Tổng hợp kiến thức git dành cho QA/Tester

Lưu ý: Tài liệu này dành cho học viên khoá học Full-Stack automation QA với Playwright Typescript. Vui lòng không chia sẻ ra ngoài.
—

Cài đặt ban đầu

Cài đặt git

Tải file cài đặt tương ứng với hệ điều hành tại: https://git-scm.com/

Cấu hình git

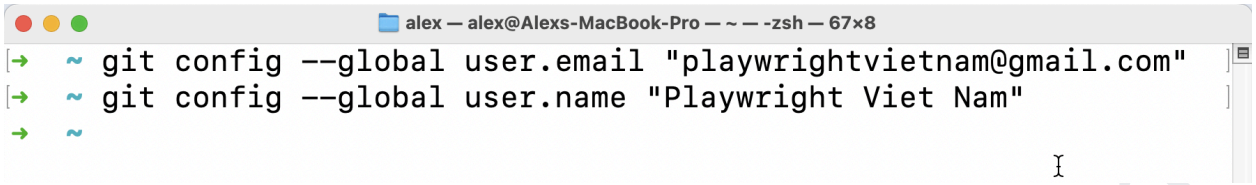
Để git biết được bạn là ai, bạn cần cấu hình tối thiểu hai thông tin:

- username
- email

Cấu hình global

Để cấu hình **mặc định** cho toàn bộ các repository được quản lý bởi git sau này, ta dùng lệnh:

```
git config --global user.name <tên_của_bạn>
git config --global user.email <email_của_bạn>
```



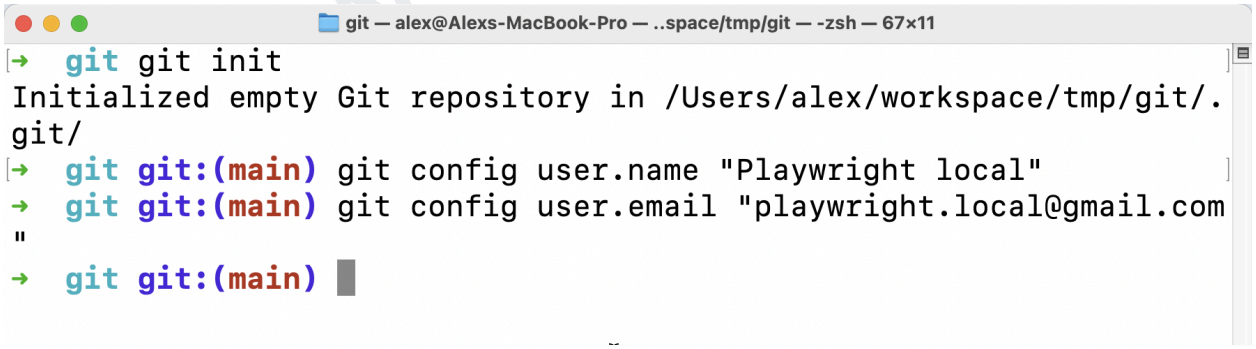
Để kiểm tra config của mình, sử dụng lệnh:

```
git config --list
```

Cấu hình local

Để cấu hình riêng cho một thư mục cụ thể, ta cần **mở terminal ở thư mục đó, khởi tạo git** và bỏ flag `--global` ở các câu lệnh đi.
Ví dụ:

```
git config user.name “Playwright local”
git config user.email “playwright.local@gmail.com”
```



Khi kiểm tra cài đặt (`git config --list`), ta sẽ thấy local config nằm ở phía dưới global config

Default branch

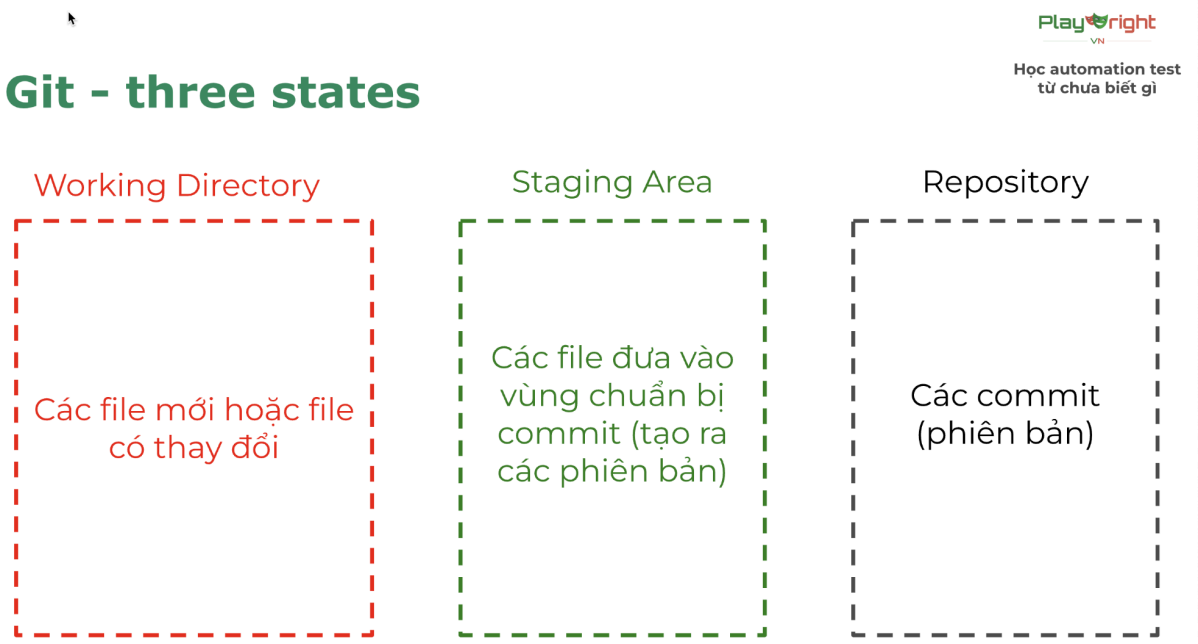
Trước đây, branch mặc định của các repo git là “master”. Tuy nhiên gần đây, một số provider đổi lại thành “main”.
Có một câu chuyện vui, là người ta đổi tên để tránh xa từ “master-slave” (ông chủ - nô lệ).
Để cài đặt branch mặc định, bạn dùng lệnh sau:

```
git config --global init.defaultBranch main
```

Concept quan trọng: ba vùng trạng thái

git quản lý thông qua ba vùng trạng thái:

- Working directory
- Staging
- Repository



Trong đó:

- **Working Directory**: chứa các file:
  - Mới tạo, **chưa được git quản lý**
  - Hoặc các file **đã được git quản lý** mà **có sự thay đổi**.
- **Staging Area**: vùng chứa các file để chuẩn bị được commit
- **Repository**: Vùng chứa các commit

Concept quan trọng: branching

Các câu lệnh thường dùng

Phần này được cấu trúc thành 2 phần:

- Tổng hợp lệnh thường dùng ở dạng bảng
- Giải thích chi tiết từng câu lệnh với ví dụ

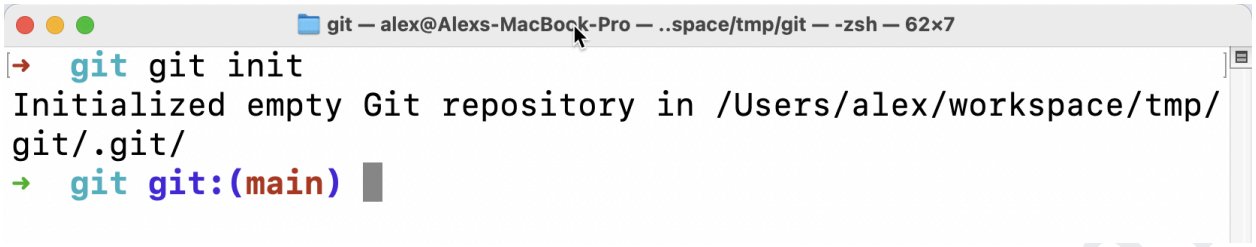
Tổng hợp lệnh thường dùng

Câu lệnh	Công dụng	Ví dụ
git status	Xem trạng thái các file	`git status`
git add <tên_file>	Thêm một file vào vùng staging	`git add file1.txt`

git add .	Thêm toàn bộ các file ở vùng working directory vào vùng repository	`git add .`
git commit -m”<message>”	Đưa tất cả các file ở vùng staging vào vùng repository và tạo commit với <message> tương ứng	`git commit -m”feat: add login test case”`
git branch <branch_name>	Tạo một nhánh mới với tên <branch_name>	`git branch feat/feature-A`
git branch	Liệt kê các nhánh hiện có	`git branch`
git checkout <branch_name>	Chuyển sang nhánh có tên <branch_name>	`git checkout feat/feature-A`
git checkout -b <branch_name>	Tạo một nhánh mới có tên <branch_name> và chuyển sang nhánh đó	`git checkout -b <branch_name>`
git clone <url>	Clone code từ remote về	`git clone git@github.com:playwrightvn/udemy-ts-course.git`
git remote add <name> <url>	Thêm mới một remote với tên <name> ở địa chỉ <url>	`git remote add origin git@github.com:playwrightvn/udemy-ts-course.git`
git push <name> <branch>	Đưa code ở remote <name> lên <branch>	`git push origin main`
git pull <name> <branch>	Lấy code từ remote <name> branch <branch> về nhánh hiện tại	`git pull origin main`

Khởi tạo repo

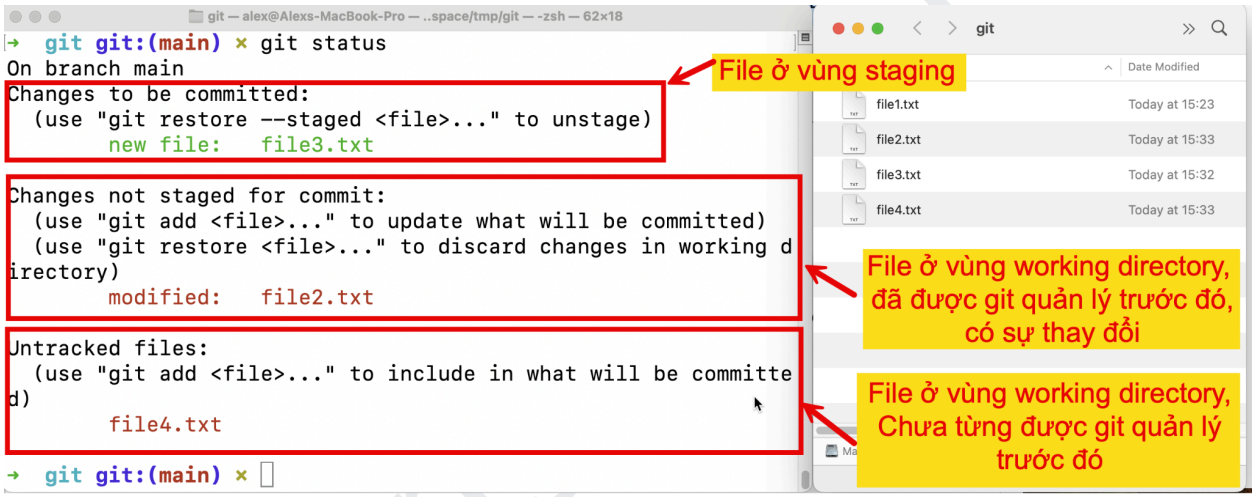
Để khởi tạo 1 repo, ta gõ lệnh git init tại thư mục cần khởi tạo.



Sau khi init, thư mục chính thức được git quản lý và hình thành ba vùng trạng thái.

Xem trạng thái file

Gõ lệnh `git status`



Đưa file vào vùng staging

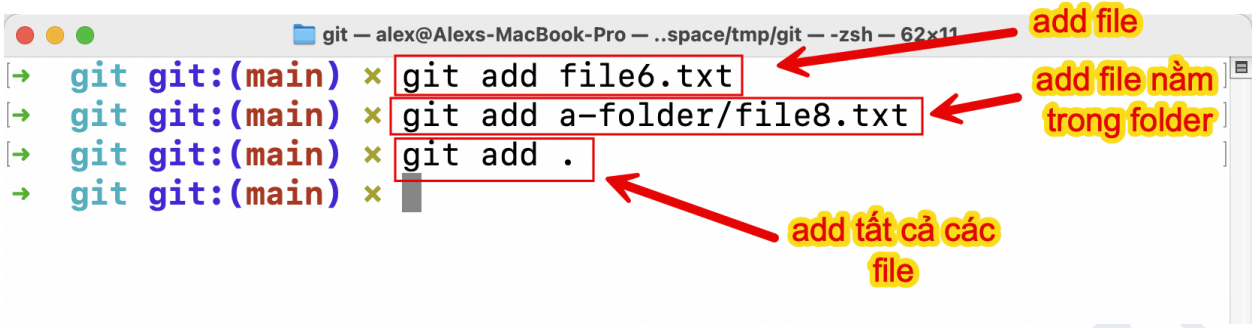
Để đưa file từ vùng working directory vào vùng staging, bạn dùng lệnh

git add <tên_file>

Trong đó:

- Nếu file không nằm trong thư mục hiện tại, bạn cần đưa đường dẫn đầy đủ của file vào
- Nếu bạn muốn thêm toàn bộ các file vào, thay tên file bằng dấu chấm (.)

  Tức là `git add .`

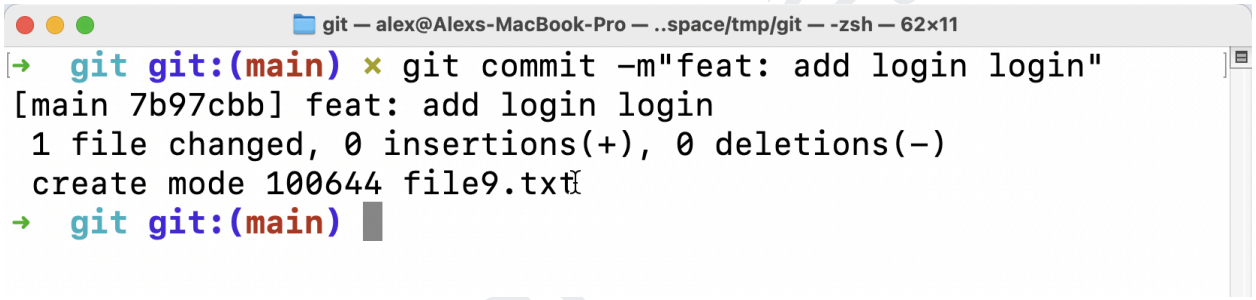


Đưa file vào vùng repository

Để đưa file từ vùng staging vào vùng repository, ta dùng lệnh

```
git commit -m"<message>"
```

Trong đó: message là nội dung commit.
Nội dung commit nên đi theo chuẩn conventional commit của thế giới. Xem mục "Commit convention" để biết thêm chi tiết.



Tạo nhánh

Để tạo nhánh, sử dụng lệnh: git checkout <tên_branch>

Xem các nhánh

- Branching
  - git branch <branch_name>
  - git checkout -b ư
- Stashing
- Undo things
  - git commit --amend
  - git push
  - git pull
  - git fetch
  - git rebase, squash

Đọc thêm: Commit convention

Convention là các bộ quy tắc. Commit convention là các bộ quy tắc để viết message cho commit.
Viết viết commit message theo convention có nhiều lợi ích:

- Commit theo 1 format nhất định, dễ nhìn.
- Dễ đoán được ý đồ của commit.
- Giúp hệ thống tự động trích xuất các nội dung của commit.

Đầy đủ và chi tiết nhất, bạn có thể tham khảo tại trang chính chủ:
https://www.conventionalcommits.org/en/v1.0.0/

Ở đây, mình đề xuất một bộ quy tắc giản lược cho khóa học như sau:
Commit message sẽ có định dạng: `<type>: <short_description>`
Trong đó:

- `<type>`: kiểu commit.
  - Các type chấp nhận:
    - **`feat`**: Viết tắt của feature. Dùng khi bạn thêm code cho một test case mới, hoặc cập nhật code cho tính năng mới của test case.
    - `fix`: Dùng khi bạn fix code cho test case cũ.
    - `chore`: Dùng khi bạn sửa các lỗi nhỏ, không đáng kể như: đổi tên file, sửa chính tả, thêm comment code,...
    - `conf`: Viết tắt của configuration. Dùng khi bạn thay đổi config của hệ thống.
- `<short_description>`: mô tả ngắn về commit làm gì
  - Phần này free text, bạn có thể viết theo ý mình, tuy nhiên nên ngắn gọn, dưới 50 kí tự.

Ví dụ:

Tình huống	Phân loại	Commit message
Bạn thêm code cho test case login mới	Tính năng mới	`feat: add login test case`
Bạn sửa code cho case login fail đã tồn tại: thêm phần verify error message	Tính năng mới	`feat: add logic verify error message`
Bạn thêm config cho screenshot	Update cấu hình	`conf: update screenshot config`
Bạn sửa chính tả cho 1 dòng code	Sửa lỗi nhỏ	`chore: correct syntax`
Bạn sửa code cho 1 test case bị	Sửa lỗi	`fix: update wrong XPath`

fail do sai XPath		

Lưu ý:

- Nên là một người viết commit message có tâm, không nên viết một cách tạm bợ. Sẽ hình thành thói quan xấu
  - Good commits:
  - Bad commits:
- Commit chỉ nên bao gồm các file và thay đổi liên quan tới commit đó.
  - Nếu có nhiều thay đổi khác nhau, hãy tạo nhiều commit khác nhau.

Stash

`Stash` = lưu trữ những công việc đang làm dở vào vùng nhớ tạm.
`Unstash` = đưa lại những công việc đang làm dở trở lại vào vùng working directory.

Stash thường dùng khi bạn đang làm dở công việc ở nhánh A, nhưng có một việc chen ngang ở nhánh B muốn xử lý trước.

Một số câu lệnh thường dùng với stash:

Câu lệnh	Giải thích	Ví dụ
git stash	Lưu trữ lại công việc đang làm, chỉ lưu các file đã được tracked	git stash
git stash list	Hiển thị danh sách các stash đang lưu trữ	git stash list
git stash save "<name>"	Lưu trữ lại công việc đang làm với tên <name>	git stash save "WIP javascript exercises"
git stash save --all	Stash cả untracked file	git stash save --all
git stash pop	Unstash nội dung stash mới nhất. Sau khi unstash xong thì xóa stash đó đi.	git stash pop
git stash pop <stash_id>	Unstash nội dung có stash id tương ứng. Sau khi unstash thì xóa stash đó đi.	git stash pop stash@{2}

git stash apply	Unstash nội dung stash mới nhất. Sau khi unstash vẫn giữ stash đó lại. Thường dùng khi muốn apply change ở multiple branch	git stash apply
git stash apply <stash_id>	Unstash nội dung stash mới theo stash_id. Sau khi unstash vẫn giữ stash đó lại. Thường dùng khi muốn apply change ở multiple branch	git stash apply stash@{2}