



DevOps education program

VCS

Git, GitHub

Lecture 1.2

Module 1. DevOps Introduction

Andrii Kostromytskyi



Agenda

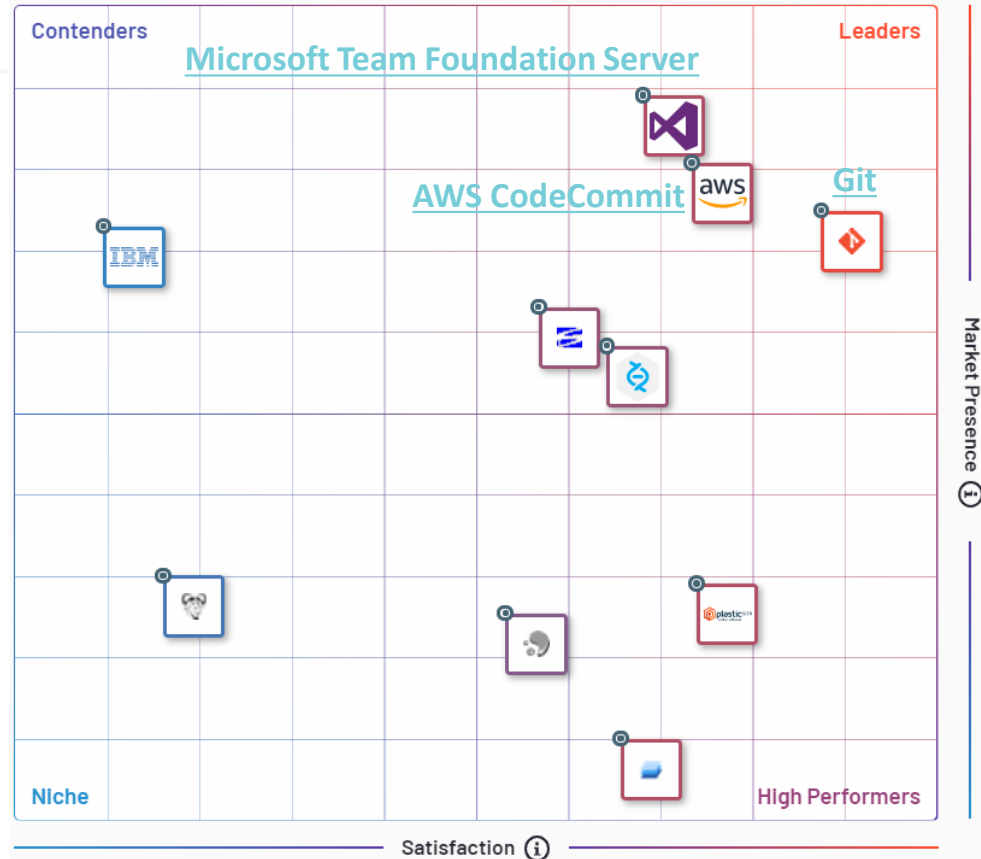
- What is a VCS. VCS types
- Installing and configuring git
- Basic commands overview
- Basic workflows
- Git hosting services
- Markdown language
- Q&A

What's a VCS?

Version control is a system that records changes to a set of files over time so that you can recall specific versions later

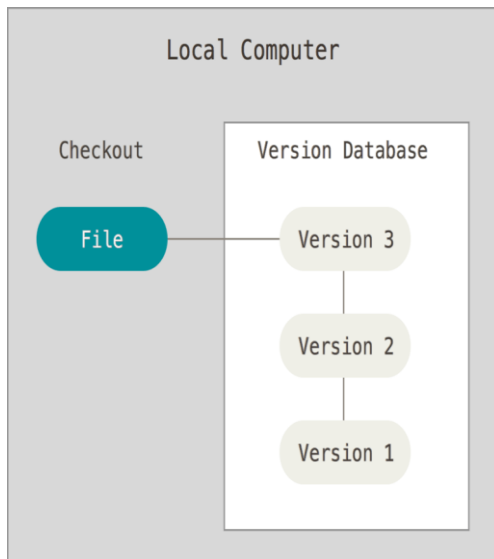
Also usually it is used for

- Collaboration
- CI/CD interactions
- Analysis
- Backups and release management

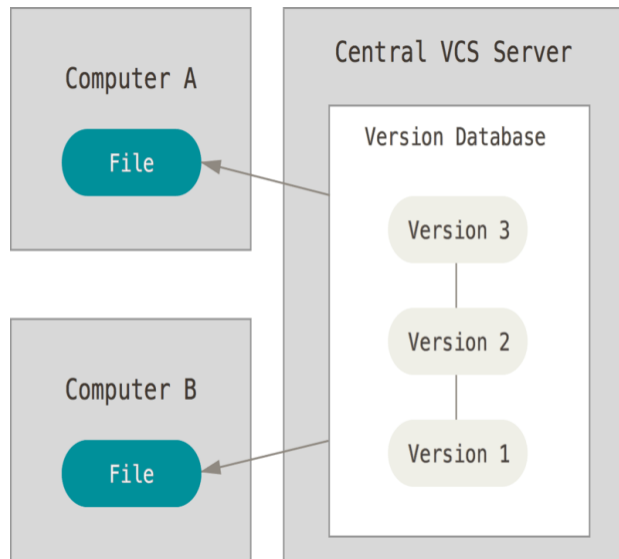


VCS types

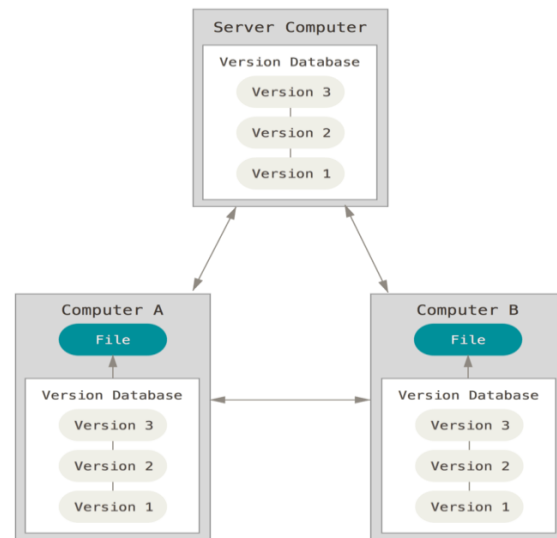
Local version control



Centralized version control



Distributed version control



Installing and configuring Git

Linux

- Via binary installer:

```
$ sudo yum install git-all
```

- If you're on a Debian-based distribution like Ubuntu, try:

```
$ sudo apt-get install git-all
```

Windows

<http://git-scm.com/download/win>

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email johndoe@example.com
```

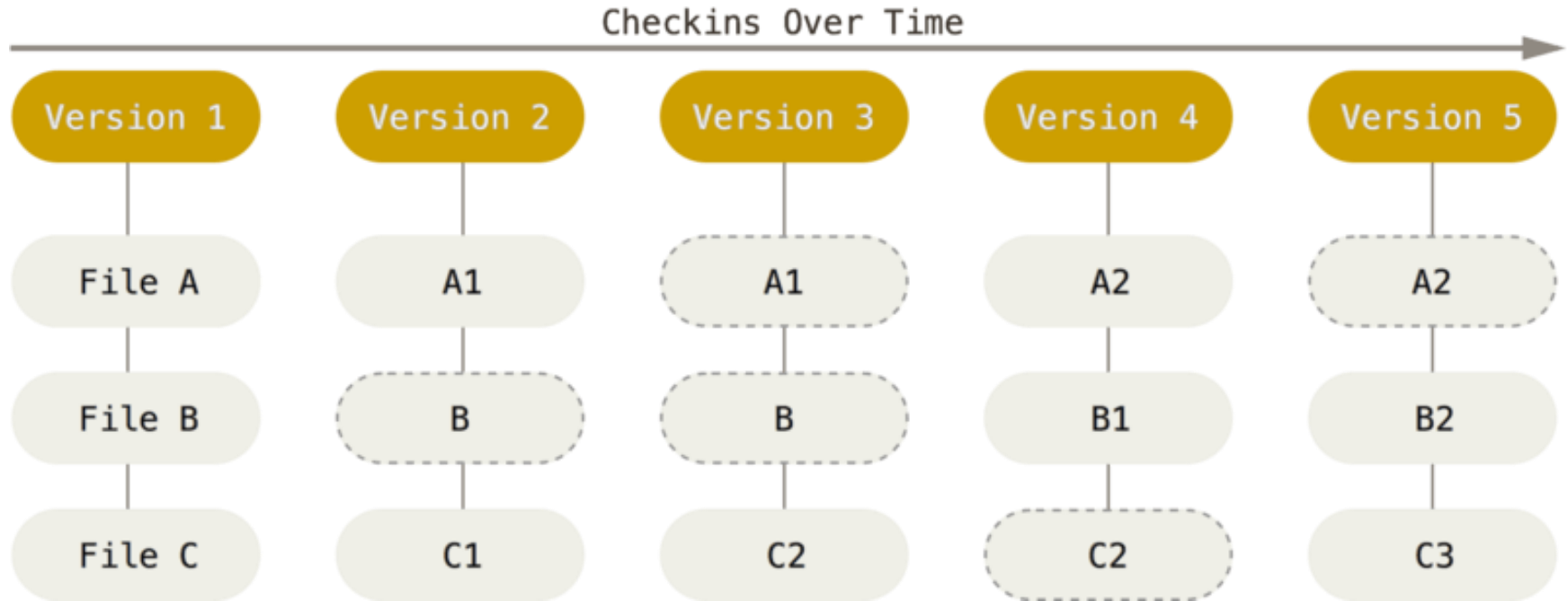
```
$ git help <verb>
```

```
$ git <verb> --help
```

```
$ git <verb> -h
```



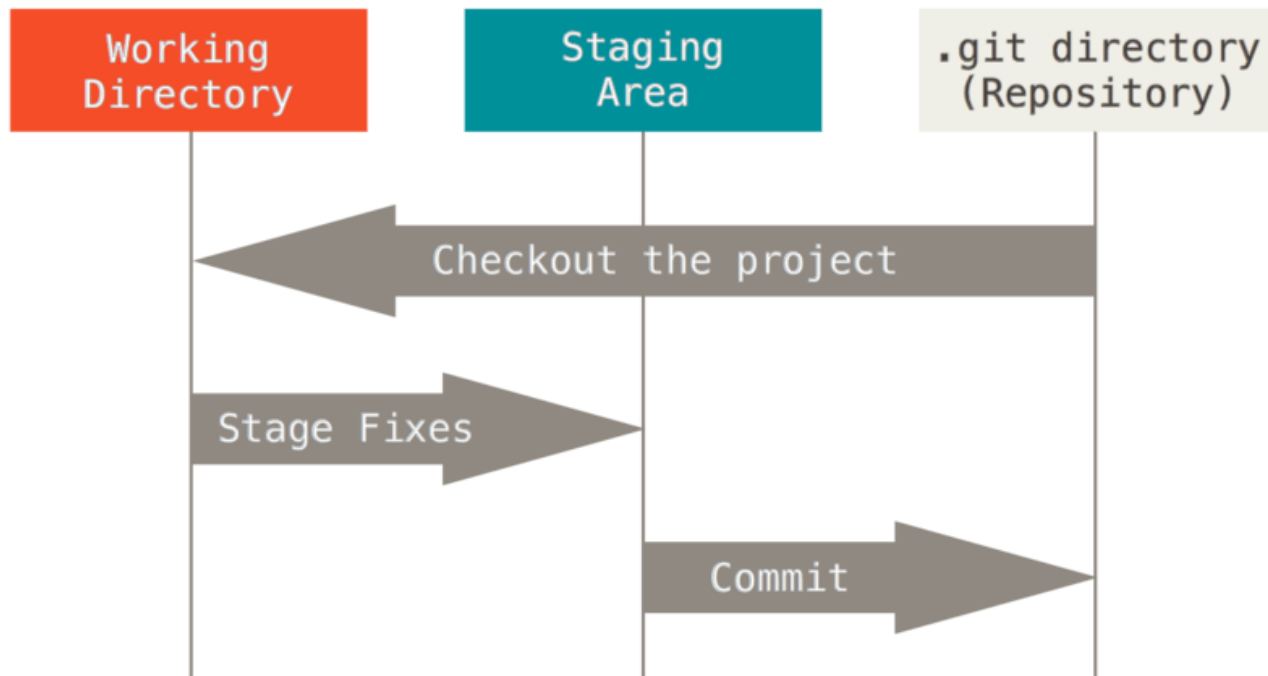
Storing data as snapshots of the project over time



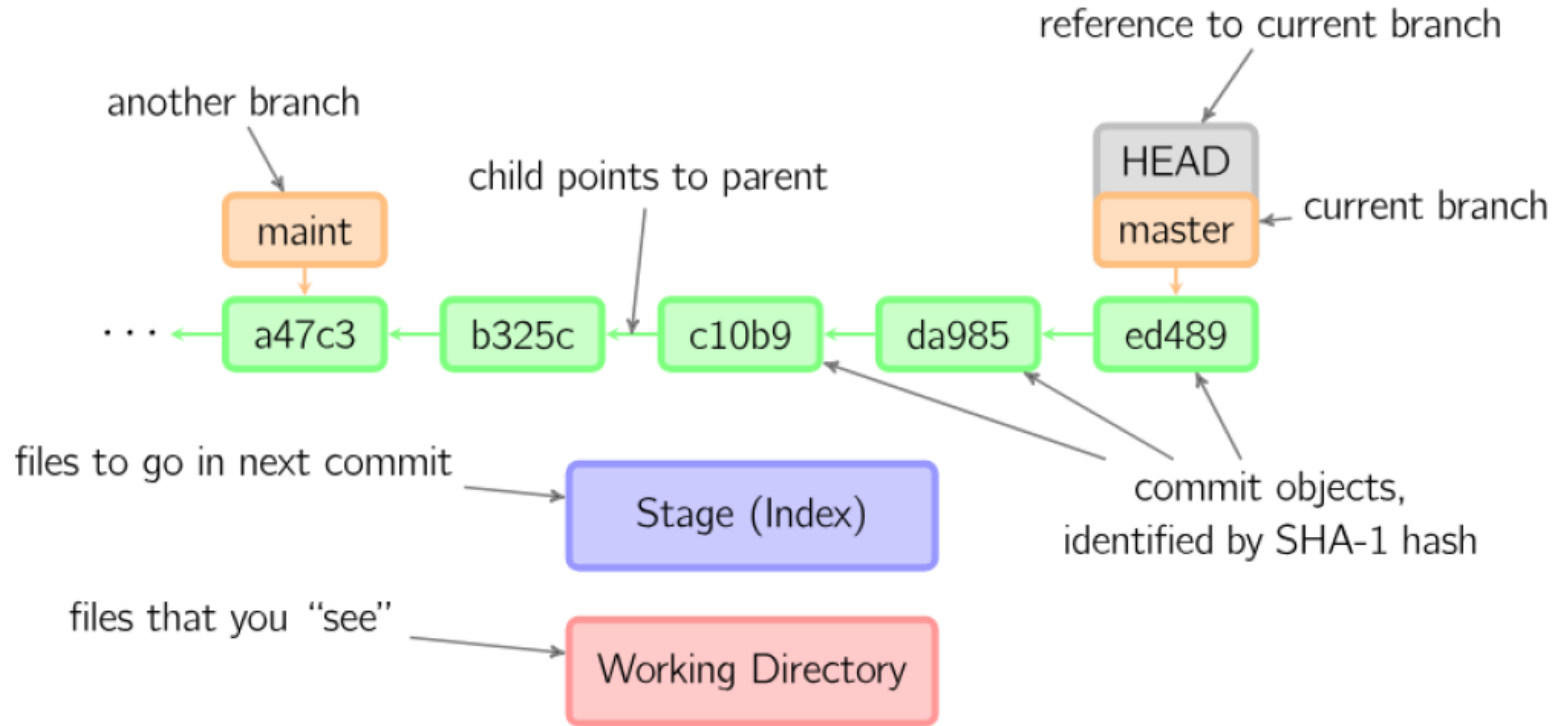
Working tree

The Three States

- Modified
- Staged
- Committed



Git terminology. Very briefly



Creating git repository

- Init

```
$ git init
```

```
$ git init <folder>
```

- Clone

```
$ git clone <remote repository url>
```

Git configuration and help

- config

```
$ git config _name "_value" -set property
$ git config --global -for current user
$git config --system -for all users
$git config --globaluser.name "IvanIvanov"
$gitconfig--list -get config for current repo
$ gitconfig--globalcore.editor"C:/Program
Files (x86)/Notepad++/notepad++.exe" -
multiInst-notabbar-nosession-noPlugin"
```

- help

```
$ githelp <verb>
$ git<verb> --help
$ mangit-<verb>
```

.gitignore

This is a file, which you could create in the root of your repository. All files, which are match patterns from gitignore, would be untracked by default. This could be binary files; files, which are generated by IDE, logs, ect. So all of this files exist in you project directory, but you will never want to commit them to repository.

The rules for the patterns you can put in the .gitignorefile are as follows:

- Blank lines or lines starting with # are ignored.
- Standard glob patterns work.
- You can start patterns with a forward slash (/) to avoid recursivity.
- You can end patterns with a forward slash (/) to specify a directory.
- You can negate a pattern by starting it with an exclamation point (!).

```
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a files above
!lib.a
# ignore all files in the build/ directory
build/
# ignore all .pdf files in the doc/ directory
doc/**/*.*pdf
```

Git aliases

```
$ git config --global alias.co checkout
```

```
$ git config --global alias.br branch
```

```
$ git config --global alias.ci commit
```

```
$ git config --global alias.st status
```

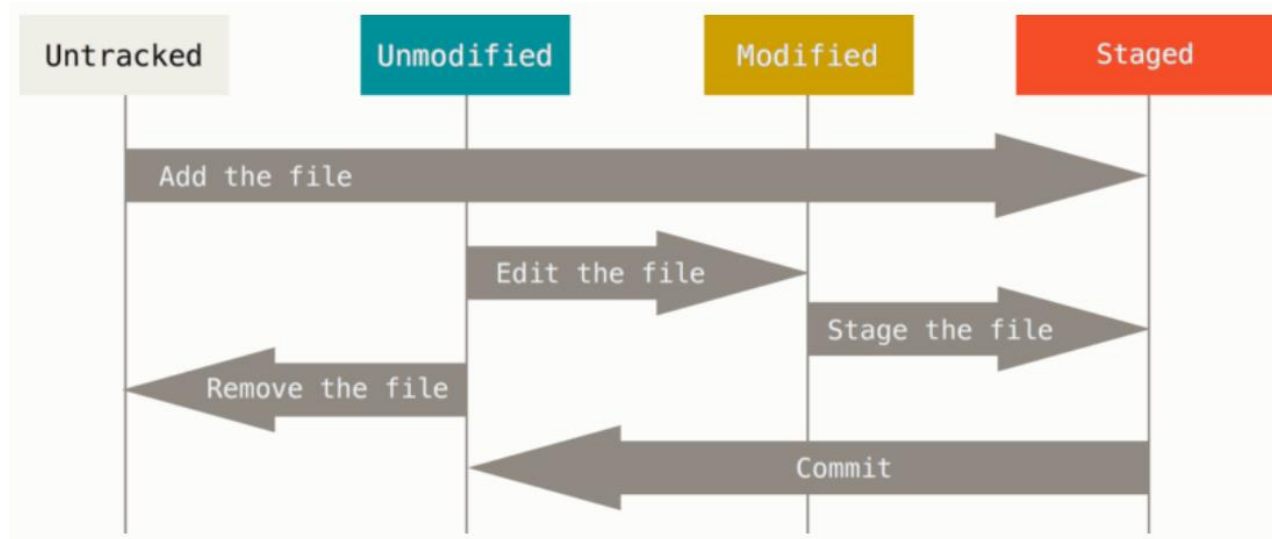
```
$ git config --global alias.ll= log --  
pretty=format:"%C(yellow)%h%Cred%d\\%Creset%s%Cblue\\[%cn]" --decorate --numstat
```

```
$ git config --global alias.ld= log --  
pretty=format:"%C(yellow)%h\\%C(green)%ad%Cred%d\\%Creset%s%Cblue\\[%cn]" --  
decorate --date=short --graph
```

```
$ git config --global alias.ls = log --  
pretty=format:"%C(green)%h\\%C(yellow)[%ad]%Cred%d\\%Creset%s%Cblue\\[%cn]" --  
decorate --date=relative
```

State lifecycle

- **New (Untracked)**: file was recently created
- **Modified**: you have changed the file but have not committed it to your local database
- **Staged**: you have marked a modified file in its current version to go into your next commit snapshot.
- **Committed (Unmodified)**: the data is safely stored in your local database



Commonly used commands

For local operations

- git status
- git add
- git commit
- git checkout
- git branch
- git reset
- git merge
- git rebase
- git stash

For viewing history and analysis

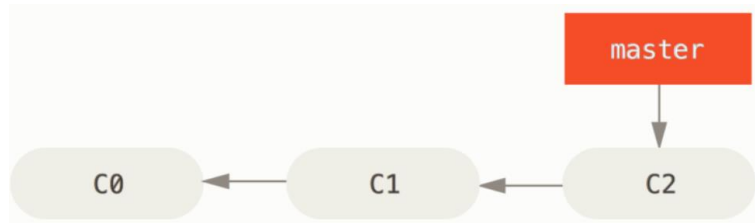
- git log
- git reflog
- git diff

For interactions with remotes

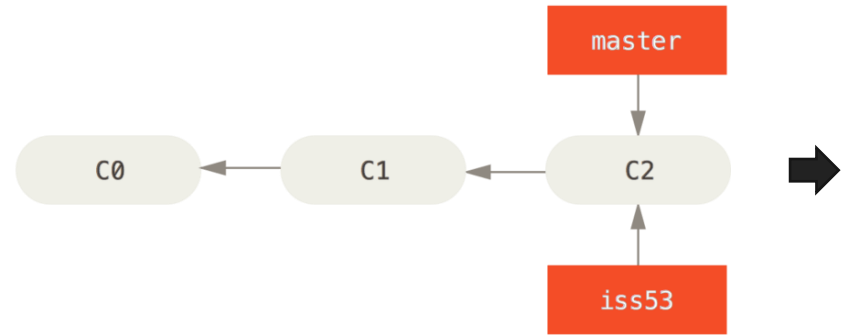
- git fetch
- git pull
- git push

Branching & merging workflow (1)

A simple commit history



Creating a new branch pointer



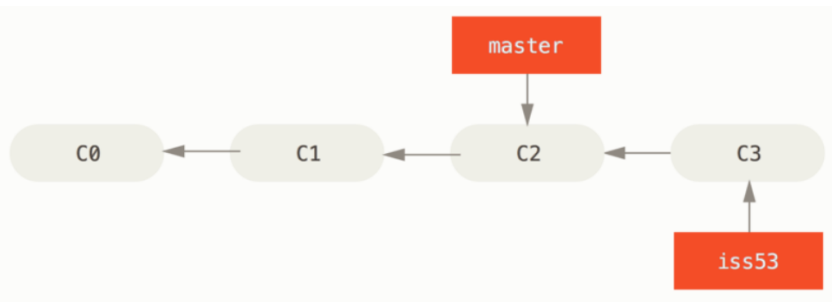
```
$ git checkout -b iss53
```

Switched to a new branch 'iss53'

```
[working on iss53]  
$ git commit -a -m 'issue53 add footer'
```

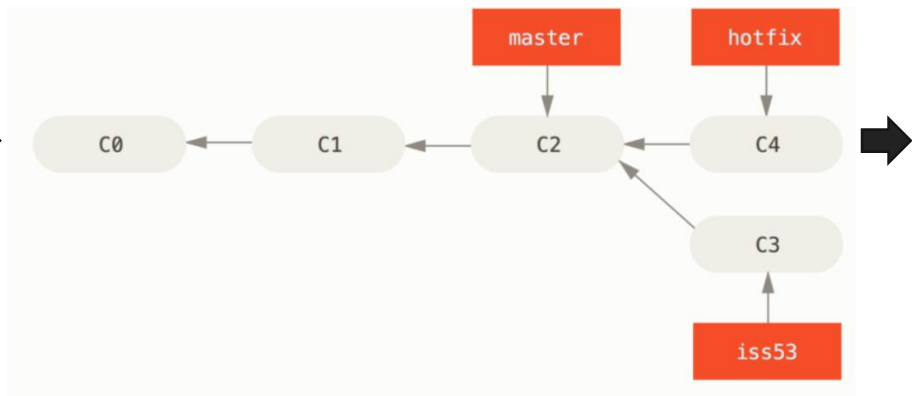
Branching & merging workflow (2)

The **iss53** branch has moved forward with your work



```
$ git checkout master  
Switched to branch 'master'
```

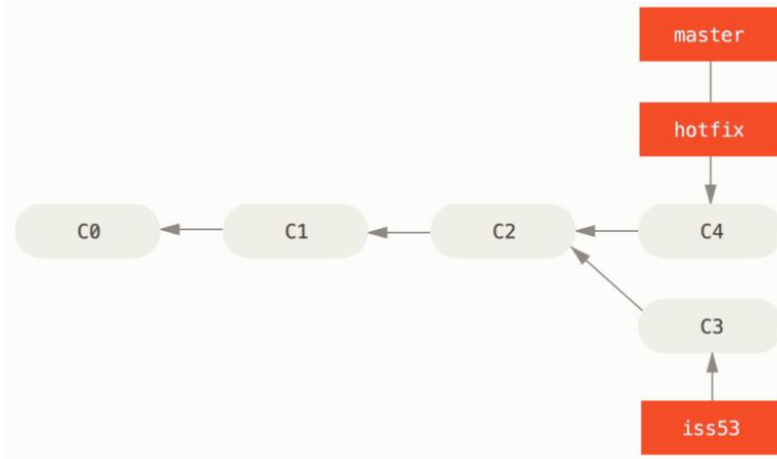
Hotfix branch based on **master**



```
$ git checkout -b hotfix  
Switched to a new branch 'hotfix'  
[do some fixes]  
$ git commit -a -m 'fix something'
```

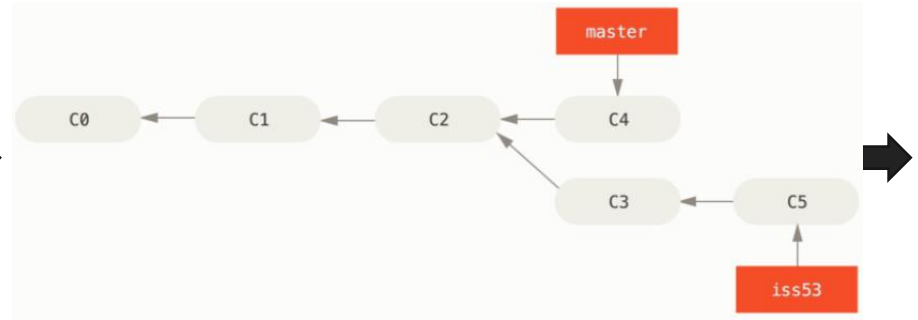

Branching & merging workflow (3)

master is fast-forwarded to **hotfix**



```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
```

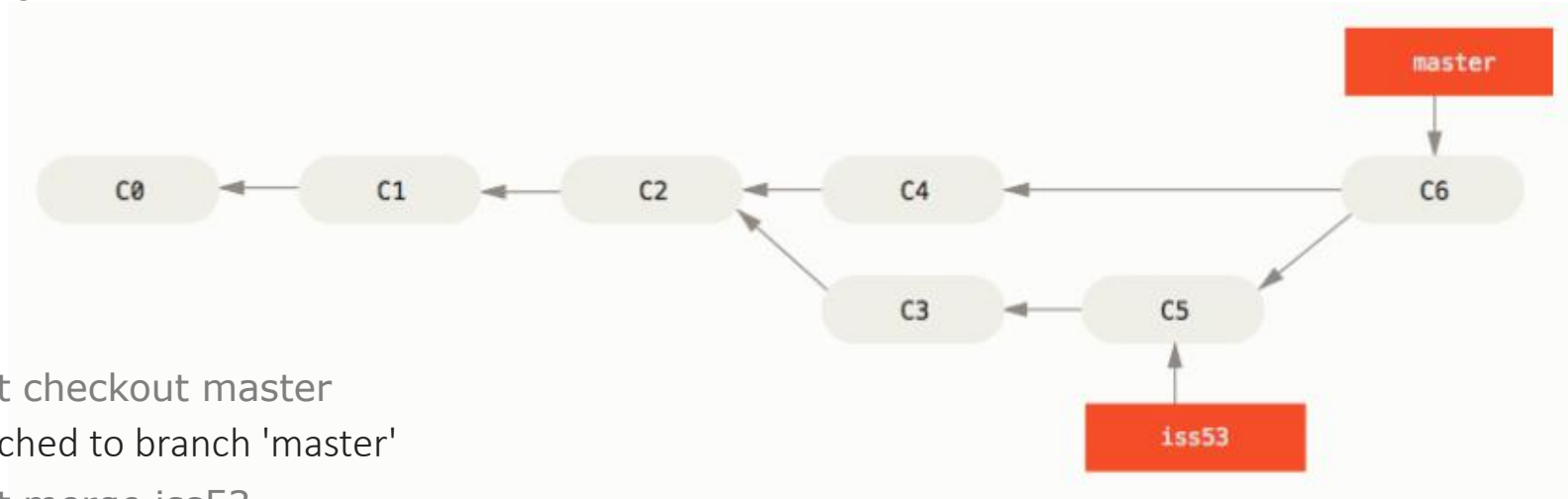
Work continues on **iss53**



```
$ git branch -d hotfix
Deleted branch hotfix (was 3a0874c).
$ git checkout iss53
Switched to branch 'iss53'
[Finish working on iss53]
$ git commit -a -m 'finish [issue 53]'
```

Branching & merging workflow (4)

A merge commit



```
$ git checkout master
```

Switched to branch 'master'

```
$ git merge iss53
```

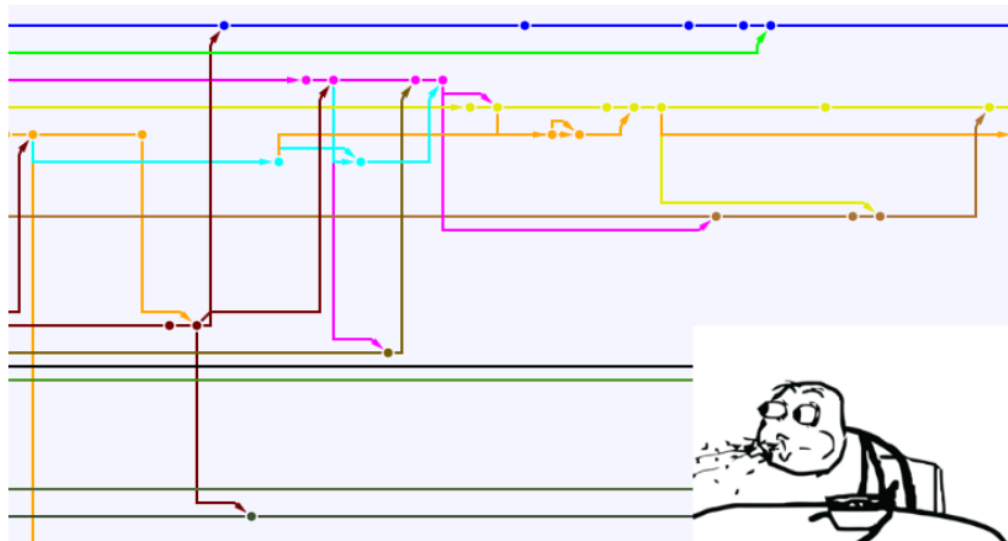
Merge made by the 'recursive' strategy.

index.html | 1 +

1 file changed, 1 insertion(+)

```
$ git branch -d iss53
```

Merge hell



Git hosting services

- Visual tools for team collaboration
- Code review tools
- Integrations with bug tracking systems



Markdown language

Markdown is a lightweight markup language with plain text formatting syntax. Its design allows it to be converted to many output formats, but the original tool by the same name only supports HTML. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.



The largest heading

The second largest heading

****This is bold text****

Some basic Git commands are:

...

git status

git add

git commit

...

This site was built using [GitHub Pages](https://pages.github.com/).

![Image of Yaktocat](https://octodex.github.com/images/yaktocat.png)

```
git status
git add
git commit
```

This site was built using [GitHub Pages](#).

Useful links

- <https://git-scm.com/>
- <https://learngitbranching.js.org/>
- <https://guides.github.com/activities/hello-world/>
- <https://git-scm.com/book/ru/v2>
- <https://github.com/github/gitignore>

- <https://daringfireball.net/projects/markdown/syntax>
- <https://guides.github.com/features/mastering-markdown/>
- <https://help.github.com/en/articles/basic-writing-and-formatting-syntax>
- <https://www.markdownguide.org/getting-started>

- <https://www.digitalocean.com/community/tutorials/how-to-use-git-hooks-to-automate-development-and-deployment-tasks>

Q&A



Thank you!