## GETTING STARTED WITH GIT

### Create your first repository, then add and commit files

Tại dòng lệnh, trước tiên hãy xác minh rằng bạn đã cài đặt Git:

Trên tất cả các hệ điều hành:

| git --version |
| --- |

Trên các hệ điều hành giống UNIX:

| **which git** |
| --- |

Nếu không có gì được trả về hoặc lệnh không được nhận dạng, 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.

Sau khi cài đặt Git, hãy định cấu hình tên người dùng và địa chỉ email của bạn. Làm điều này trước khi thực hiện một cam kết.

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 ống nước cần thiết để Git hoạt động.

Tiếp theo, kiểm tra xem Git sẽ thêm những tập tin nào vào kho lưu trữ mới của bạn; Bước này đáng được quan tâm đặc biệt:

| **git status** |
| --- |

Xem lại danh sách kết quả của các tập tin; 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 "giai đoạn" 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.

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 .

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

| **git commit** -m "Initial commit" |
| --- |

|  |
| --- |

|  |
| --- |

|  |
| --- |

|  |
| --- |

|  |
| --- |

### 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

### "Regular" Git Log

### 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