# GETTING STARTED WITH GIT

## Cài đặt GIT

Bạn có thể phải cài đặt Git trên hệ thống của mình bằng cách tải xuống và chạy trình cài đặt. Xem **Trang chủ Git** để có hướng dẫn cài đặt đặc biệt rõ ràng và dễ dàng.



### Kiểm tra phiên bản của GIT

Tại CMD, kiểm tra hệ thống Git đã cài đặt.

Trên tất cả các hệ điều hành:

| git --version |
| --- |

Trên các hệ điều hành giống UNIX/LINUX:

| which git |
| --- |

### Cấu hình tên người dùng và địa chỉ email

Sau khi cài đặt Git, hãy định cấu hình tên người dùng và địa chỉ email của bạn. Làm điều này là bắt buộc trước khi thực hiện một **commit**.

Khai báo tài khoản người dùng:

|  |
| --- |
| git config --global user.name "tên\_tài\_khoản\_github" |

Khai báo email đăng nhập:

|  |
| --- |
| git config --global user.email "email\_đăng\_nhập" |

Xem lại thông tin cấu hình:

|  |
| --- |
| git config --list |

Trong đó:

* Tham số “–global” để cấu hình cho người dùng hiện tại.
* Tham số “–system” để cấu hình cho tất cả người dùng của máy.

### Khởi tạo Git

Sau khi Git được cài đặt, hãy điều hướng đến thư mục bạn muốn đặt dưới sự kiểm soát phiên bản và tạo kho lưu trữ Git trống:

|  |
| --- |
| git init |

Việc này sẽ tạo ra một thư mục ẩn, .git, chứa hệ thống cần thiết để Git hoạt động.

### Kiểm tra trạng thái của các file trong Git

Tiếp theo, kiểm tra xem Git sẽ thêm những tập tin nào vào kho lưu trữ mới của bạn.

|  |
| --- |
| git status |

### Thêm file làm việc vào VCS

Bạn có thể cho Git biết nên đặt tệp nào vào kiểm soát phiên bản (tránh thêm tệp có thông tin bí mật như mật khẩu hoặc tệp chỉ làm lộn xộn kho lưu trữ):

|  |
| --- |
| git add <file/directory name *#1> <file/directory name #2> < ... >* |

Nếu tất cả các tệp trong danh sách phải được chia sẻ với tất cả những người có quyền truy cập vào kho lưu trữ, thì một lệnh duy nhất sẽ thêm mọi thứ trong thư mục hiện tại của bạn và các thư mục con của nó:

|  |
| --- |
| git add . |

Điều này sẽ "stage " tất cả các tệp sẽ được thêm vào kiểm soát phiên bản, chuẩn bị cho chúng được cam kết trong lần cam kết đầu tiên của bạn.

**Lưu ý:** Đối với các tệp mà bạn không bao giờ muốn kiểm soát phiên bản, hãy tạo và điền vào tệp có tên .gitignore trước khi chạy lệnh thêm .

### Thực hiện commit

Cam kết tất cả các tệp đã được thêm vào, cùng với thông báo cam kết:

|  |
| --- |
| git commit -m "Message Commit" |

"Việc này tạo ra một commit mới với thông điệp bạn cung cấp. Một commit giống như việc lưu trạng thái hoặc ảnh chụp của toàn bộ dự án của bạn. Sau đó, bạn có thể đẩy (push) nó lên một kho lưu trữ từ xa (remote repository) và sau này bạn có thể quay lại đó nếu cần thiết.

Nếu bạn không sử dụng tham số -m, trình soạn thảo mặc định của bạn sẽ mở và bạn có thể chỉnh sửa và lưu thông điệp commit ở đó.

Để thêm một kho lưu trữ từ xa mới, bạn sử dụng lệnh git remote add trong cửa sổ dòng lệnh, trong thư mục chứa kho lưu trữ của bạn."

Lệnh git remote add cần hai tham số:

Tên máy chủ từ xa, ví dụ: origin

Địa chỉ URL máy chủ từ xa, ví dụ: https://<địa chỉ-dịch-vụ-git-của-bạn>/người-dùng/kho-lưu-trữ.git Ví dụ: git remote add origin https://<địa chỉ-dịch-vụ-git-của-bạn>/chủ-sở-hữu/thu-viện.git

**Lưu ý**: Trước khi thêm remote, bạn cần tạo kho lưu trữ (repository) cần thiết trên dịch vụ Git của bạn. Sau khi thêm remote, bạn sẽ có khả năng push/pull các commit.

## Sao chép một kho lưu trữ (Clone a repository)

Lệnh git clone được sử dụng để sao chép một kho lưu trữ Git đã tồn tại từ máy chủ về máy tính cục bộ.

Các bước để sao chép một dự án trên GitHub:

1. Sao chép đường dẫn dự án trên GitHub.

***Example***: https:**//**github.com**/**username**/**projectname.git

1. Di chuyển tới thư mục muốn chứa dự án.

***Example***: "C:\Program Files\Git\Repo"

1. Chạy lệnh: git clone < https:**//**github.com**/**username**/**projectname.git>

(Lúc này, đường dẫn thư mục sẽ như sau: "C:\Program Files\Git\Repo\projectname "

Điều này sẽ tạo ra một thư mục có tên projectname trên máy tính cục bộ, chứa tất cả các tệp trong kho lưu trữ Git từ xa. Bao gồm các tệp nguồn cho dự án, cũng như một thư mục con .git chứa toàn bộ lịch sử và cấu hình cho dự án.

Để chỉ định tên thư mục khác, ví dụ: MyFolder: Chạy lệnh:

|  |
| --- |
| git clone https://github.com/username/projectname.git MyFolder |

Hoặc để sao chép vào thư mục hiện tại: Chạy lệnh:

|  |
| --- |
| git clone https://github.com/username/projectname.git |

**Lưu ý:**

1. Khi sao chép vào một thư mục được chỉ định, thư mục đó phải trống hoặc không tồn tại.
2. Bạn cũng có thể sử dụng phiên bản **ssh** của lệnh: Chạy lệnh:

|  |
| --- |
| git clone git@github.com:username/projectname.git |

Phiên bản sử dụng https và phiên bản sử dụng ssh là tương đương. Tuy nhiên, một số dịch vụ lưu trữ như GitHub khuyên bạn nên sử dụng https thay vì ssh.

## Chia sẻ mã nguồn (Sharing code)

Để chia sẻ mã nguồn của bạn, bạn cần tạo một kho lưu trữ trên một máy chủ từ xa, sau đó bạn sẽ sao chép kho lưu trữ cục bộ của bạn lên đó.

Để giảm tối đa việc sử dụng không gian trên máy chủ từ xa, bạn cần tạo một kho lưu trữ trần trống (bare repository): một kho lưu trữ chỉ chứa các đối tượng .git và không tạo bản sao làm việc trong hệ thống tệp. Như một điểm cộng, bạn có thể đặt máy chủ từ xa này làm máy chủ gốc (upstream server) để dễ dàng chia sẻ các bản cập nhật với các lập trình viên khác.

Trên máy chủ từ xa:

| **git init** --bare **/**path**/**to**/**repo.git |
| --- |

Trên máy tính cục bộ:

| **git remote** add origin ssh:**//**username**@**server:**/**path**/**to**/**repo.git |
| --- |

(Lưu ý rằng ssh: chỉ là một cách có thể để truy cập vào kho lưu trữ từ xa.)

Bây giờ bạn hãy sao chép kho lưu trữ cục bộ của bạn lên kho lưu trữ từ xa:

| **git push** --set-upstream origin master |
| --- |

Thêm --set-upstream (hoặc -u) đã tạo một tham chiếu nguồn (tracking) được sử dụng bởi các lệnh Git không có đối số, ví dụ, git pull.

### Thiết lập tên và email của bạn

Bạn cần thiết lập Tên trước khi tạo bất kỳ commit nào. Điều này sẽ cho phép các commit có tên tác giả và email chính xác được liên kết với họ.

Nó không có gì liên quan đến xác thực khi đẩy lên kho lưu trữ từ xa (ví dụ, khi đẩy lên kho lưu trữ từ xa sử dụng tài khoản GitHub, BitBucket hoặc GitLab của bạn).

Để khai báo danh tính đó cho tất cả các kho lưu trữ, hãy sử dụng: git config --global

Điều này sẽ lưu trữ cài đặt trong tệp .gitconfig của người dùng của bạn, ví dụ: $HOME/.gitconfig trên hệ thống Unix hoặc %USERPROFILE%.gitconfig trên Windows.

|  |
| --- |
| **git config** --global user.name "Your Name" **git config** --global user.email mail**@**example.com |

Để khai báo một danh tính cho một kho lưu trữ cụ thể, hãy sử dụng git config bên trong kho lưu trữ đó. Điều này sẽ lưu cài đặt trong kho lưu trữ cụ thể, trong tệp $GIT\_DIR/config. Ví dụ: /path/to/your/repo/.git/config.

|  |
| --- |
| **cd /**path**/**to**/**my**/**repo **git config** user.name "Your Login At Work" **git config** user.email mail\_at\_work**@**example.com |

Các cài đặt lưu trong tệp cấu hình của kho lưu trữ sẽ được ưu tiên hơn so với cấu hình toàn cầu khi bạn sử dụng kho lưu trữ đó.

Mẹo: nếu bạn có các danh tính khác nhau (một cho dự án mã nguồn mở, một ở nơi làm việc, một cho kho lưu trữ riêng tư, ...), và bạn không muốn quên thiết lập đúng danh tính cho mỗi kho lưu trữ khác nhau mà bạn đang làm việc:

**Xóa một danh tính toàn cầu**

|  |
| --- |
| **git config** --global --remove-section user.name **git config** --global --remove-section user.email |

arduinoCopy code

git config --global --remove-section user.name git config --global --remove-section user.email

Phiên bản ≥ 2.8

Để buộc git chỉ tìm kiếm danh tính của bạn trong cài đặt của một kho lưu trữ, không phải trong cấu hình toàn cầu:

|  |
| --- |
| **git config** --global user.useConfigOnly **true** |

Như vậy, nếu bạn quên thiết lập **user.name** và **user.email** cho một kho lưu trữ cụ thể và cố gắng tạo một commit, bạn sẽ thấy:

| no name was given and auto-detection is disabled no email was given and auto-detection is disabled |
| --- |

### Thiết lập kết nối với máy chủ gốc

Nếu bạn đã sao chép một bản sao nhánh (ví dụ: một dự án mã nguồn mở trên Github), bạn có thể không có quyền đẩy (push) lên máy chủ gốc, vì vậy bạn cần cả bản sao của bạn và có khả năng truy xuất (fetch) dự án gốc.

Đầu tiên, kiểm tra tên kết nối (remote):

|  |
| --- |
| $ **git remote** -v  Origin https:**//**github.com**/**myusername**/**repo.git **(**fetch**)** Origin https:**//**github.com**/**myusername**/**repo.git **(**push**)**  upstream *# this line may or may not be here* |

Nếu đã có kết nối "upstream" (điều này xảy ra trong một số phiên bản Git), bạn cần thiết lập URL (hiện tại nó đang trống):

|  |
| --- |
| $ **git remote** set-url upstream https:**//**github.com**/**projectusername**/**repo.git |

Nếu "upstream" chưa tồn tại hoặc nếu bạn muốn thêm một bản sao nhánh của bạn hoặc người bạn/người đồng nghiệp (họ hiện không tồn tại):

|  |
| --- |
| $ **git remote** add upstream https:**//**github.com**/**projectusername**/**repo.git $ **git remote** add dave https:**//**github.com**/**dave**/**repo.git |

tps://github.com/dave/repo.git

### Tìm hiểu về một lệnh

Để có thêm thông tin về bất kỳ lệnh Git nào - tức là chi tiết về công việc của lệnh, các tùy chọn có sẵn và tài liệu khác - sử dụng tùy chọn --help hoặc lệnh help. Ví dụ, để lấy tất cả thông tin có sẵn về lệnh git diff, sử dụng:

bashCopy code

git diff --help git help diff

Tương tự, để lấy tất cả thông tin có sẵn về lệnh status, sử dụng:

luaCopy code

git status --help git help status

Nếu bạn chỉ muốn được hướng dẫn nhanh về ý nghĩa của các tùy chọn thường sử dụng trên dòng lệnh, sử dụng -h:

Copy code

git checkout -h

Phần 1.7: Thiết lập SSH cho Git

Nếu bạn đang sử dụng Windows, hãy mở Git Bash. Nếu bạn đang sử dụng Mac hoặc Linux, hãy mở Terminal của bạn. Trước khi tạo một khóa SSH, bạn có thể kiểm tra xem bạn có bất kỳ khóa SSH hiện có nào không. Liệt kê nội dung của thư mục ~/.ssh của bạn:

cssCopy code

$ ls -al ~/.ssh # Liệt kê tất cả các tệp trong thư mục ~/.ssh của bạn

### Clone a repository

Lệnh git clone được sử dụng để sao chép kho lưu trữ Git hiện có từ máy chủ sang máy cục bộ.

Ví dụ: để sao chép dự án GitHub:

| cd <đường dẫn nơi bạn muốn bản sao tạo thư mục>  git clone https://github.com/username/projectname.git |
| --- |

Để sao chép dự án BitBucket:

| cd <đường dẫn nơi bạn muốn bản sao tạo thư mục>  git clone https://yourusername@bitbucket.org/username/projectname.git |
| --- |

Điều này tạo ra một thư mục có tên projectname trên máy cục bộ, chứa tất cả các tệp trong kho Git từ xa. Điều này bao gồm các tệp nguồn cho dự án cũng như thư mục con .git chứa toàn bộ lịch sử và cấu hình cho dự án.

Để chỉ định tên khác của thư mục, ví dụ MyFolder:

| git clone https://github.com/username/projectname.git MyFolder |
| --- |

Hoặc để sao chép trong thư mục hiện tại:

| git clone https://github.com/username/projectname.git |
| --- |

Ghi chú:  
Trên máy cục bộ:  
1. Khi sao chép vào một thư mục được chỉ định, thư mục đó phải trống hoặc không tồn tại.

| git clone git@github.com:tên người dùng/tên dự án.git |
| --- |

Phiên bản https và phiên bản ssh tương đương nhau. Tuy nhiên, một số dịch vụ lưu trữ như GitHub khuyên bạn nên rằng bạn sử dụng https thay vì ssh.

### Sharing code

Để chia sẻ mã của mình, bạn tạo một kho lưu trữ trên máy chủ từ xa mà bạn sẽ sao chép kho lưu trữ cục bộ của mình vào đó.

Để giảm thiểu việc sử dụng dung lượng trên máy chủ từ xa, bạn tạo một kho lưu trữ trống: một kho lưu trữ chỉ có các đối tượng .git và không tạo bản sao hoạt động trong hệ thống tệp. Như một phần thưởng, bạn đặt điều khiển từ xa này làm máy chủ ngược tuyến để dễ dàng chia sẻ thông tin cập nhật với các lập trình viên khác.

Trên máy chủ từ xa:

| git init --bare /path/to/repo.git |
| --- |

Trên máy cục bộ:

| **git remote** add origin ssh:**//**username**@**server:**/**path**/**to**/**repo.git |
| --- |

(Lưu ý rằng ssh: chỉ là một cách khả thi để truy cập kho lưu trữ từ xa.)

Bây giờ sao chép kho lưu trữ cục bộ của bạn vào điều khiển từ xa:

| git push --set-upstream Origin master |
| --- |

Việc thêm --set-upstream (hoặc -u) đã tạo một tham chiếu ngược dòng (theo dõi) được sử dụng bởi các lệnh Git không có đối số, ví dụ như git pull.

### Setting your user name and email

|  |
| --- |

### Setting up the upstream remote

### Learning about a command

### Set up SSH for Git

### Git Installation

# BROWSING THE HISTORY

## Lệnh Git Log

Các tham số của lệnh git log và ý nghĩa của chúng:

* -q, --quiet: Im lặng, không hiển thị kết quả diff
* --source: Hiển thị nguồn gốc của commit
* --use-mailmap: Sử dụng file mailmap để thay đổi thông tin người commit
* --decorate[=...]: Tùy chọn trang trí
* -L <n,m:file>: Hiển thị log cho dòng cụ thể trong file, bắt đầu từ dòng n đến dòng m. Cũng hiển thị diff.
* --show-signature: Hiển thị chữ ký của các commit đã được ký
* -i, --regexp-ignore-case: Khớp biểu thức chính quy mà không phân biệt chữ hoa chữ thường

### Mục 2.1: "Log thông thường" của Git

Lệnh git log sẽ hiển thị tất cả các commit với Tên tác giả và Thời gian. Mỗi commit sẽ được hiển thị trên nhiều dòng. Sử dụng phím q để thoát khỏi lệnh git log.

Theo mặc định, lệnh git log sẽ hiển thị các lệnh commit mới nhất lên trước tiên.

VD:

|  |
| --- |
| C:\Users\LENOVO\OneDrive\Desktop\Home\Git\Git>git log  commit 7f2ca9d86bd62b70b8b0397ef39f7cc7c305ac33 (HEAD -> main, origin/main, origin/HEAD)  Author: Hao Nguyen <haonp1999@gmmail.com>  Date: Fri Apr 26 20:06:51 2024 +0700  ok  commit f54a64859eeec5b0e924a7a8a84d93a82baf45e5  Author: BV\_hao.nguyen-phi <hao.nguyen-phi@banvien.com.vn>  Date: Tue Jan 30 14:08:58 2024 +0700  ok  commit 1278d3d4916cc300a18d4af6310f9fabf8148458  Author: BV\_hao.nguyen-phi <hao.nguyen-phi@banvien.com.vn>  Date: Fri Dec 1 11:56:40 2023 +0700  ok  commit 0f81e4e9b3b6ccf2c210f2c70c51fce71379c7ec  Author: BV\_hao.nguyen-phi <hao.nguyen-phi@banvien.com.vn>  Date: Thu Nov 9 14:44:06 2023 +0700  ok |

Nếu muốn giới hạn số lượng commit, có thể chuyển tham số.

**VD**: git log -2 để hiển thị 2 commit gần nhất.

### Log đẹp hơn Để xem log dưới dạng đồ thị:

|  |
| --- |
| git log --decorate --oneline –graph |

Ví dụ kết quả:

* e0c1cea (HEAD -> maint, tag: v2.9.3, origin/maint) Git 2.9.3
* 9b601ea Hợp nhấnh nhánh 'jk/difftool-in-subdir' vào maint |\  
  | \* 32b8c58 sử dụng Git::\_ thay vì truyền trạng thái | \* 98f917e tránh $GIT\_DIR và $GIT\_WORK\_TREE |\_ 9ec26e7 sửa đối số trong thư mục con
* | f4fd627 Hợp nhánh 'jk/reset-ident-time-per-commit' vào maint

Có thể đặt alias để rút gọn lệnh:

git config --global alias.lol "log --decorate --oneline --graph"

Sử dụng:

* git lol: xem lịch sử nhánh hiện tại
* git lol HEAD develop origin/master: xem lịch sử kết hợp
* git lol --all: xem toàn bộ lịch sử

### 2.3 Tô màu log

git log --graph --pretty=format:'%C(đỏ)%h%Creset -%C(vàng)%d%Creset %s %C(xanh lá)%cr%C(vàng)<%an>%Creset'

Cú pháp:

* %C(màu): tô màu cho phần sau
* %h: rút gọn mã commit
* %d: tên nhánh
* %s: chủ đề (subject)
* %cr: ngày commit
* %an: tên tác giả

### 2.4 Log một dòng

|  |
| --- |
|  |

| **git log** --oneline |
| --- |

Log một dòng sẽ hiển thị tất cả commit với chỉ một phần hash và message trên một dòng:

git log --oneline

Ví dụ:

87ef97f Hợp nhập pull request #7724 eb8b729 Sửa lỗi chính tả 'its' trong Where Art Thou e50ff0d Hợp nhập pull request #7718 ...

Có thể giới hạn số lượng commit hiển thị, ví dụ:

git log -2 --oneline

2.5 Tìm kiếm log

* Tìm kiếm thêm/bớt chuỗi cụ thể:

git log -S"#define SAMPLES"

* Tìm thay đổi trong dòng chứa chuỗi:

git log -G"#define SAMPLES"

2.6 Liệt kê đóng góp theo tác giả

git shortlog tóm tắt git log và nhóm theo tác giả

Nếu không có tham số, sẽ liệt kê tất cả các commit nhóm theo tác giả theo thứ tự thời gian:

$ git shortlog Tác giả 1 (<số\_commit>): Commit message 1 Commit message 2

### Prettier log

|  |
| --- |
|  |

|  |
| --- |
|  |

### Colorize Logs

|  |
| --- |
|  |

|  |
| --- |
|  |

### Oneline log

|  |
| --- |
|  |

|  |
| --- |
|  |

### Log search

|  |
| --- |
|  |

|  |
| --- |
|  |

### List all contributions grouped by author name

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Searching commit string in git log

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Log for a range of lines within a file

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Filter logs

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Log with changes inline

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Log showing commited files

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Show the contents of a single commit

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Git Log Between Two Branches

|  |
| --- |
|  |

|  |
| --- |
|  |

### One line showing commiter name and time since commit

# WORKING WITH REMOTES

### Deleting a Remote Branch

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Changing Git Remote URL

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### List Existing Remotes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Removing Local Copies of Deleted Remote Branches

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Updating from Upstream Repository

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### ls-remote

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Adding a New Remote Repository

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Set Upstream on a New Branch

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Getting Started

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Renaming a Remote

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Show information about a Specific Remote

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Set the URL for a Specific Remote

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Get the URL for a Specific Remote

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Changing a Remote Repository

# STAGING

### Staging All Changes to Files

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Unstage a file that contains changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Add changes by hunk

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Interactive add

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Show Staged Changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Staging A Single File

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

### Stage deleted files

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# IGNORING FILES AND FOLDERS

## Ignoring files and directories with a .gitignore file

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Checking if a file is ignored

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Exceptions in a .gitignore file

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## A global .gitignore file

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignore files that have already been committed to a Git repository

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignore files locally without committing ignore rules

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignoring subsequent changes to a file (without removing it)

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignoring a file in any directory

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Prefilled .gitignore Templates

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignoring files in subfolders (Multiple gitignore files)

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Create an Empty Folder

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Finding files ignored by .gitignore

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignoring only part of a file [stub]

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Ignoring changes in tracked files. [stub]

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Clear already committed files, but included in .gitignore

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# GIT DIFF

## Show dierences in working branch

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Show changes between two commits

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Show dierences for staged files

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Comparing branches

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Show both staged and unstaged changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Show dierences for a specific file or directory

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Viewing a word-di for long lines

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Show dierences between current version and last version

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Produce a patch-compatible di

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## dierence between two commit or branch

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Using meld to see all modifications in the working directory

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Di UTF-16 encoded text and binary plist files

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# UNDOING

## Return to a previous commit

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Undoing changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Using reflog

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Undoing merges

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Revert some existing commits

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Undo / Redo a series of commits

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# MERGING

## Automatic Merging

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Finding all branches with no merged changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Aborting a merge

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Merge with a commit

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Keep changes from only one side of a merge

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Merge one branch into another

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# SUBMODULES

## Cloning a Git repository having submodules

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Updating a Submodule

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Adding a submodule

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Setting a submodule to follow a branch

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Moving a submodule

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Removing a submodule

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# COMMITTING

## Stage and commit changes

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Good commit messages

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Amending a commit

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Committing without opening an editor

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Committing changes directly

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Selecting which lines should be staged for committing

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Creating an empty commit

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Committing on behalf of someone else

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## GPG signing commits

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

## Commiting changes in specific files

## Committing at a specific date

## Amending the time of a commit

## Amending the author of a commit

# ALIASES

## Simple aliases

## List / search existing aliases

## Advanced Aliases

## Temporarily ignore tracked files

## Show pretty log with branch graph

## See which files are being ignored by your .gitignore configuration

## Updating code while keeping a linear history

## Unstage staged files

# REBASING

## Local Branch Rebasing

## Rebase: ours and theirs, local and remote

## Interactive Rebase

## Rebase down to the initial commit

## Configuring autostash

## Testing all commits during rebase

## Rebasing before a code review

## Aborting an Interactive Rebase

## Setup git-pull for automatically perform a rebase instead of a merge

## Pushing after a rebase

# CONFIGURATION

## Setting which editor to use

## Auto correct typos

## List and edit the current configuration

## Username and email address

## Multiple usernames and email address

## Multiple git configurations

## Configuring line endings

## configuration for one command only

## Setup a proxy

# BRANCHING

## Creating and checking out new branches

## Listing branches

## Delete a remote branch

## Quick switch to the previous branch

## Check out a new branch tracking a remote branch

## Delete a branch locally

## Create an orphan branch (i.e. branch with no parent commit)

## Rename a branch

## Searching in branches

## Push branch to remote

## Move current branch HEAD to an arbitrary commit

# REV-LIST

List Commits in master but not in origin/master

# SQUASHING

Squash Recent Commits Without Rebasing

Squashing Commit During Merge

Squashing Commits During a Rebase

Autosquashing and fixups

Autosquash: Committing code you want to squash during a rebase

# CHERRY PICKING

## Copying a commit from one branch to another

## Copying a range of commits from one branch to another

## Checking if a cherry-pick is required

## Find commits yet to be applied to upstream

# RECOVERING

## Recovering from a reset

## Recover from git stash

## Recovering from a lost commit

## Restore a deleted file after a commit

## Restore file to a previous version

## Recover a deleted branch

# GIT CLEAN

## Clean Interactively

## Forcefully remove untracked files

## Clean Ignored Files

## Clean All Untracked Directories

# USING A .GITATTRIBUTES FILE

## Automatic Line Ending Normalization

## Identify Binary Files

## Prefilled .gitattribute Templates

## Disable Line Ending Normalization

# MAILMAP FILE: ASSOCIATING CONTRIBUTOR AND EMAIL ALIASES

## Merge contributers by aliases to show commit count in shortlog

# ANALYZING TYPES OF WORKFLOWS

## Centralized Workflow

## Gitflow Workflow

## Feature Branch Workflow

## GitHub Flow

## Forking Workflow

# PULLING

## Pulling changes to a local repository

## Updating with local changes

## Pull, overwrite local

## Pull code from remote

## Keeping linear history when pulling

## Pull, "permission denied"

# HOOKS

## Pre-push

## Verify Maven build (or other build system) before committing

## Automatically forward certain pushes to other repositories

## Commit-msg

## Local hooks

## Post-checkout

## Post-commit

## Post-receive

## Pre-commit

## Prepare-commit-msg

## Pre-rebase

## Pre-receive

## Update

# CLONING REPOSITORIES

## Shallow Clone

## Regular Clone

## Clone a specific branch

## Clone recursively

## Clone using a proxy

# STASHING

## What is Stashing?

## Create stash

## Apply and remove stash

## Apply stash without removing it

## Show stash

## Partial stash

## List saved stashes

## Move your work in progress to another branch

## Remove stash

## Apply part of a stash with checkout

## Recovering earlier changes from stash

## Interactive Stashing

## Recover a dropped stash

# SUBTREES

## Create, Pull, and Backport Subtree

# RENAMING

## Rename Folders

## Rename a local and the remote branch

## Renaming a local branch

# PUSHING

## Push a specific object to a remote branch

## Push

## Force Pushing

## Push tags

## Changing the default push behavior

# INTERNALS

## Repo

## Objects

## HEAD ref

## Refs

## Commit Object

## Tree Object

## Blob Object

## Creating new Commits

## Moving HEAD

## Moving refs around

## Creating new Refs

# GIT-TFS

## git-tfs clone

## git-tfs clone from bare git repository

## git-tfs install via Chocolatey

## git-tfs Check In

## git-tfs push

# EMPTY DIRECTORIES IN GIT

## Git doesn't track directories

# GIT-SVN

## Cloning the SVN repository

## Pushing local changes to SVN

## Working locally

## Getting the latest changes from SVN

## Handling empty folders

# ARCHIVE

## Create an archive of git repository

## Create an archive of git repository with directory prefix

## Create archive of git repository based on specific branch, revision, tag or directory

# REWRITING HISTORY WITH FILTER-BRANCH

## Changing the author of commits

## Setting git committer equal to commit author

# MIGRATING TO GIT

## SubGit

## Migrate from SVN to Git using Atlassian conversion utility

## Migrating Mercurial to Git

## Migrate from Team Foundation Version Control (TFVC) to Git

## Migrate from SVN to Git using svn2git

# SHOW

## Overview

# RESOLVING MERGE CONFLICTS

## Manual Resolution

# BUNDLES

## Creating a git bundle on the local machine and using it on another

# DISPLAY COMMIT HISTORY GRAPHICALLY WITH GITK

## Display commit history for one file

## Display all commits between two commits

## Display commits since version tag

# BISECTING/FINDING FAULTY COMMITS

## Binary search (git bisect)

## Semi-automatically find a faulty commit

# BLAMING

## Only show certain lines

## To find out who changed a file

## Show the commit that last modified a line

## Ignore whitespace-only changes

# GIT REVISIONS SYNTAX

## Specifying revision by object name

## Symbolic ref names: branches, tags, remote-tracking branches

## The default revision: HEAD

## Reflog references: <refname>@{<n>}

## Reflog references: <refname>@{<date>}

## Tracked / upstream branch: <branchname>@{upstream}

## Commit ancestry chain: <rev>^, <rev>~<n>, etc

## Dereferencing branches and tags: <rev>^0, <rev>^{<type>}

## Youngest matching commit: <rev>^{/<text>}, :/<text>

# WORKTREES

## Using a worktree

## Moving a worktree

# GIT REMOTE

## Display Remote Repositories

## Change remote url of your Git repository

## Remove a Remote Repository

## Add a Remote Repository

## Show more information about remote repository

## Rename a Remote Repository

# GIT LARGE FILE STORAGE (LFS)

## Declare certain file types to store externally

## Set LFS config for all clones

## Install LFS

# GIT PATCH

## Creating a patch

## Applying patches

# GIT STATISTICS

## Lines of code per developer

## Listing each branch and its last revision's date

## Commits per developer

## Commits per date

## Total number of commits in a branch

## List all commits in pretty format

## Find All Local Git Repositories on Computer

## Show the total number of commits per author

# GIT SEND-EMAIL

## Use git send-email with Gmail

## Composing

## Sending patches by mail

# GIT GUI CLIENTS

## gitk and git-gui

## GitHub Desktop

## Git Kraken

## SourceTree

## Git Extensions

## SmartGit

# REFLOG - RESTORING COMMITS NOT SHOWN IN GIT LOG

## Recovering from a bad rebase

# TORTOISEGIT

## Squash commits

## Assume unchanged

## Ignoring Files and Folders

## Branching

# EXTERNAL MERGE AND DITOOLS

## Setting up KDi3 as merge tool

## Setting up KDi3 as di tool

## Setting up an IntelliJ IDE as merge tool (Windows)

## Setting up an IntelliJ IDE as di tool (Windows)

## Setting up Beyond Compare

# UPDATE OBJECT NAME IN REFERENCE

## Update Object Name in Reference

# GIT BRANCH NAME ON BASH UBUNTU

## Branch Name in terminal

# GIT CLIENT-SIDE HOOKS

## Git pre-push hook

## Installing a Hook

# GIT RERERE

## Enabling rerere

# CHANGE GIT REPOSITORY NAME

## Change local setting

# GIT TAGGING

## Listing all available tags

## Create and push tag(s) in GIT

# TIDYING UP YOUR LOCAL AND REMOTE REPOSITORY

## Delete local branches that have been deleted on the remote

# DI-TREE

## See the files changed in a specific commit

## Usage

## Common di options

# CREDITS

# YOU MAY ALSO LIKE

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated