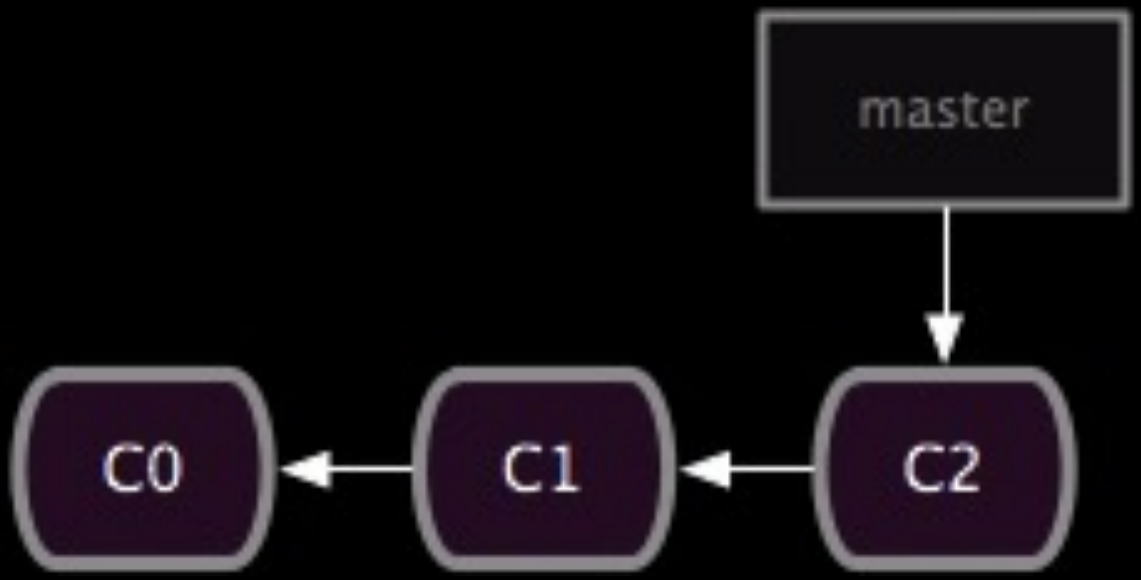
Git & GitLab Guides

Why track/manage revisions?

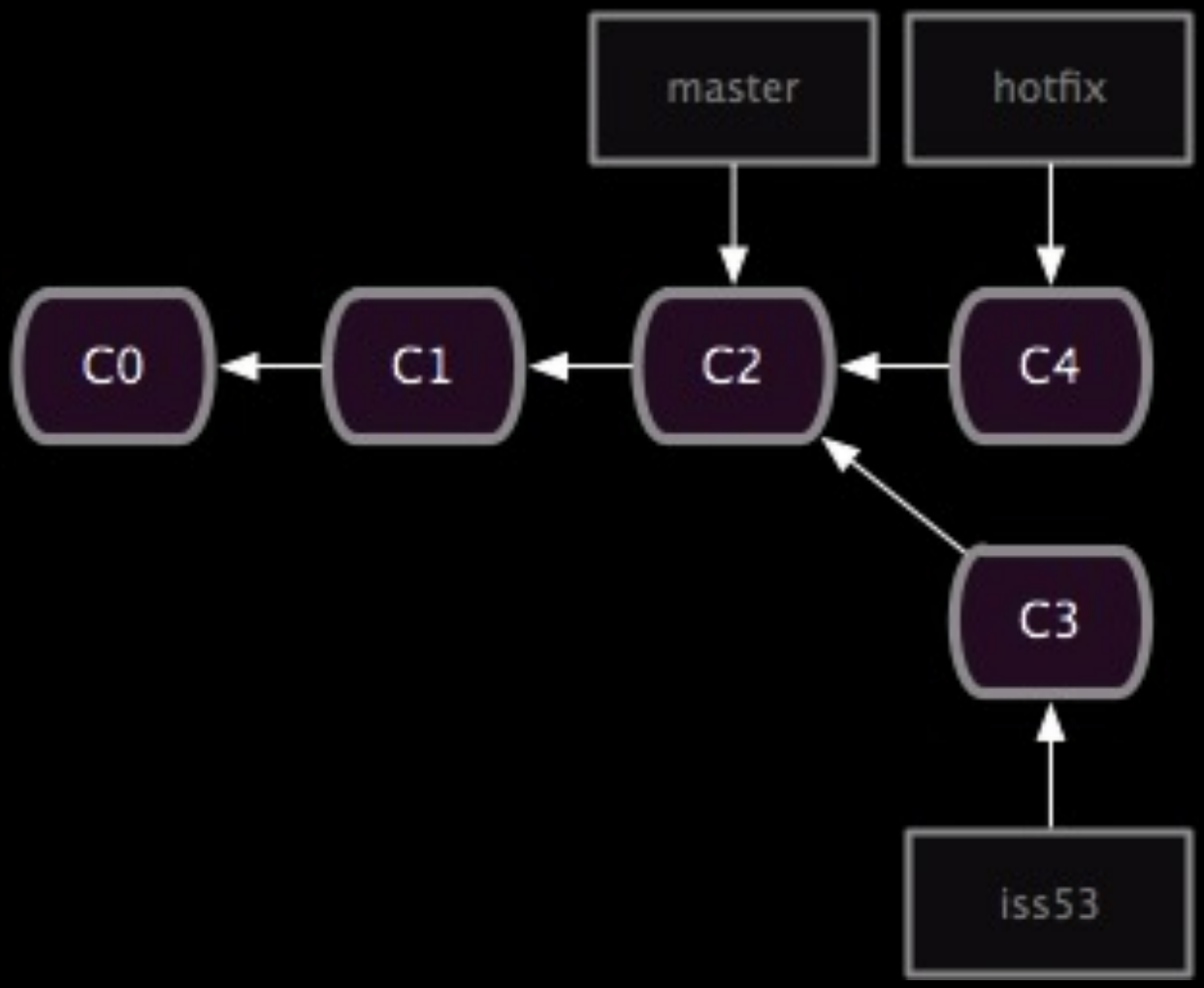
Backup: Undo or refer to old stuff



http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

Why track/manage revisions?

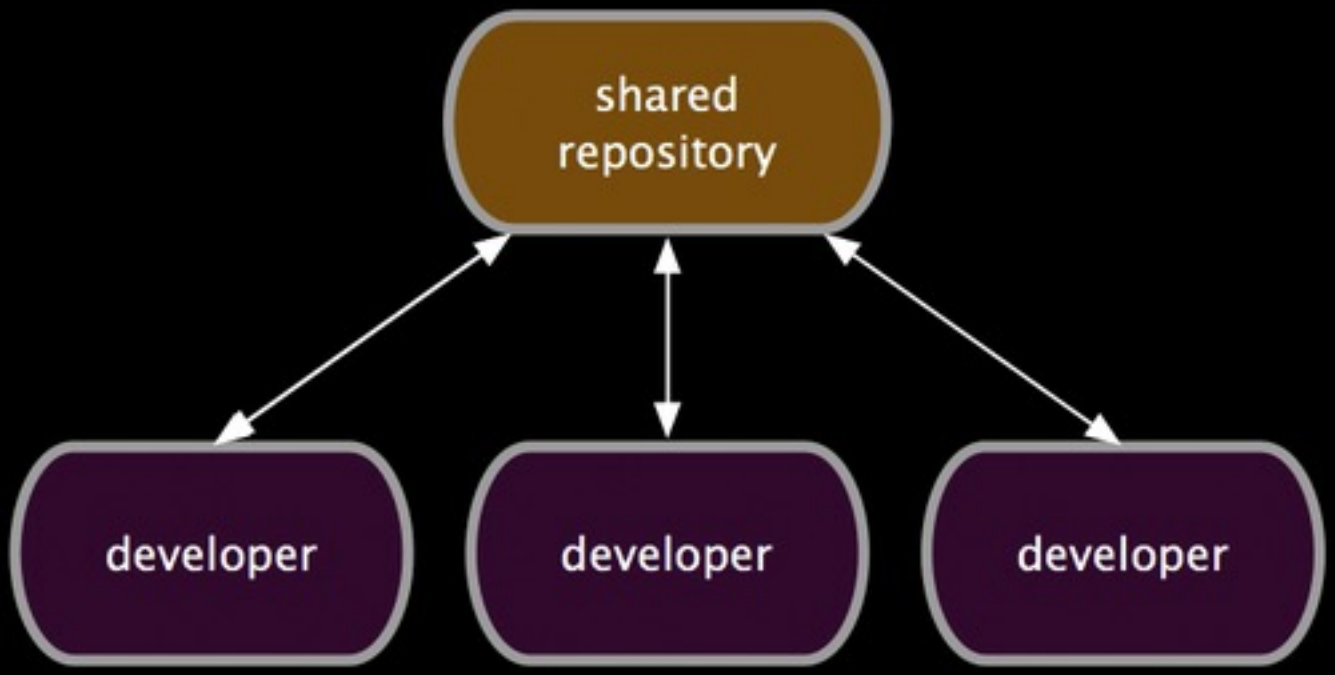
Branch: Maintain old release while working on new



http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

Why track/manage revisions?

Collaborate: Work in parallel with teammates



http://git-scm.com/book/en/Distributed-Git-Distributed-Workflows

Version Control Systems (VCSs)

- Help you track/manage/distribute revisions
- Standard in modern development
- Examples:
  - Revision Control System (RCS)
  - Concurrent Versions System (CVS)
  - Subversion (SVN)
  - Git

older

newer

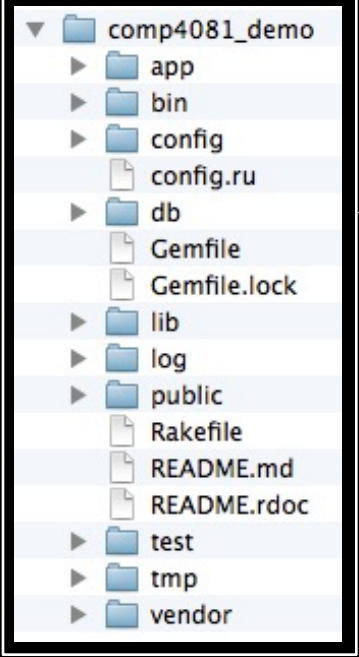
Our focus

GitLab-User Perspective

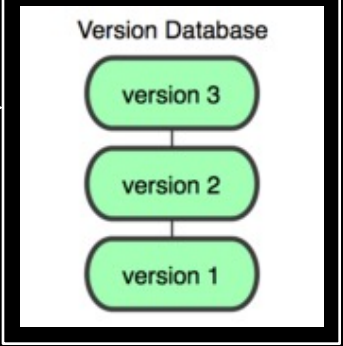
You

GitLab

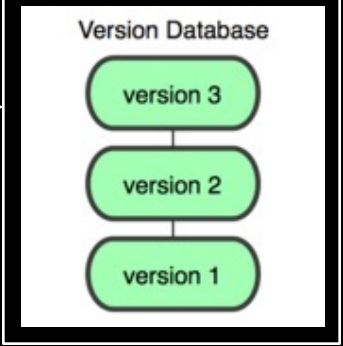
Working Dir



Local Repos



Remote Repos



Let’s begin with an example…

You

GitLab

Log into GitHub and create a repos

(with add README option)

You

GitLab

Remote Repos

Configure your Git client

- Install Git:

```
Download appropriate version from: https://git-scm.com
```

- Create local repos & check config info:

```
$ mkdir comp4081_demo
$ cd comp4081_demo
$ git init
$ git config --list
user.name=Kien Nguyen
user.email=kiennt@gmail.com
```

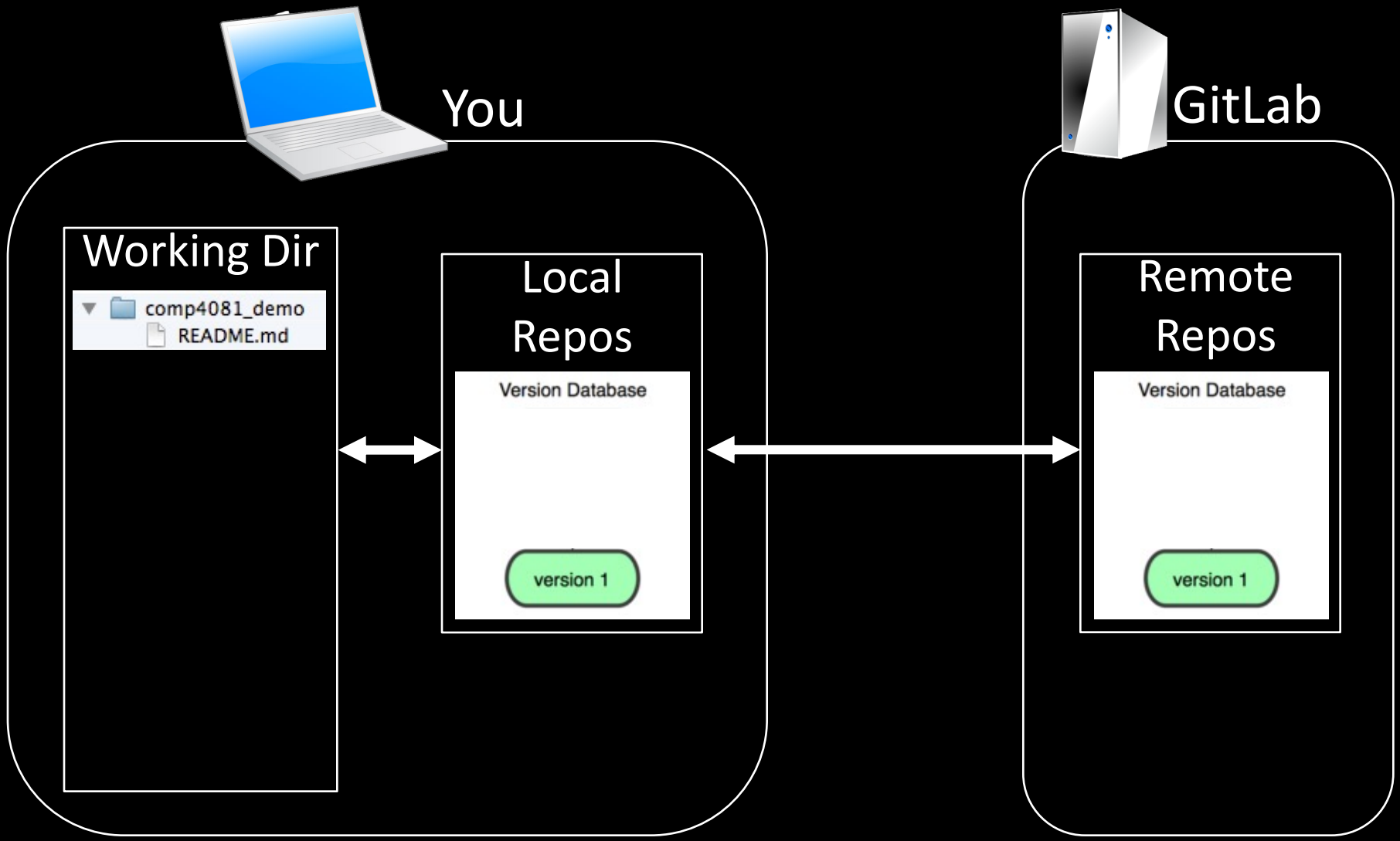
- Fix if necessary:

```
$ git config --global user.name ”Kien Nguyen"
$ git config --global user.email kiennt@fpt.edu.vn
$ git config --global init.defaultBranch main
$ git config --global core.excludesfile ~/.gitignore
```

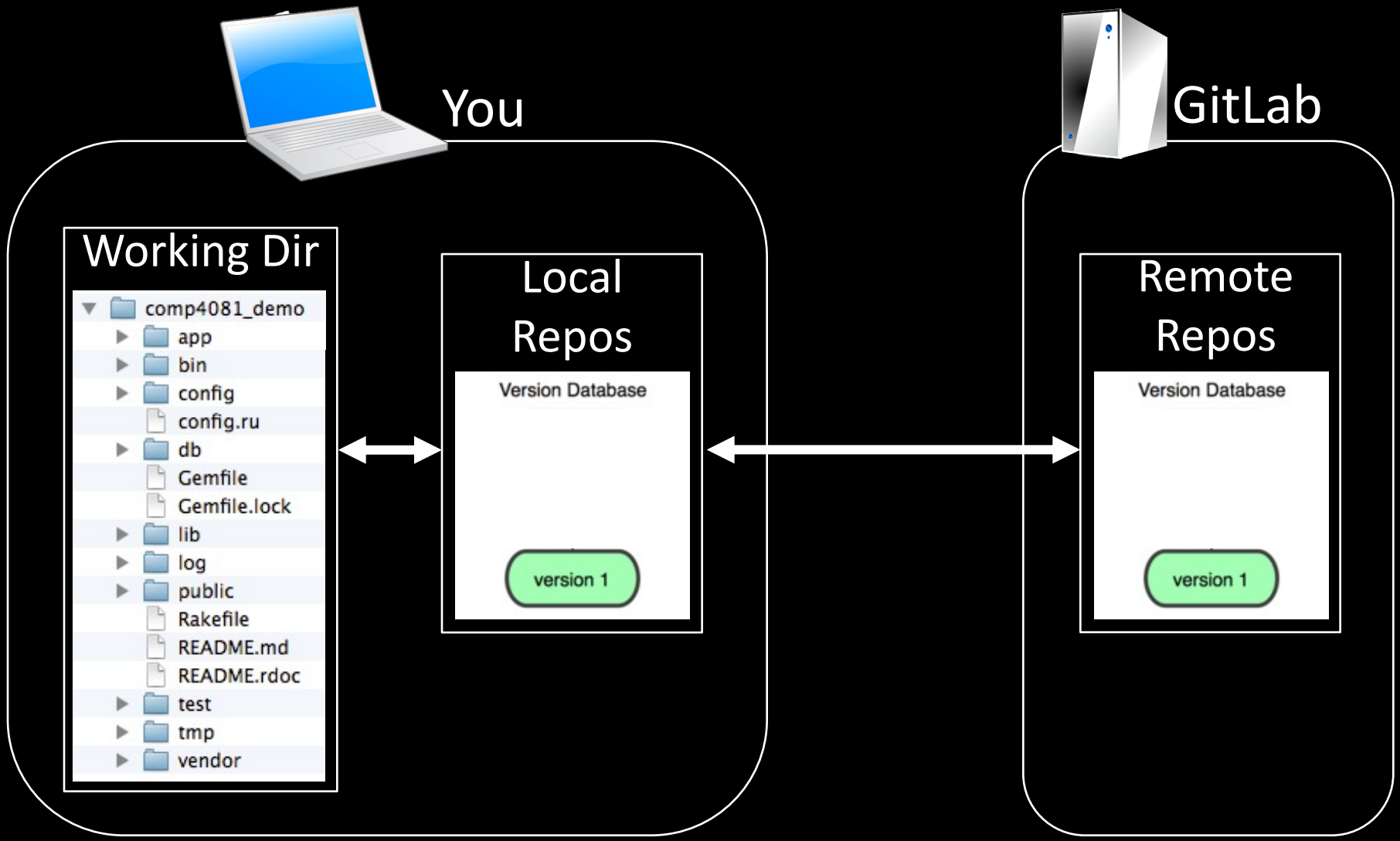
- Connect to Remote repos:

```
$ git remote add origin https://gitlab.com/kienpmp/demo.git
```

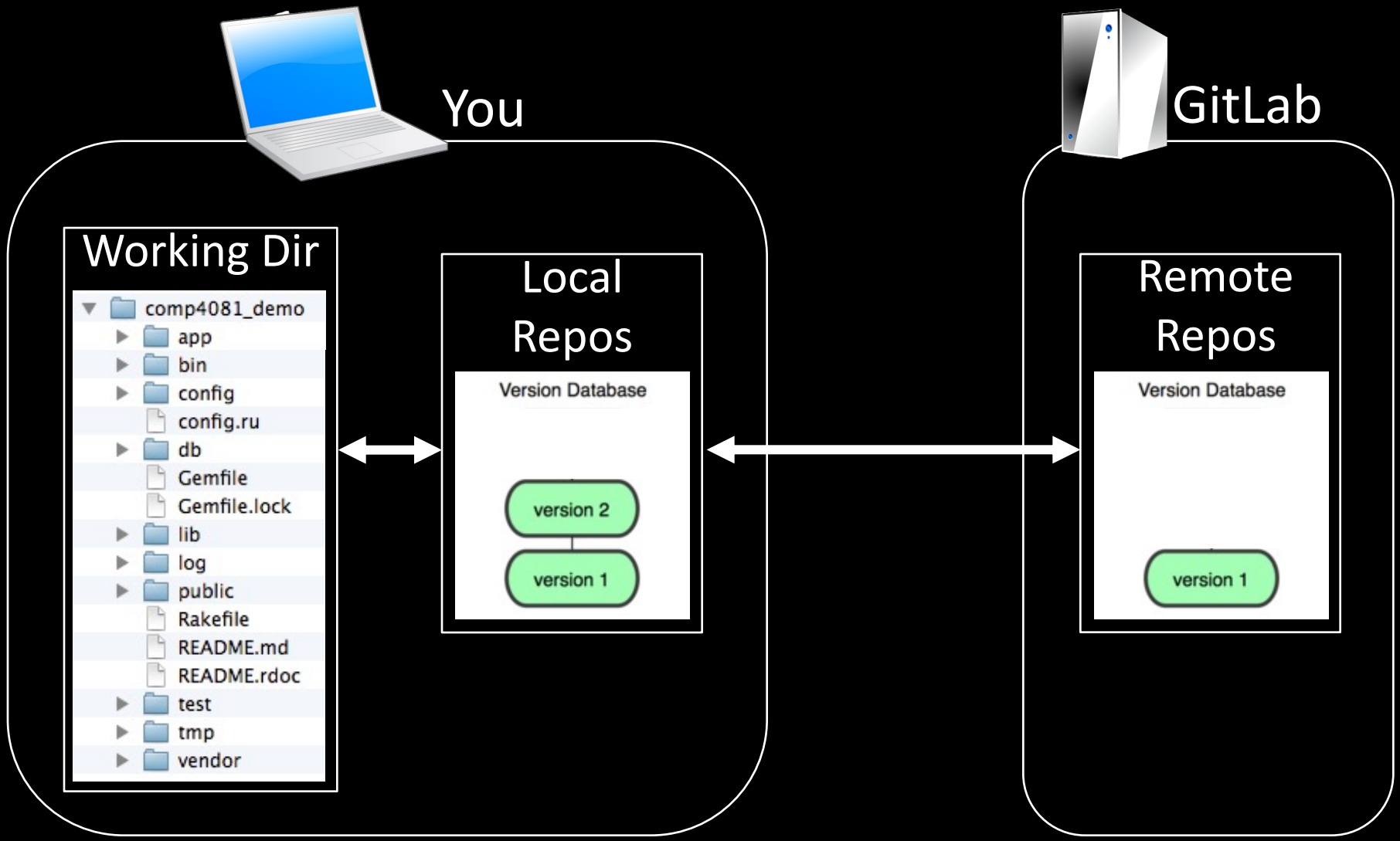
```
$ git clone https://gitlab.com/kienpmp/demo.git
```



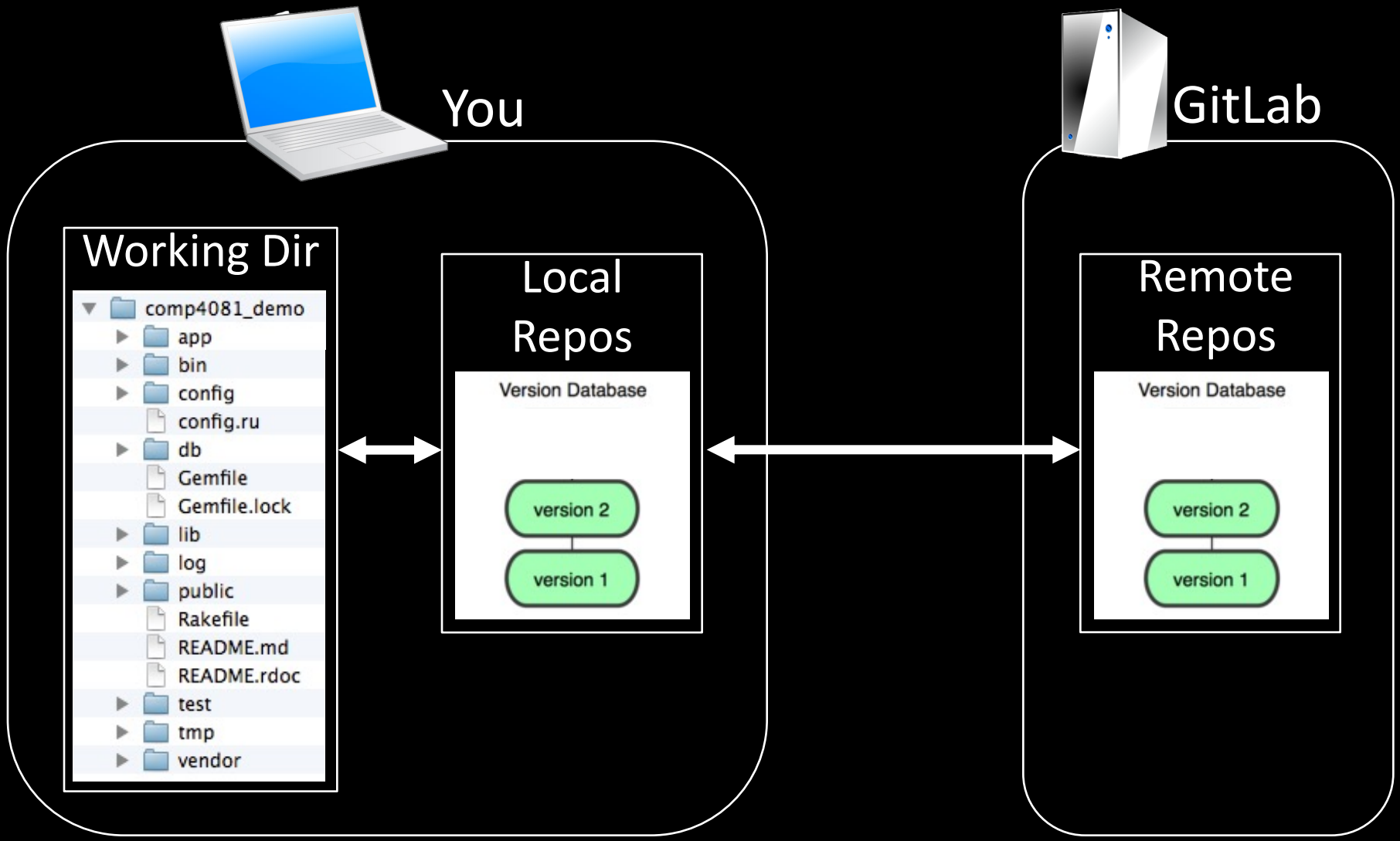
Create project code skeleton



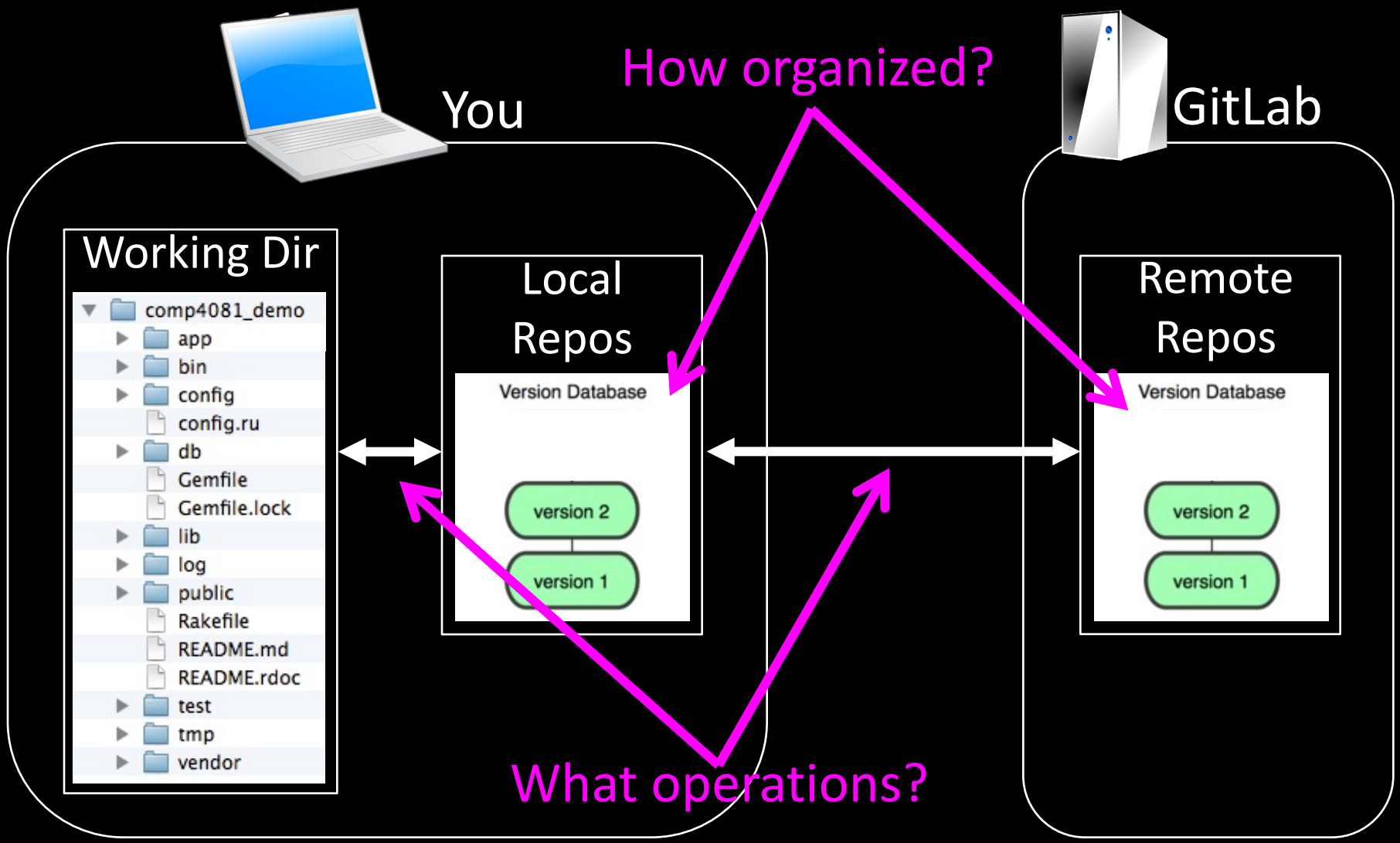
```
$ cd comp4081_demo
$ git add -A
$ git commit -m "Created project skeleton"
```



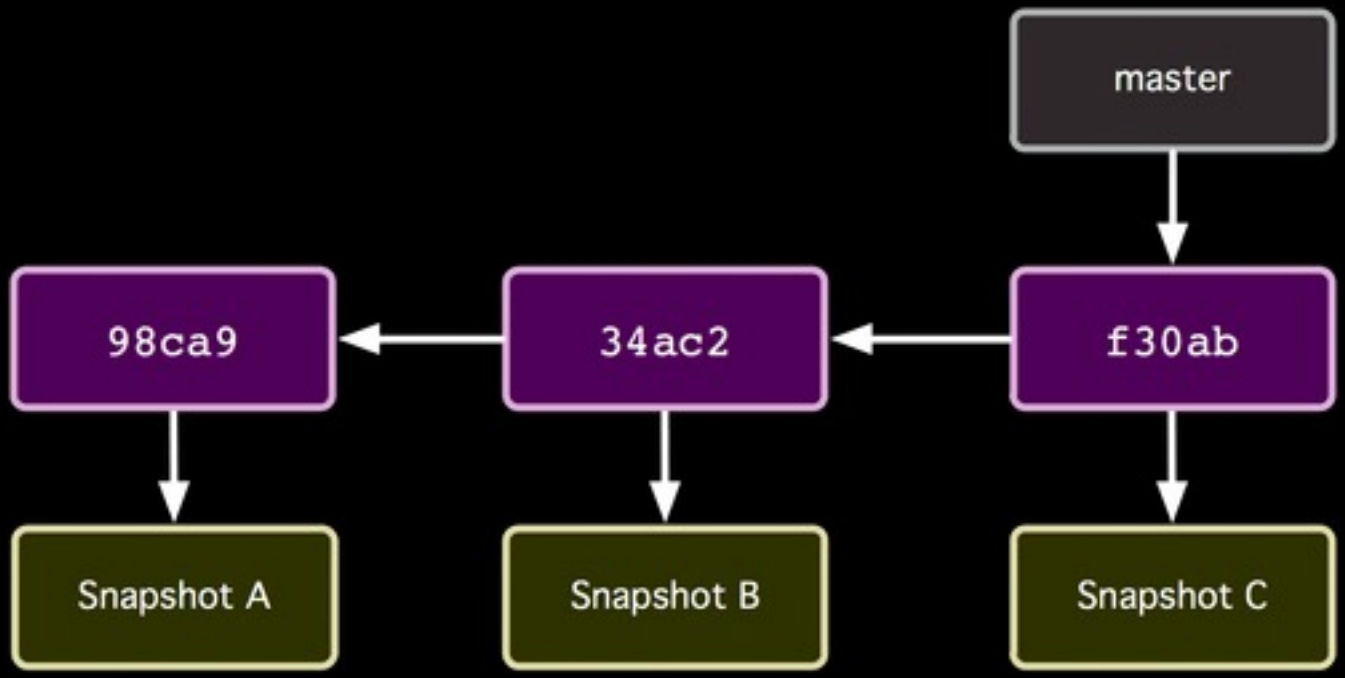
```
$ git push
```


You
GitLab
Working Dir
comp4081_demo
app
bin
config
config.ru
db
Gemfile
Gemfile.lock
lib
log
public
Rakefile
README.md
README.rdoc
test
tmp
vendor
Local Repos
Version Database
version 2
version 1
Remote Repos
Version Database
version 2
version 1

Questions to answer



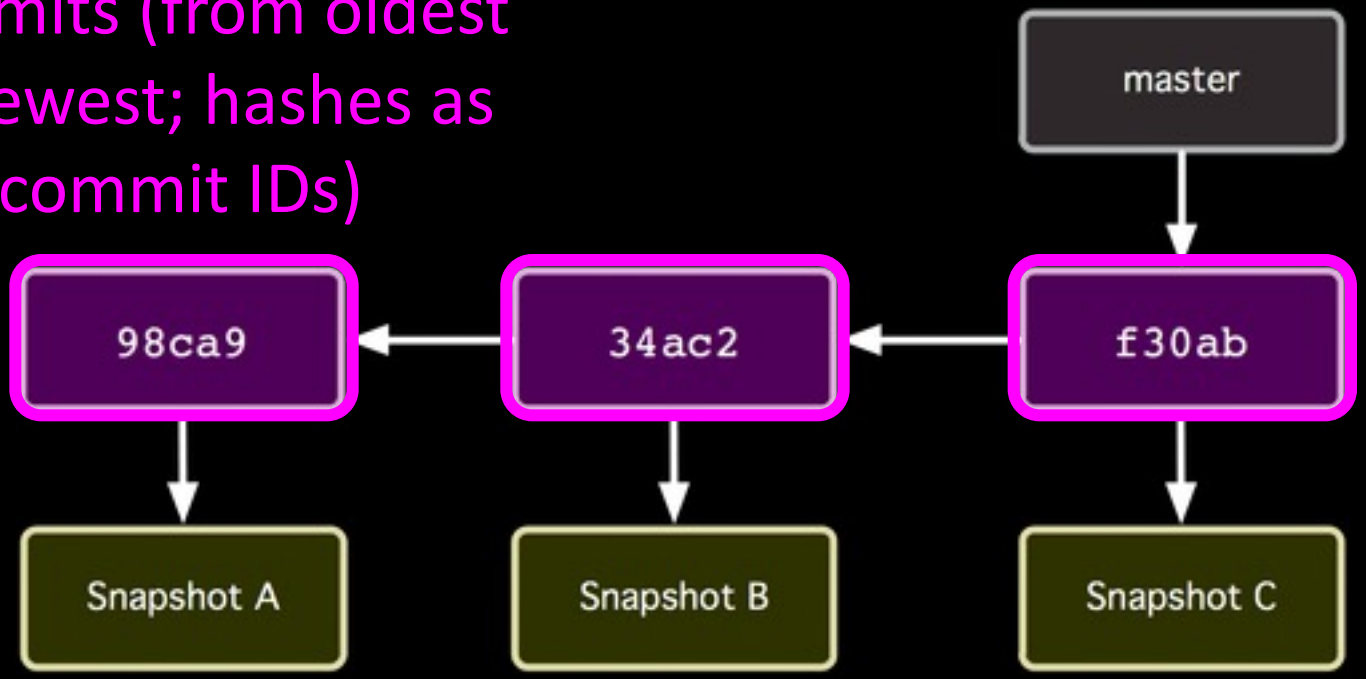
How the repos is organized



http://git-scm.com/book/

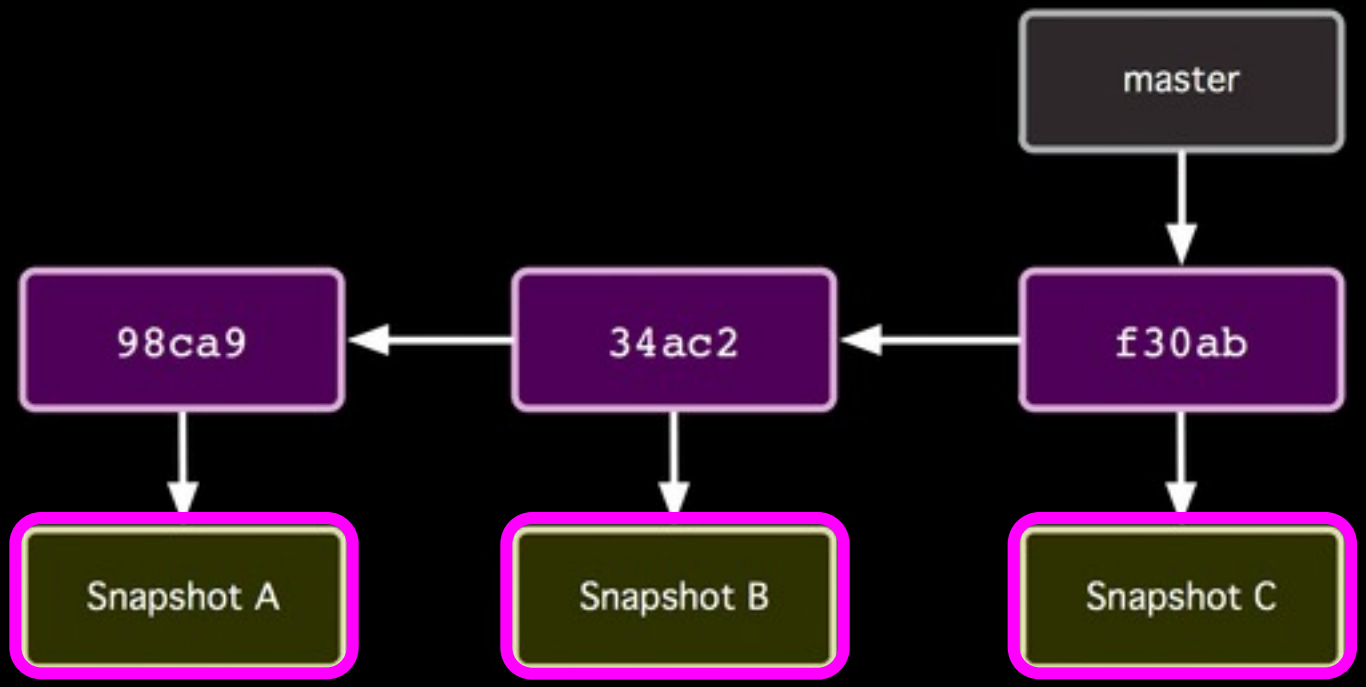
How the repos is organized

Commits (from oldest to newest; hashes as commit IDs)



http://git-scm.com/book/

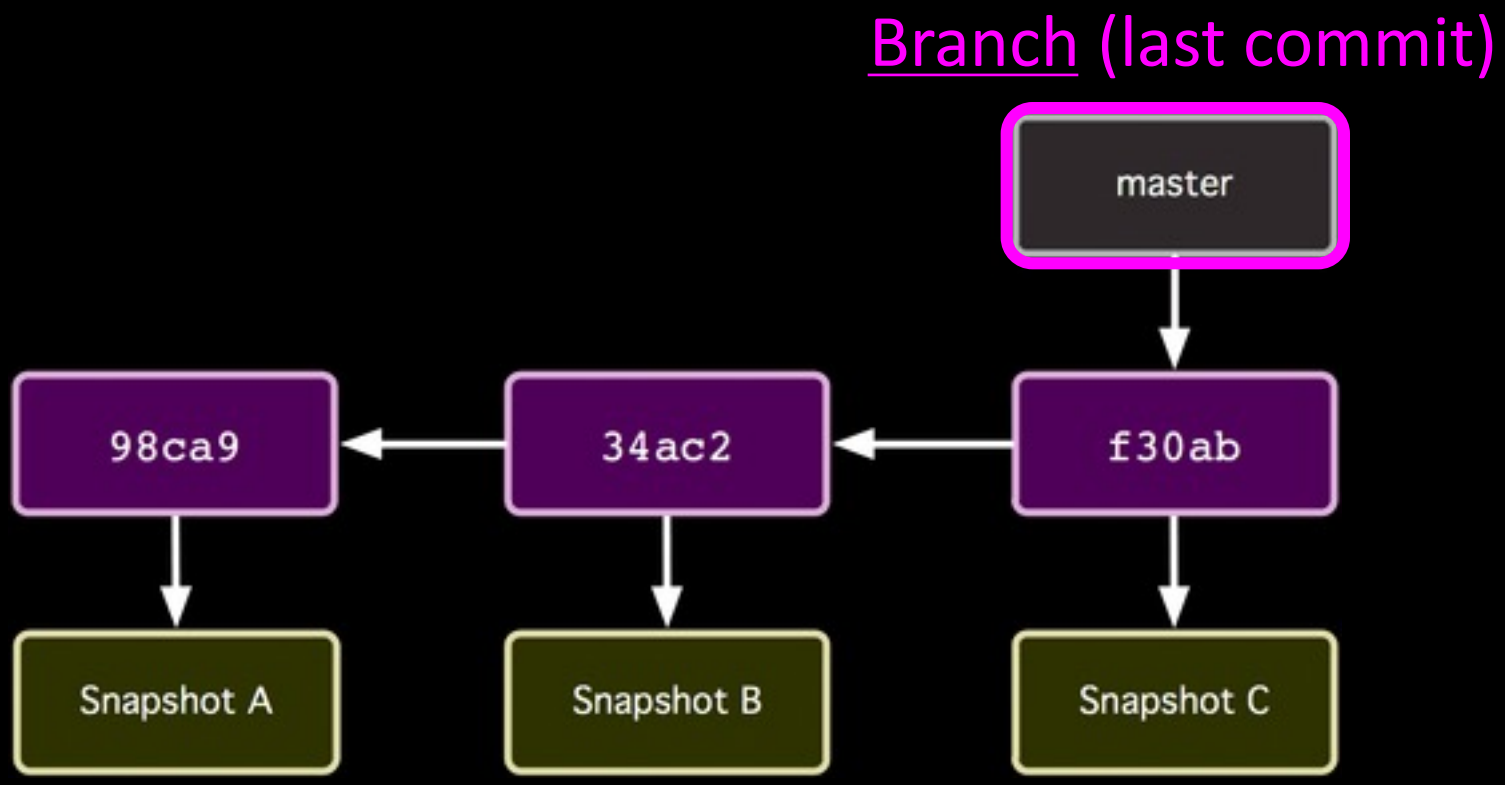
How the repos is organized



Snapshot of all files
at each commit

http://git-scm.com/book/

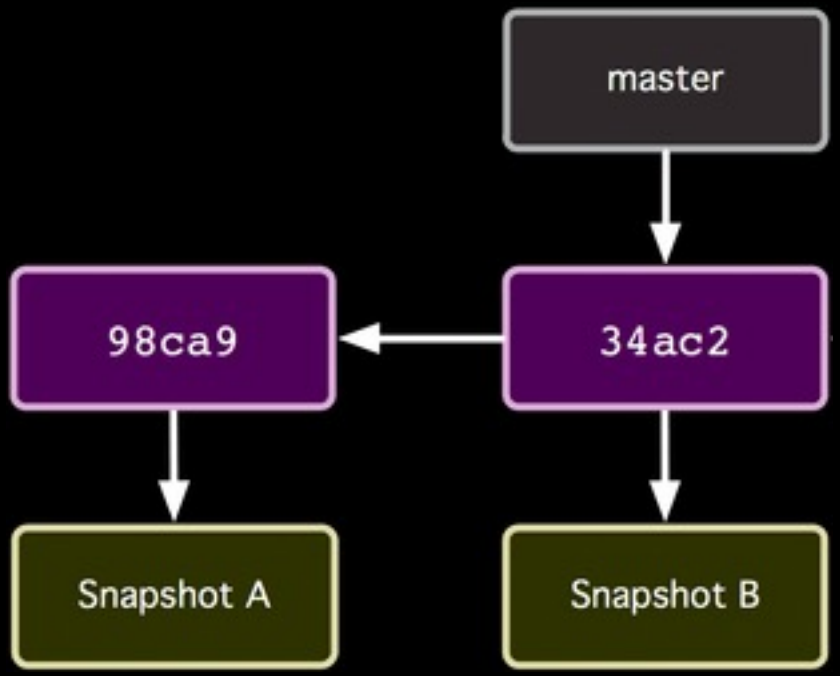
How the repos is organized



http://git-scm.com/book/

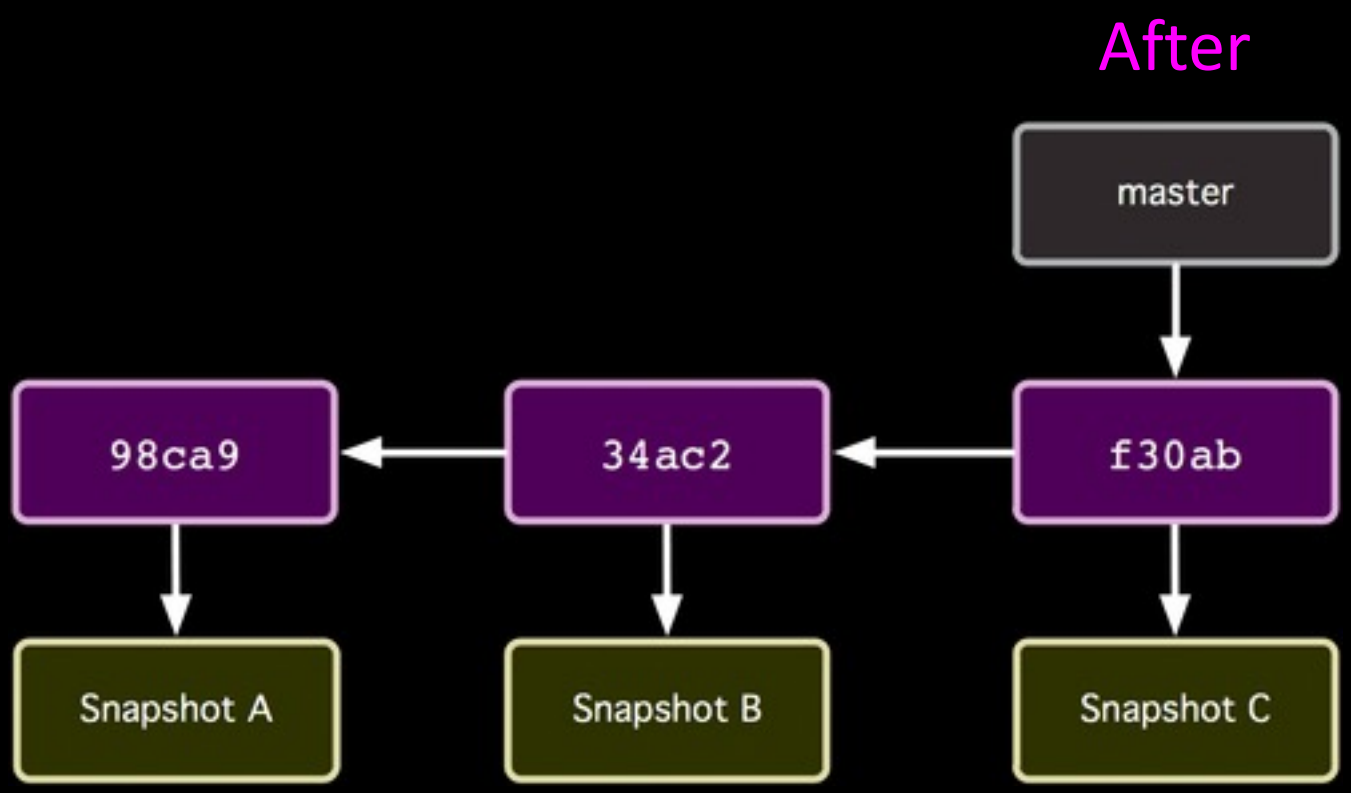
How commit works

Before



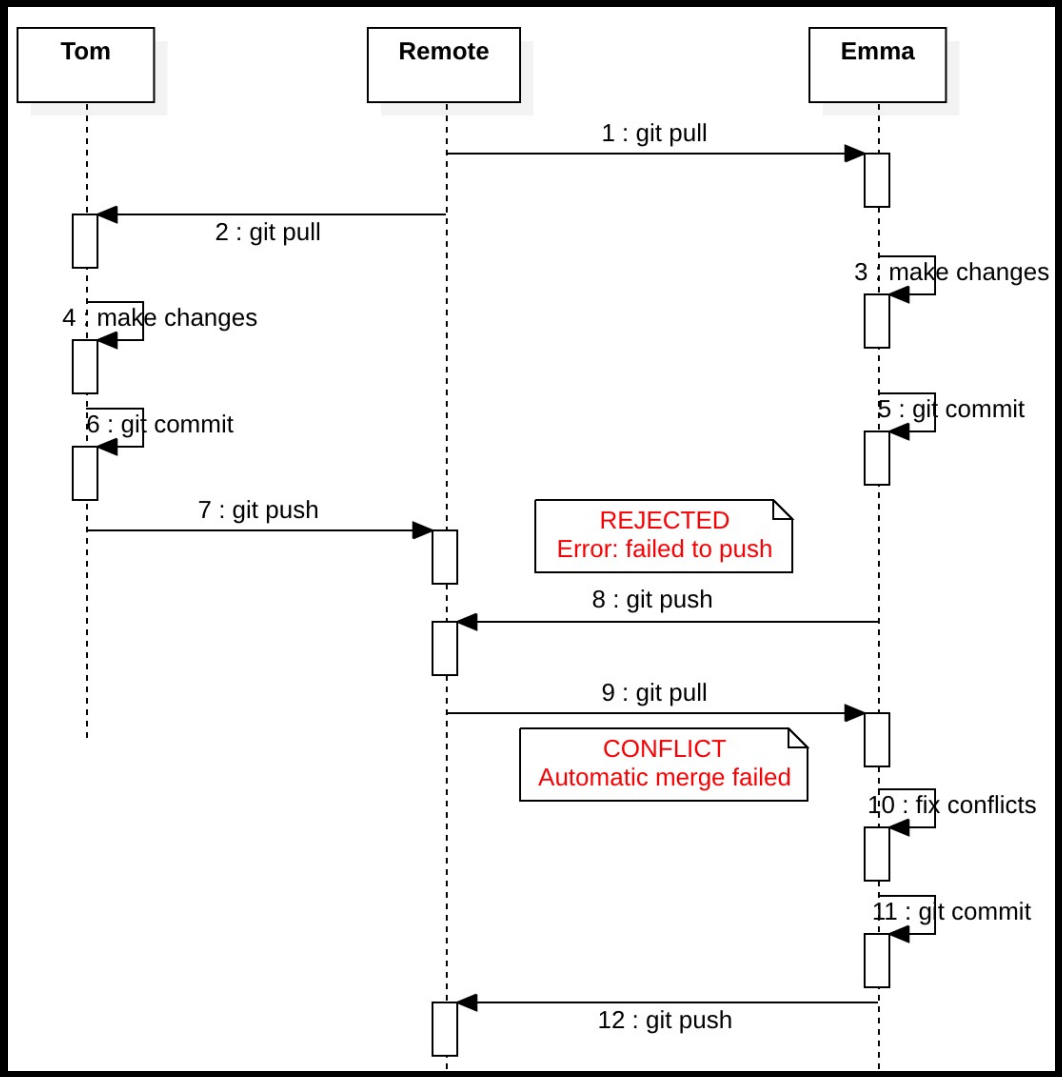
http://git-scm.com/book/

How commit works



http://git-scm.com/book/

Common Workflow #1



1_Tom & Emma: pull to have file7, the latest version (line1)
2_Tom: add line2 to file7 & push => OK
3_Emma: add line3 to file7 & push => NOK
4_Emma: pull to have updated file7 => conflict

```
file7_line1
<<<<<<< HEAD
file7_line3
=======
file7_line2
>>>>>>> 09cbbd
```

5_Memma: fix conflicts as below, then add, commit, and push: OK

```
file7_line1
file7_line2
file7_line3
```

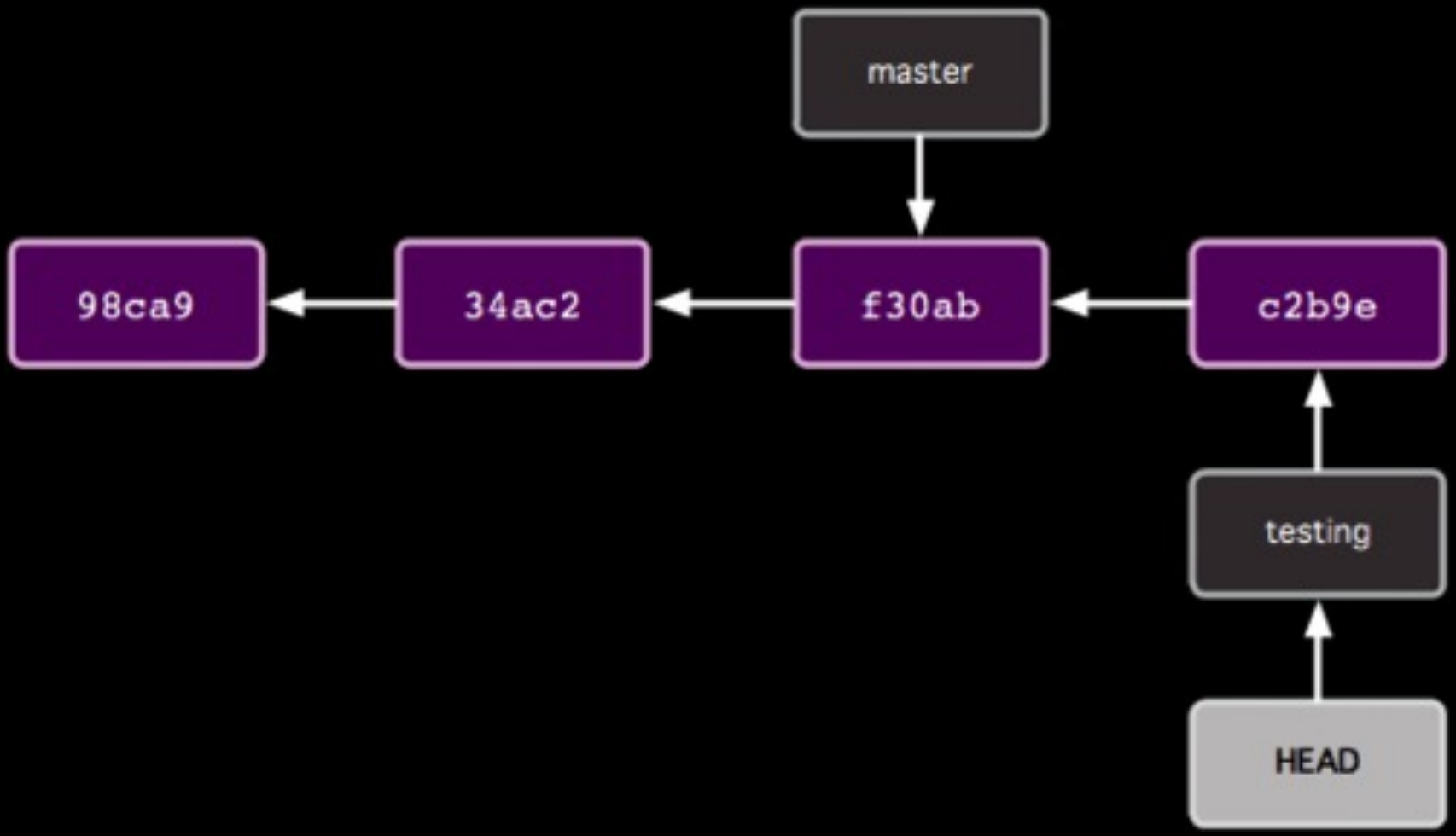
Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

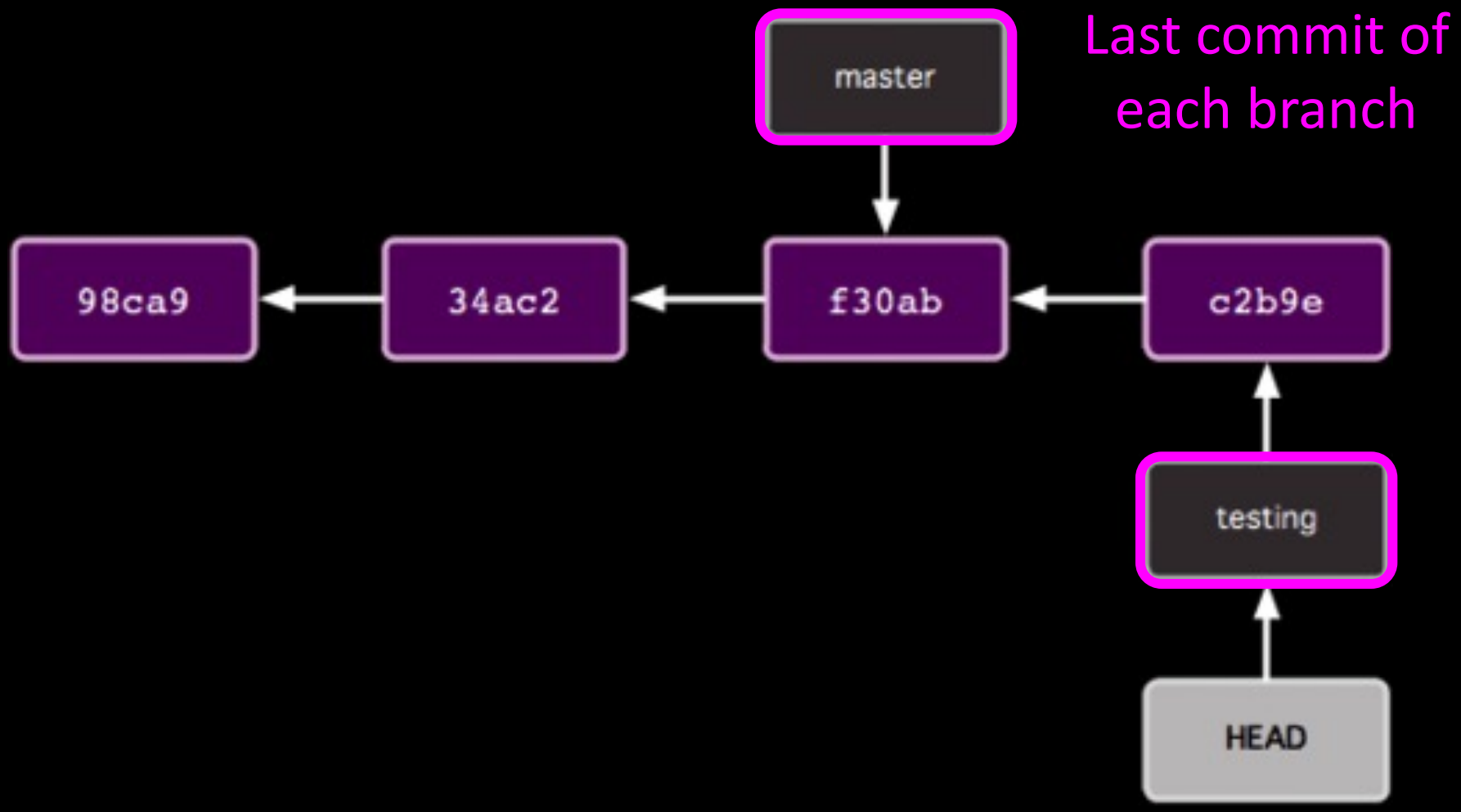
Make changes in local branch

Merge with GitHub repos

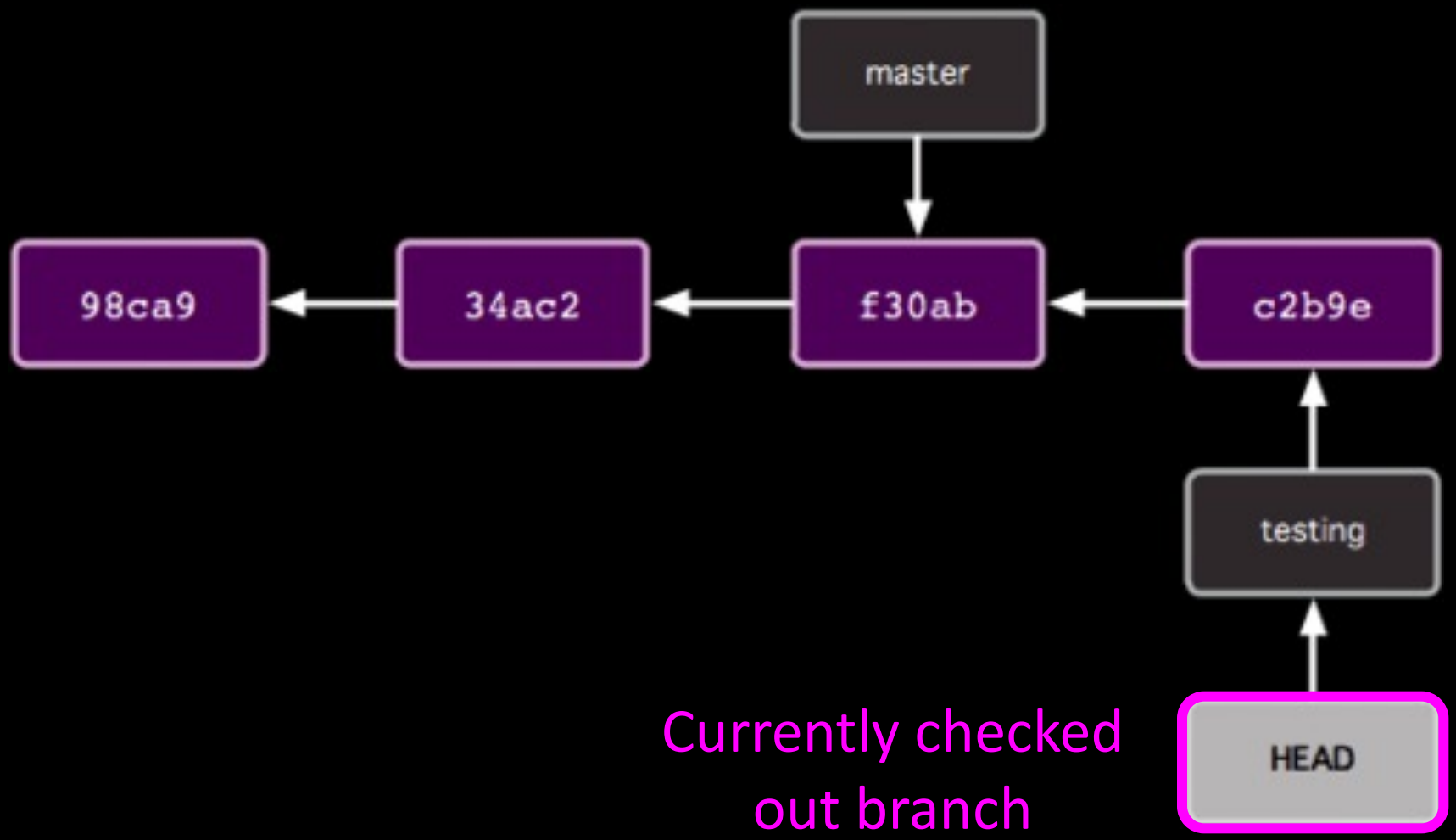
Organization with two branches



Organization with two branches



Organization with two branches



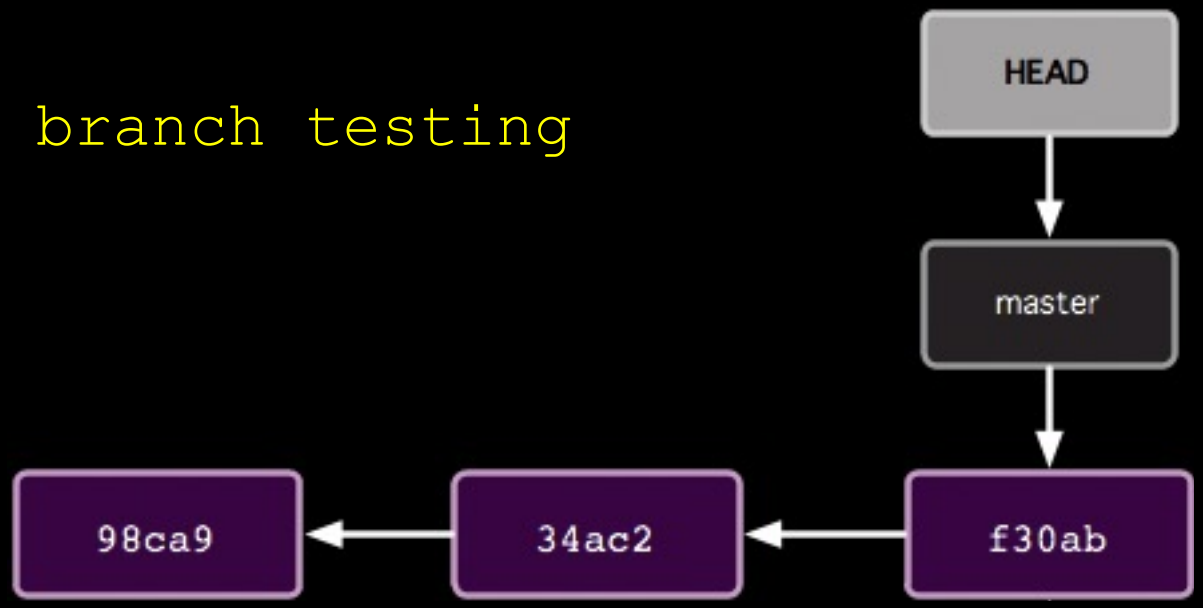
Currently checked out branch

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git branch works

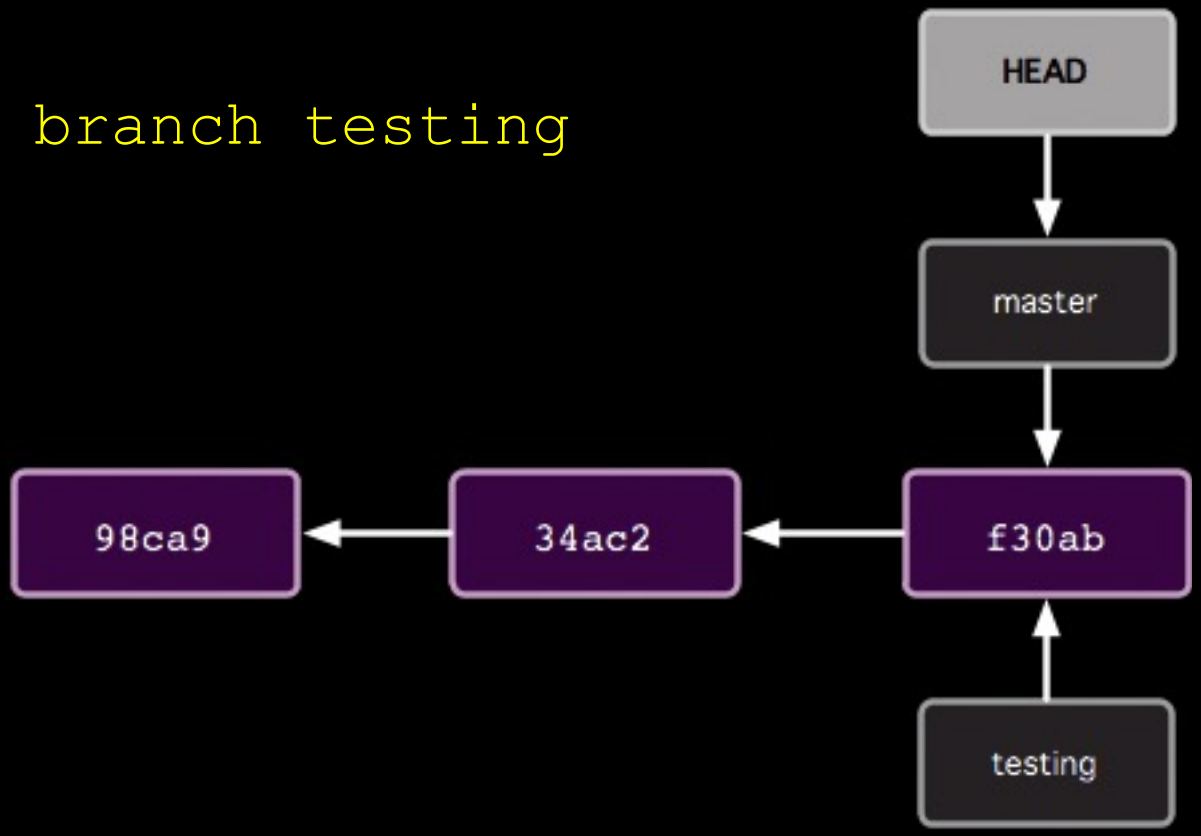
```
$ git branch testing
```



Before

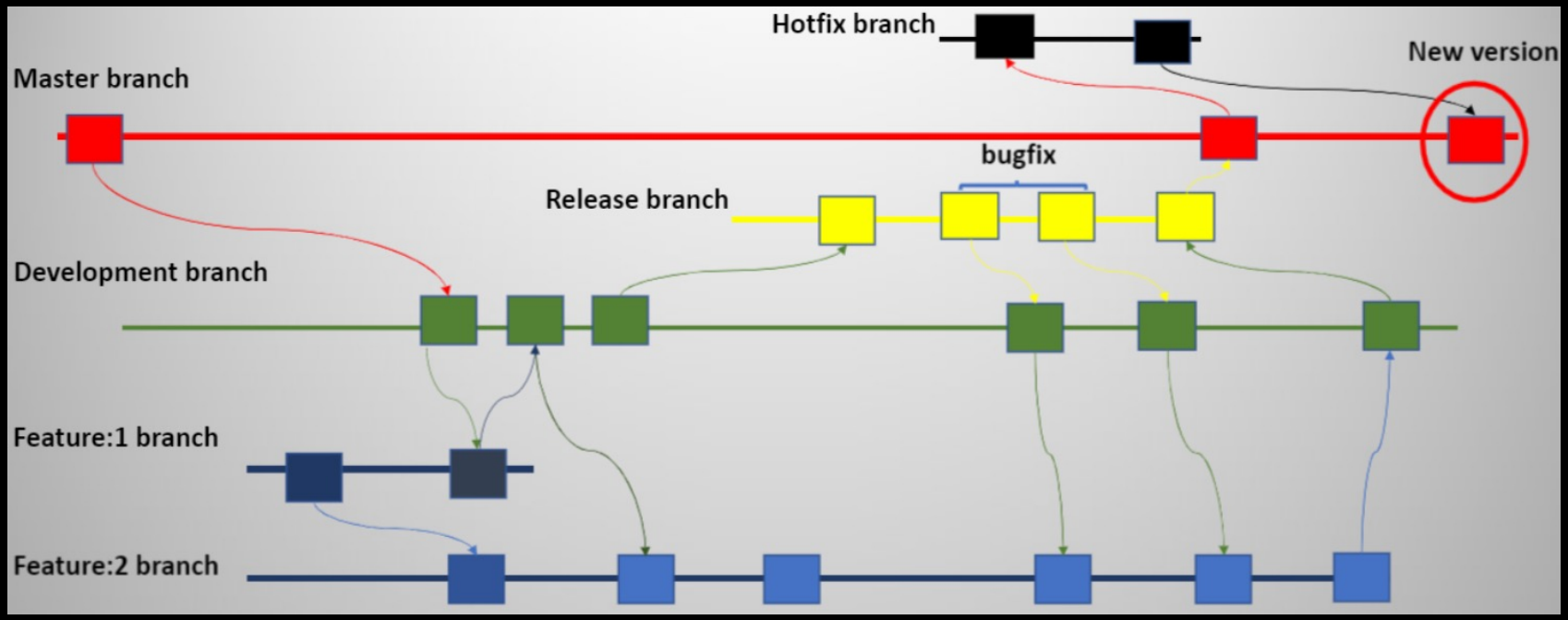
How git branch works

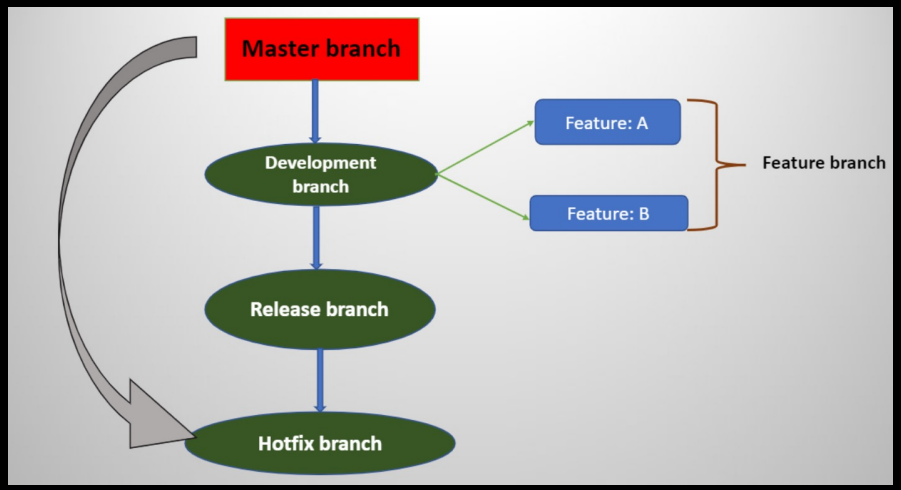
```
$ git branch testing
```



After

How to work with branch?





Common branch commands

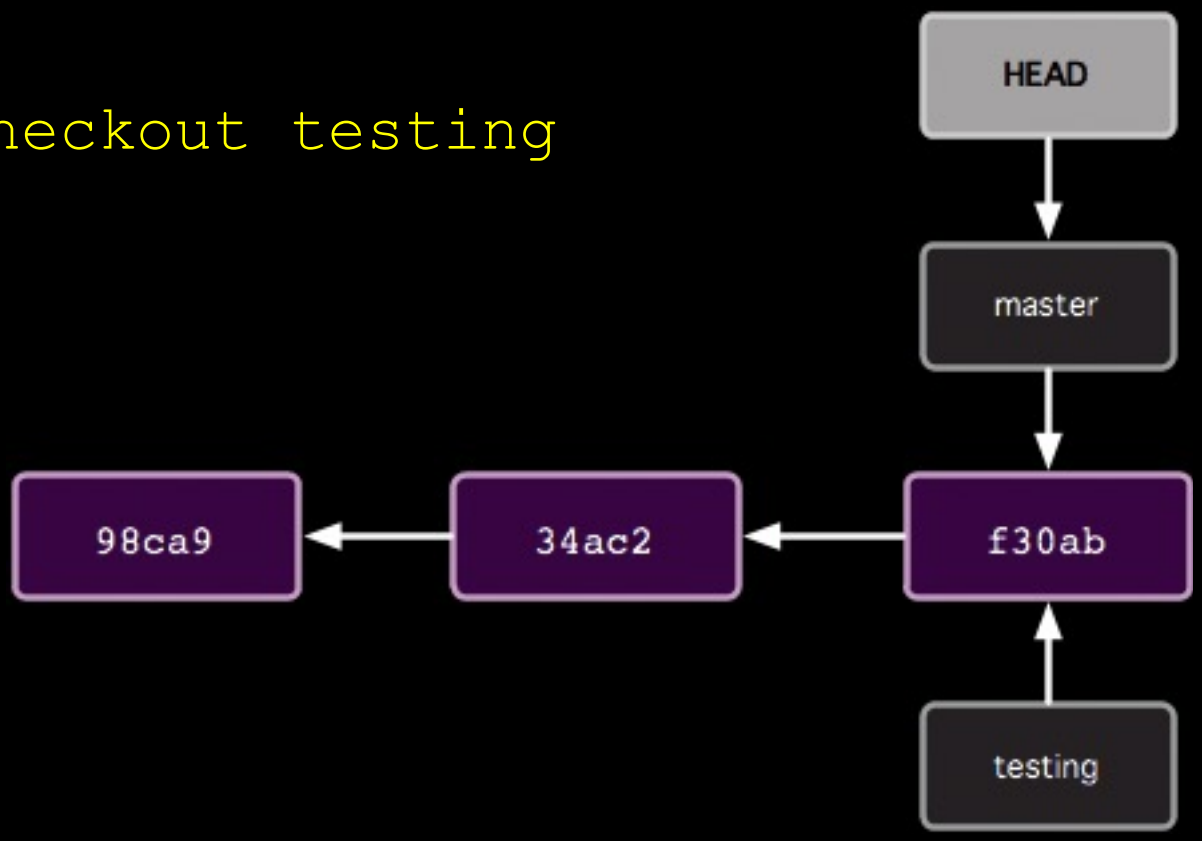
- git branch -a
- git branch <new_branch>
- git checkout -b <br_name>
- git checkout <br_name>
- git branch -d <br_name>
- git merge <br_name>

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git checkout works

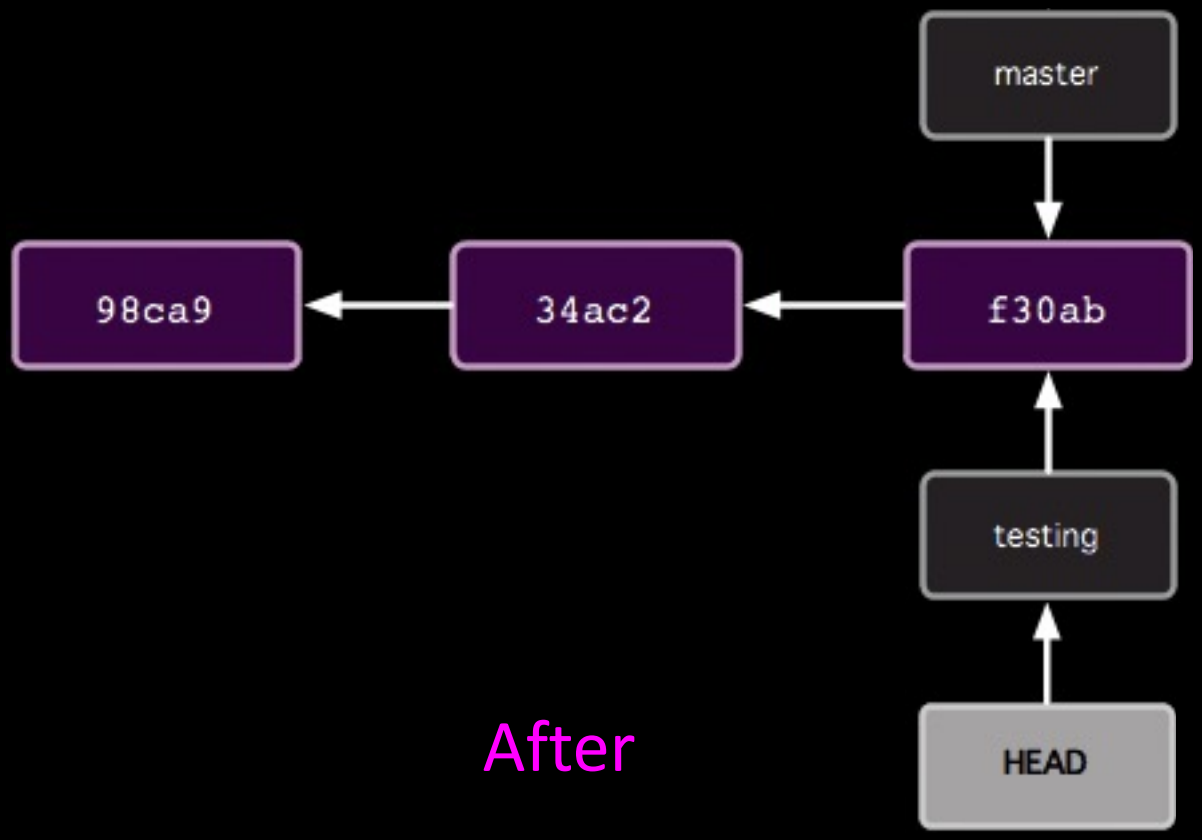
```
$ git checkout testing
```



Before

How git checkout works

```
$ git checkout testing
```



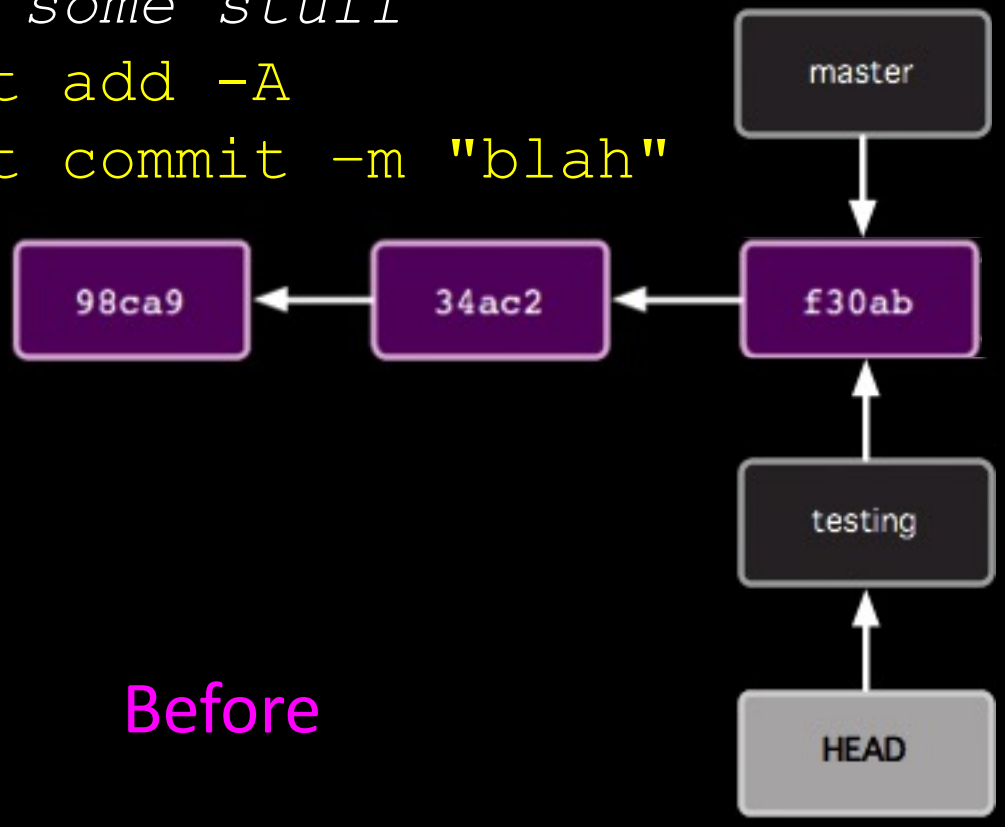
After

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git commit works with multiple branches

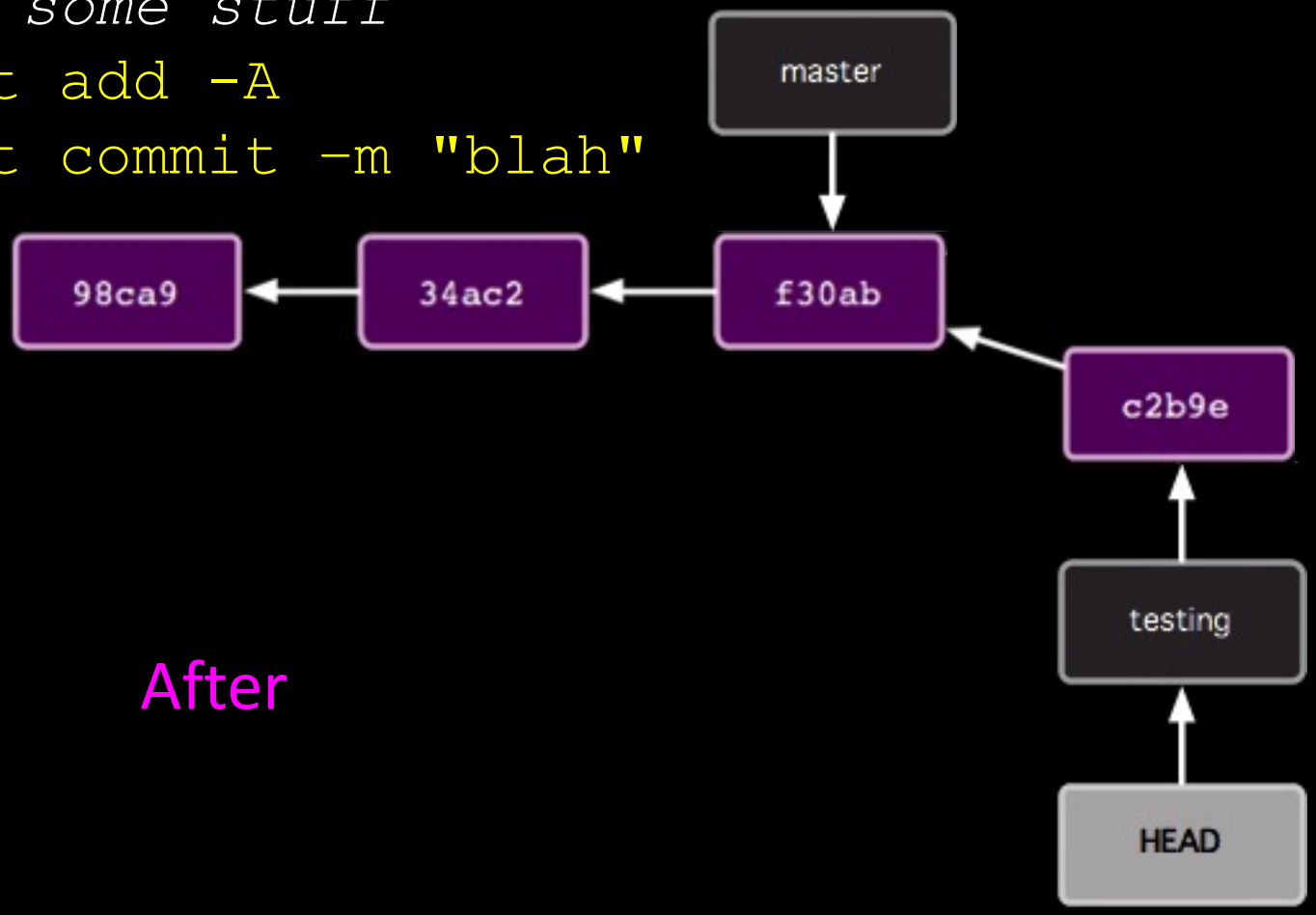
```
Edit some stuff
$ git add -A
$ git commit -m "blah"
```



Before

How git commit works with multiple branches

```
Edit some stuff
$ git add -A
$ git commit -m "blah"
```



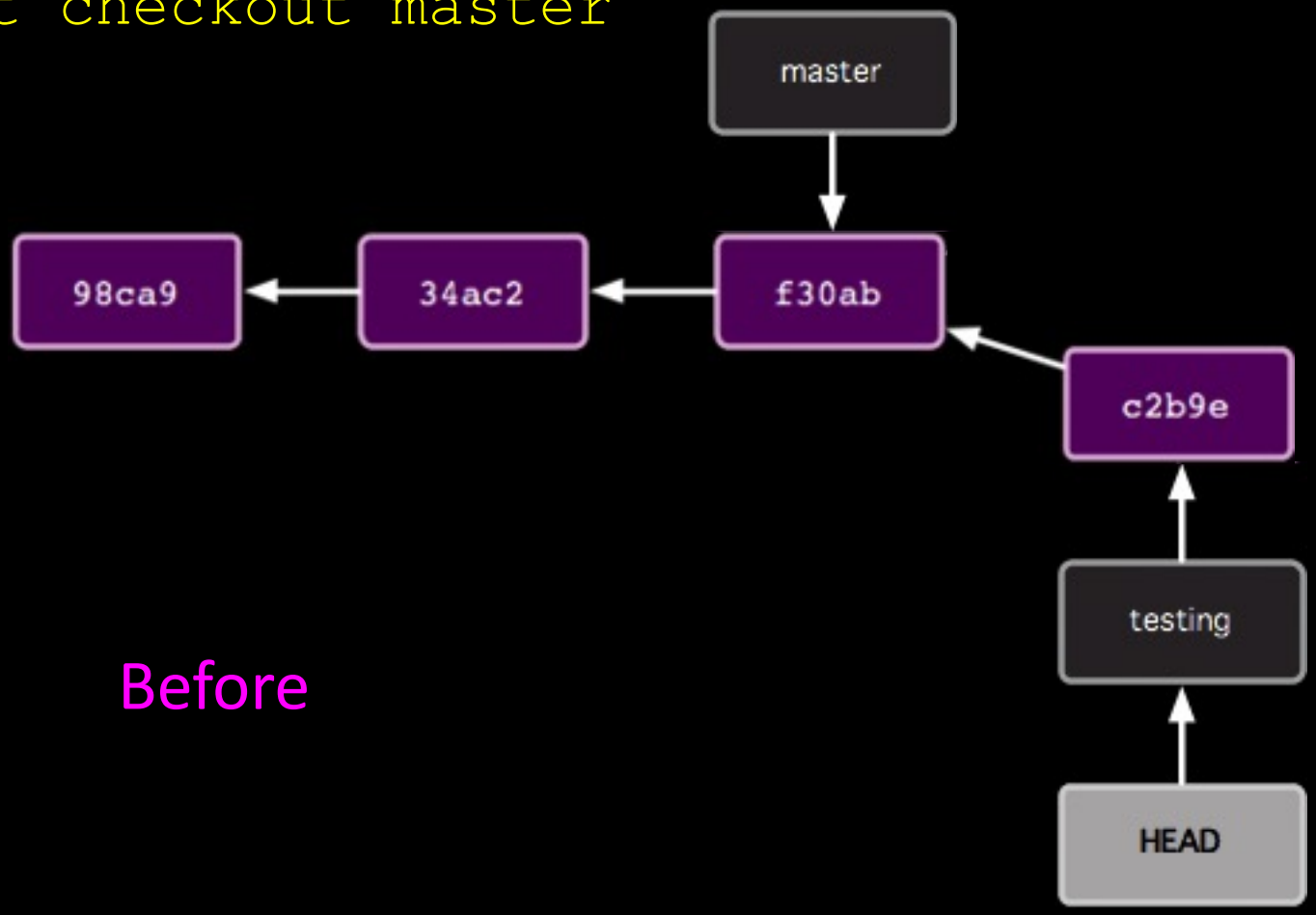
After

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git checkout works

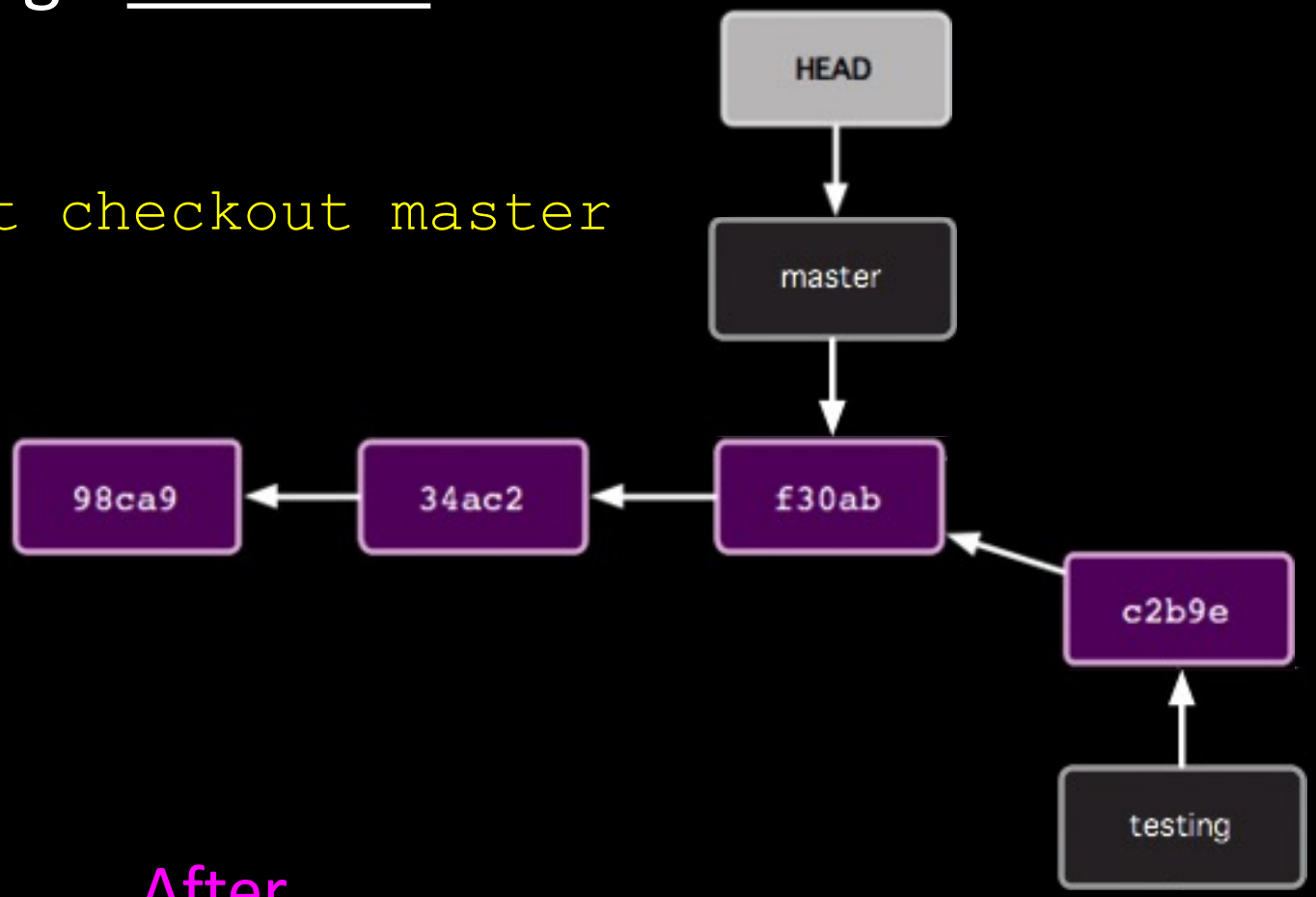
```
$ git checkout master
```



Before

How git checkout works

```
$ git checkout master
```



After

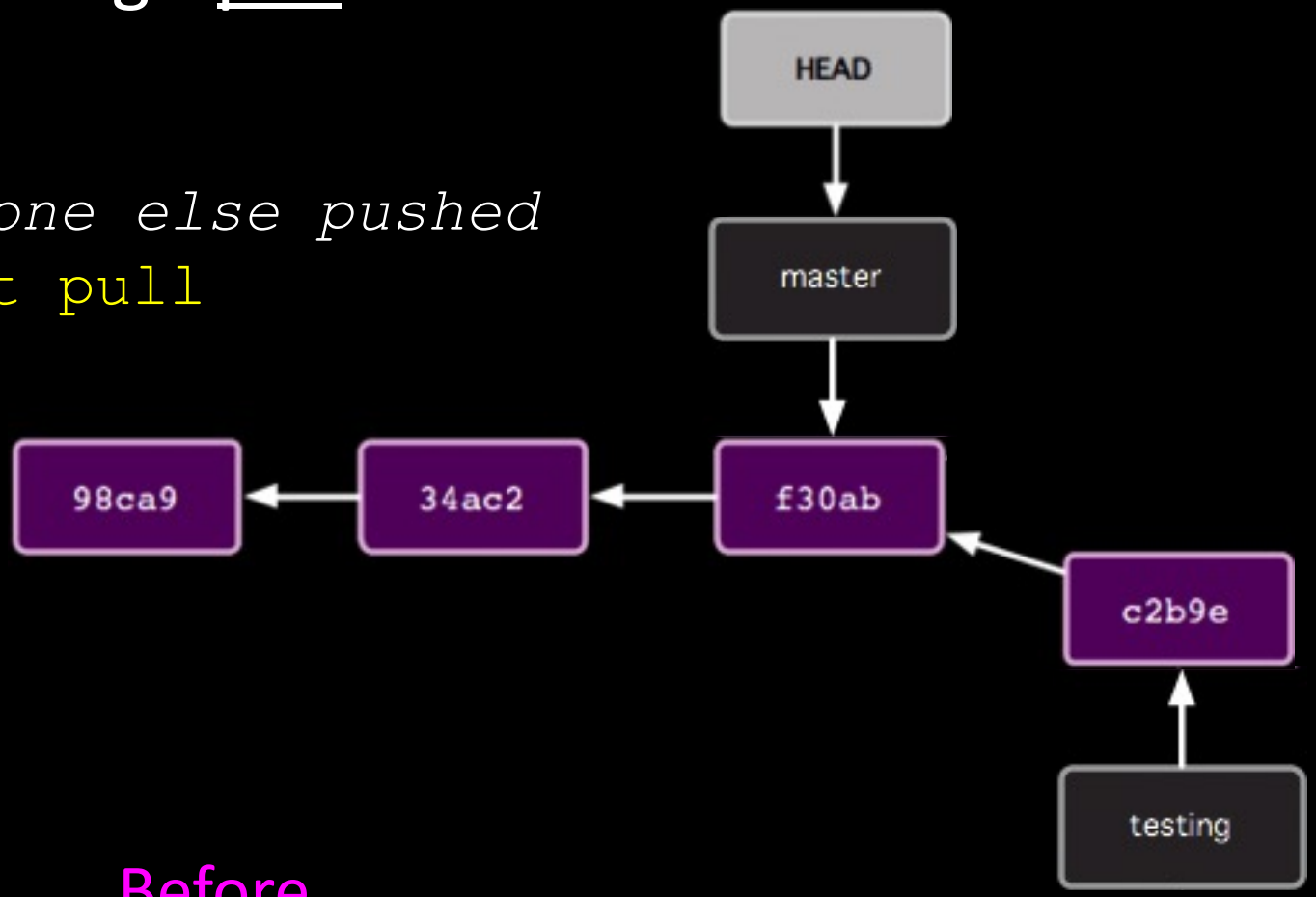
Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git pull works

Someone else pushed

```
$ git pull
```

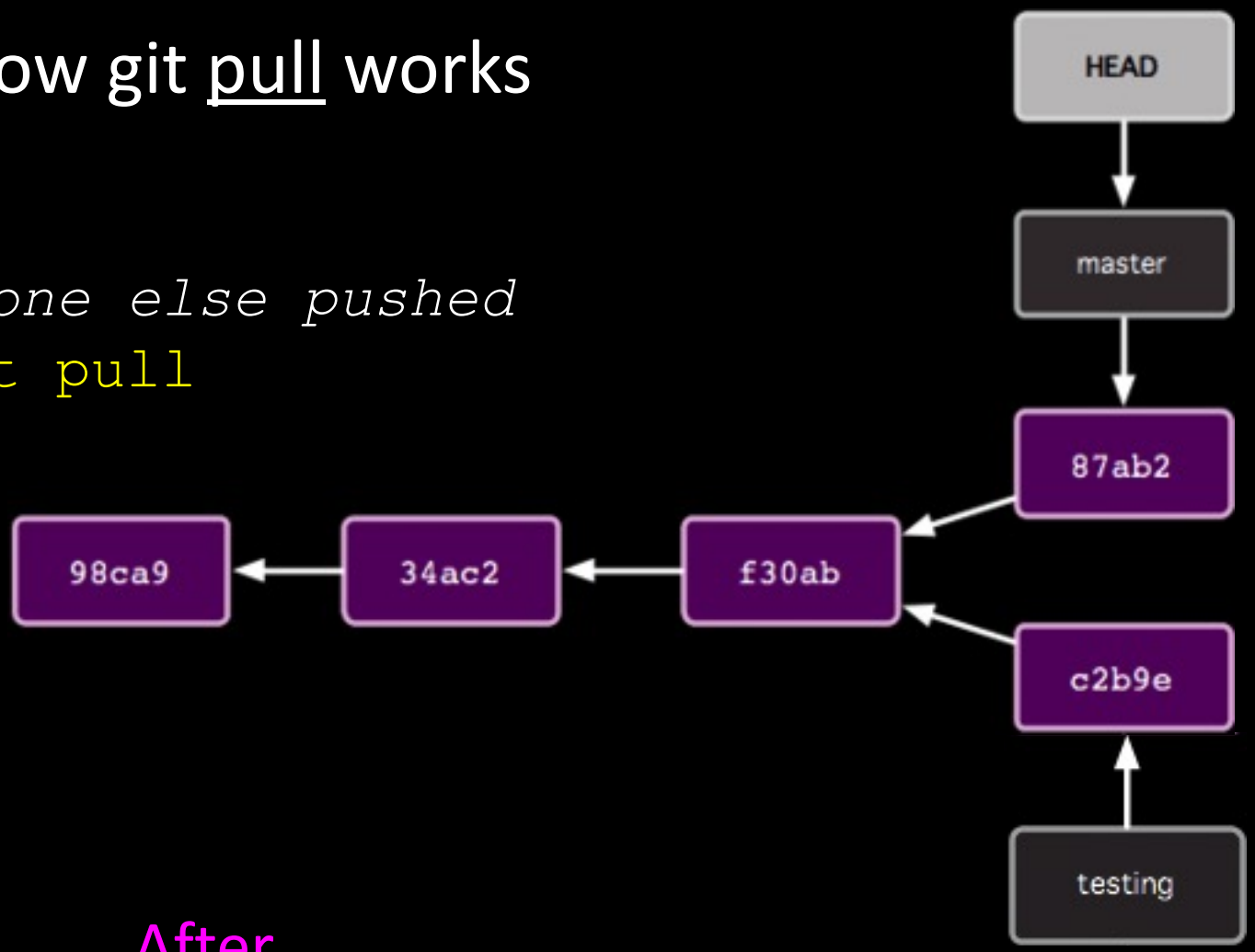


Before

How git pull works

Someone else pushed

```
$ git pull
```



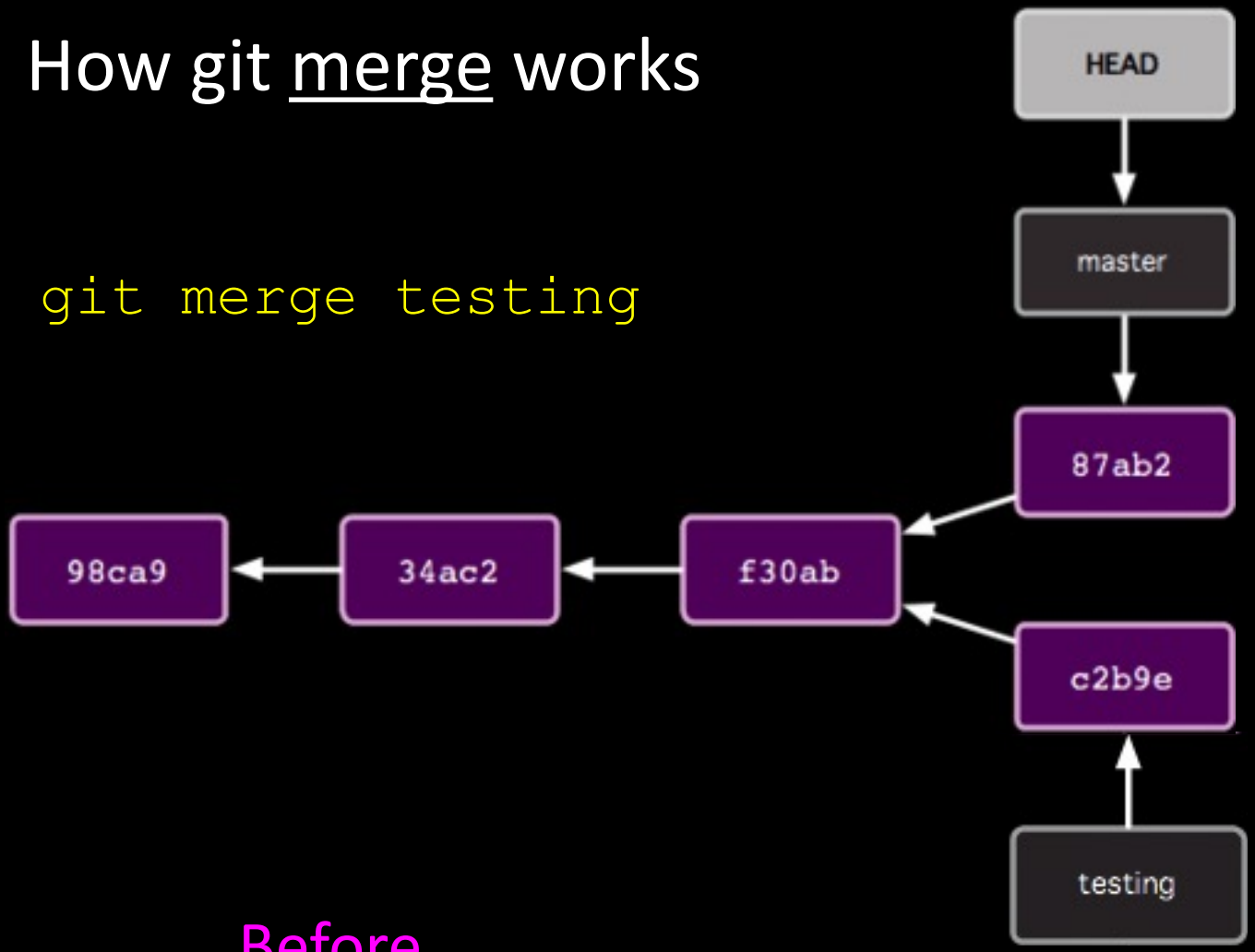
After

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git merge works

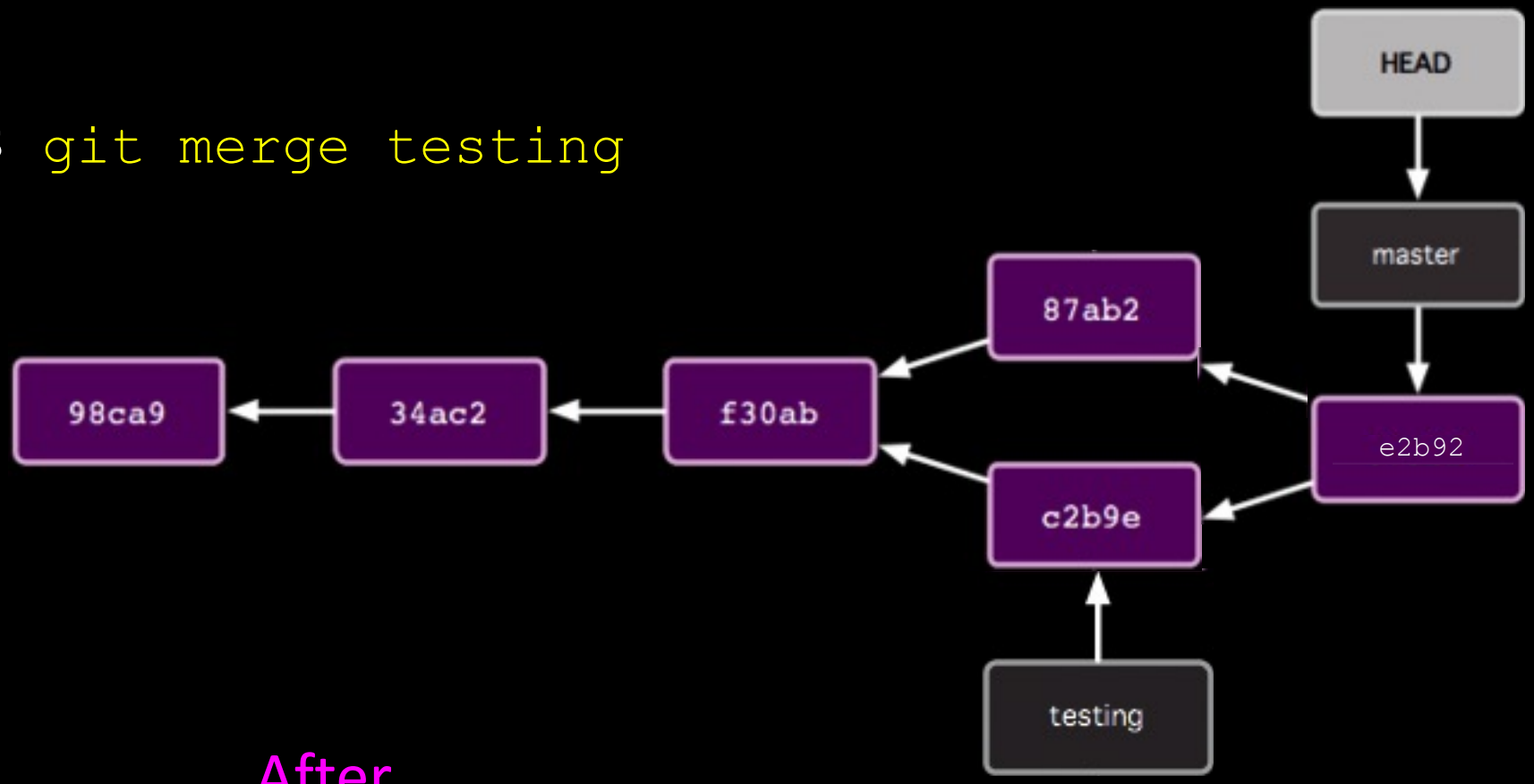
```
$ git merge testing
```



Before

How git merge works

```
$ git merge testing
```



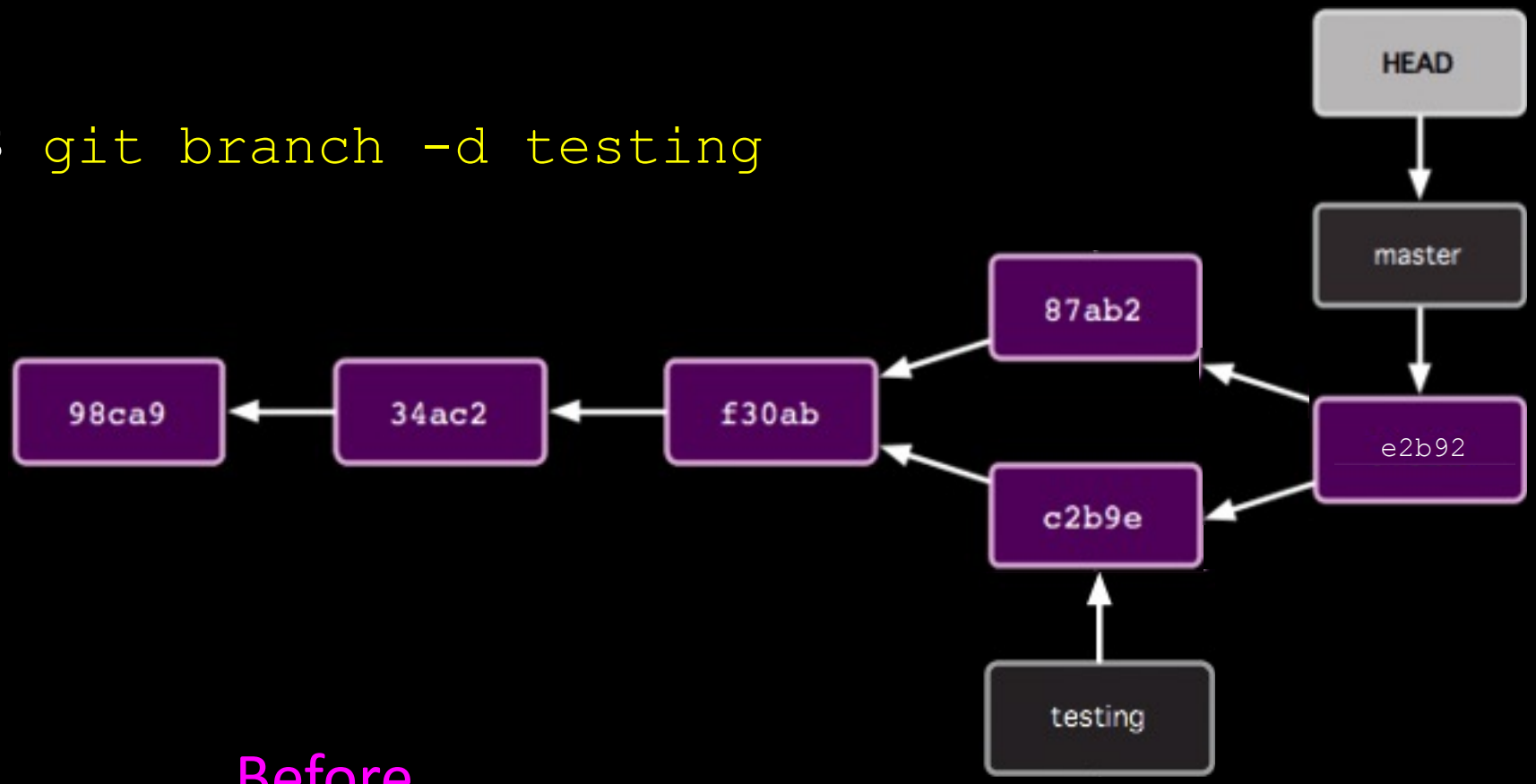
After

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How to delete branches

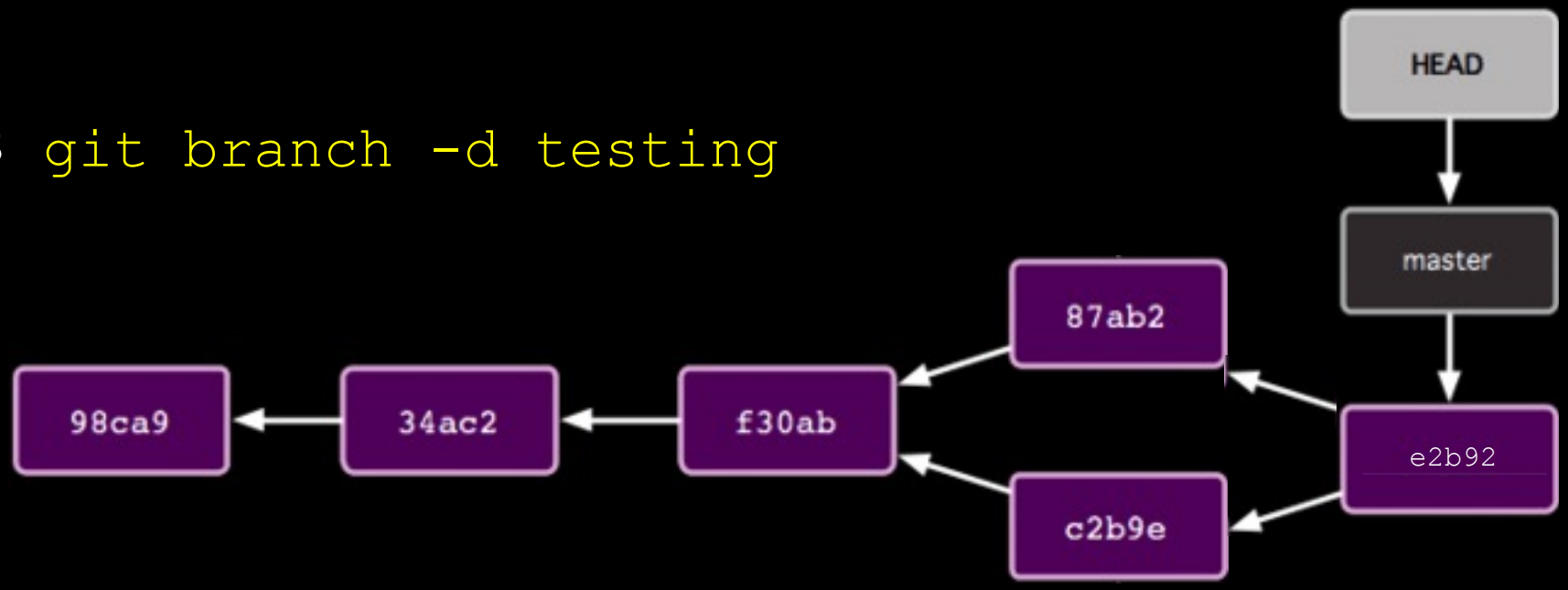
```
$ git branch -d testing
```



Before

How to delete branches

```
$ git branch -d testing
```



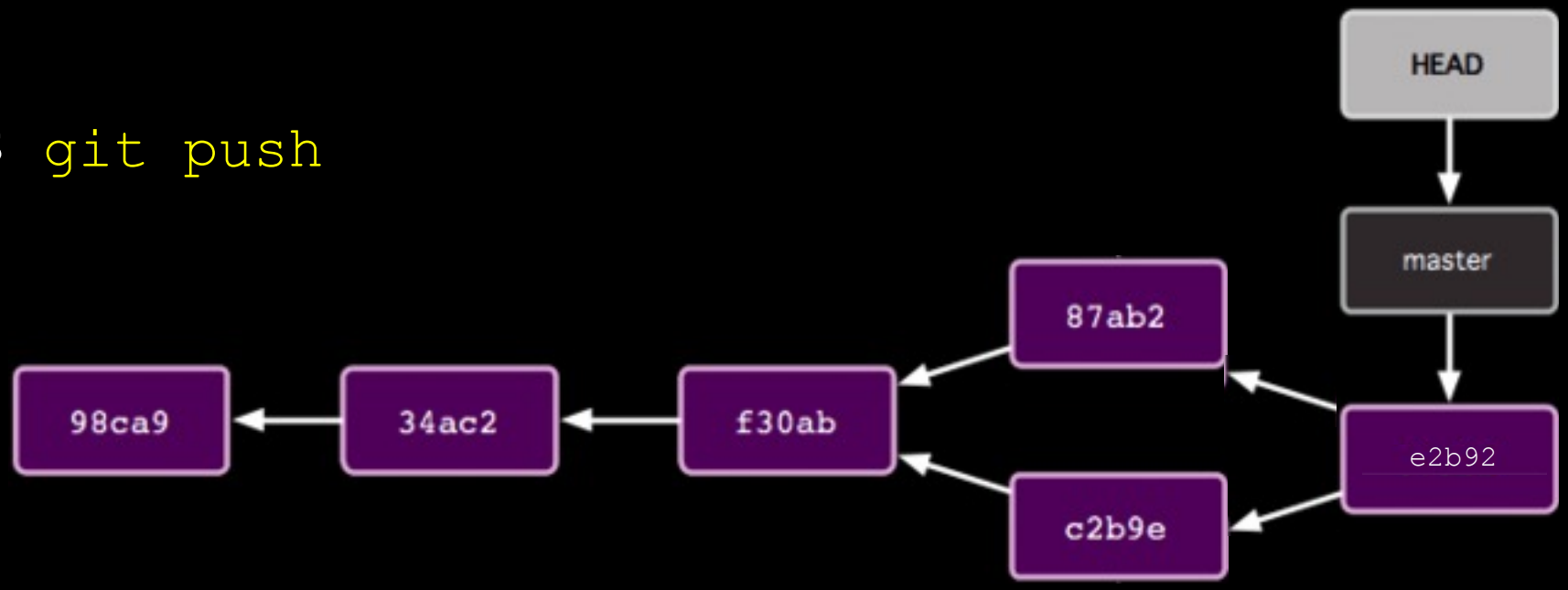
After

Common Workflow #2

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git push works

```
$ git push
```



Should update server repos

(if no one else has pushed commits to master branch since last pull)

What if…

Alice did this (on the branch ***myfix)***:

app/models/micropost.rb

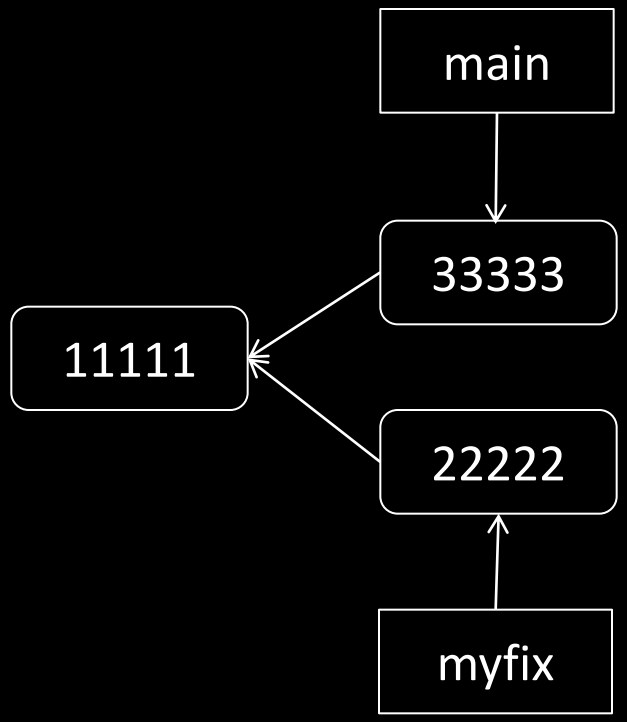
```
class Micropost < ActiveRecord::Base
  validates :content, length: { maximum: 140 }
end
```

Bob did this (on the branch ***main)***:

app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  validates :content, length: { maximum: 120 }
end
```

What if Alice did this?




```
$ git checkout main
$ git merge myfix
```

$ git merge myfix

Auto-merging app/models/micropost.rb

Automatic merge failed; fix conflict and then commit result.

app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
<<<<<<< HEAD
  validates :content, length: { maximum: 140 }
=======
  validates :content, length: { maximum: 120 }
>>>>>>> myfix
end
```

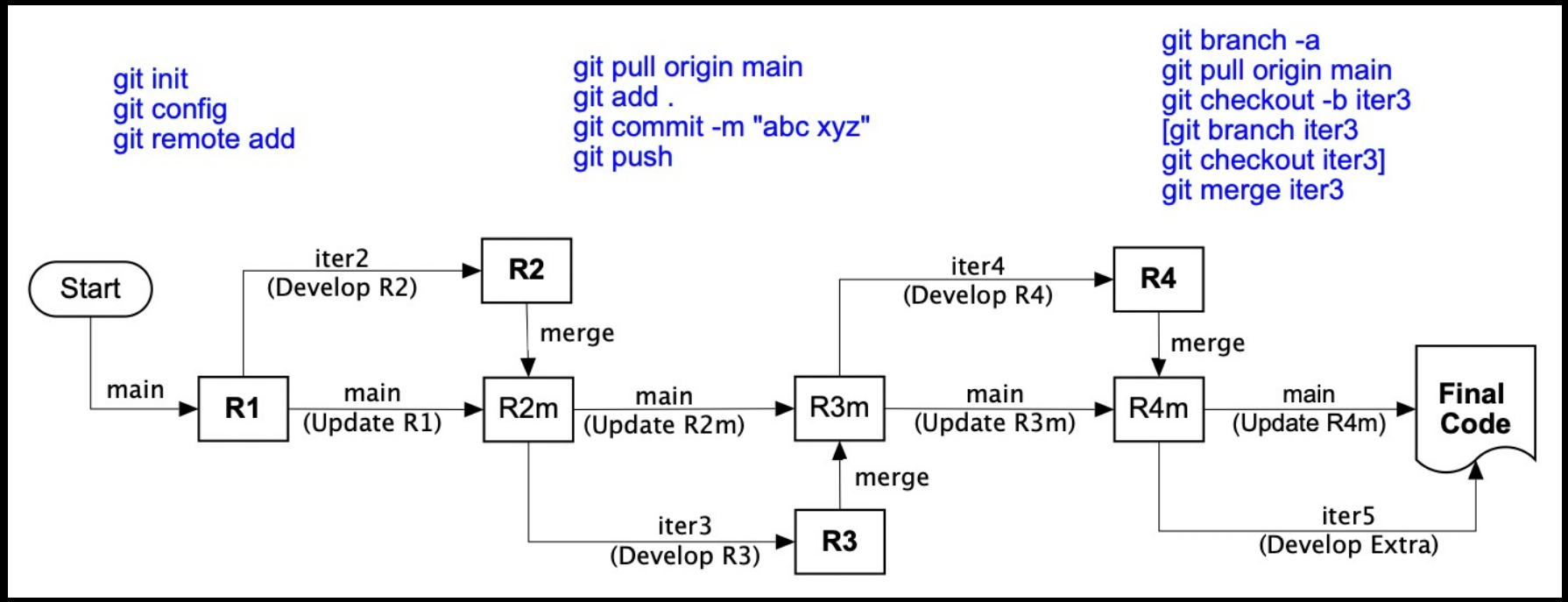
To resolve:

Manually fix the file; git add and commit

Common Working with Git in Summary

```
$ git config --global user.name ”Kien Nguyen"
$ git config --global user.email kiennt@fpt.edu.vn
$ git config --global init.defaultBranch main
$ git config --global core.excludesfile ~/.gitignore
```

```
$ git remote add origin https://gitlab.com/kienpmp/demo.git
```

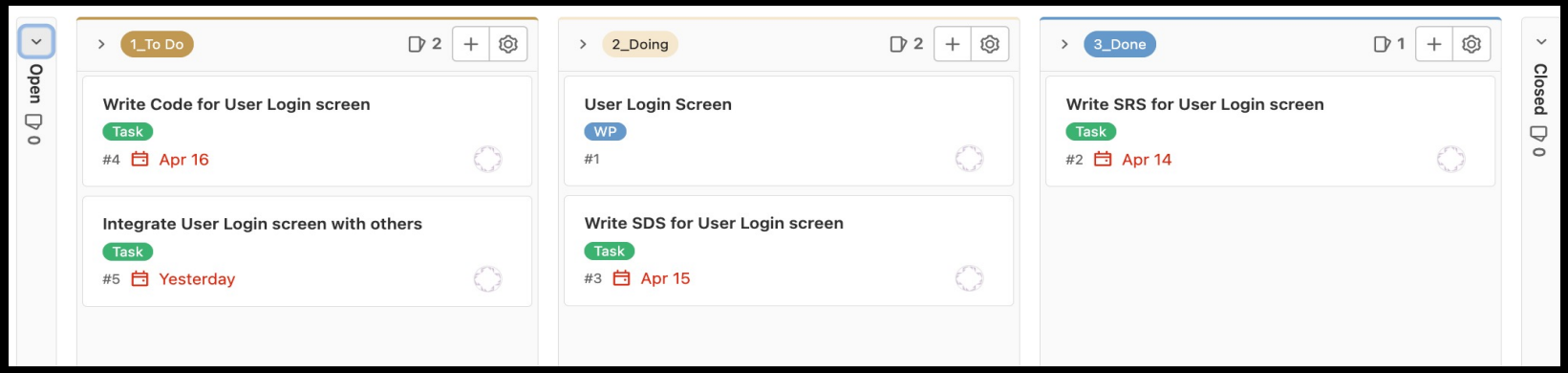


Git Practices & Tips

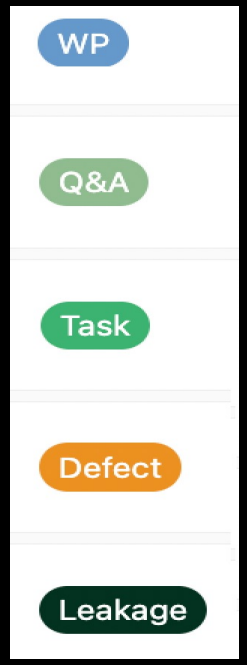
- Ignore files/folders by configuring the file .gitignore
- Pull before starting temp branch or change a branch contents
- Set good commit comments, push right after you done
- Merging may not be as easy as I showed
  - E.g.: Multiple collabs updated same parts of file
- Git output contains lots of hints
  - git status is your friend!
  - git log --oneline to show commit history
  - git diff .. To show the differences
- Team communication important!

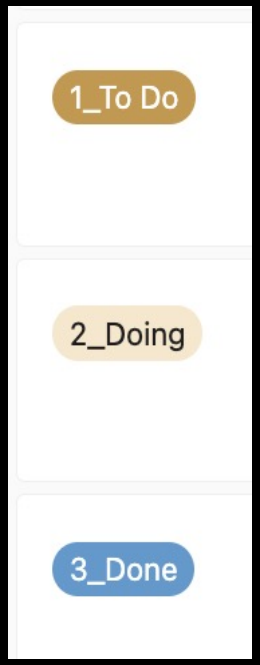
$ vim .gitignore
Cm + Shift + . to show hidden files
VIM (mini tutorial):
i - start editing
ESC - get back to normal mode
:w - save :q - quit
:wq – save and then quit
:q! – quit without save the file

Project Tracking & Monitoring with GitLab



- Project Milestones -> Issues / Milestones
- Project Issues:
  - Requirements -> Issues / Label = WP
  - Tasks -> Issues / Label = Task
  - Issues: Q&A, Issue, Defect, Leakage,..
- - Issue Status: [Open: 1_To do, 2_Doing, 3_Done], [Closed]





Q&A