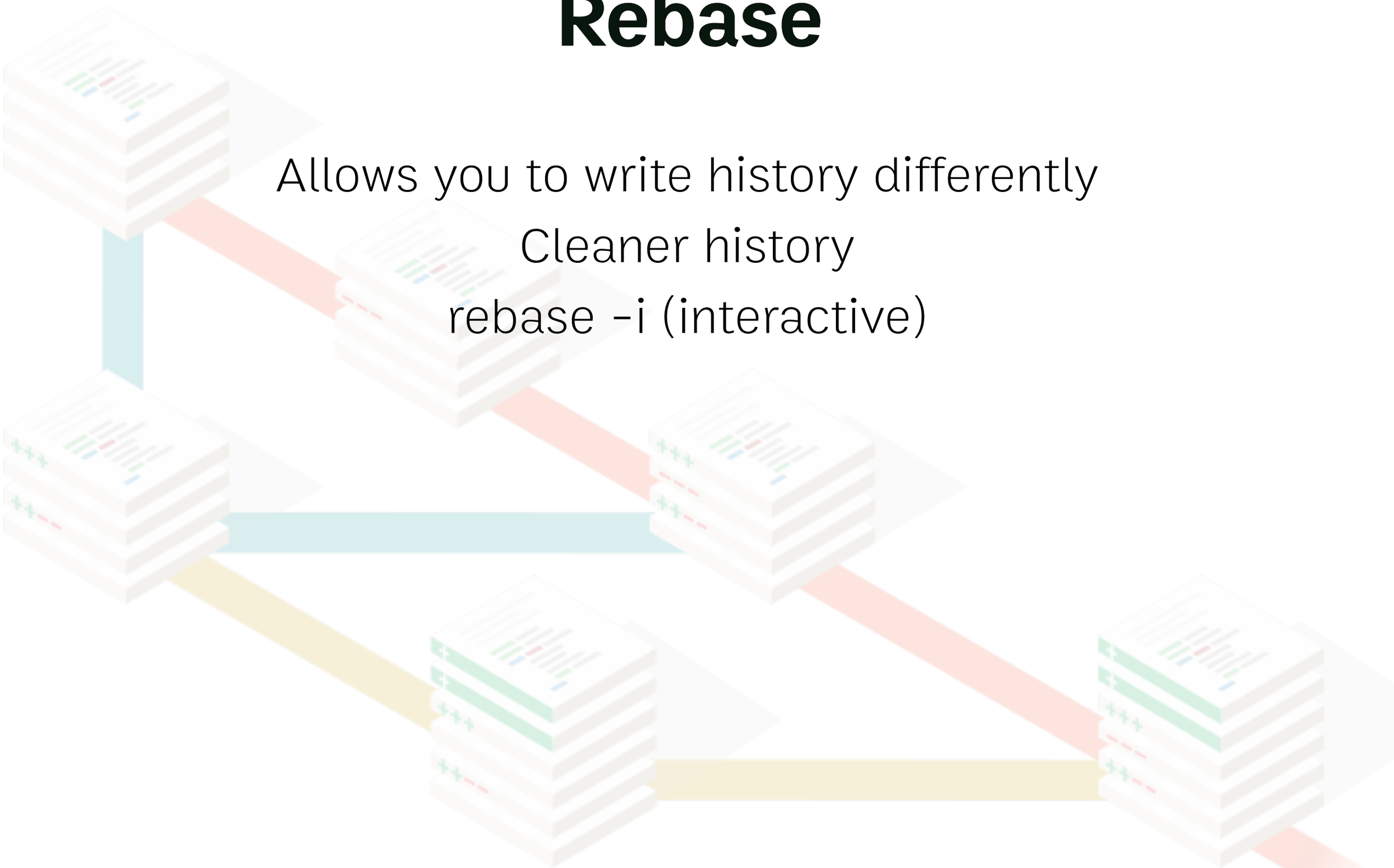
Mob programming

Rebase

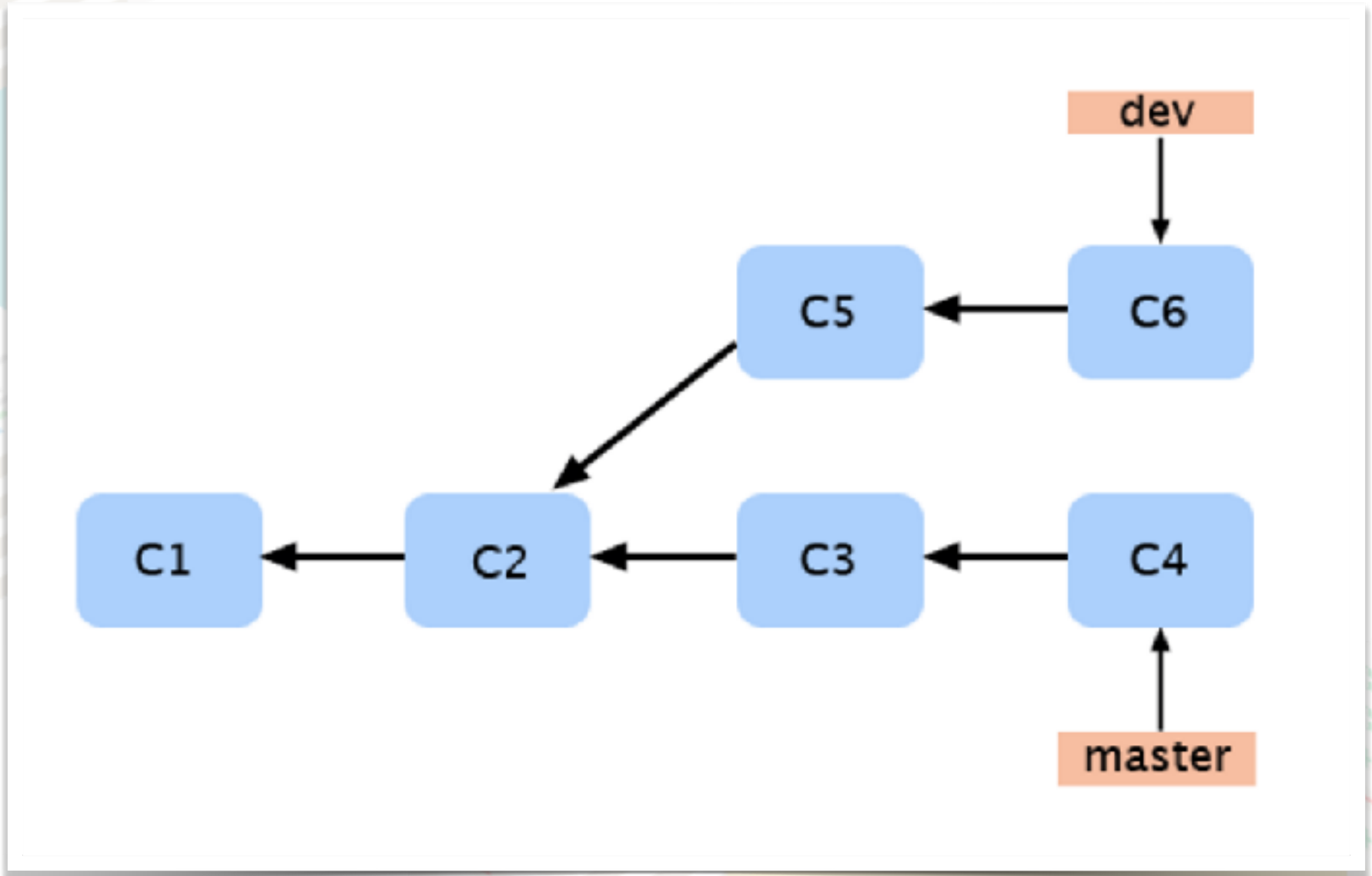
Allows you to write history differently

Cleaner history

rebase -i (interactive)

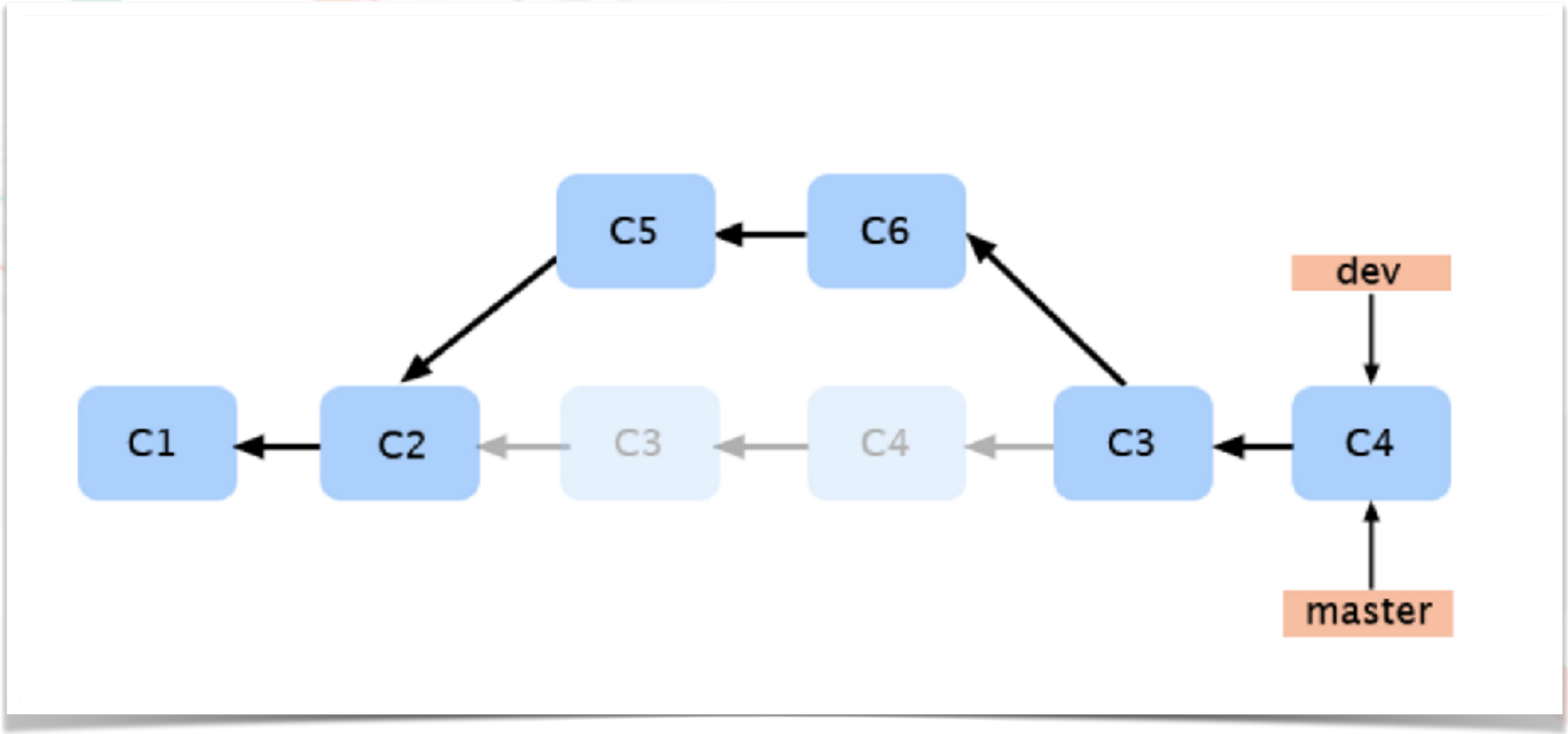


Start before **Rebase**

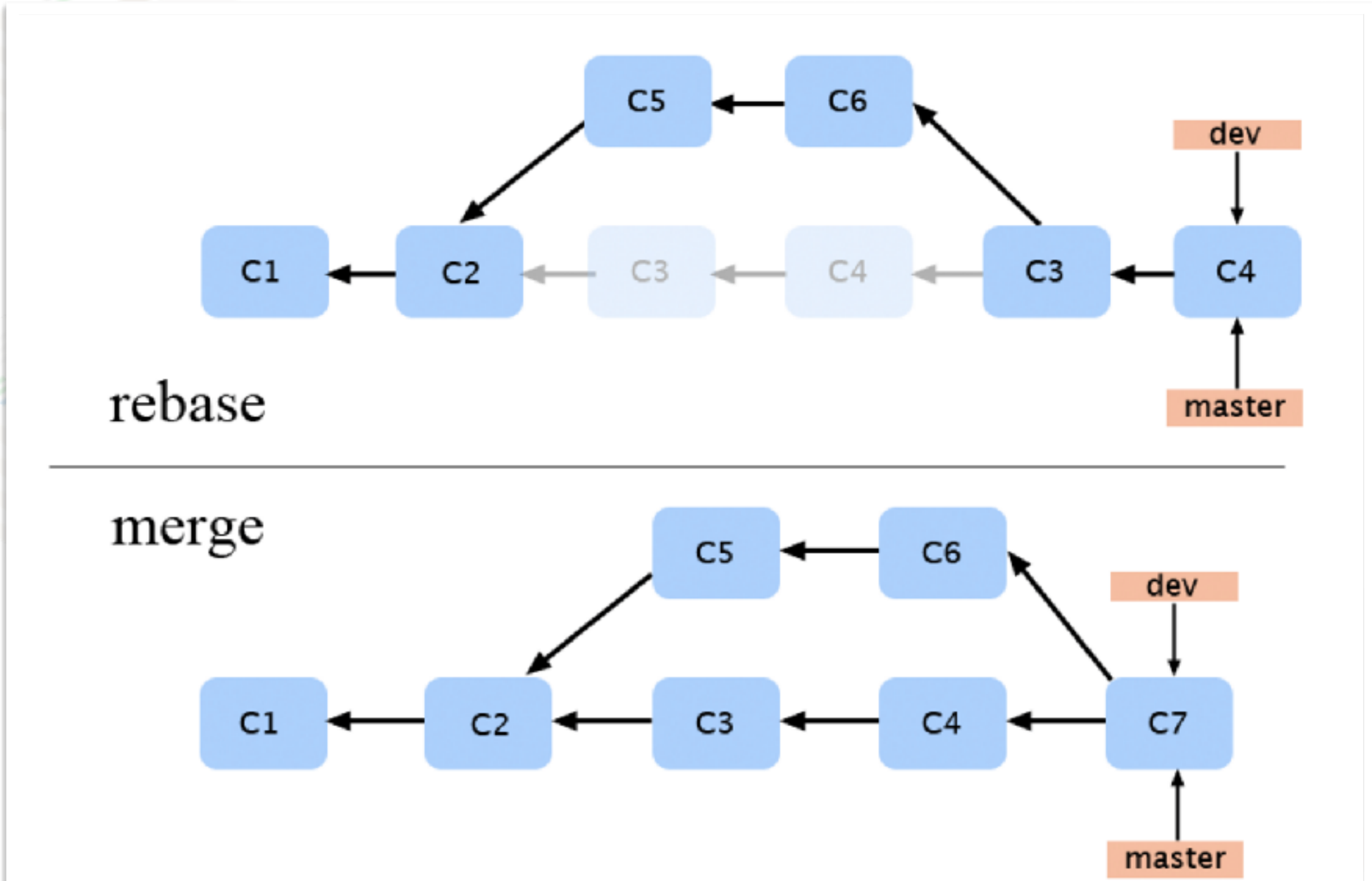


After Rebase

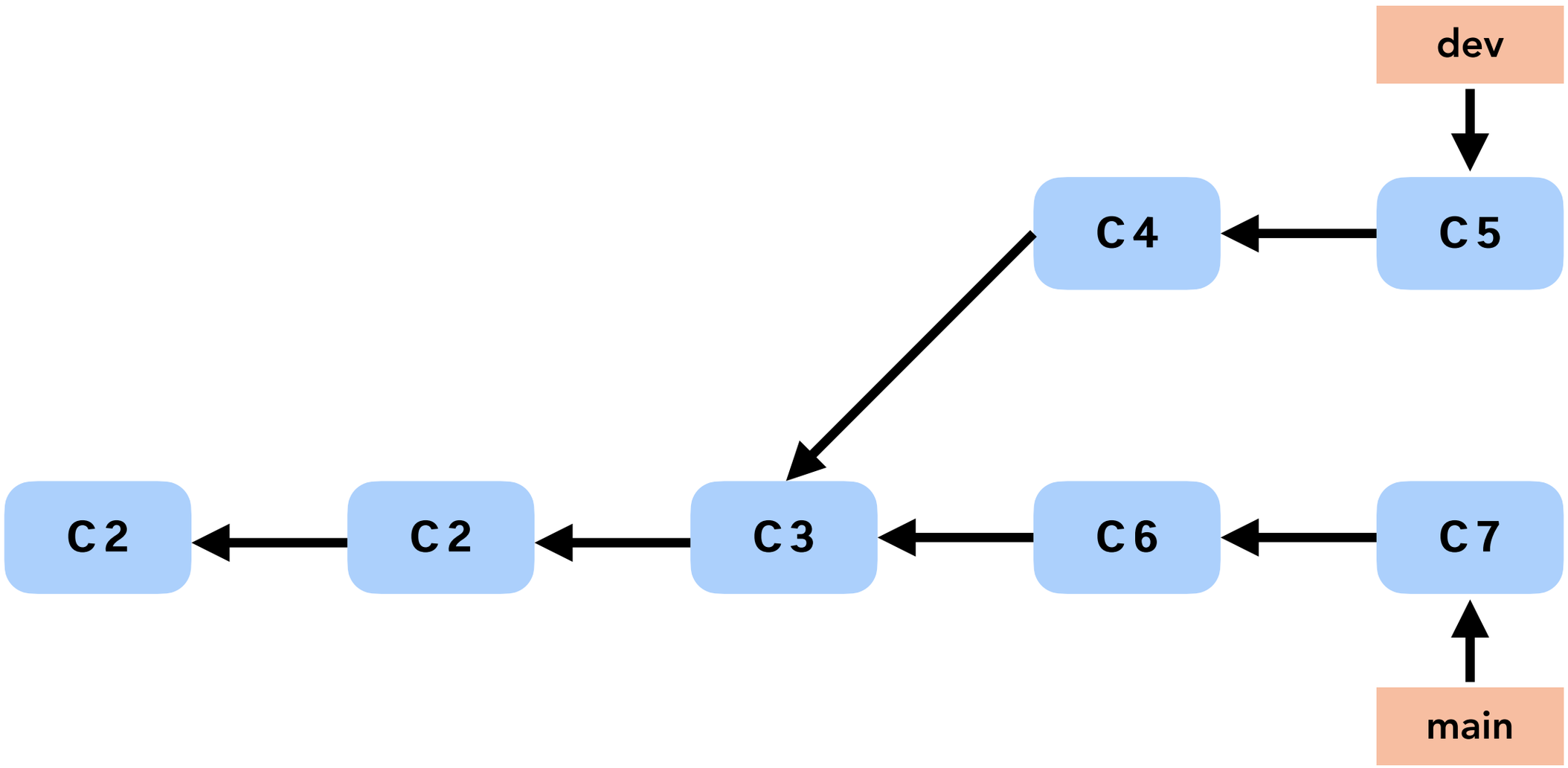
$git **rebase** *[BRANCH NAME]*



Merge vs **Rebase**



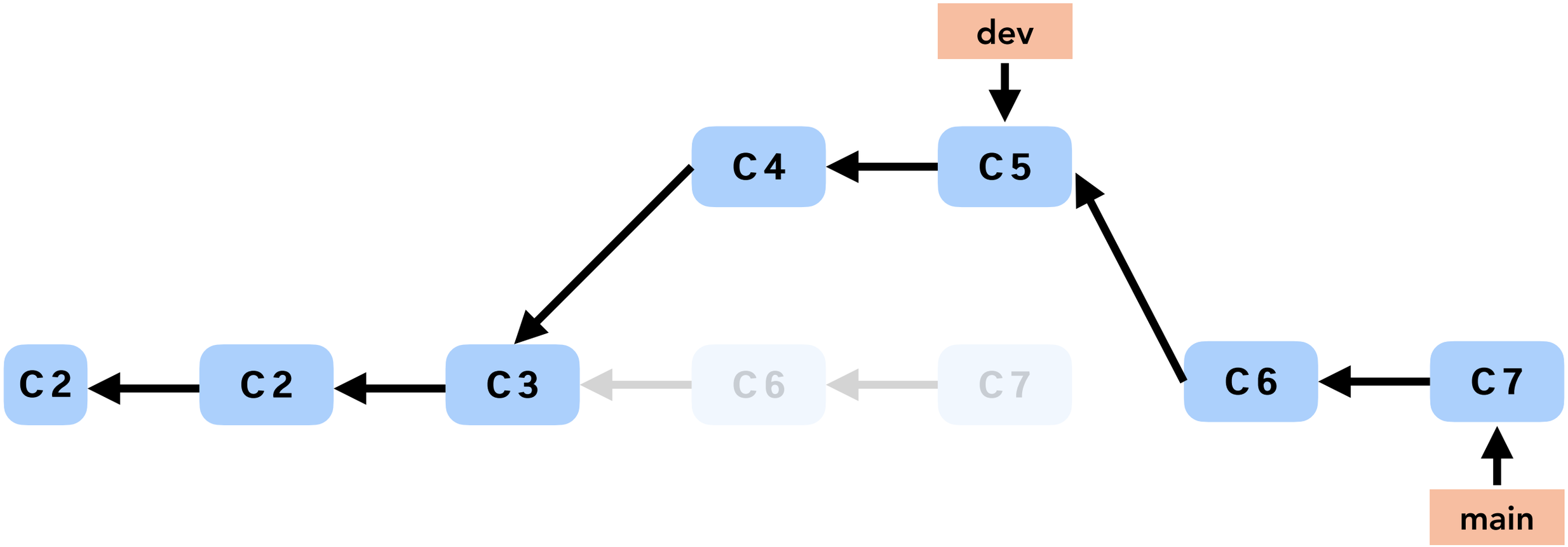
Rebase in action



Rebase in action

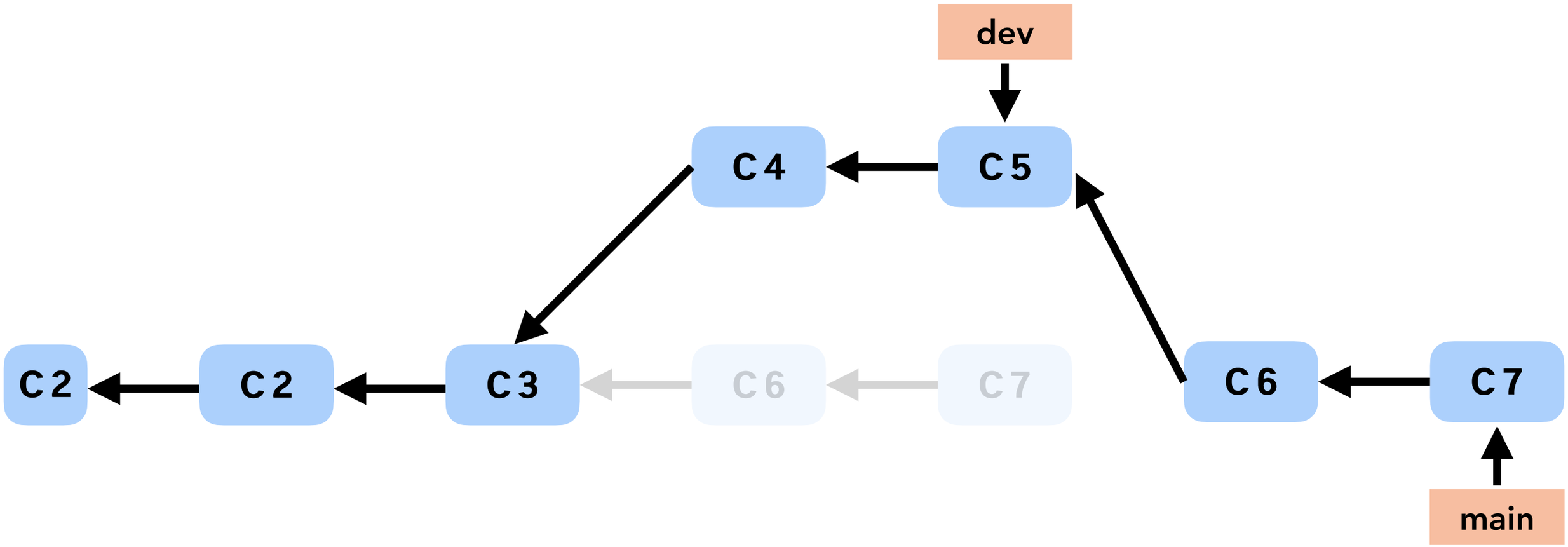
$git **checkout** *main*

$git **rebase** *dev*

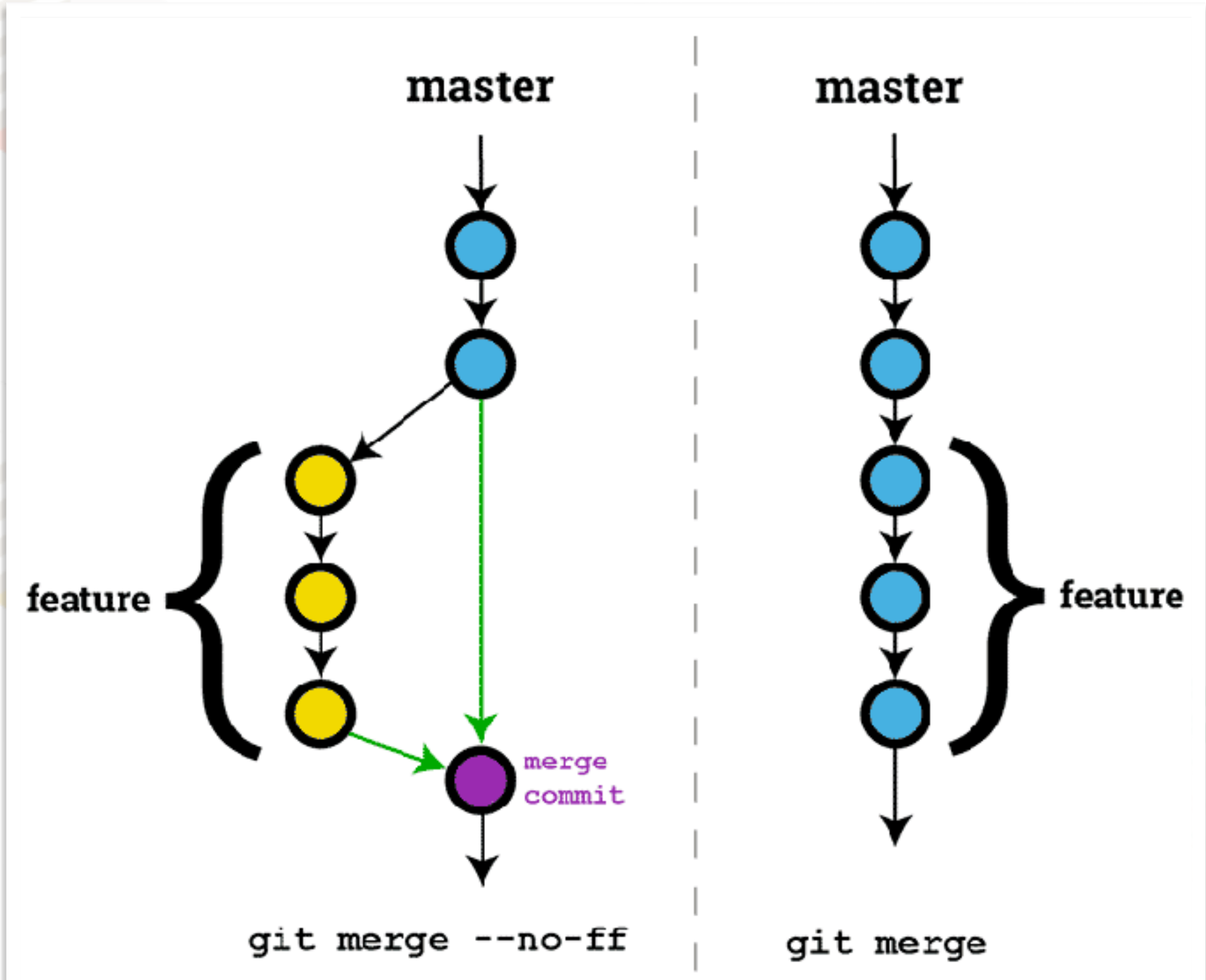


Rebase in action

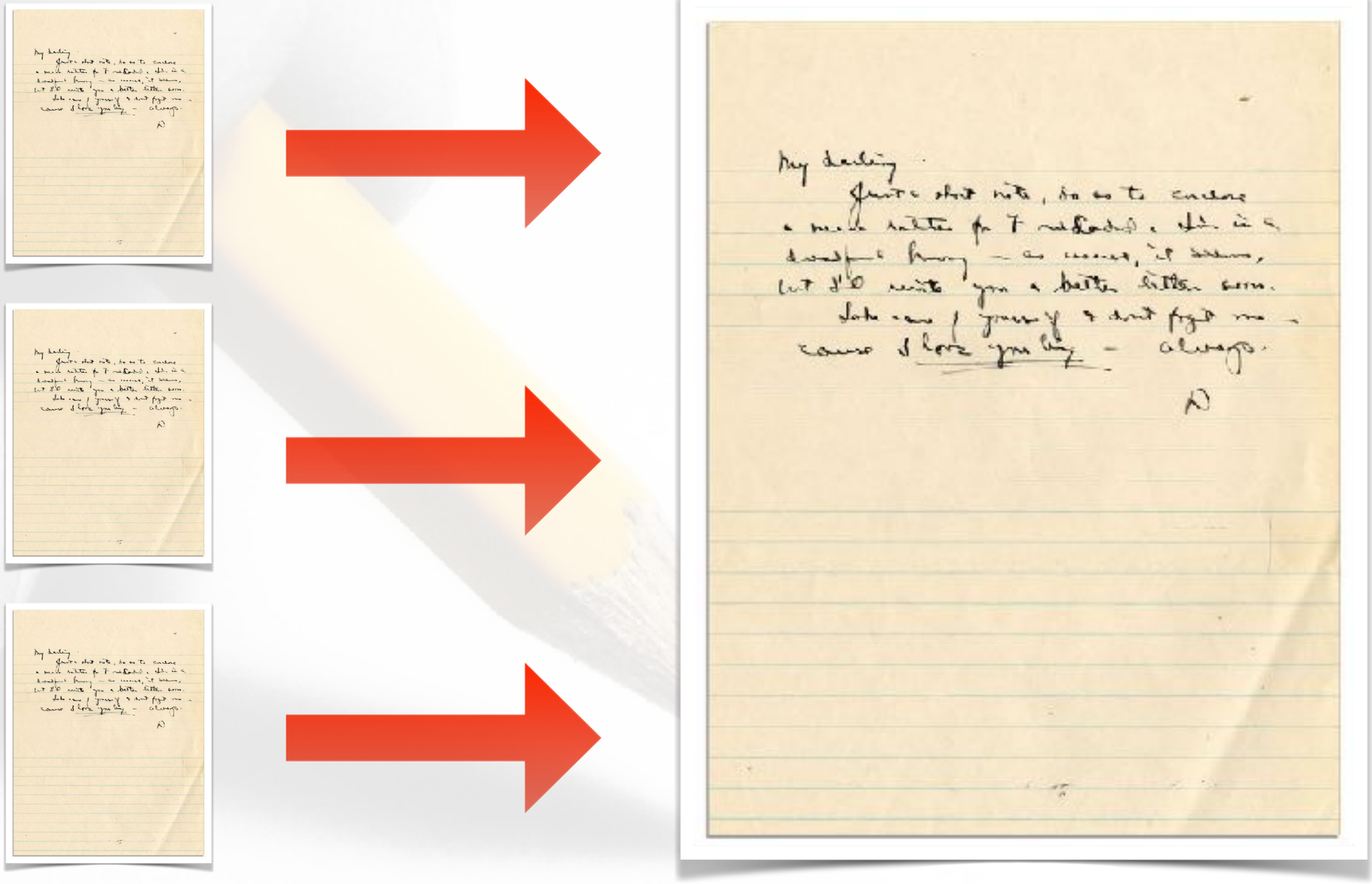
$git **log --oneline --graph**



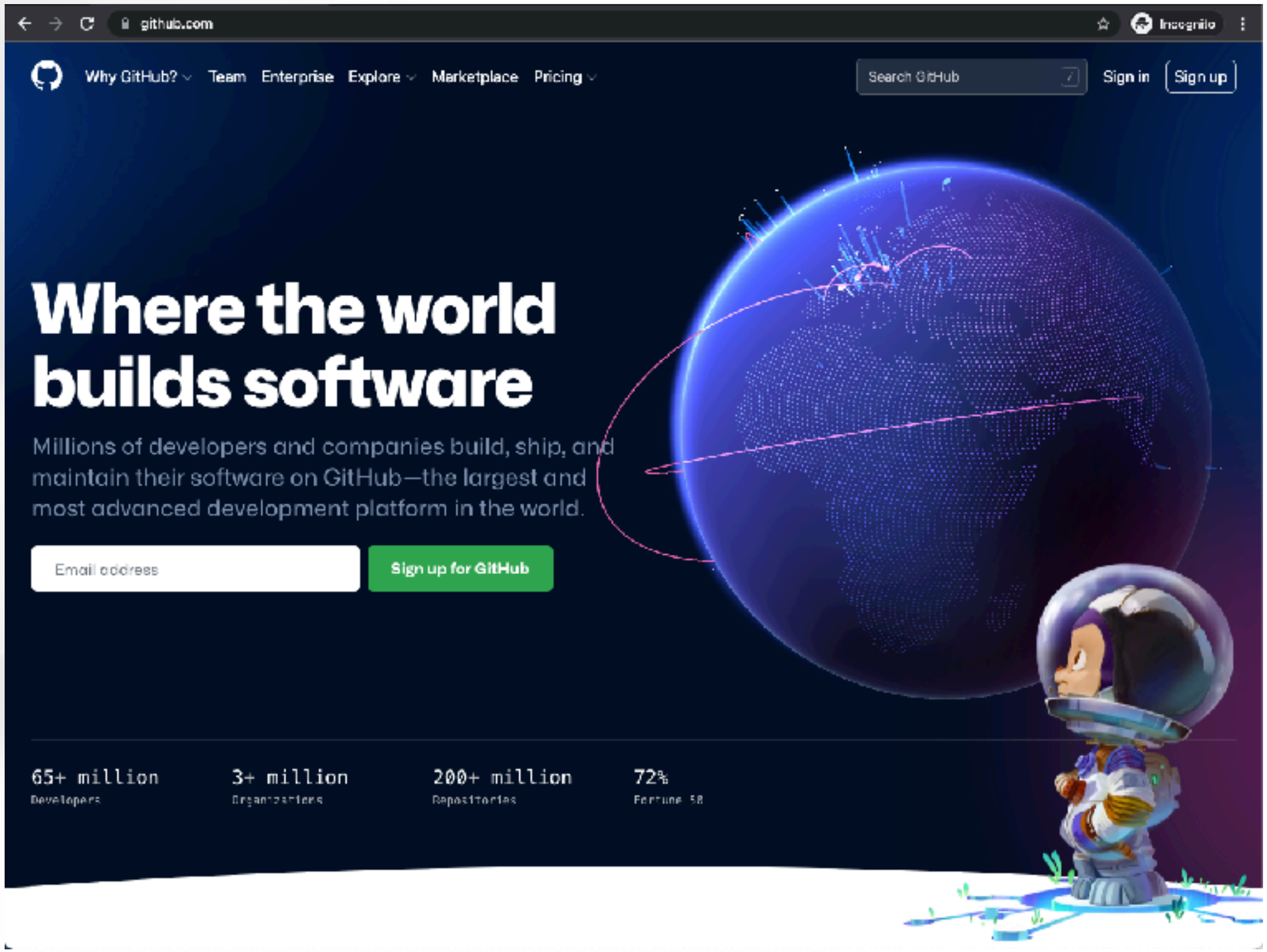
Merge and **Rebase**



All become One



Working on GitHub



Generate ssh key and add to Github

```
$ssh-keygen
```

COPY KEY TO CLIP BOARD

Mac

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

Windows

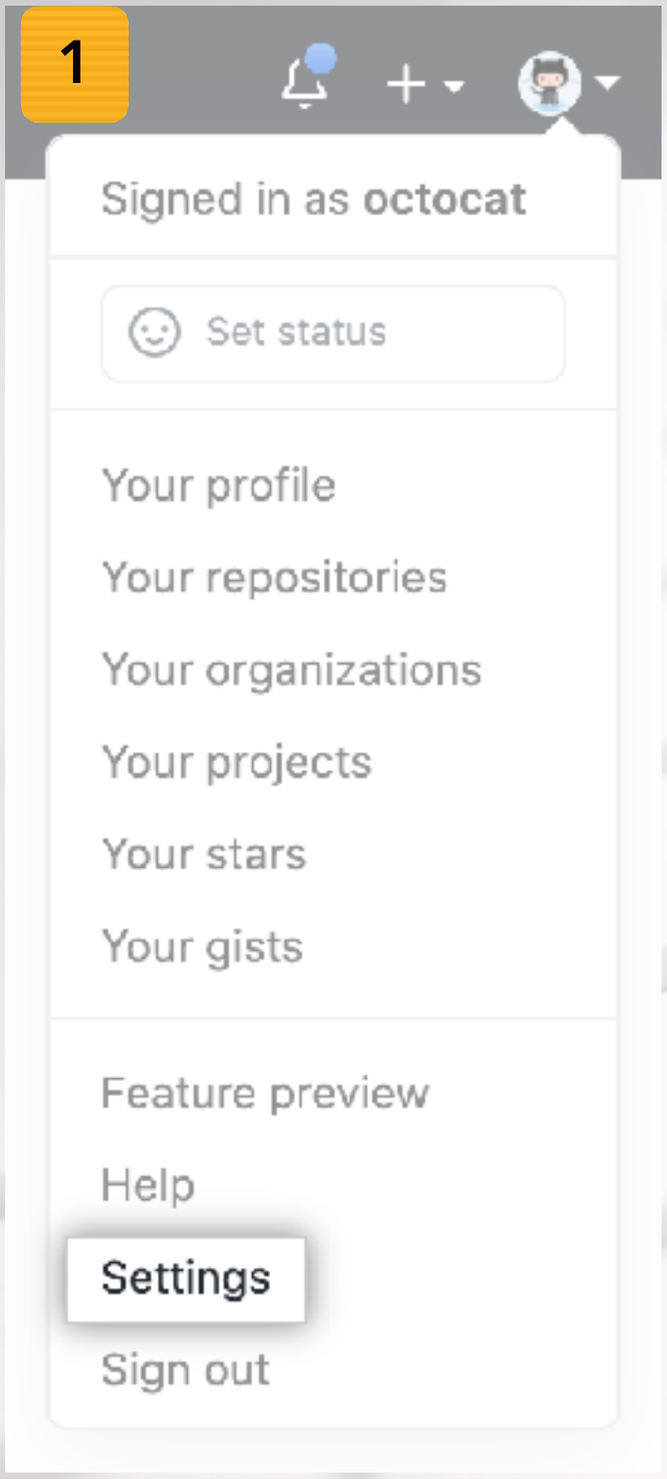
```
$clip < ~/.ssh/id_rsa.pub
```

Linux

```
$cat < ~/.ssh/id_rsa.pub
# Then select and copy the contents of the id_ed25519.pub file
# displayed in the terminal to your clipboard
```

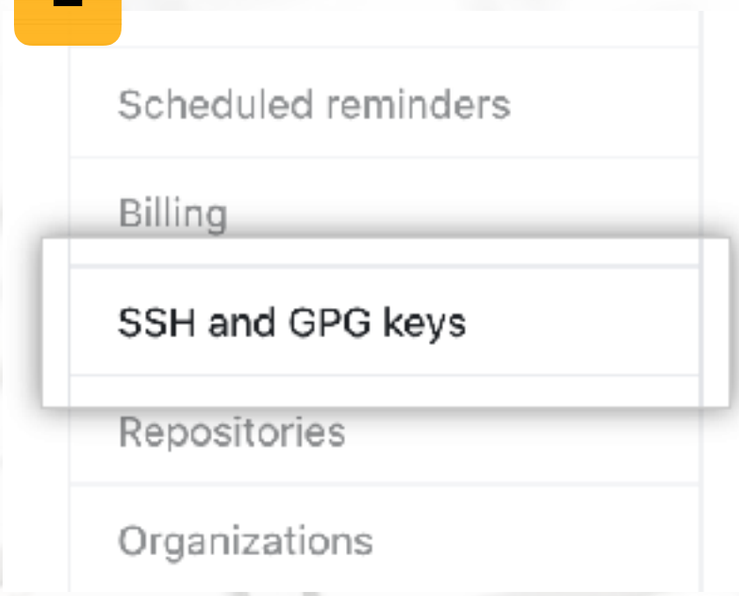
Generate ssh key and add to Github

1

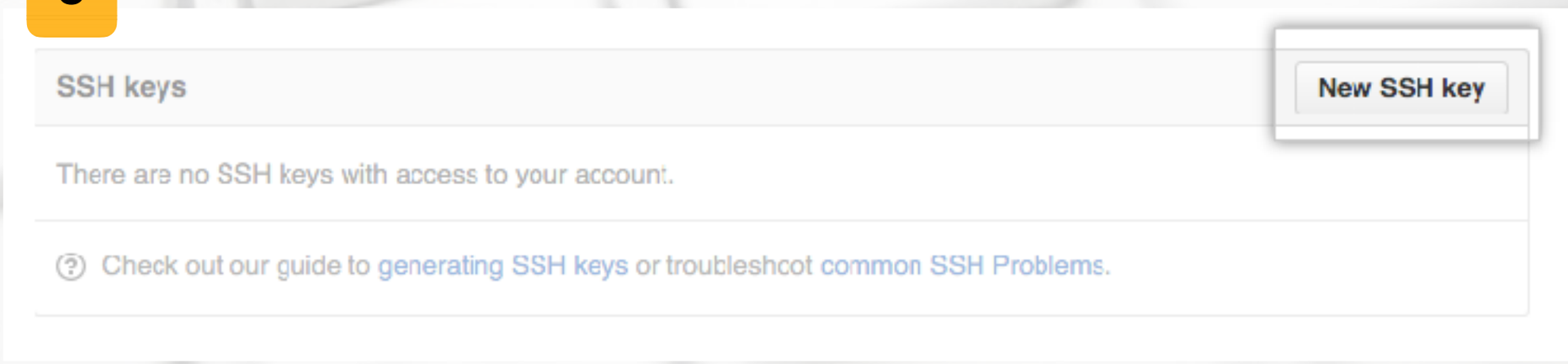


open web browser: go to https://github.com

2



3



Generate ssh key and add to Github

open web browser: go to https://github.com

4

SSH keys

New SSH key

There are no SSH keys with access to your account.

Title

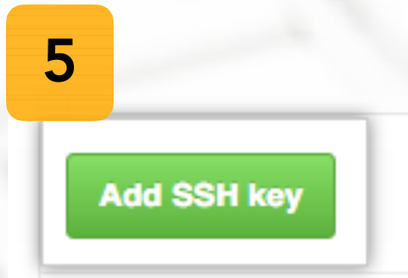
Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

5



6

Confirm password to continue

Password

Forgot password?

Confirm password

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)?
```

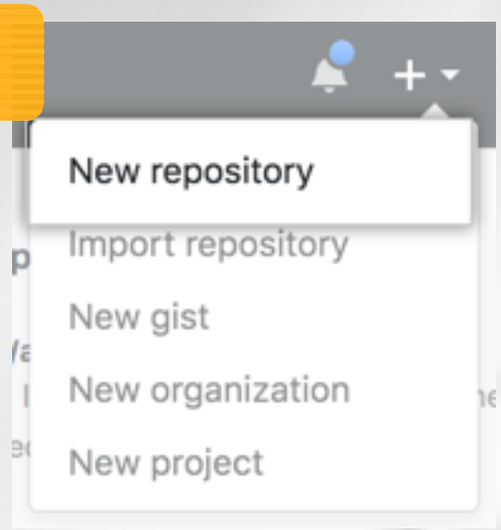
```
The authenticity of host 'github.com (192.30.252.1)' can't be established.
RSA key fingerprint is nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)?
```

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

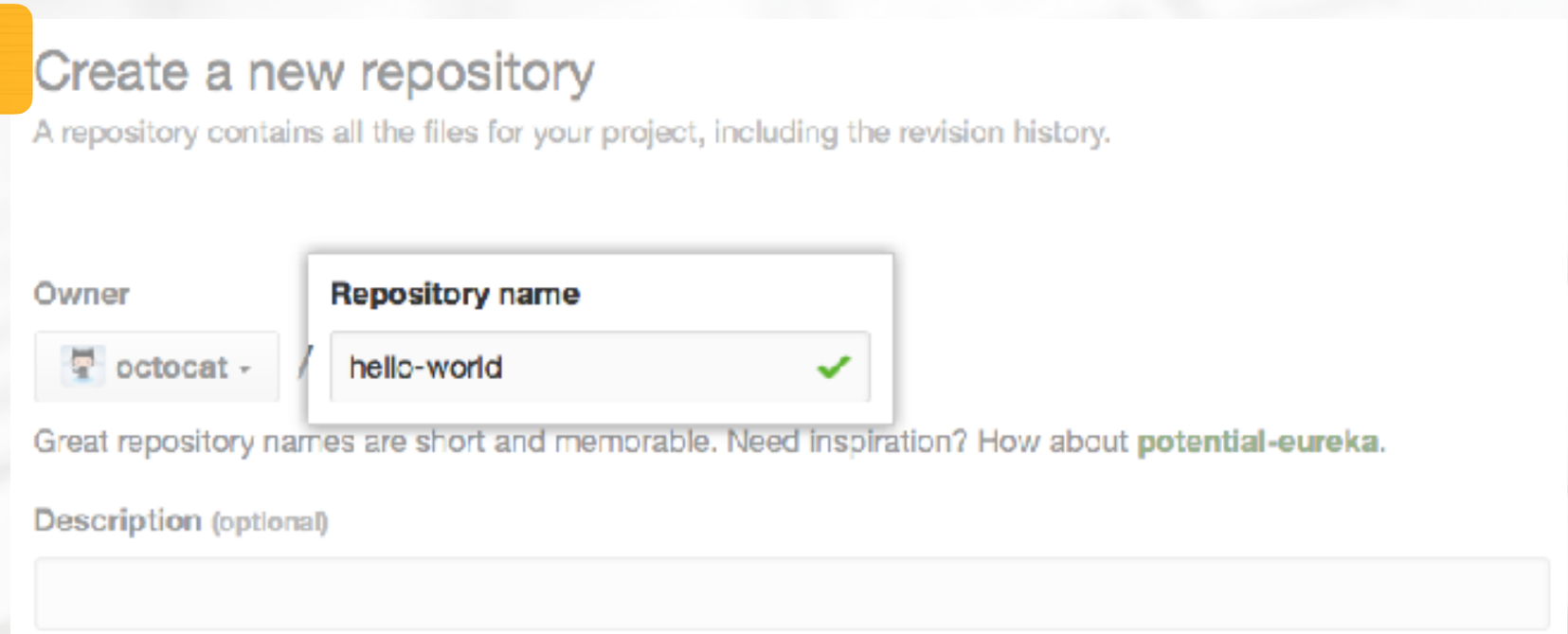
Create your repository

open web browser: go to https://github.com

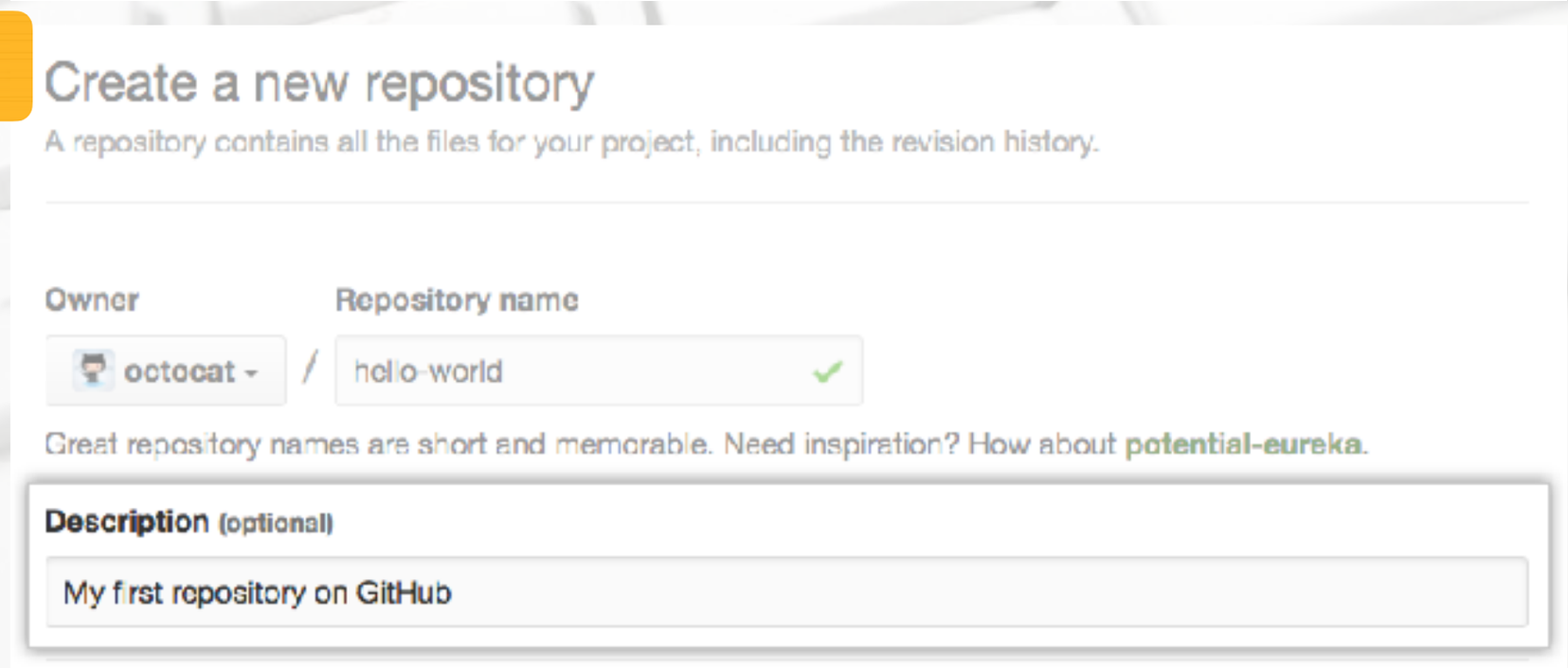
1



2



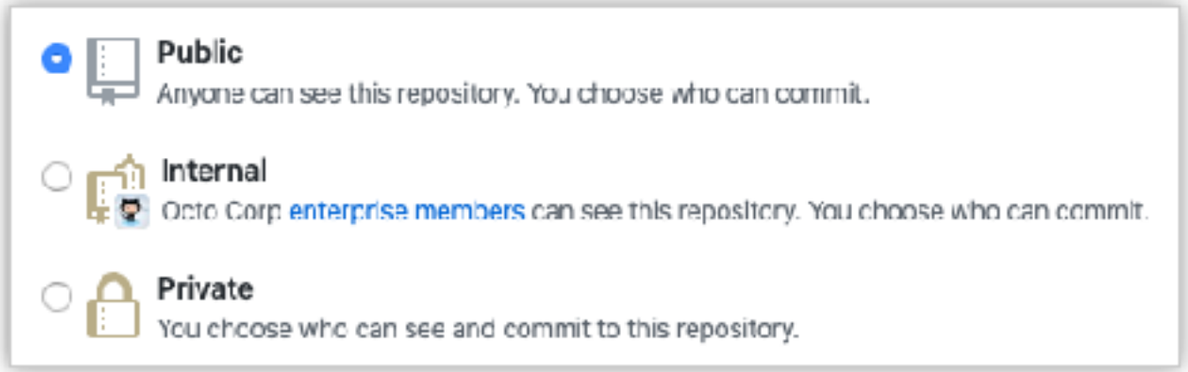
3



Create your repository

4

Description (optional)



Skip this step if you're importing an existing repository.

5

This will let you immediately clone the repository to your computer.

Add .gitignore: None

Add a license: None

Create repository

Create your repository

6

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:<accout name>/hello-world.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# hello-world" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:<account name>/hello-world.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:<account name>/hello-world.git
git branch -M main
git push -u origin main
```

Add and **Rename** remote

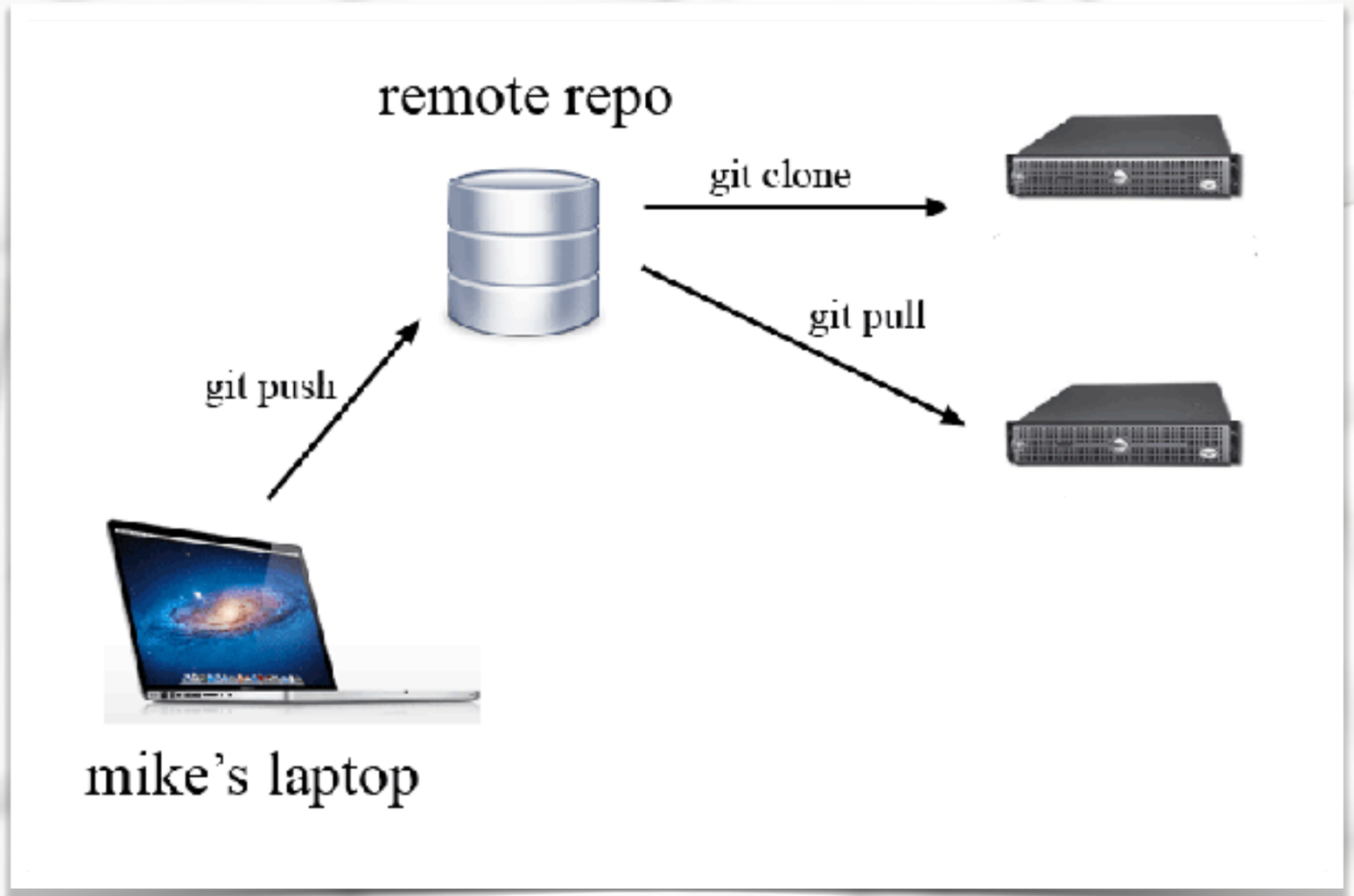
$git **remote add** [*NICKNAME] [REMOTE URL]*

$git **remote rm** [*NICKNAME]*

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

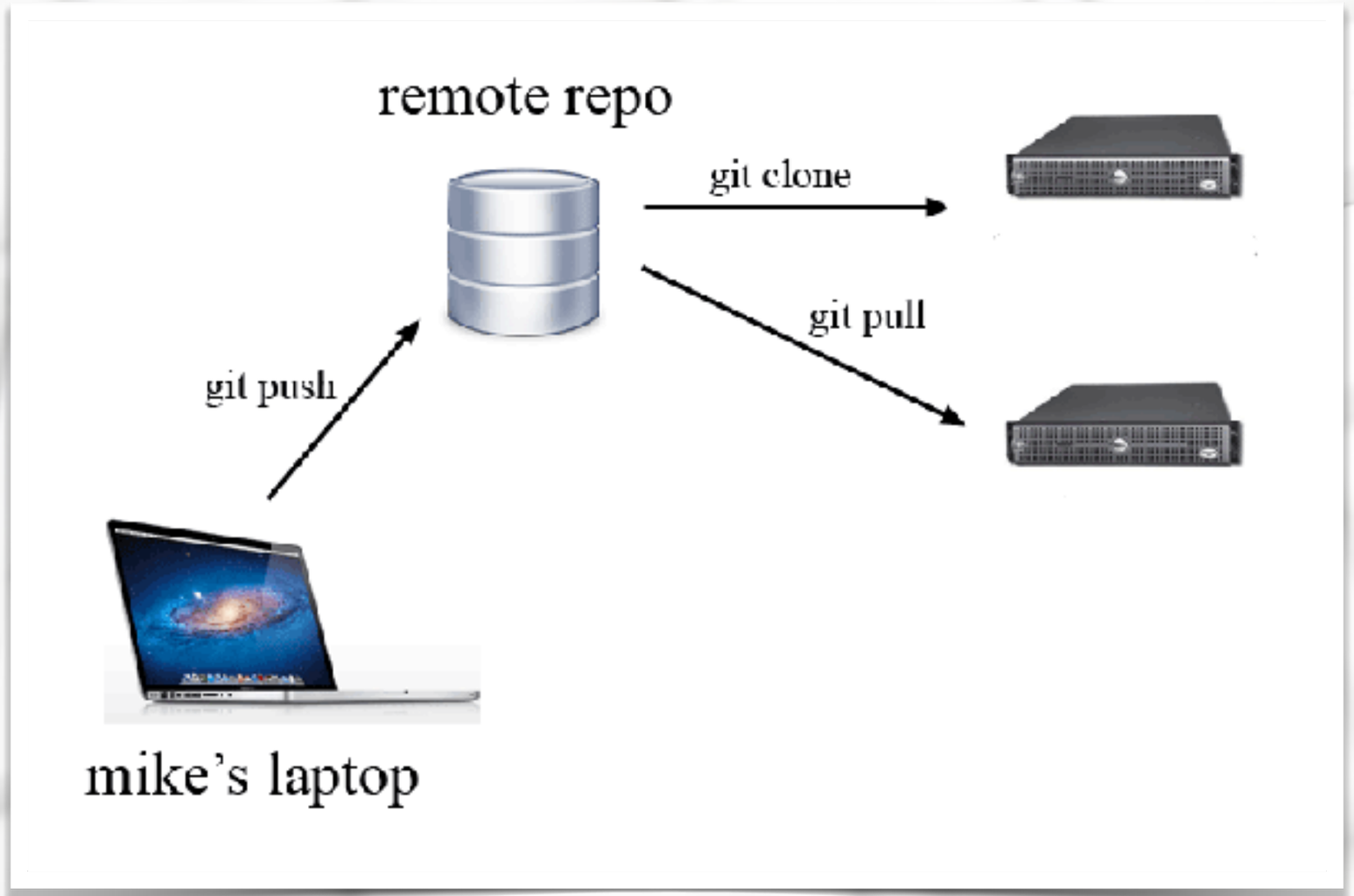
Push to remote

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



Fetch and **Merge**

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



Fetch and **Merge**

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

$git fetch [NICKNAME]

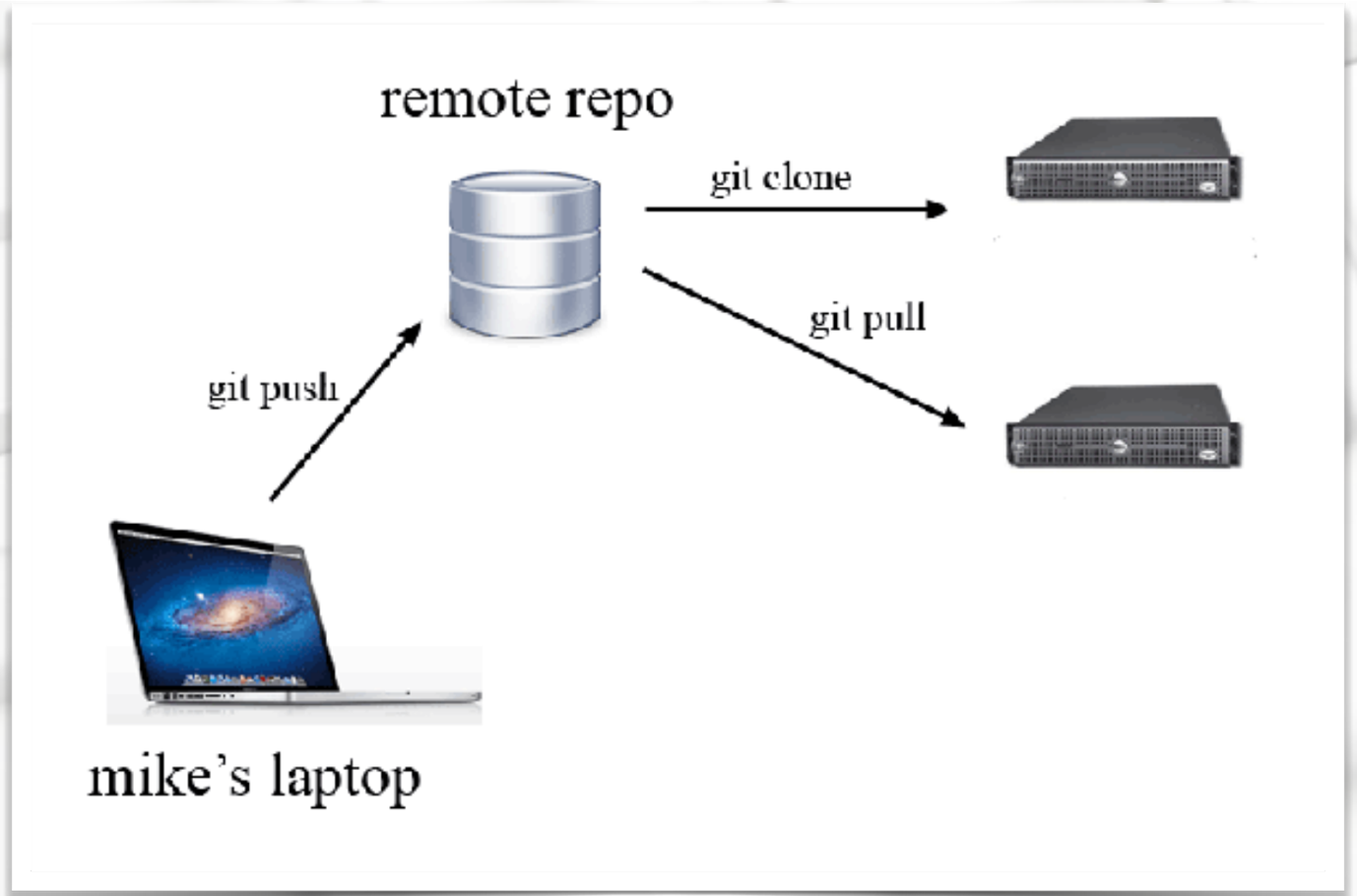
$git merge [BRANCH]

Clone project

$git **clone** *git-url*

$git **clone** *git@github.com:<account name>/hello-world.git*

$git **clone** https://*github.com/<account name>/hello-world.git*




TIPS

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**

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

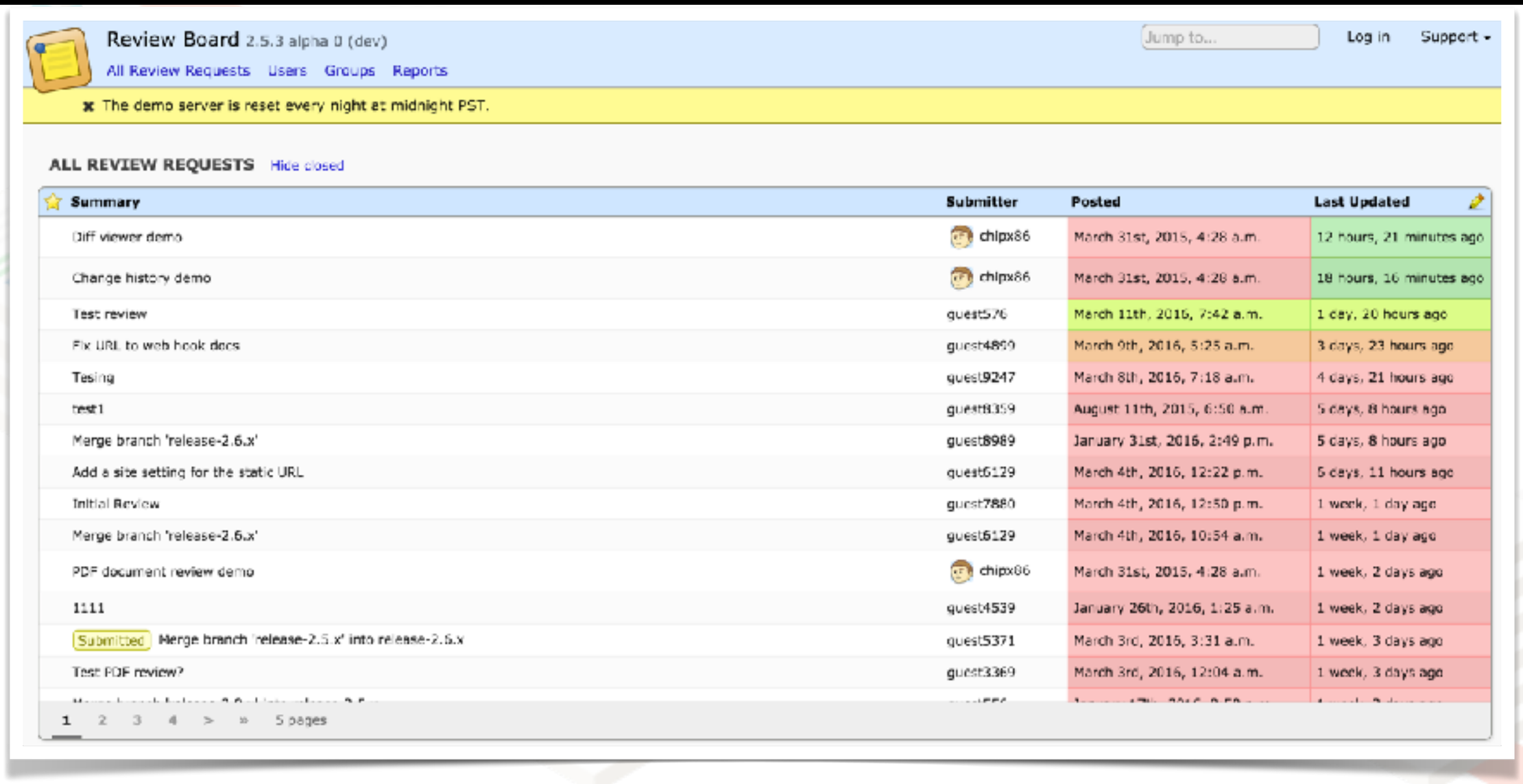
This is a list of software that helps software developers conduct and manage code reviews.

Software	Maintainer	Development status	License	VCS supported	Platforms supported	Workflow
Swascan	Swascan	actively developed	Proprietary	Any	Java, PHP, csharp, cee, erlang, flex, groovy, javascript, python, semgit, semsvn, 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

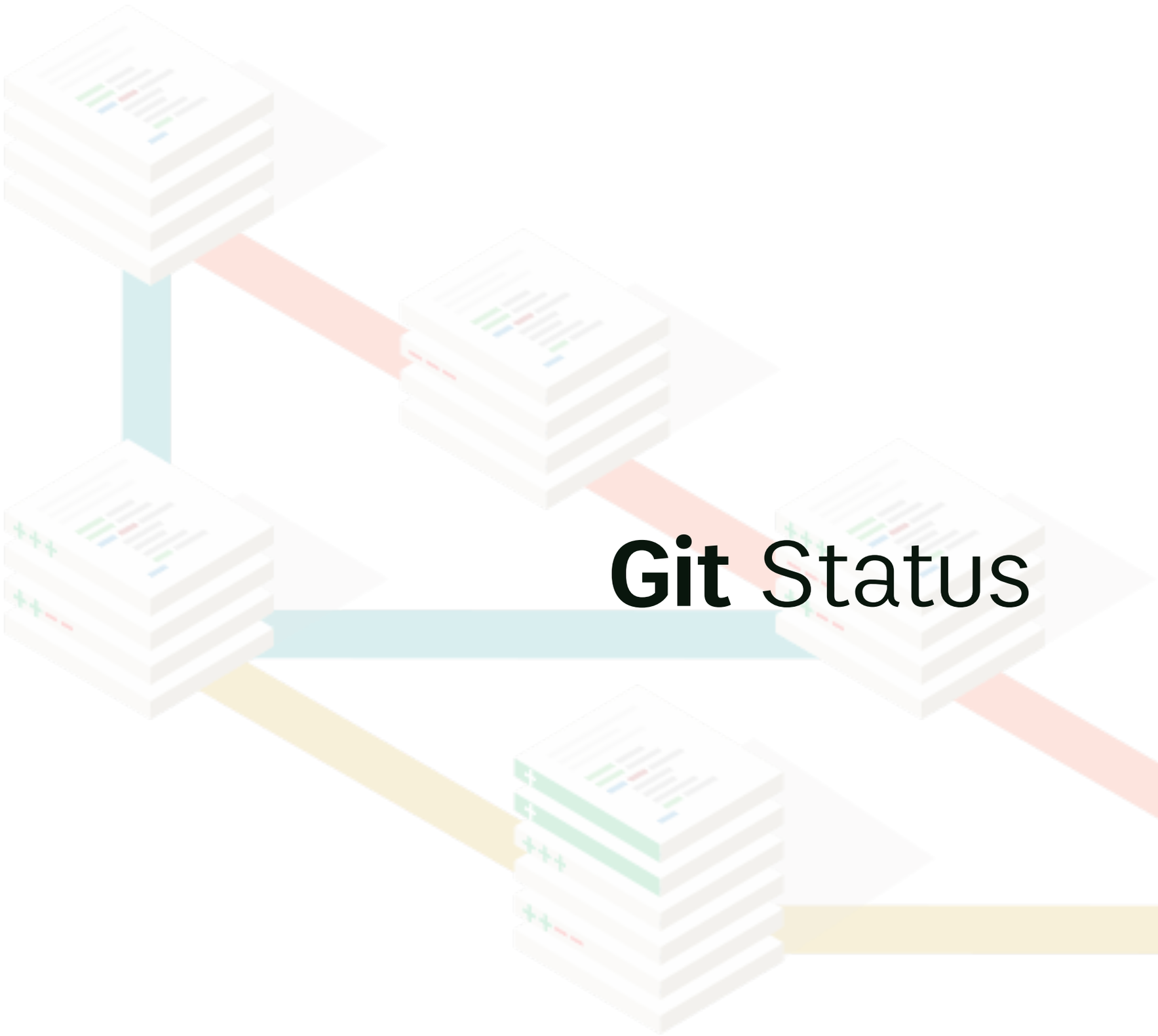
Tips Learn More

Professional review code to find out **ERRORS**



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

Git Log

Log in one line

$git **log --oneline**

Log in pretty mode

$git **log --oneline --decorate**

Log in graph mode

$git **log --oneline --graph**

See diff from log

$git **log --state**

$git **log -p**

Log by user

$git **log --author=<author name>**

Log by message

$git **log --grep=<message>**

Log in date range

$git **log -after=<YYYY-MM-DD> --before=<YYYY-MM-DD>**

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

Filter merge commit

$git **log --no-merges**

$git **log --merges**

Shortly

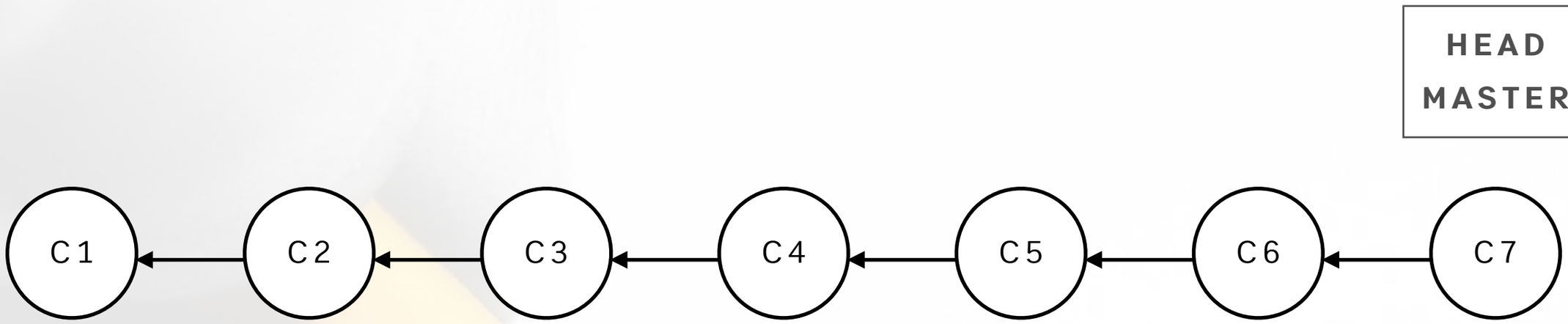
$git **shortly**

Help

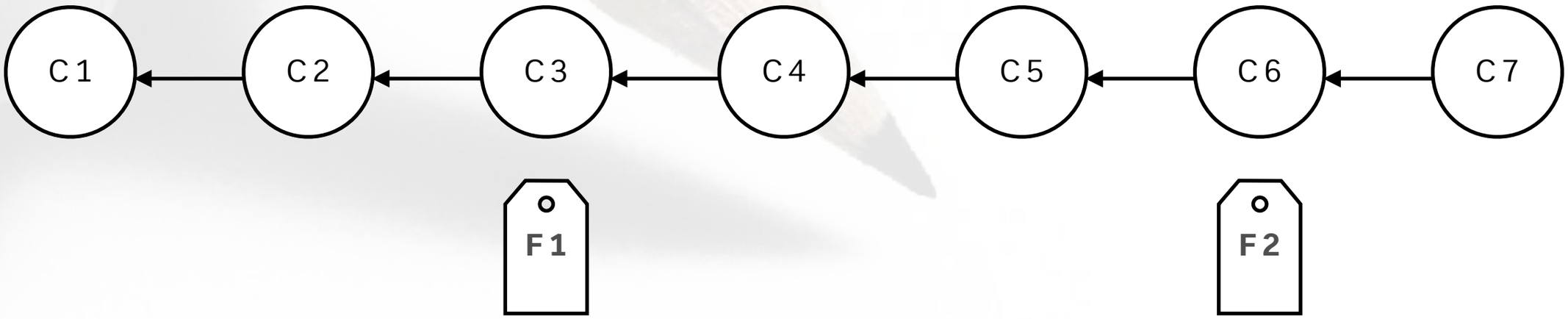
$git **help**

Git tags

Git tree



Git tag



Tag command[local]

Show tags

git tag

Add new tag

git tag <tag_name>

git tag -a <tag_name> <commit_id>

git tag -am "Reason for tag" <tag_name> <commit_id>

Remove tag

git tag -d <tag_to_remove>

Remove all tag

git tag | xargs -n1 git tag -d

Tag command[remote]

Share tags to remote

git push <remote_name> --tags

git push <remote_name> <tag_name>

Remove tag on remote

git push <remote_name> :<tag_to_remove>

git push --delete <remote_name> <tag_to_remove>

Fetch tags from remote

git fetch --all --tags

Git branch

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

master

keep master running

Need to fix bug ?

$git checkout **feature**

feature

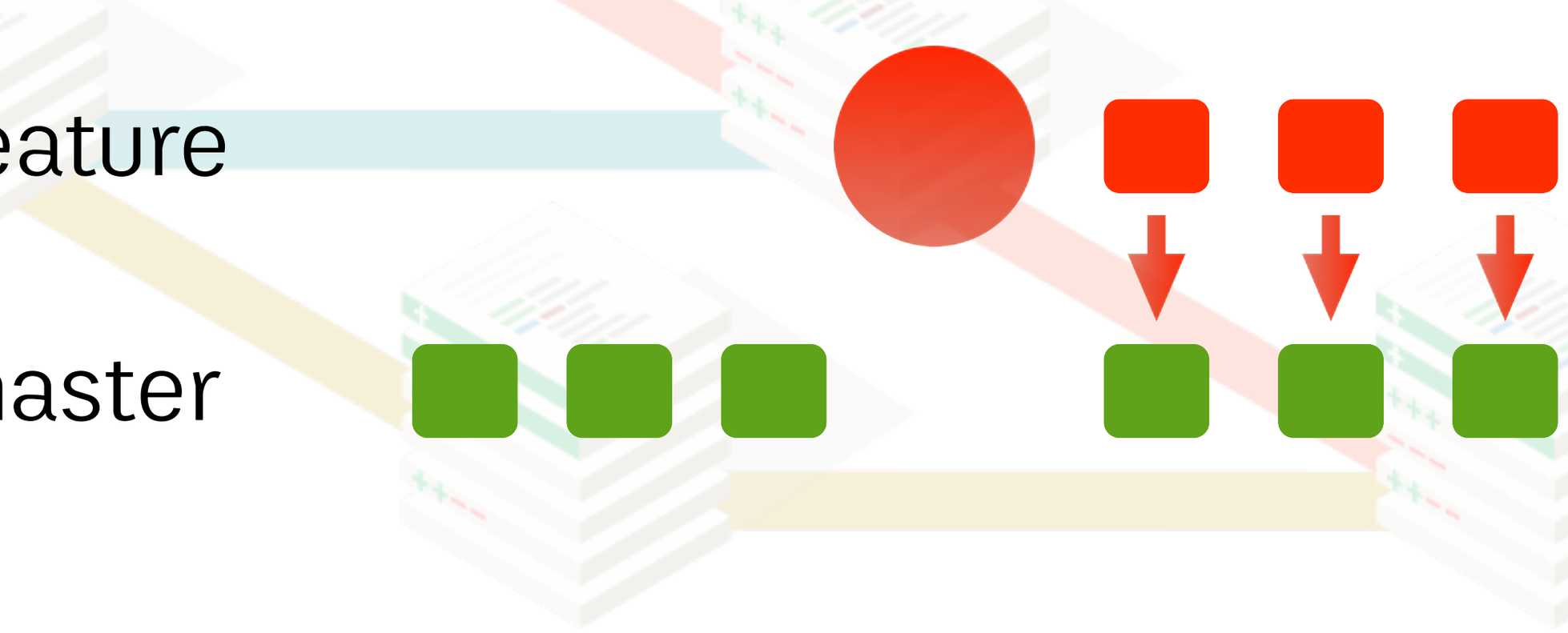
master

Git checkout -b **feature**

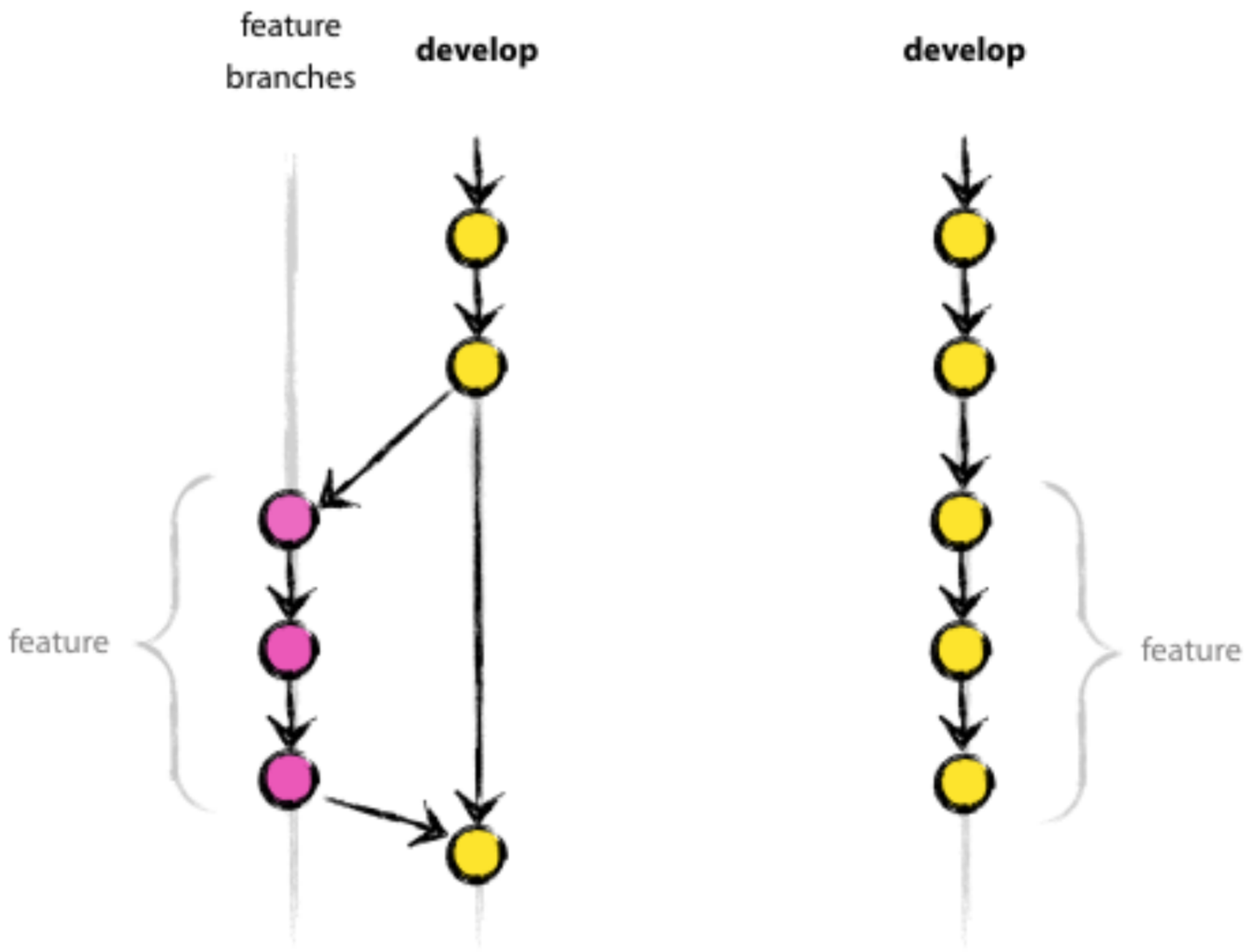
When ready to merge into master

How ?

$git merge **feature** master



$git merge --no-ff



branching is cheap

painless development

delete branch **if it’s not work**

delete branch **if it’s not use**

Branching Strategy

Maintainable Git

CODE is KING

Developer cost

$$$$$$$$$$

Developer cost

$$$$$$$$$$

Branching cost

(~40 bytes) $

Poor branching strategy

$$$$$$$$$$$$$$$$$$$$

branching is cheap

painless development

delete branch **if it’s not work**

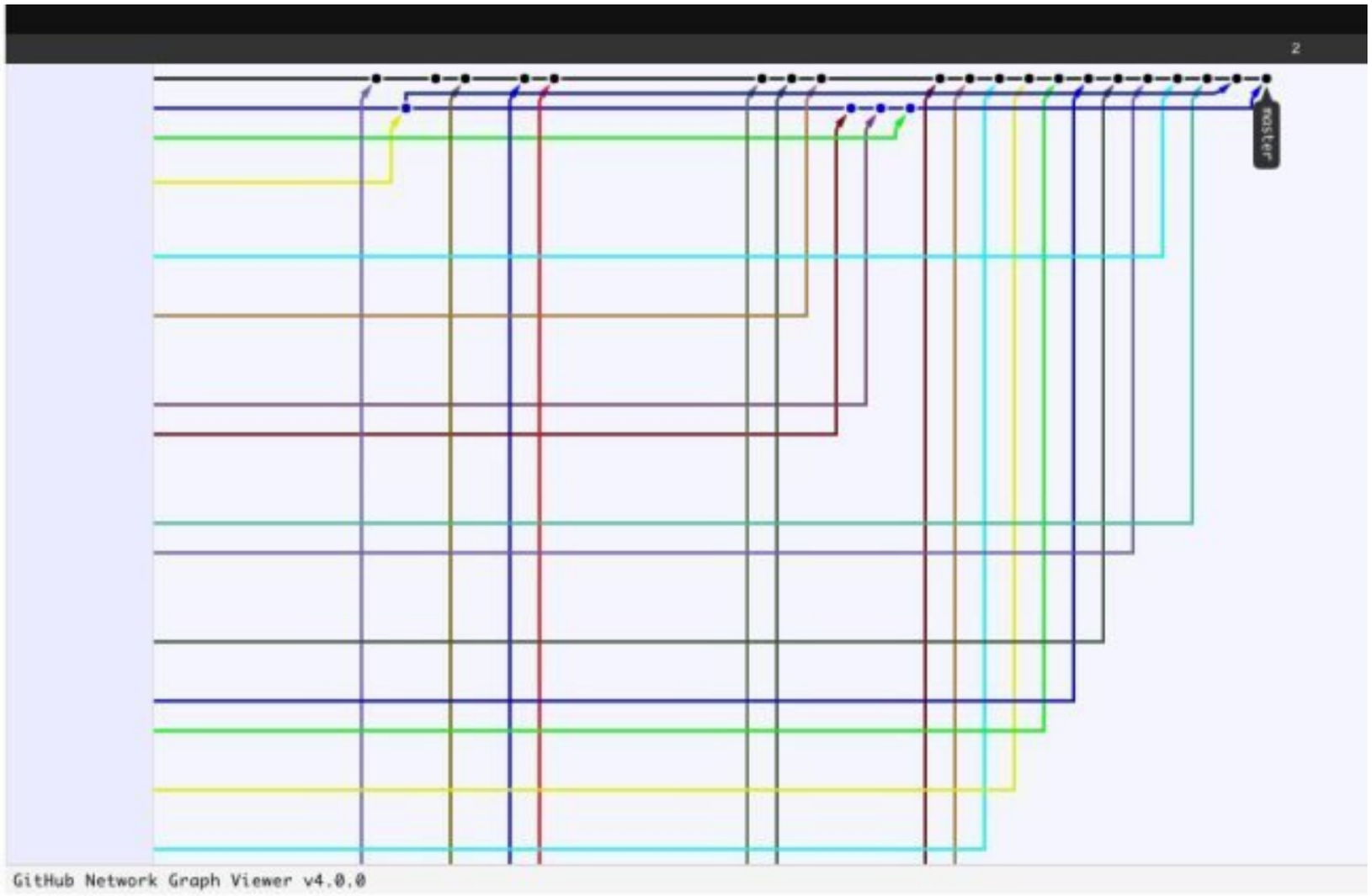
delete branch **if it’s not use**

Branching strategies

What is being worked on ?

How do you switch tasks ?

When do you integrated ?


2
master
GitHub Network Graph Viewer v4.0.0

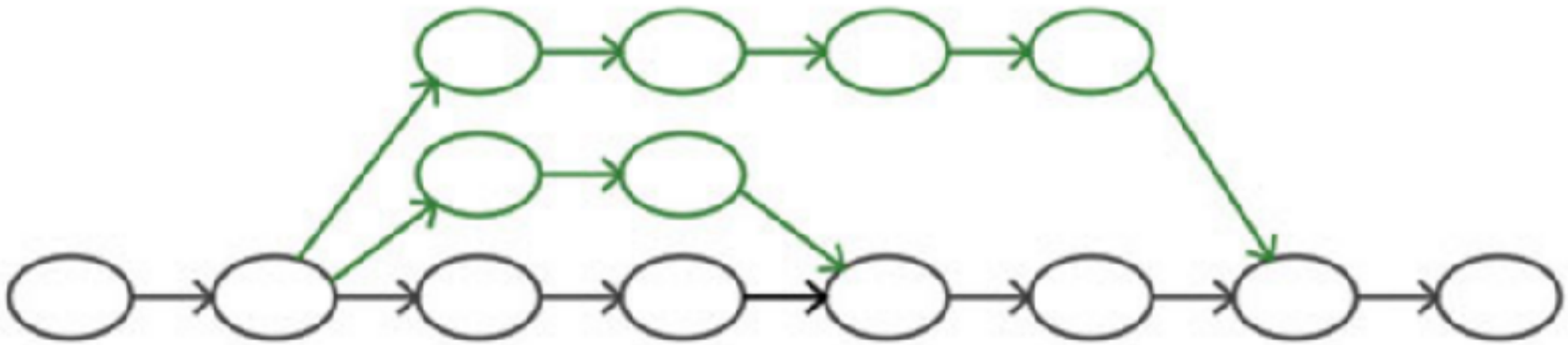
ทำอย่างไรดี ?

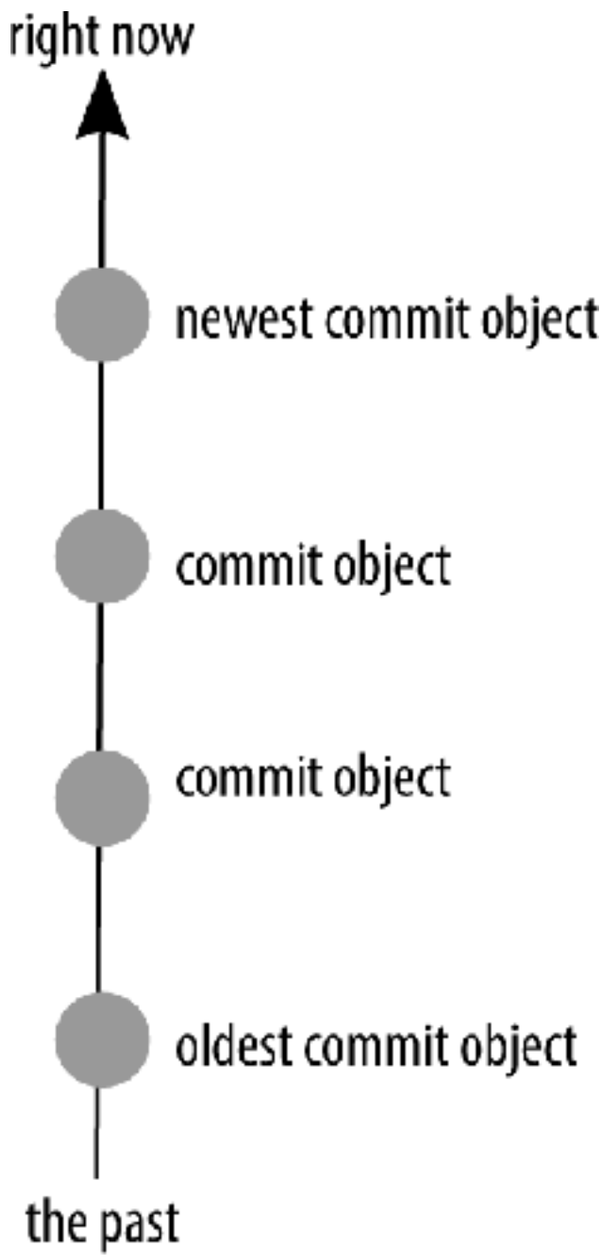
Release Process ?

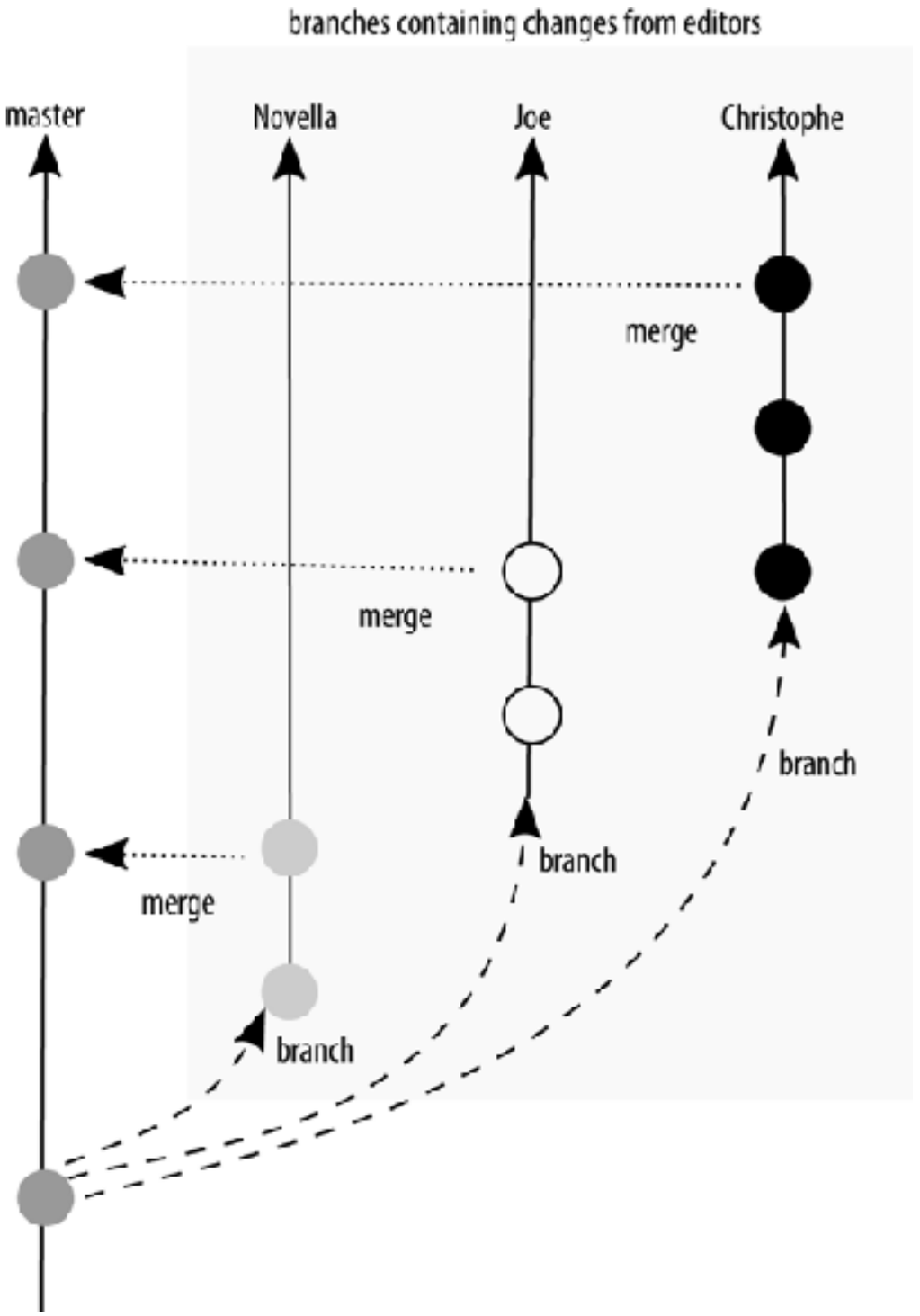
Create a new branch

when working on any feature, issue or bug

Merged back when it is finished

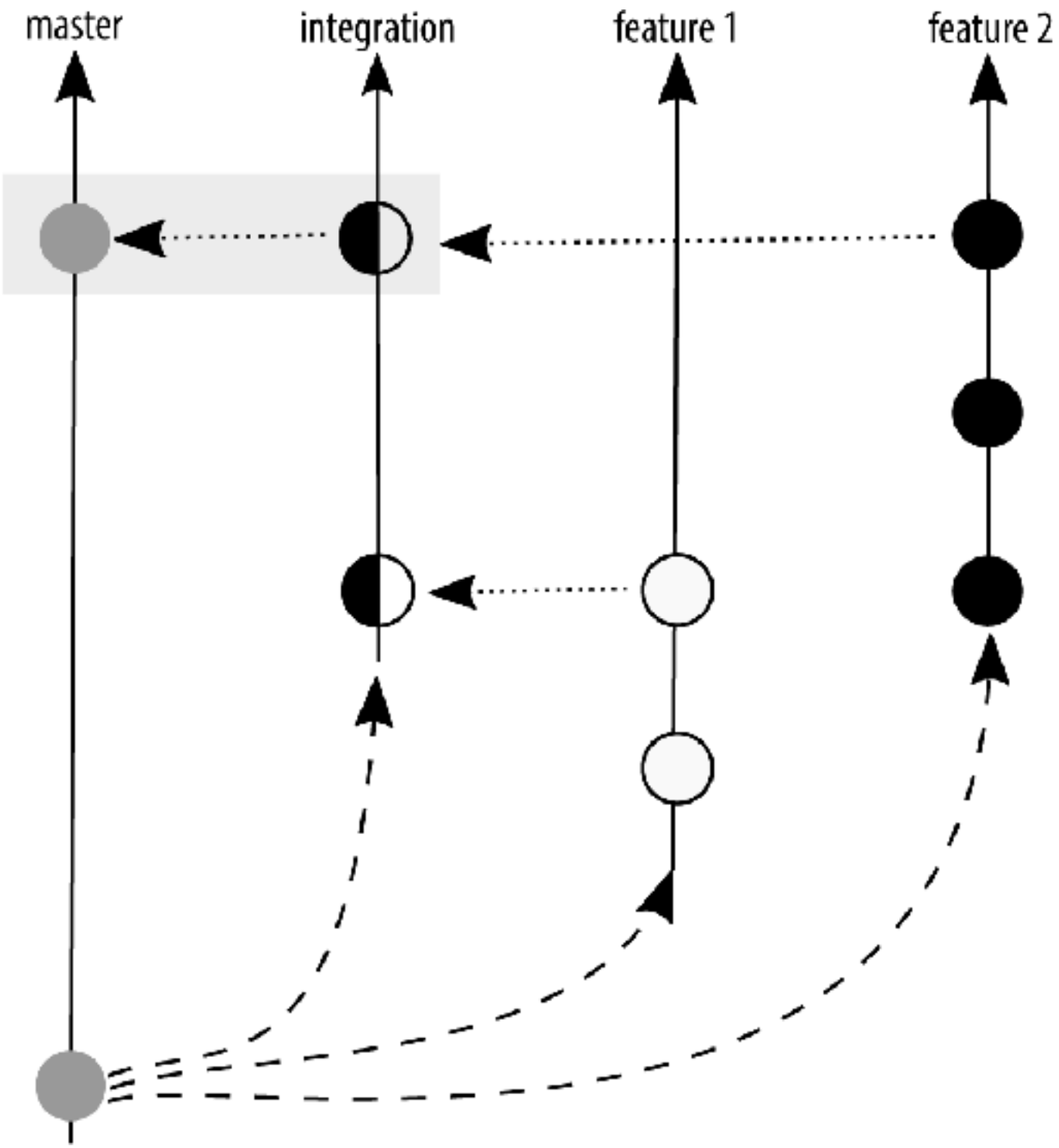


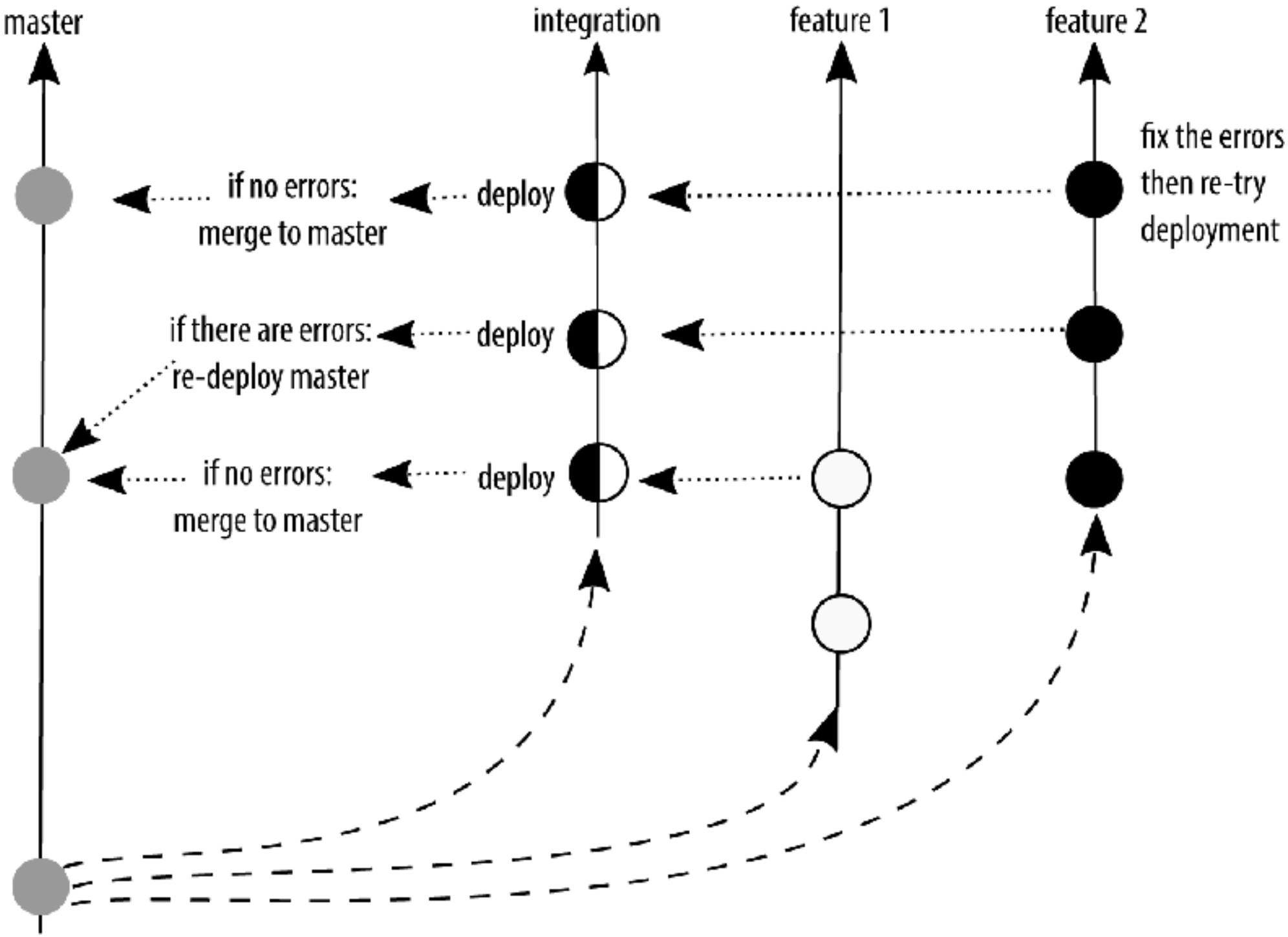

right now
newest commit object
commit object
oldest commit object
the past


branches containing changes from editors
master
Novella
Joe
Christophe
merge
merge
branch
branch
merge
branch

Let’s discuss with Your team

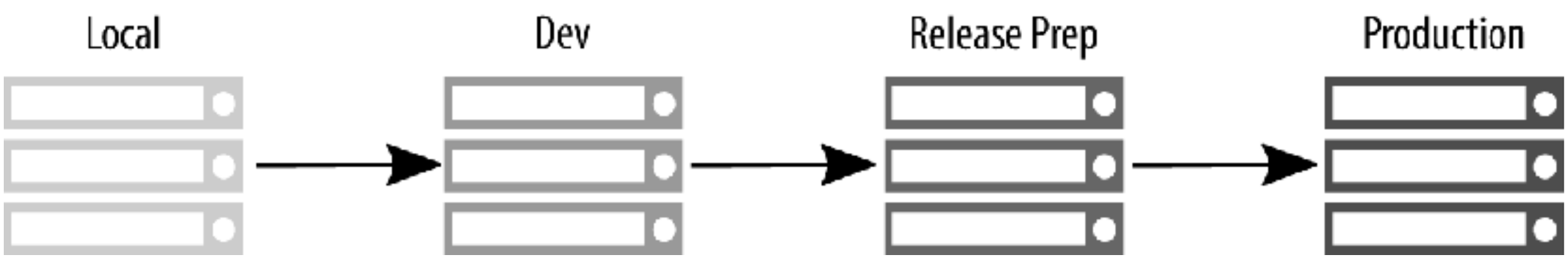
Assuming feature 1 and feature 2 integrate well: merge the combined changes in the integrration branch into the master branch.

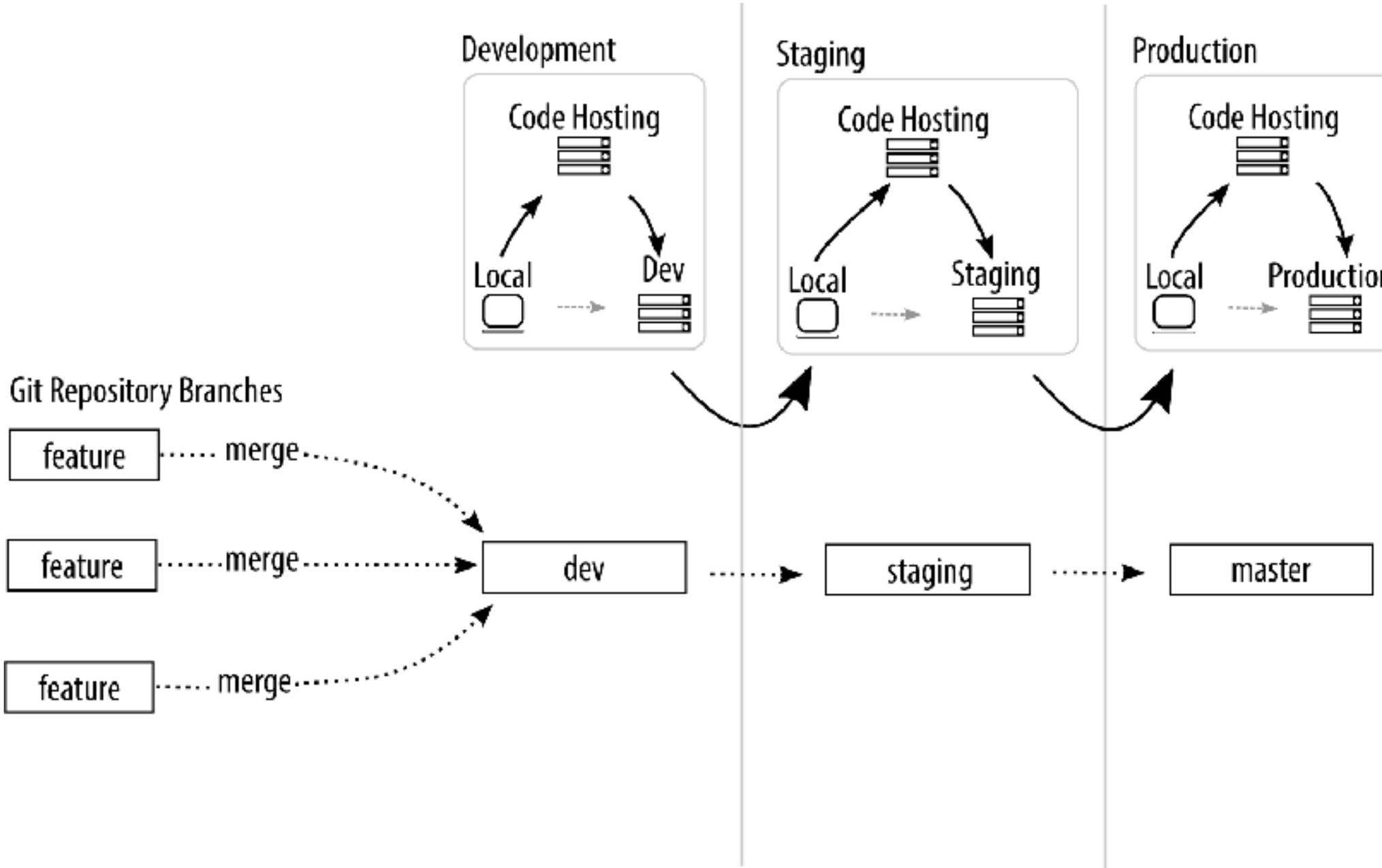



master
integration
feature 1
feature 2
fix the errors
then re-try
deployment
if no errors:
merge to master
deploy
if there are errors:
re-deploy master
deploy
if no errors:
merge to master
deploy

Let’s discuss with Your team

One branch per platform

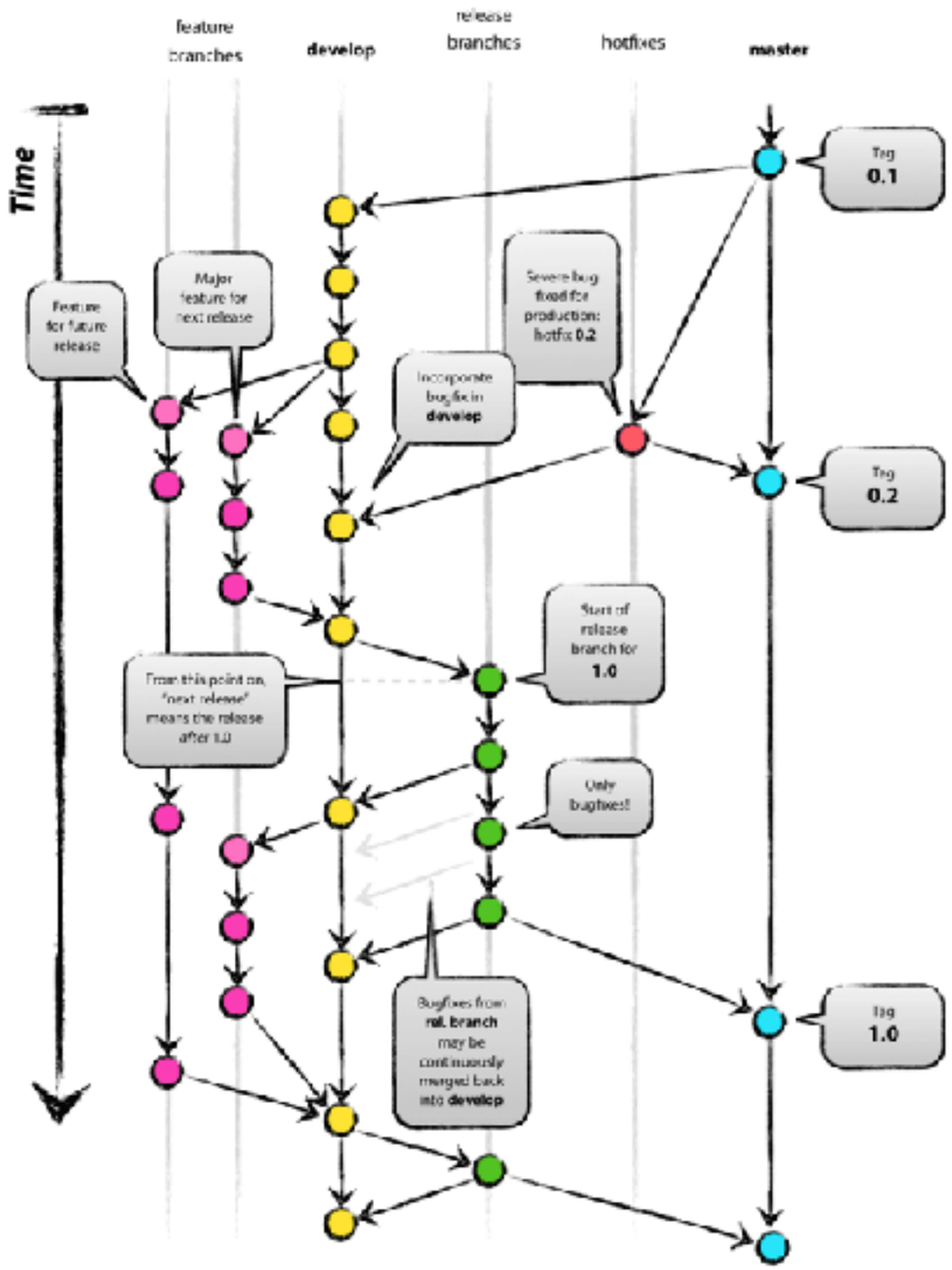



Development
Code Hosting
Local
Dev
Staging
Code Hosting
Local
Staging
Production
Code Hosting
Local
Production
Git Repository Branches
feature
merge
feature
merge
feature
merge
dev
staging
master

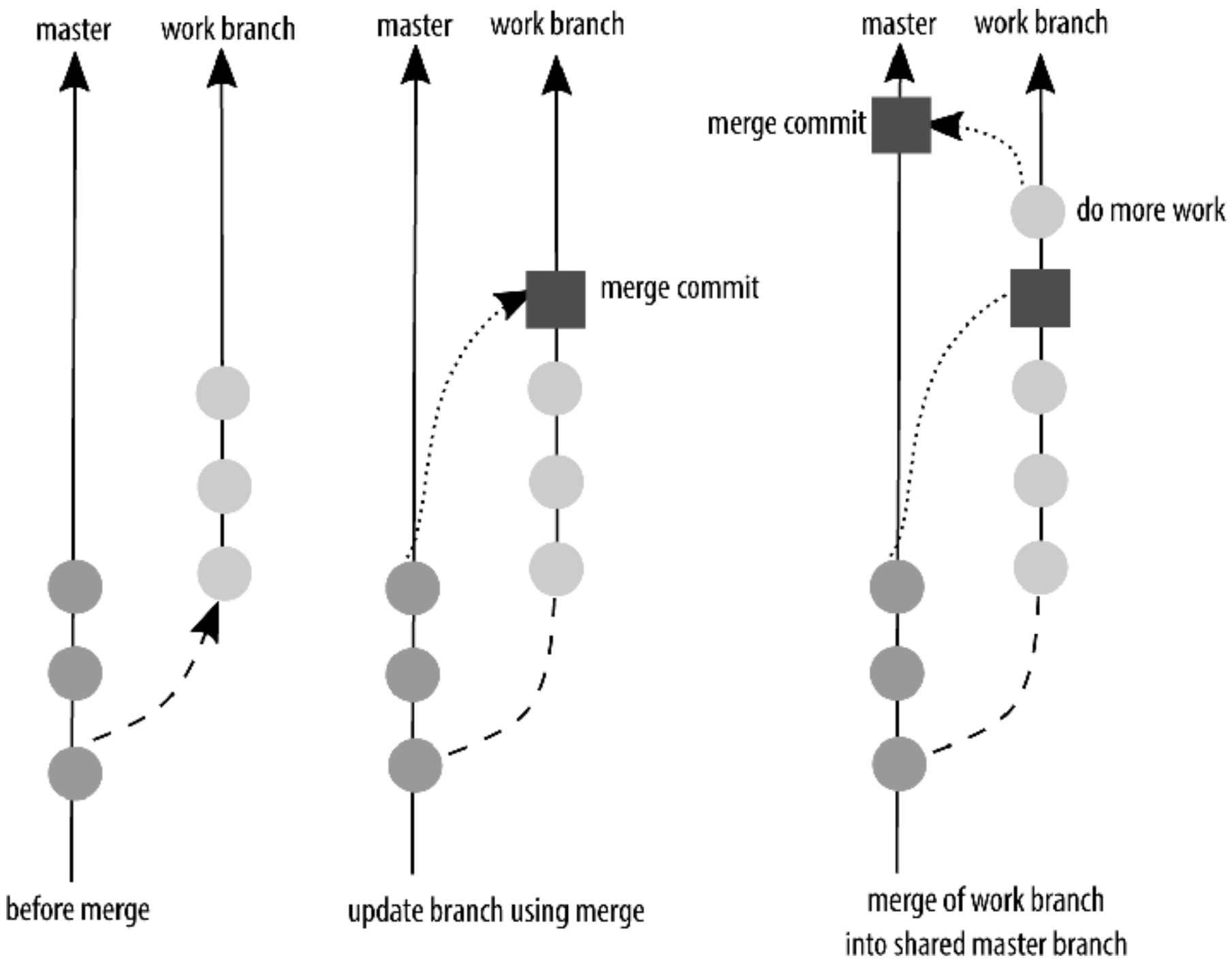
Let’s discuss with Your team

Git flow

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



Let’s discuss with Your team


master
work branch
before merge
master
work branch
merge commit
update branch using merge
master
work branch
merge commit
do more work
merge of work branch
into shared master branch

Branch by ENV

vs

Feature Toggles

Requirements

/api/v1/greeting?name=<yourname>

Method: GET

Response:

Content-Type: text/html

Body: "Hello, <yourname>"

/api/v2/greeting

Method: POST

Content-Type: application/json

Body: { "name": "<yourname>" }

Response:

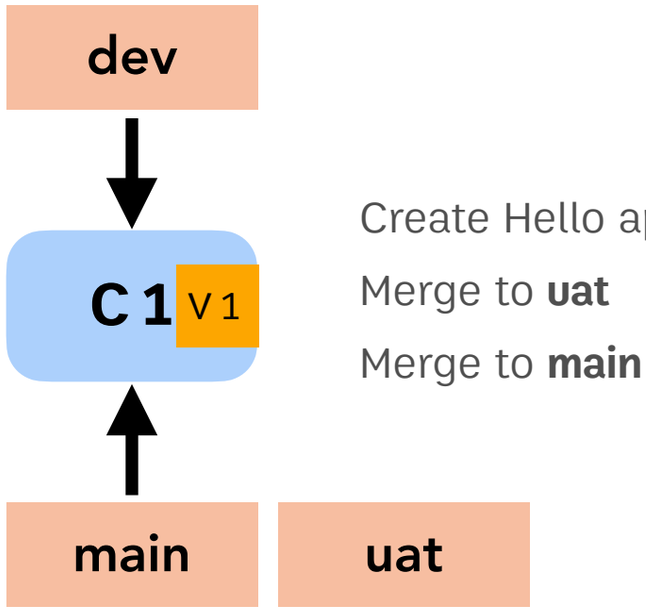
Content-Type: application/json

Body: { "message": "Hello, <yourname>" }

Timeline

1. Create Hello api/v1: on **dev**
    1. Merge to **uat**
    2. Merge to **main**
2. Create Hello api/v2: on **dev**
    1. Merge **dev** to **uat**
3. Fixed bug "add **!** to message" on **uat**
    1. Merge **fixed bug** to **main** and **dev**
4. Move message func to greeting package
5. Fixed bug "add **,** after Hello" on **main**
    1. Merge **fixed bug** to **uat** and **dev**
6. Refactor **dev** and merge **dev** to **uat**

Timeline

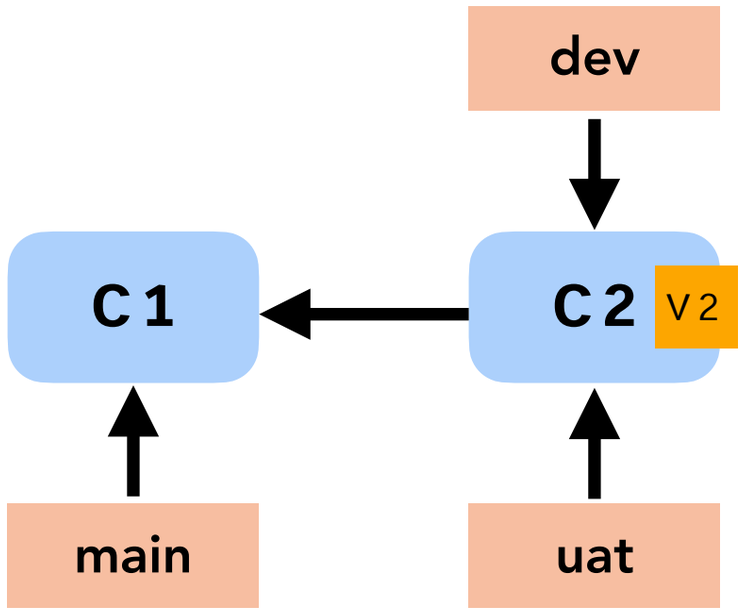


Create Hello api/v1: on **dev**

Merge to **uat**

Merge to **main**

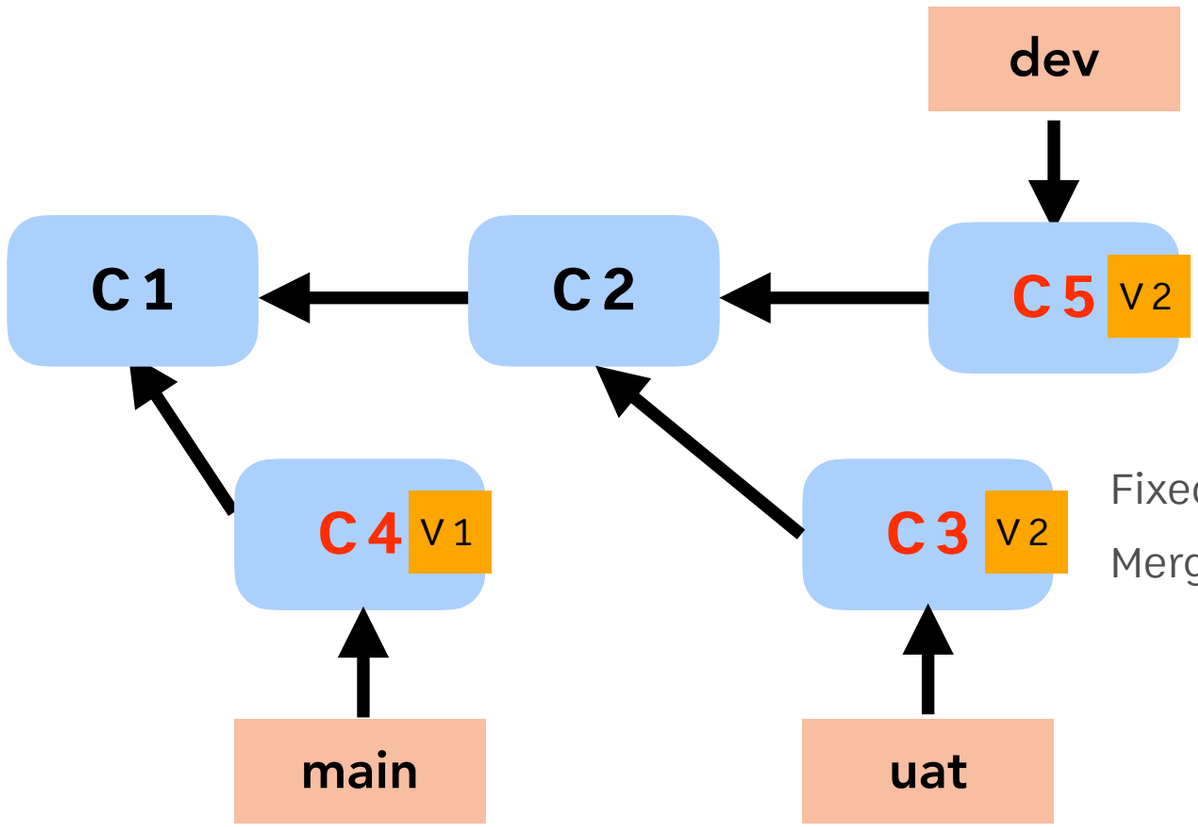
Timeline



Create Hello api/v2: on **dev**

Merge **dev** to **uat**

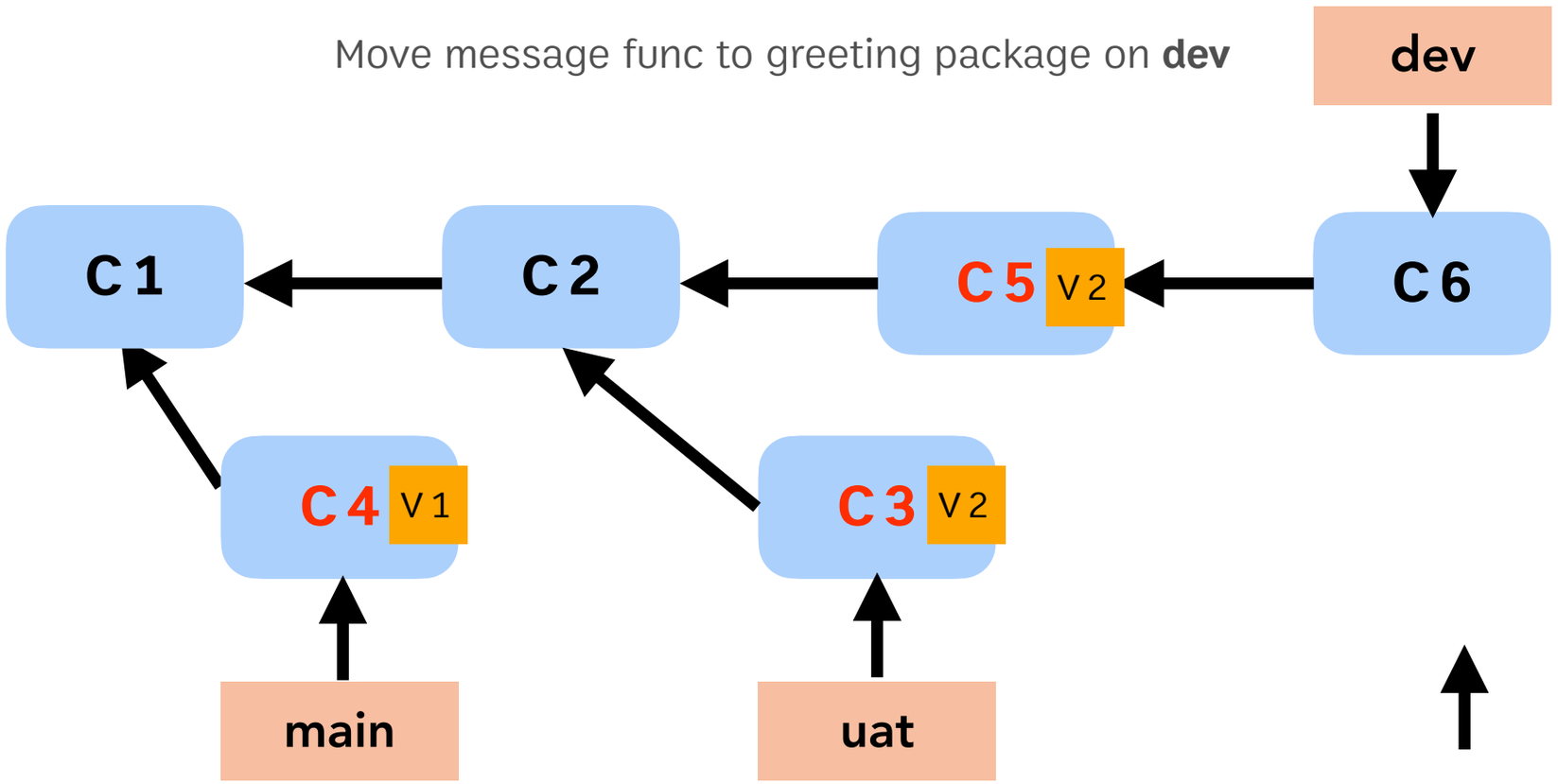
Timeline



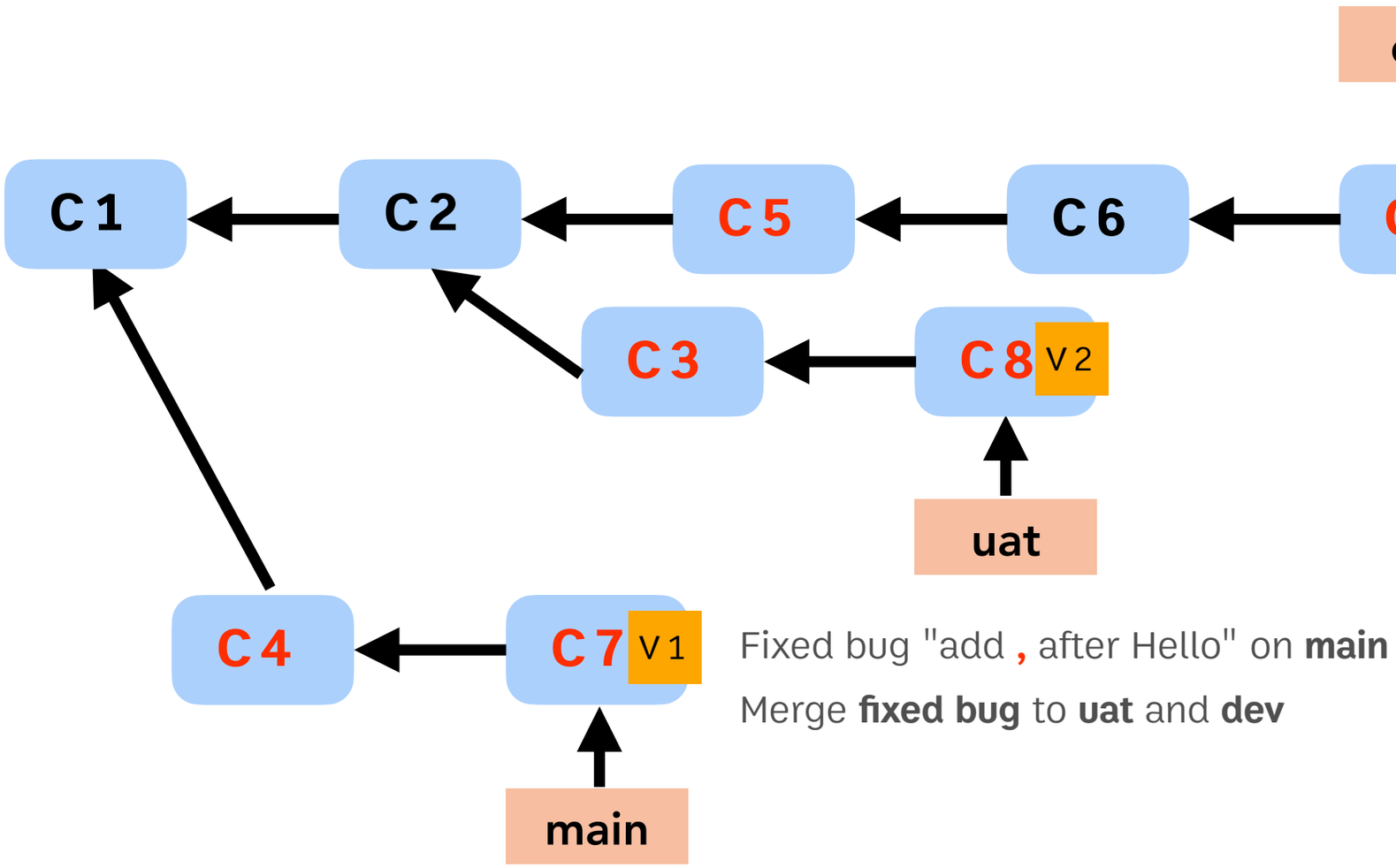
Fixed bug "add **!** to message" on **uat**

Merge **fixed bug** to **main** and **dev**

Timeline


Move message func to greeting package on **dev**
dev
C1
C2
C5 V2
C6
C4 V1
C3 V2
main
uat

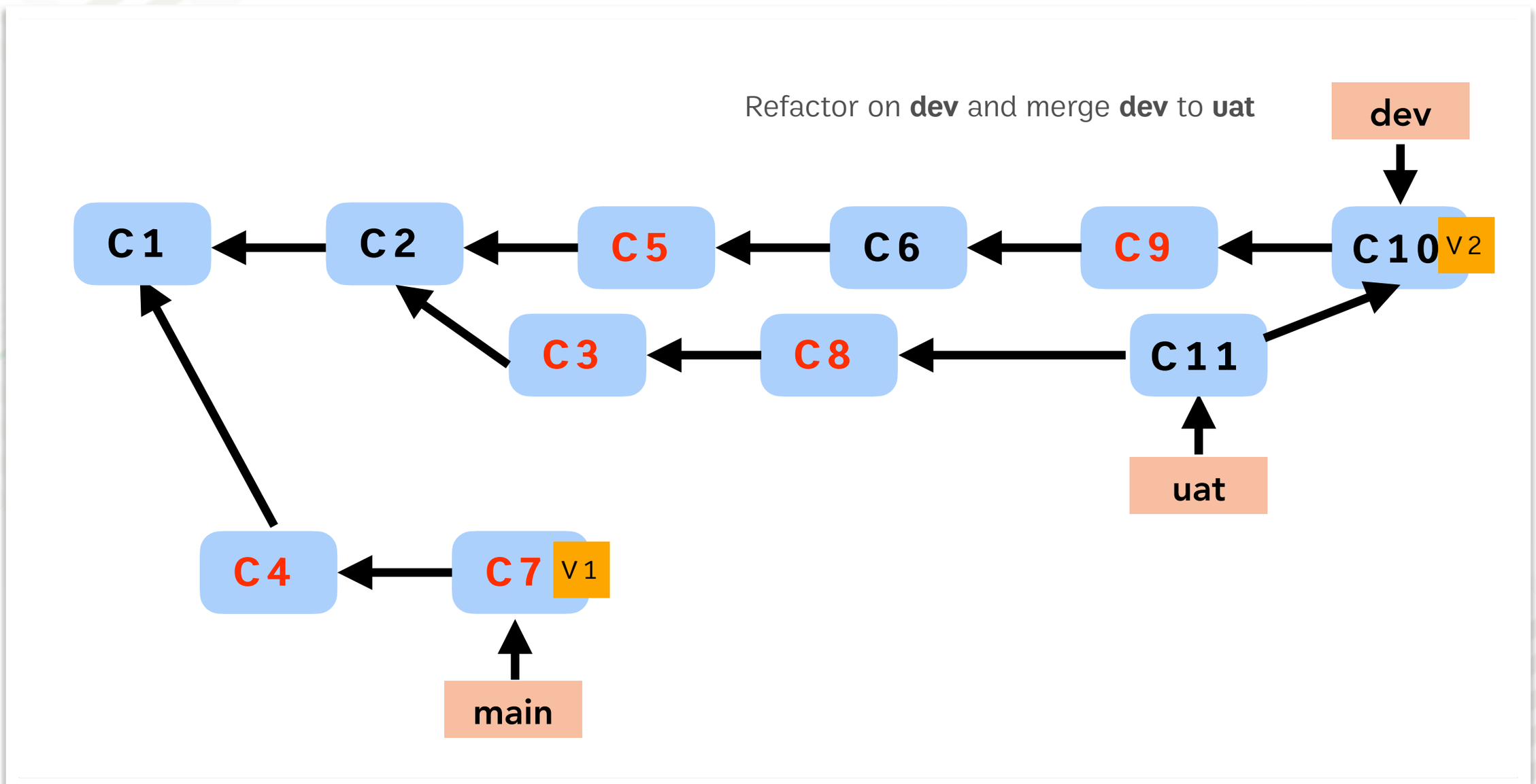
Timeline



Fixed bug "add , after Hello" on **main**

Merge **fixed bug** to **uat** and **dev**

Timeline


Refactor on dev and merge dev to uat
dev
C1
C2
C5
C6
C9
C10
V2
C3
C8
C11
C4
C7
V1
uat
main

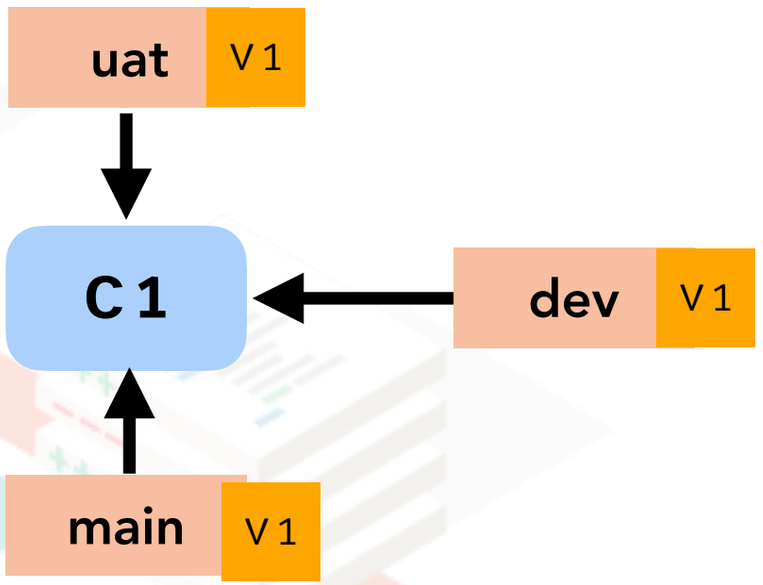
Timeline: Toggle

Create Hello api/v1:

release to **dev**

release to **uat env**

release to **main env**

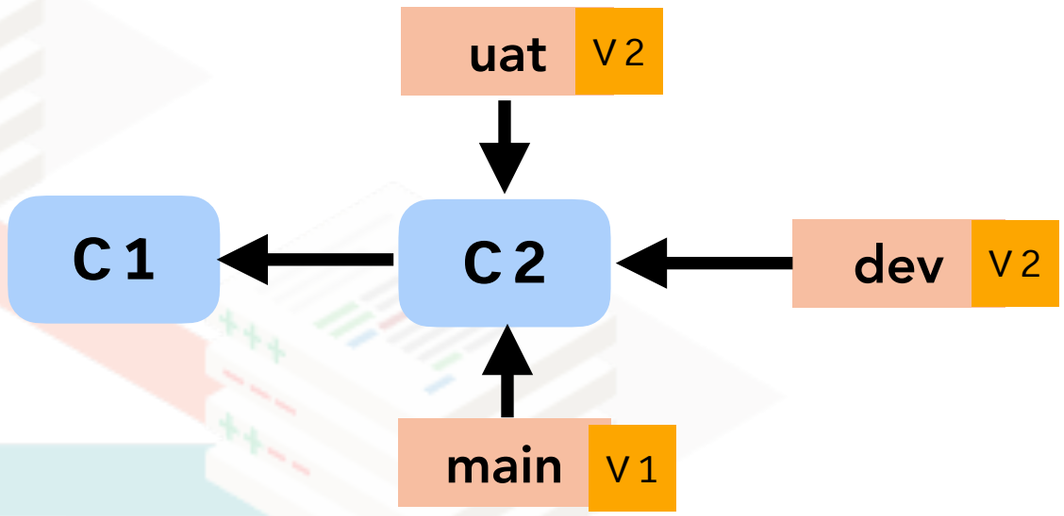


Timeline: Toggle

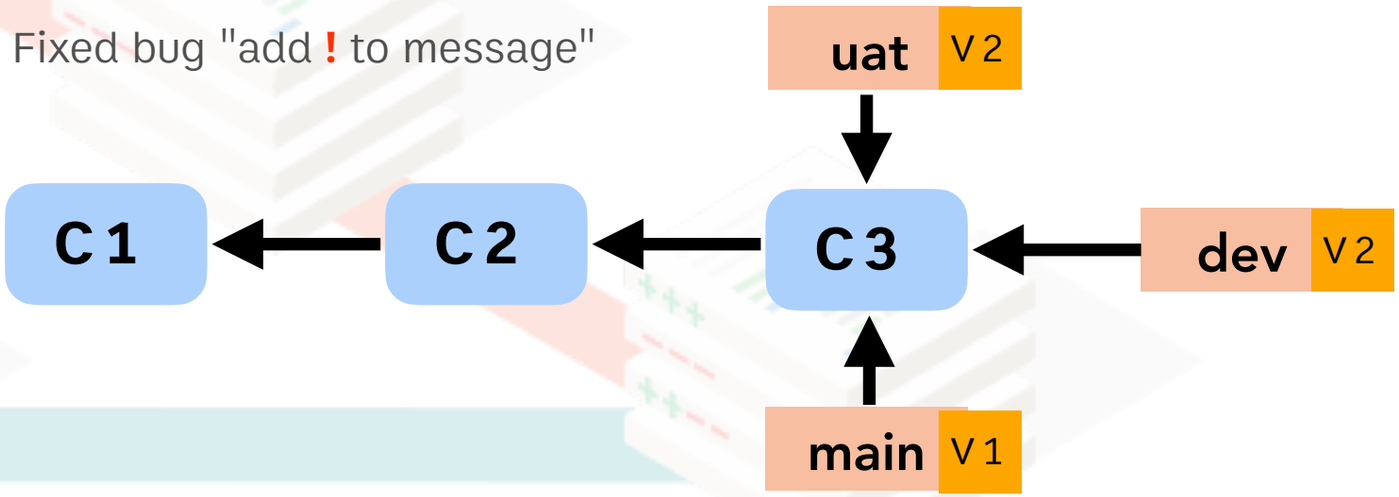
Create Hello api/v2:

release to **dev**

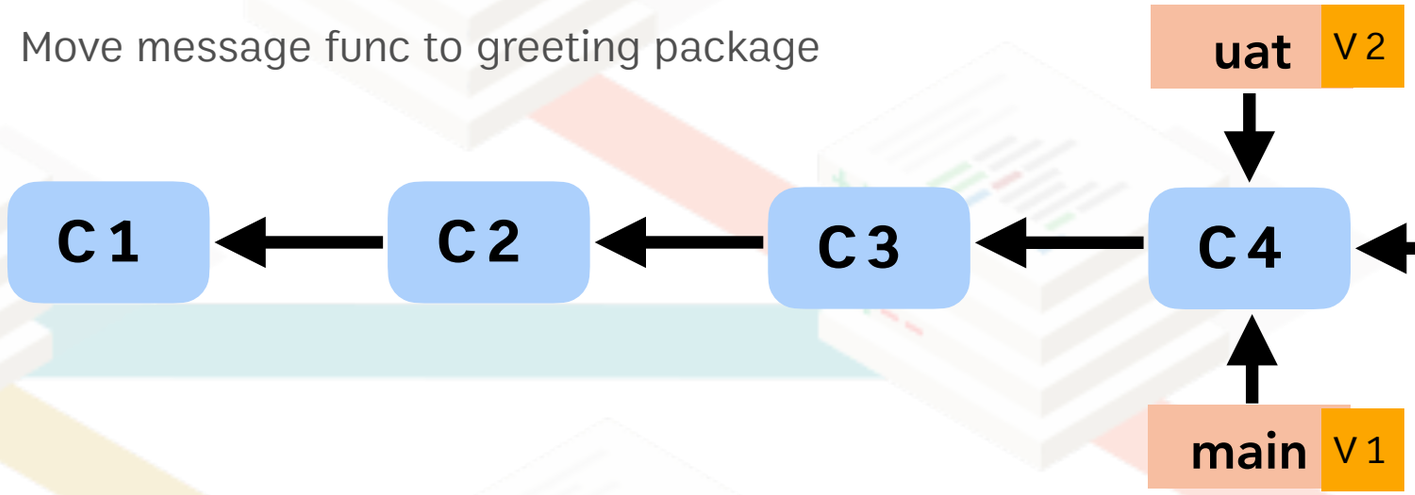
release to **uat**



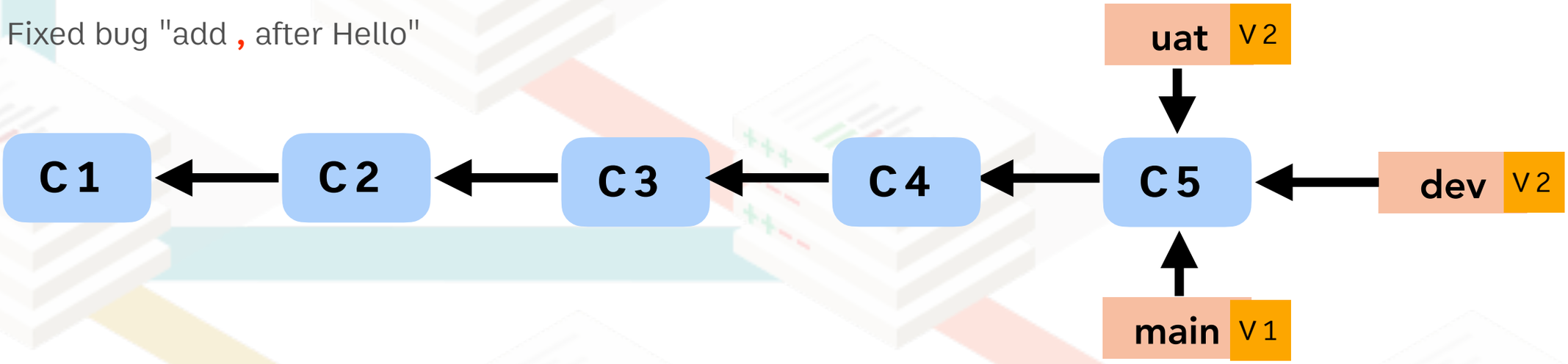
Timeline: Toggle



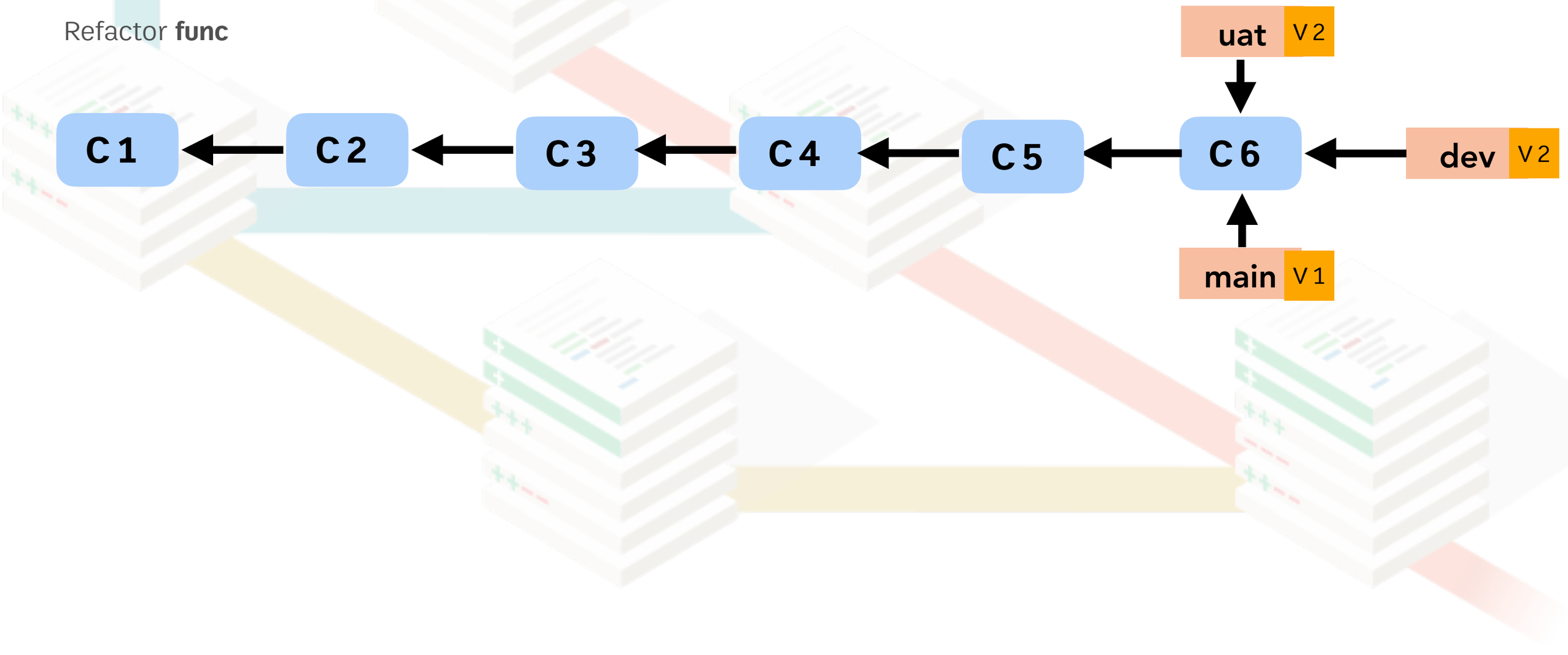
Timeline: Toggle



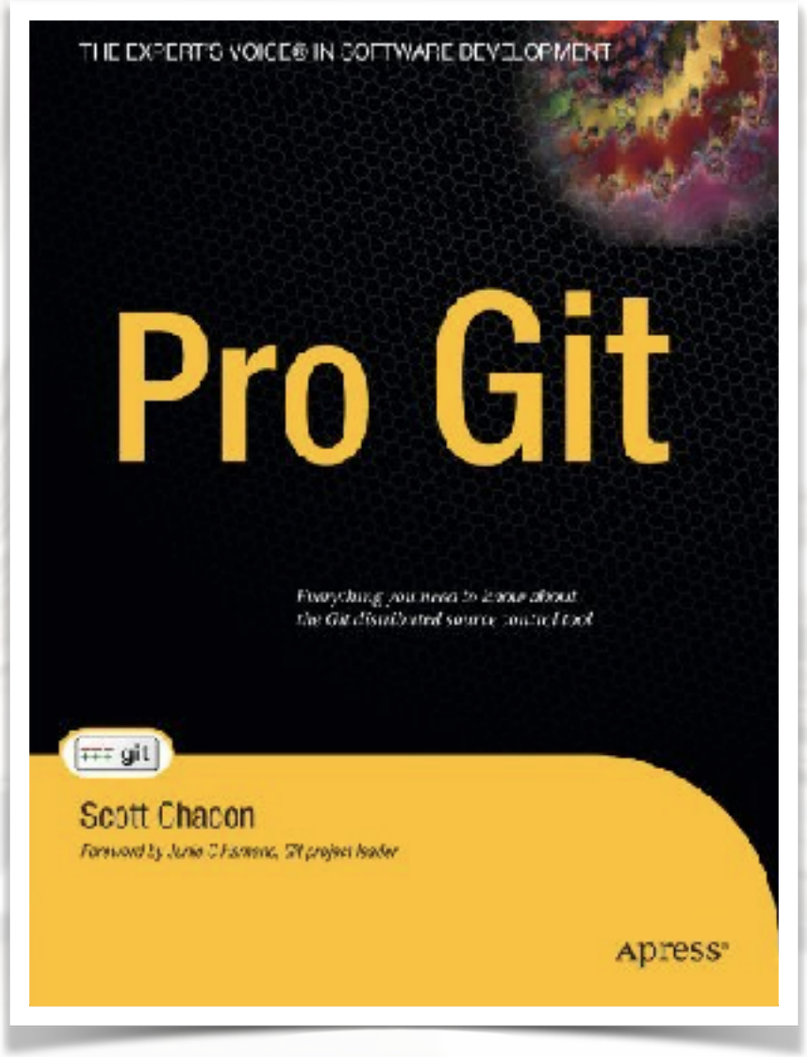
Timeline: Toggle



Timeline: Toggle



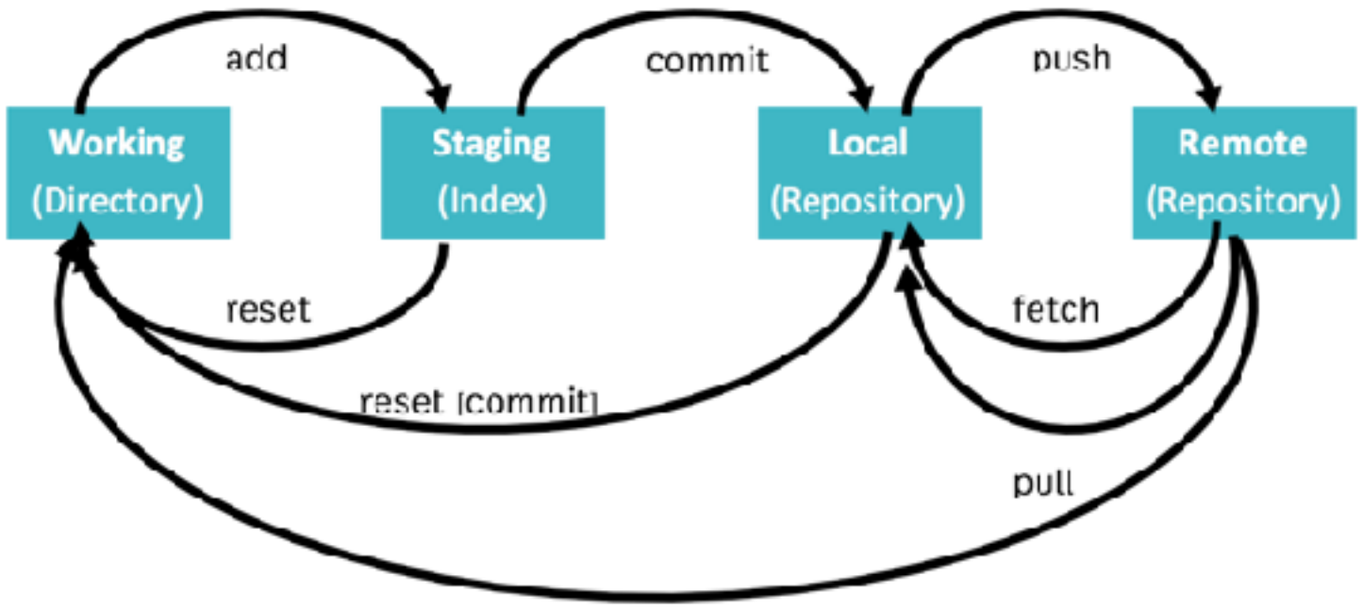
Books to Read and Practice



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

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

cheatsheet



CONFIGURE TOOLING

กำหนดข้อมูลสำหรับใช้ใน local repositories

$ git config --global user.name "[name]"

ตั้งค่าชื่อที่ใช้แสดงผลในการ commit

$ git config --global user.email "[email address]"

ตั้งค่าอีเมล์ที่ใช้แสดงผลในการ commit

$ git config --global color.ui auto

เปิดใช้งานตัวอักษรสีในชุดคำสั่งต่าง ๆ เพื่อง่ายต่อการ

CREATE REPOSITORIES

เริ่มต้นสร้าง repository หรือนำเข้าข้อมูลจาก repository URL อื่นๆ

$ git init [project-name]

สร้างและกำหนดชื่อ local repository

$ git clone [url]

ดาวน์โหลดข้อมูลจาก repository URL และประวัติการควบคุมการเปลี่ยนแปลง (version history)

cheatsheet

SYNCHRONIZE CHANGES

การปรับ local repository ให้เท่ากับ remote repository

$ git fetch [nickname]

ดาวน์โหลดทุกรายการประวัติของเวอร์ชันจาก remote repository สู่ local repository

$ git merge [nickname]/[branch]

รวม remote repository จาก branch ที่กำหนด เข้าสู่ local repository

$ git push [nickname] [branch]

อัพโหลดข้อมูลจาก local branch เข้าสู่ remote branch

$ git pull

ดาวน์โหลดประวัติของ remote repository และการเปลี่ยนแปลงต่าง ๆ เข้าสู่ local repository

MAKE CHANGES

ตรวจสอบการแก้ไขข้อมูลและ commit

$ git status

แสดงสถานะการเปลี่ยนแปลง ณ ปัจจุบัน

$ git add [file]

บันทึกไฟล์ข้อมูลเข้าสู่ index เพื่อเตรียมทำเวอร์ชัน

$ git reset [file]

ยกเลิกการเปลี่ยนแปลงไฟล์

$ git diff

แสดงความแตกต่างของไฟล์ที่ยังไม่ได้ทำ index

$ git diff --staged

แสดงความแตกต่างของไฟล์ที่ได้ทำ index แล้ว กับไฟล์ที่ได้แก้ไขครั้งล่าสุด

$ git commit -m "[descriptive message]"

บันทึกไฟล์ลงประวัติการแก้ไข

cheatsheet

REVIEW HISTORY

เรียกดูและตรวจสอบวิวัฒนาการของข้อมูล

$ git log

แสดงประวัติเวอร์ชันทั้งหมดของ branch ที่กำลังใช้งาน

$ git log --follow [file]

แสดงประวัติเวอร์ชันทั้งหมดของไฟล์และการเปลี่ยนแปลงชื่อไฟล์ด้วย

$ git diff [first-branch]...[second-branch]

แสดงความแตกต่างของข้อมูลระหว่าง 2 branches

$ git show [commit]

แสดง metadata และการเปลี่ยนข้อมูลของ

REDO COMMITS

ลบข้อผิดพลาด และประวัติการบันทึก

$ git reset [commit]

ยกเลิกการ commit ทั้งหมดจนถึง commit ที่ระบุไว้,แต่ยังคงการเปลี่ยนแปลงของข้อมูลใน index

$ git reset --hard [commit]

ยกเลิกการ commit ทั้งหมดจนถึง commit ที่ระบุไว้

cheatsheet

GROUP CHANGES

ชุดของการ commits ข้อมูล และการรวบรวมชุดต่าง ๆ เข้าด้วยกัน

$ git branch

แสดงรายการทั้งหมดของ local branches ใน repository ที่กำลังใช้งานอยู่

$ git branch [branch-name]

สร้าง branch ใหม่

$ git checkout [branch-name]

สลับการใช้งาน branch และปรับปรุงข้อมูลของ working directory

$ git merge [branch]

รวบรวม branch ที่กำหนด เข้ากับ branch ที่กำลังใช้งาน

REFACTOR FILENAMES

เปลี่ยนแปลงไฟล์ในเวอร์ชันที่ใช้งาน

$ git rm [file]

ลบไฟล์จาก working directory และ index

$ git rm --cached [file]

ลบไฟล์ออกจาก Git แต่ไม่ลบออกจาก working directory

$ git mv [file-original] [file-renamed]

เปลี่ยนชื่อไฟล์