Set Up

Set author name
git config --global user.name "<Firstname Lastname>"

Set email address
git config --global user.email "<email>"

Change author name and email for the last commit
git commit --amend --author="<Firstname Lastname> <<email>>"

Set automatic CLI coloring
git config --global color.ui auto

Configure line-ending (CRLF)
Show current CRLF config:
git config core.autocrlf

Set line-ending style:
git config --global core.autocrlf <input|true|false>
- input: converts crlf to lf, but not the other way around
- false: turn off crlf
- true: converts lf endings into crlf

Set default editor
git config --global core.editor "vim"

List all config settings
git config --list

Remove

Local branch
(using -D instead of -d forces deletion)
git branch -d <branch_name>

Local branches that aren't on remote repository
git remote prune <remote_name> --dry-run
to check the branches that will be deleted, before deleting them

Remote branch
git push --delete <remote_name> <branch_name>

Remove file from working directory and Git repo
git rm <path/to/file.txt>

Tag from local repository
git tag -d <tag_name>

Tag from remote repository
git push --delete origin <tag_name> or:
git push <remote_name> :refs/tags/<tag_name>

Specific stash
(remove the [stash_id] or the last one if none is provided)
git stash drop [stash_id]

All stored stashes
git stash clear

Untracked files
(remove untracked files. Modified files are unchanged)
git clean -f

Untracked files and directories
(remove untracked files and directories. Modified files are unchanged)
git clean -f -d

Remote from a repository
git remote rm <remote_name>

Workflow

Init project
cd ~/project_directory

Create repository from local data:
git init

Create from existing remote (e.g. GitHub) repository:
git clone <repository_url>

Create new branch based on [base_branch], switch to it
git checkout -b <branch_name> [base_banch]

Create new branch based on current branch, switch to it
git checkout -b <branch_name>

Create a new branch, stay on current branch
git branch <branch_name>

Get new changes, branches, tags from remote repo
(just get the changes, doesn't merge them)
git fetch <remote_name>

Check changes on current branch
(show the status of the working directory)
git status

Add a local changed file to stage area
git add <path/to/file.txt>

Add all local changes to stage area
git add .

Interactively asks to stage, and show changes
git add -p

Commit staged local changes on current local branch
git commit -m "Commit message"

Add all local files to stage area and commit
git commit -am "Commit message"

Switch to another local branch
git checkout <branch_name>

Back to previous branch
git checkout -

Get a remote branch locally
git fetch origin
git checkout <remote_branch_name>
or:
git checkout -t <remote_name>/<remote_branch_name>

Merge another branch on current local branch
git merge <branch_name_to_merge>

Merge remote branch changes on current local branch
(fetch from remote and merge into local to keep up-to-date)
git pull <remote_name> <remote_branch_name>

Conflicts
Resolve merge conflicts in favor of pulled changes during pull:
git checkout --theirs <path/to/file.txt>

Resolve merge conflicts in favor of local changes during pull:
git checkout --ours <path/to/file.txt>
e.g.: git checkout --ours package-lock.json

Pick one or more commits and apply to the current local branch
git cherry-pick <commit_sha> [commit_sha]

Push local commited changes to a remote branch
git push <remote_name> <remote_branch_name>

-u|--set-upstream: also add upstream (tracking) reference
git push -u <remote_name> <remote_branch_name>

--tags: push also the tags
git push --tags <remote_name> <remote_branch_name>

--force|-f: force push - if there are changes on remote branch that aren't in local branch (command refuses to update remote), and you want to overwrite them anyway:
git push -f <remote_name> <remote_branch_name>

--force-with-lease: force push and ensure you don't overwrite work from others
git push --force-with-lease <remote_name> <branch_name>

HEAD
is the ref to current branch and last commit on local repository:

cat .git/HEAD
ref: refs/heads/<branch_name>

<remote_name>
it's usually **origin**

<commit_sha>
it can be found using **git log**
or on GitHub history
e.g. acd0f29

<stash_id>
use **git stash list** to get <stash_id>
It's usually: stash@{index}
e.g. stash@{1}

Rename

Local branch
git branch -m <old_name> <new_name>

Remote branch
Delete the current branch:
git push <remote_name> --delete <old_name>
Push the new local branch with the new name:
git push -u <remote_name> <new_name>

Existing remote name
List existing remotes urls
git remote -v
> origin https://github.com/username/reposiroty.git (fetch)
> origin https://github.com/username/repository.git (push)

Rename the remote from old_name to new_name:
git remote rename <old_name> <new_name>
e.g.: git remote rename origin production

Rename a file
git mv <old_name> <new_name>

Update

Update remote
Change the remote's URL from SSH to HTTPS:
git remote set-url origin https://github.com/username/repository.git
Change the remote's URL from HTTPS to SSH:
git remote set-url origin git@github.com:username/repository.git

Update unpushed commit message
git commit --amend -m "New commit message here"

Update last commit, keep the commit message
git commit --amend --no-edit

Notes

Add additional information (note) on commit
git notes add -m 'Note text here' [commit_sha]

Show note for HEAD
git notes show

Push all notes to remote
git push <remote_name> refs/notes/*

Show all notes
git log --show-notes="*"

Fetch all remote notes
git fetch <remote_name> refs/notes/*:refs/notes/*

Stash

Put changed files into stash
to stash with a message
git stash or: git stash push -m <message>

Include untracked files into stash
git stash -u

Add all files into stash (ignored, untracked, and tracked)
git stash -a

Apply stash content and drop it from stash
git stash pop

Apply specific stash without deleting from the stash area
git stash apply <stash_id>

Get a single file from a stash
git checkout <stash_id> -- <path/to/file.txt>

Show the content of specific stash
git stash show -p <stash_id>

Undo

Unstage a file
(keep the changes in working directory)
git reset <path/to/file.txt>

Unstage all files
(keep the changes in working directory)
git reset HEAD -- .

Discard changes on unstaged file in working directory
(changes to the modified file are discarded)
git checkout -- <path/to/file.txt>

Discard changes on all unstaged files in working directory
(changes to the modified files are discarded)
git checkout .

Undo local unpushed commit (most recent commit)
(remove the commit, keep changes on working directory)
git reset HEAD~

Revert changes
(switches the current branch to target reference, leaving a difference as an uncommitted change)
git reset [--hard] <target_reference>
to discard all changes

Discard all changes and restore to last commit
git reset --hard HEAD

Revert a specific commit
(creates a new commit, reverting changes)
git revert <commit_sha>

Revert local changes to a relative time
git reset --hard HEAD@{3.minutes.ago}

Change the last (unpushed) commit message
git commit --amend -m "New message here"

Undo deleted file
git reset HEAD <path/to/file.txt>

Untrack files without deleting on working directory
git rm --cached <path/to/file.txt>

Useful commands

Git help guide
git help -g

Git web-based UI
git instaweb --httpd=<httpd_daemon>
(default httpd daemon: lighttpd)
e.g.: git instaweb --httpd=apache2
git instaweb --httpd=nginx
git instaweb --httpd=webrick

Sync with remote, overwrite local changes
git fetch origin &&
git reset --hard <remote_name>/<branch_name> &&
git clean -f -d

Find the commit that introduced a bug using binary search
git bisect start
git bisect bad
git bisect good
git bisect skip

List

All local branches
git branch

All tracking branches, their upstreams, last commit, and if local branch is ahead, behind or both
git branch -vv

All branches merged into the specified branch
git branch --merged <branch_name>

All operations made on local repository
(commits, checkouts, pull, ... also removed commits)
git reflog

All files with conflict
git diff --name-only --diff-filter=U

Changes since a provided period
git log --no-merges --raw --since='3 weeks ago' or:
git whatchanged --since='3 weeks ago'

Search commits by content
git log -S '<content to search>'

Search in files
git grep -n <content or expression to search>

Search by commit message
git log --all --grep='content to search'
git log --oneline | grep -F 'content to search'

All changes for specific file
git log -p <path/to/file.txt>

Unstaged Changes
git diff

Staged changes (not commited)
git diff --staged or:
git diff --cached

Staged and unstaged changes
git diff HEAD

All changed files on specific commit
git diff-tree --no-commit-id --name-only -r <commit_sha>

Show all commits (Git history)
(commit history as a one-line - commit sha and message)
git log --oneline --graph --all

Summary of commits grouped by author
git shortlog

Local unpushed commits
git log @{u}..

Differences between two commits
to save in a file instead of output on terminal
git diff <commit_sha1>..<commit_sha2> > file.txt

Differences between two branches
git diff <branch_name>..<branch_name>

Show all contributors and with commits count
git shortlog -sn

All saved stashes
git stash list

@andreabohner