Intro to Git

Linux Moderators
Open Source Community

Agenda

- What is Git and why use it.
- Git Basics
- Git Branching
- Git on the server

What Is Git ??

Git is
A Free Open Source DVCS

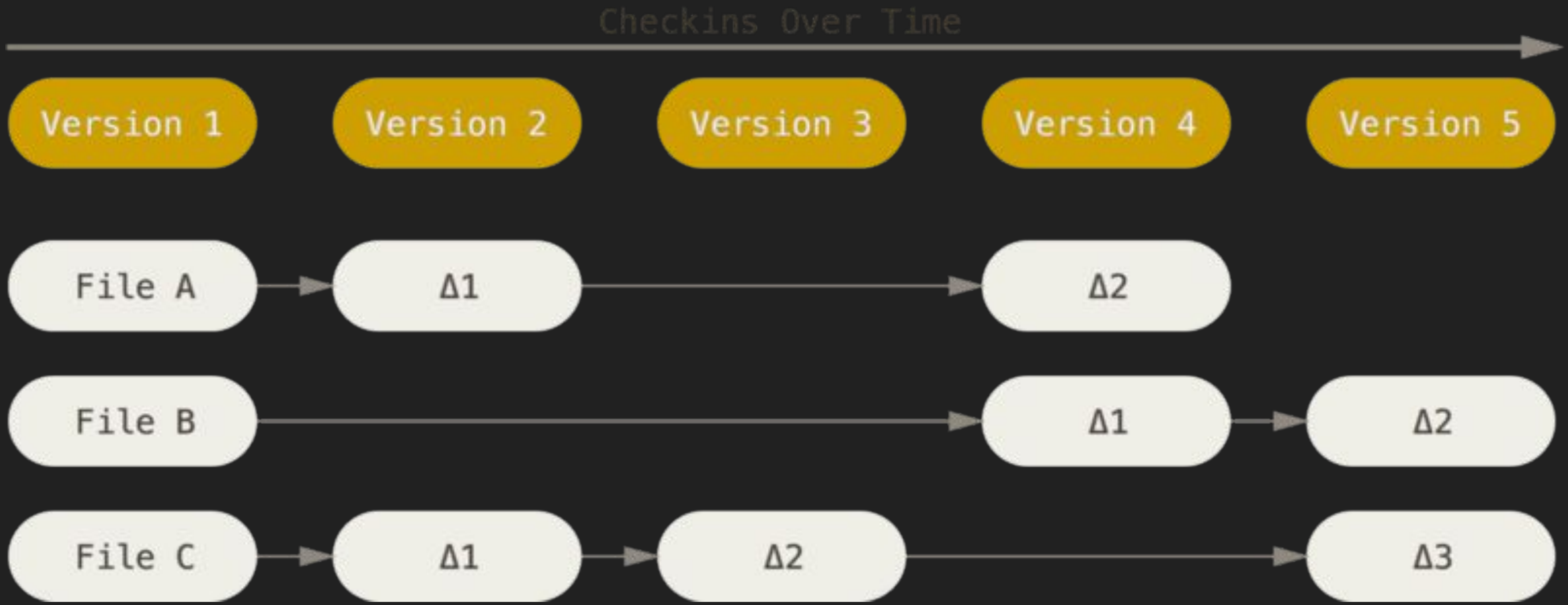
”distributed version control system ”

How it works ?

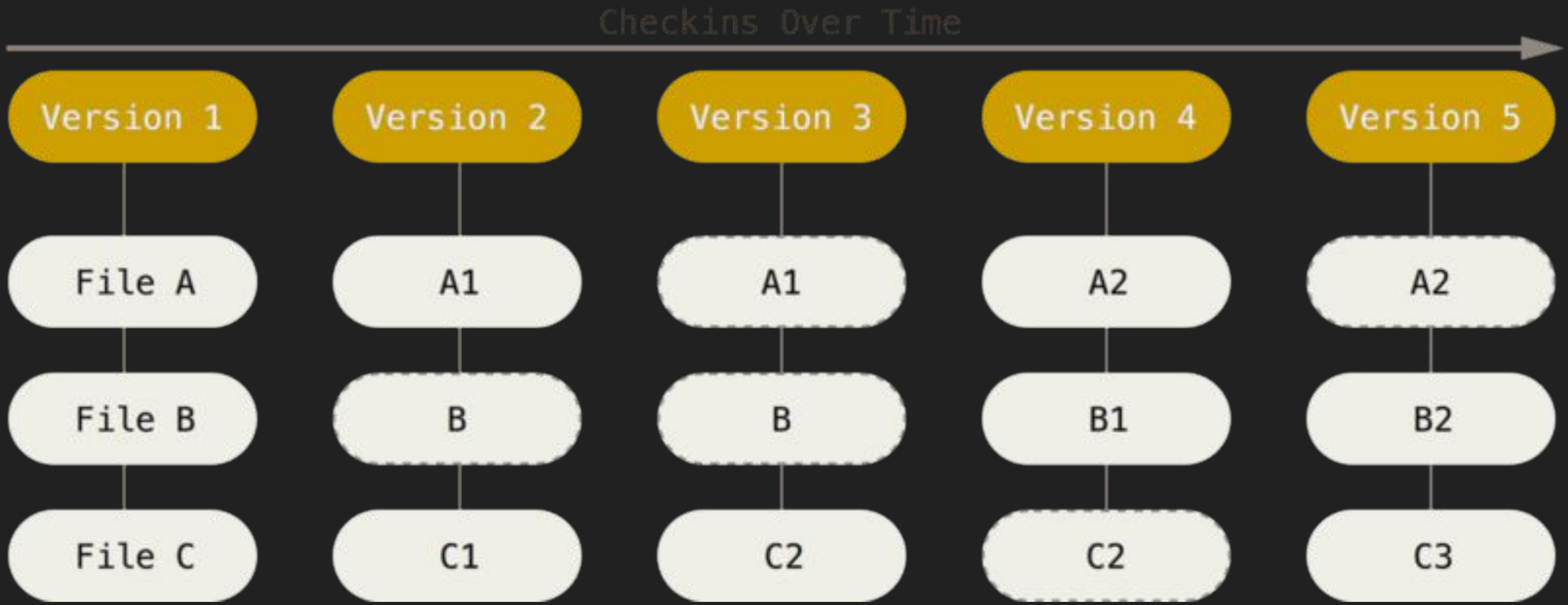
1. Snapshots, Not Differences

- Compares SHA1 sum of modified files.
    - Faster, Easier
- Saves the new file as a Whole
- Each Snapshot contains references to the files

2. Stores new versions of files only



3. Each snapshot contains each file



4. Everything is local “distributed”

- Each Computer has all the history of the project
- Makes it faster, easier
- Can work offline any time

5. Git has Integrity

- Everything in Git is CHECKSUMMED
  - fast ,reliable

- Uses SHA-1 hash
  - a 40 hexadecimal char string

- Easy to recognize corrupted files.

6. Git Generally Only Adds Data

- Everything is almost reversible
- We will talk Later about how to undo almost anything

7. Files’ Three States “Ultra Important”

- Modified
- Staged
- Committed

How to Use it?

Install Git

- sudo apt update
- sudo apt install git

First Time Configurations

- git config --global user.name "John Doe"
- git config --global user.email johndoe@example.com
- git config --global core.editor vim

OR

- git config --global core.editor nano
- git config --global merge.tool meld “later”

Git Help

- git help <verb>

  OR

- man git-<verb>
- Ex: git help commit
- git <verb> -h “summarized”

Some Important terms

- Repository
- Index “Staging area”
- working tree “Project tree”
- Commit
- Branch
- Tag
- Master
- Head

Now you’re set to GO !!

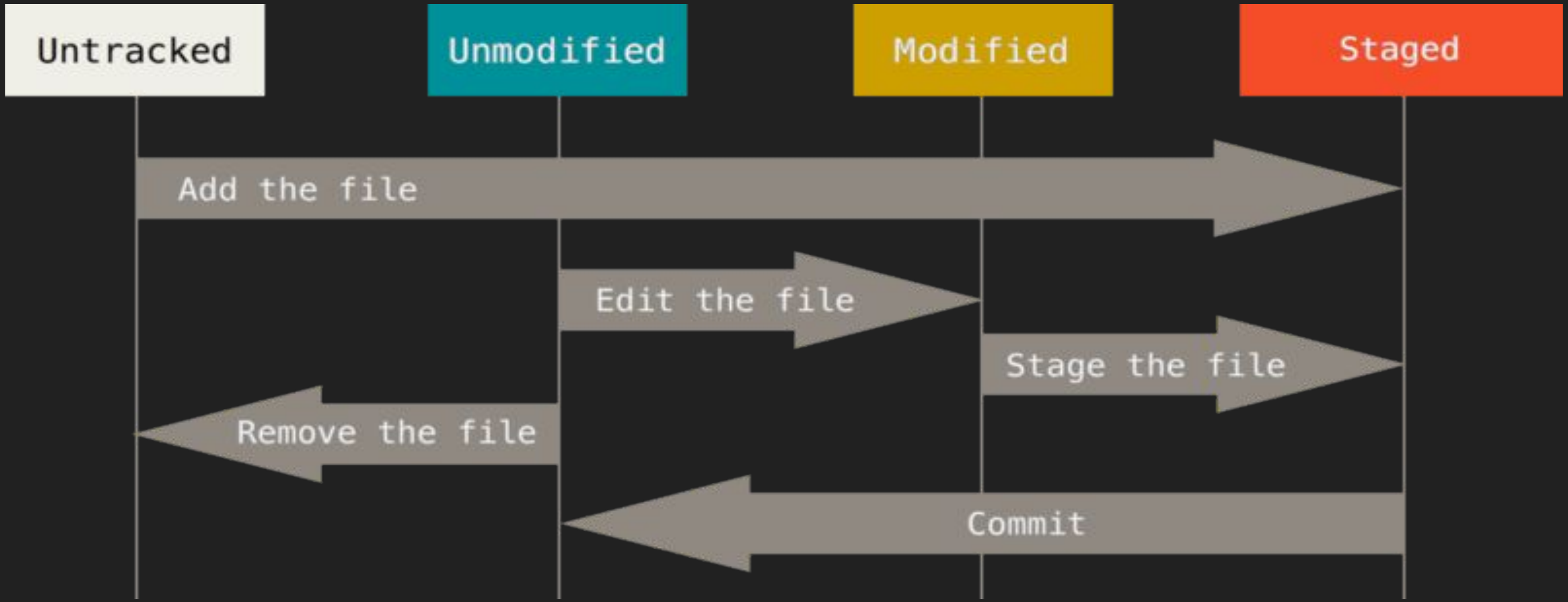
Git Basics

- Getting a Git Repository
- Recording Changes to the Repository
- Viewing the Commit History
- Undoing Things
- Working with Remotes
- Tags and Aliases

Git Repository

1. Make a new one
   - git init
2. Clone an existing one
   - git clone <URL>

Recording Changes to the Repository



Checking status of your files

- git status

  OR

- git status -s
  - ?? “ untraked ”
  - A “ staged ”
  - M “ Modified staged “
  - M “ Modified untracked “

Now try !!

- git status
- echo 'I’ > README
- git status

Staging Modified Files

- git add <files>
- Stage your files after adding not before

Now try !!

- git add README
- git status
- echo “USE” >> README
- git status
- git add README
- git status

Ignoring Files “Ultra Important”

- A file listing patterns to match them with files in your repositories
- File name “.gitignore”
- .gitignore common templates
  - https://github.com/github/gitignore
- See “man gitignore” for more information
- Linux Kernel repo has 206 gitignore files

Viewing staged and unstaged

- git status
- git diff
- git diff --staged

Now try !!

- git commit -am “cleaning the stage”
- echo “Git” >> README
- echo “Git” > Cont.md
- git add README Cont.md
- git diff
- echo “OSC” > Cont.md
- git status
- git diff

Now try !!

- git commit -am “cleaning the stage”
- echo “1” > file
- git add file && git commit -m “new file”
- echo “2” > file
- git add file
- echo “3” >file
- git diff
- git diff --staged

Committing Changes

- git commit
- git commit -am “Your Message”

Now try !!

- git commit -am “cleaning the stage”
- echo “1” > newfile
- git commit -am “new file”
- git status
- git add newfile && git commit -m “new file”
- Change “newfile” and try using commit -am “changed new file”

Remove
Move

- git rm <file name>
- git mv <file name>

Commit History

- git log
- git log -p
- Git log -<number>
- git log --since <date>
    - EX : “ 2008-01-15”
    - EX : "2 years 1 day 3 minutes ago"
- git log --until <date>

UnDoing Things

Unstaging a Staged File

“Undo add”

- git reset <files>
- git reset

Amend

- git commit --amend

Undo Commit

- git revert <commit>
- git checkout <file|commit>
- git reset
- git reset --soft <commit>
- git reset --hard <commit>
- <commit>:
  - SHA1
  - Head^,Head@{number}

Remote Repositories

Clone a repository

- git clone <URL>
    - https://github.com/
    - git://github.com/koke/grit.git
    - git@github.com:mojombo/grit.git
    - /srv/git/project.git
    - file:///srv/git/project.git

Showing Your Remotes

- git remote
- git remote -v

Add Remotes

- git remote add <Name> <URL>

Fetching and Pulling Repositories

- git fetch <remote>
- git pull <remote> <branch>

Pushing into Repositories

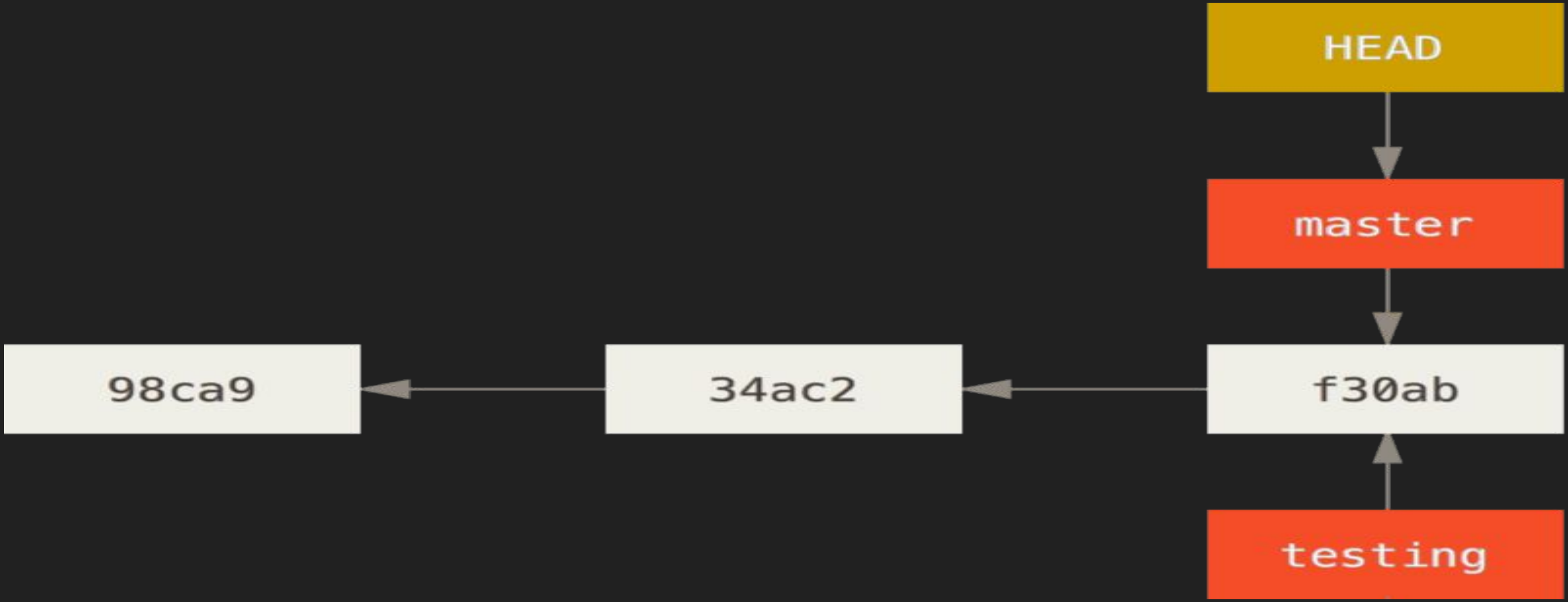
- git push <remote> <branch>

Branching

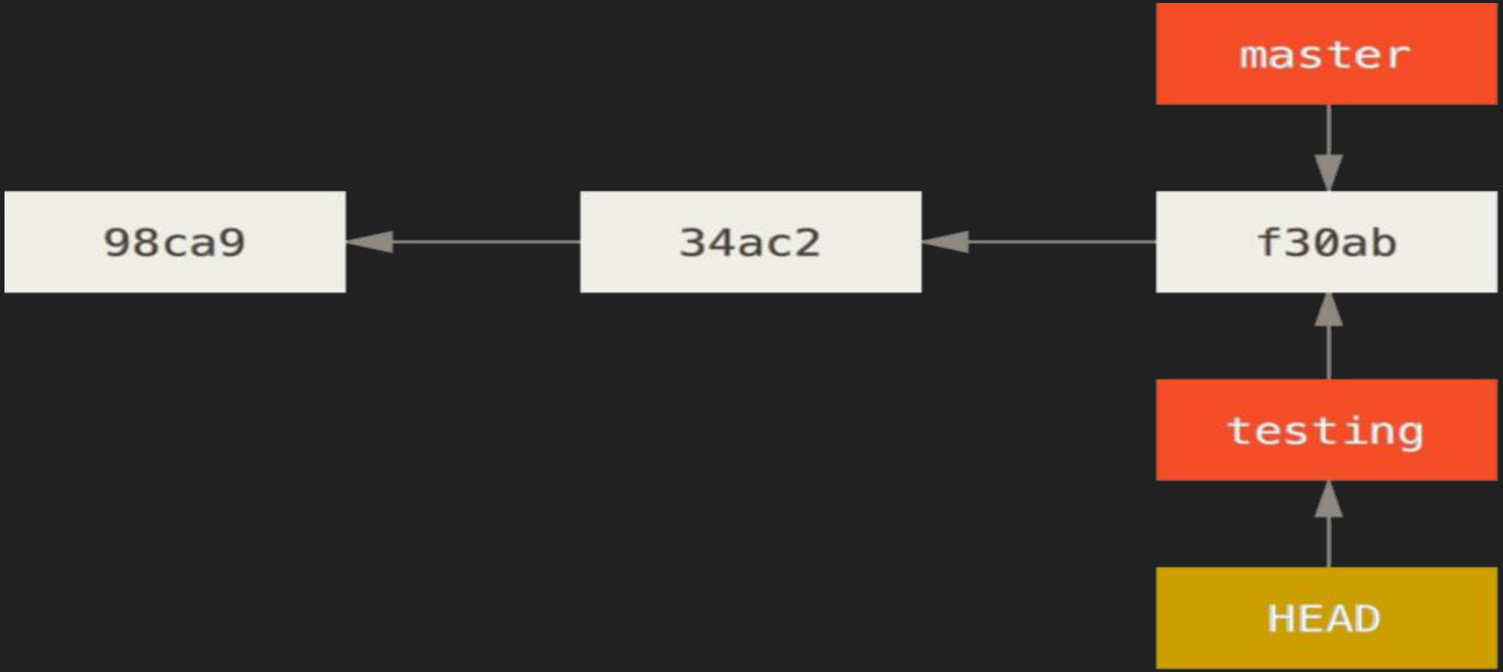
Branching

- List branches
  - git branch
  - git branch -a
- Create new branch
  - git branch <new name>
- Switch branches
  - git Checkout <branch name>
- Delete Branches
  - git branch -d <branches>

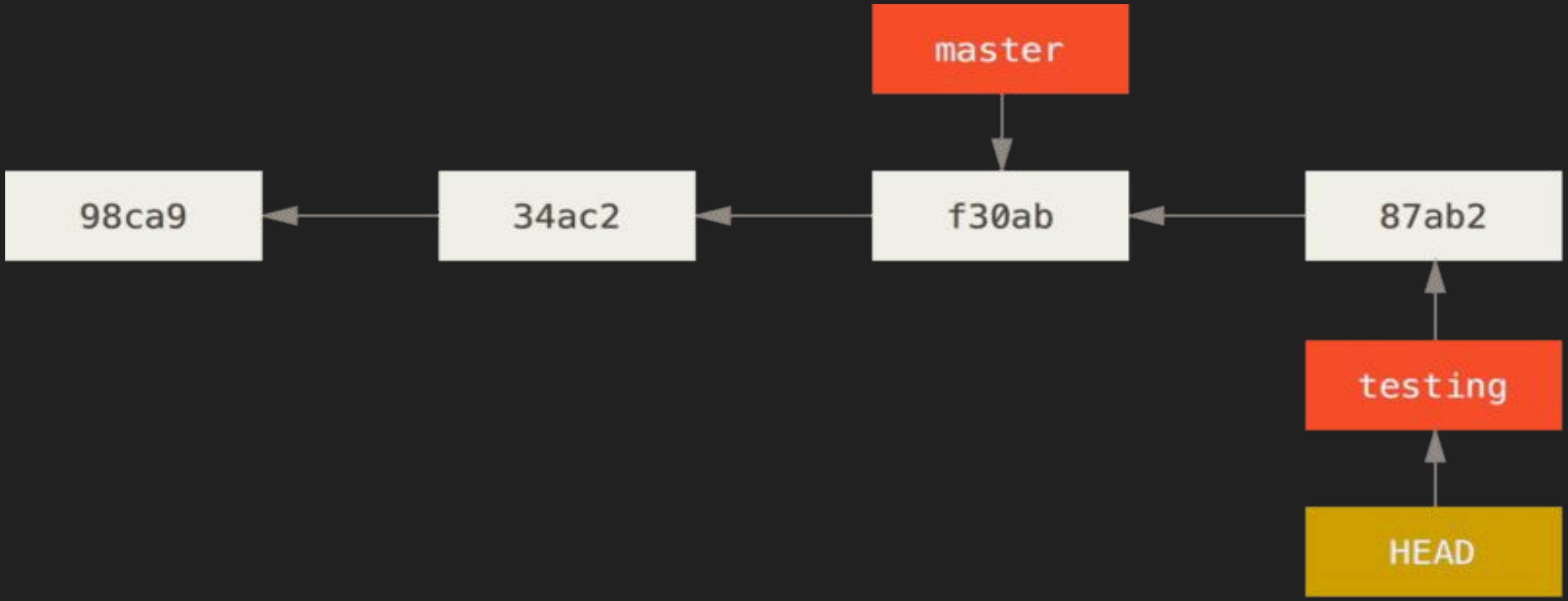
Creating a New Branch “git branch testing”



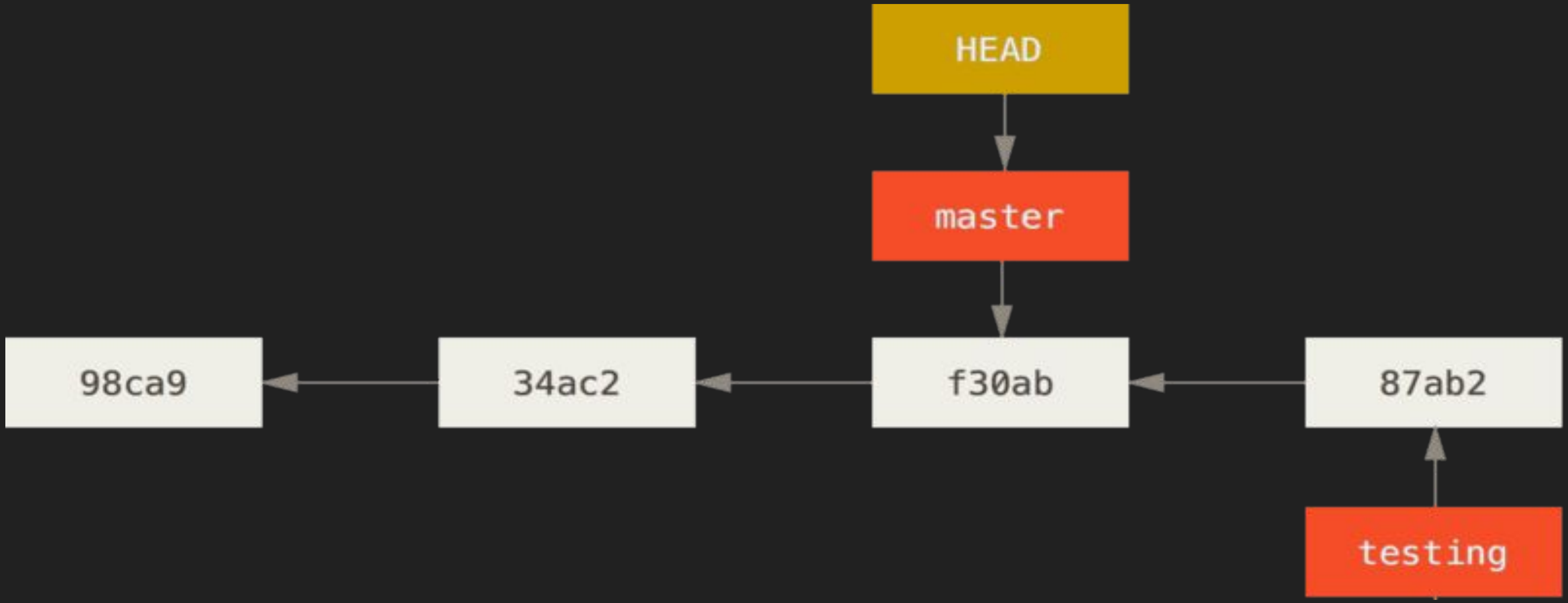
switching Branch “git checkout testing”



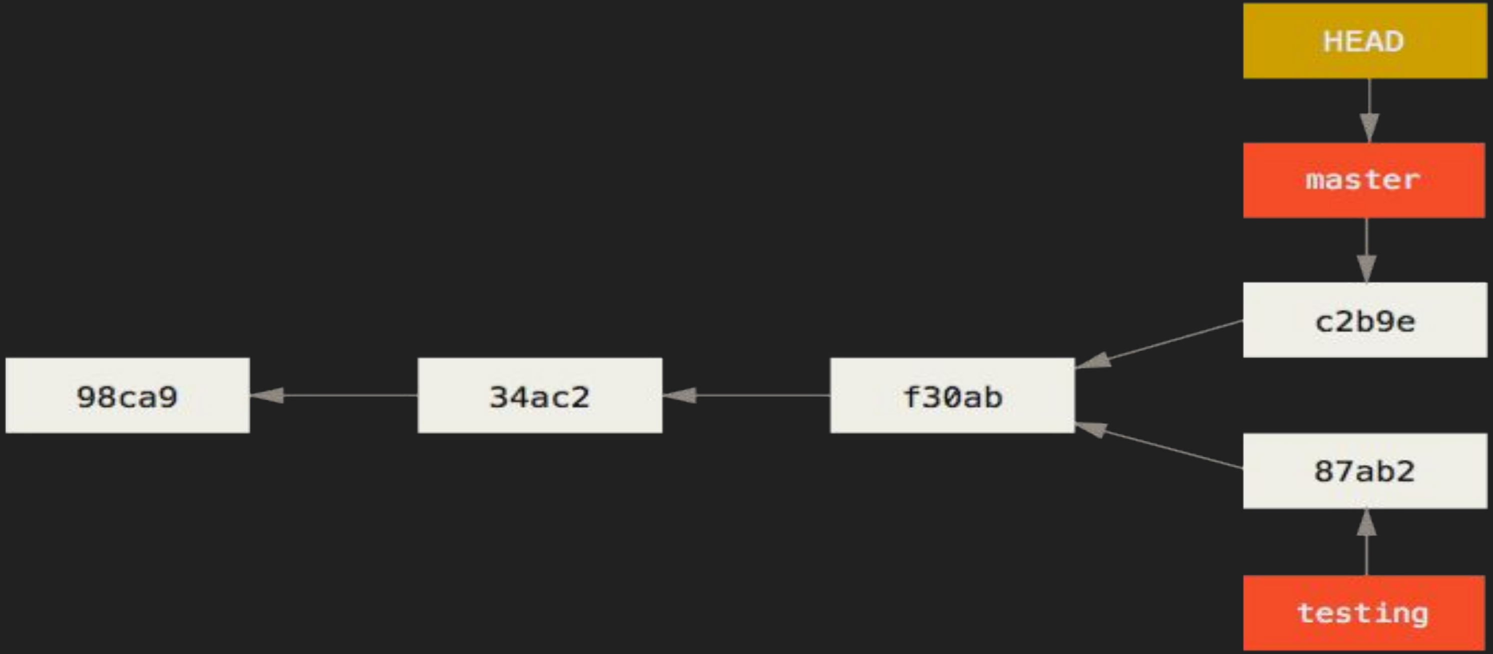
Commiting in a New Branch



Return to master Branch



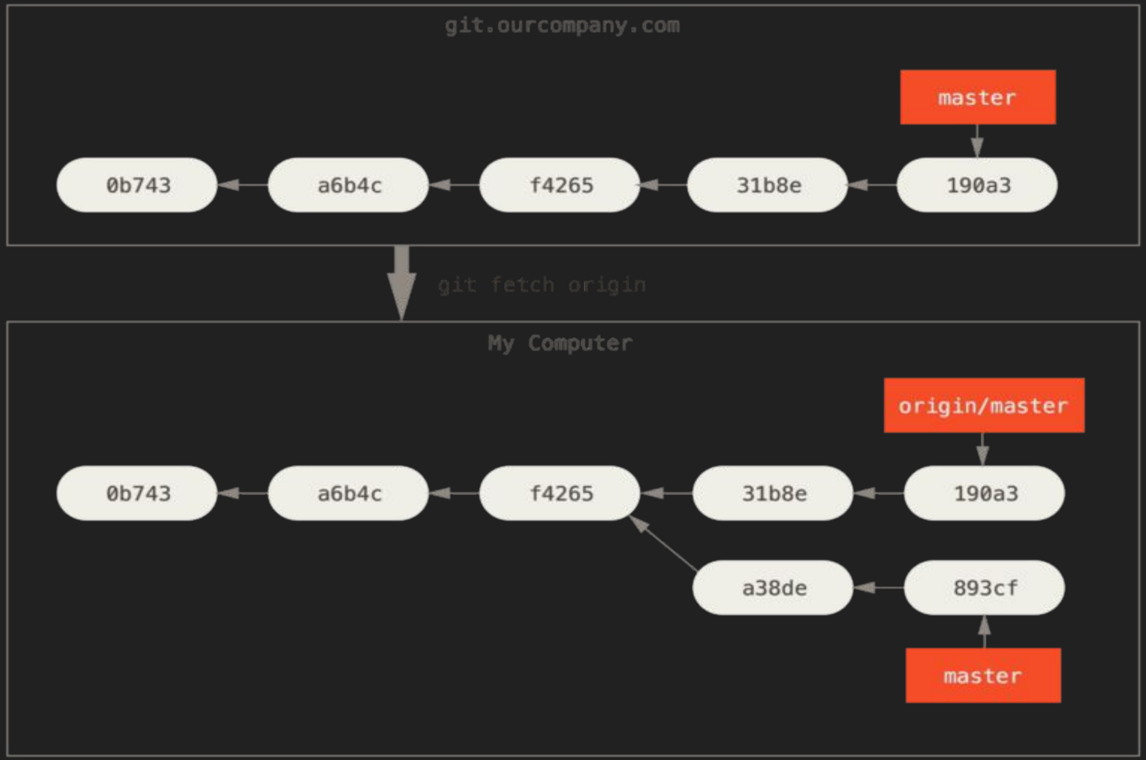
Committing again Branch



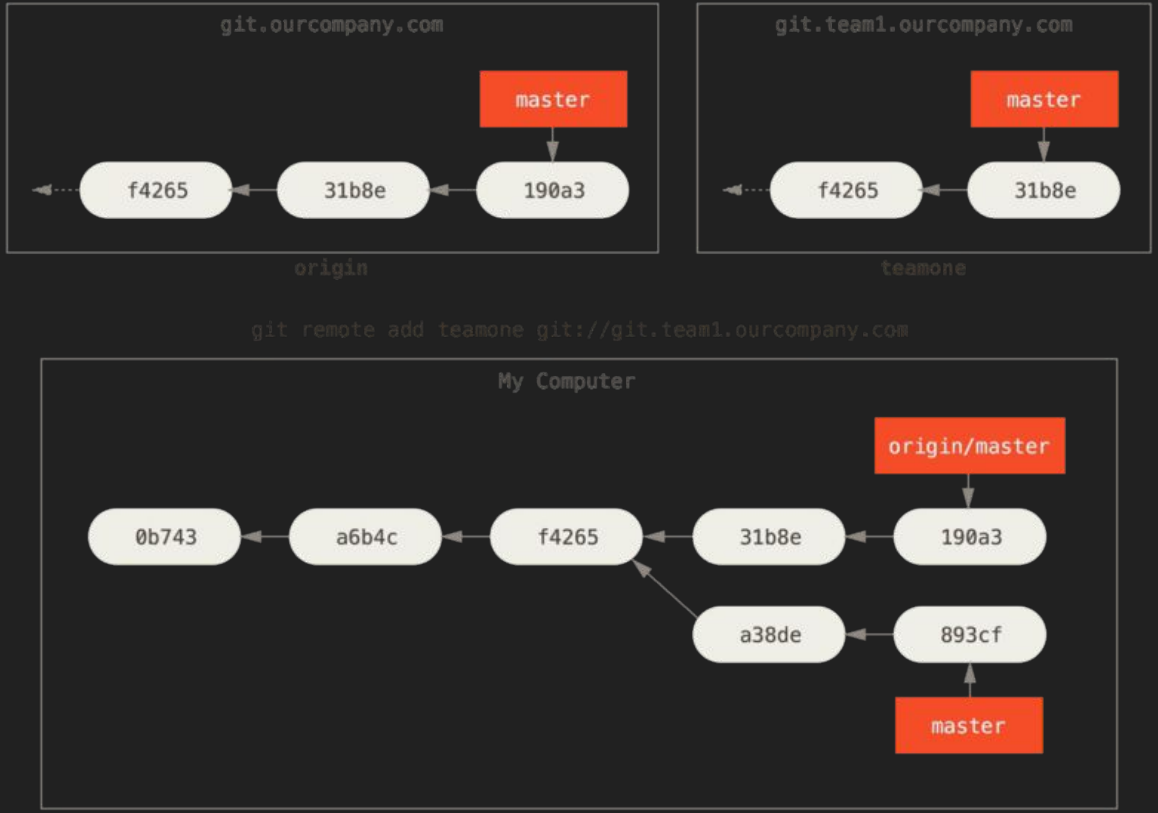
Merging Branching

- Fast-forward
- Recursive “Three Way merge”
  - git mergetool
    - meld

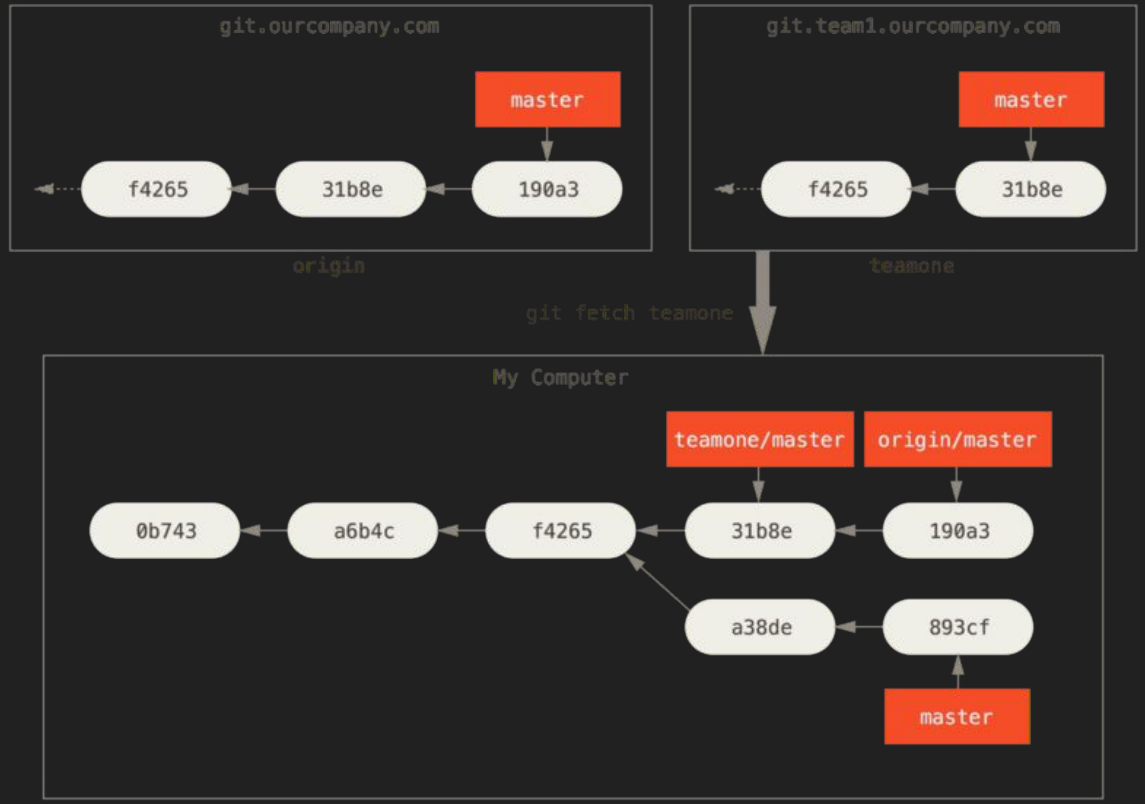
Remote Branches



Remote Branches



Remote Branches



Local Git Repositories

git daemon

1. --base-path=<path>
2. --export-all
3. --reuseaddr
4. --informative-errors
5. --verbose
6. --enable=receive-pack
7. You can use aliases

git aliases

1. git config --global alias.<name> ‘commands’
    - git config --global alias.serve '!git daemon --base-path=. --export-all --reuseaddr --informative-errors --verbose'
    - git config --global alias.hub '!git daemon --base-path=. --export-all --enable=receive-pack --reuseaddr --informative-errors --verbose'
2. git hub or git serve

resources

1. Pro GIt
    - https://git-scm.com/book/en/v2
2. git daemon
3. rest vs checkout vs revert

The End ^_^