### 1. Konfigurasi Awal (Setup)

Perintah ini biasanya hanya perlu dijalankan sekali saat Anda pertama kali menginstal Git di komputer Anda.

| Perintah | Penjelasan |
| --- | --- |
| git config --global user.name "Nama Anda" | Mengatur nama Anda yang akan muncul sebagai penulis (author) dari setiap commit. |
| git config --global user.email "email@anda.com" | Mengatur email Anda yang akan terhubung dengan setiap commit. Penting agar sesuai dengan akun GitHub/GitLab/Bitbucket Anda. |
| git config --global core.editor "code --wait" | **(Opsional)** Mengatur editor teks default yang akan dibuka Git saat butuh input teks yang panjang (misalnya, saat commit tanpa flag -m). Contoh ini untuk VS Code. |
| git config --list | Menampilkan semua konfigurasi Git yang sedang berlaku. |

### 2. Memulai & Mengambil Proyek (Starting a Project)

Langkah pertama untuk mulai bekerja dengan Git di proyek Anda.

| Perintah | Penjelasan |
| --- | --- |
| git init | Menginisialisasi sebuah repositori Git baru di dalam folder proyek Anda saat ini. Perintah ini akan membuat subfolder tersembunyi bernama .git. |
| git clone [url] | Membuat salinan (kloning) dari repositori remote yang sudah ada (misalnya dari GitHub) ke komputer lokal Anda. |

### 3. Alur Kerja Dasar Sehari-hari (The Basic Workflow)

Ini adalah perintah-perintah yang akan paling sering Anda gunakan.

| Perintah | Penjelasan |
| --- | --- |
| git status | Perintah paling penting. Menampilkan status dari *working directory* dan *staging area*. Anda bisa melihat file mana yang sudah diubah, file mana yang sudah di-*stage* (siap di-commit), dan file baru yang belum dilacak (untracked). |
| git add [nama\_file] | Menambahkan perubahan pada file spesifik ke *staging area*. *Staging area* adalah "keranjang" tempat Anda mengumpulkan perubahan sebelum di-commit. |
| git add . | Menambahkan **semua** perubahan (file baru dan file yang dimodifikasi) di direktori saat ini dan subdirektori ke *staging area*. |
| git commit -m "Pesan commit yang deskriptif" | Menyimpan perubahan yang ada di *staging area* secara permanen ke riwayat (history) repositori lokal Anda. **Pesan commit yang baik sangat penting!** |
| git pull | Mengambil (fetch) perubahan terbaru dari repositori remote dan langsung menggabungkannya (merge) ke branch lokal Anda. Ini adalah cara untuk memastikan kode Anda selalu *up-to-date*. |
| git push | Mengirim semua commit yang ada di branch lokal Anda ke repositori remote (misalnya GitHub). Ini adalah cara Anda membagikan hasil kerja Anda dengan tim. |

### 4. Bekerja dengan Cabang (Branching)

Branching adalah fitur inti Git yang memungkinkan Anda bekerja pada fitur baru atau perbaikan bug secara terisolasi tanpa mengganggu kode utama.

| Perintah | Penjelasan |
| --- | --- |
| git branch | Menampilkan semua branch yang ada di repositori lokal Anda. Branch yang aktif akan ditandai dengan tanda bintang (\*). |
| git branch [nama\_branch] | Membuat sebuah branch baru. |
| git checkout [nama\_branch] | Berpindah kerja ke branch lain yang sudah ada. |
| git checkout -b [nama\_branch] | **Shortcut populer:** Membuat branch baru dan langsung berpindah ke branch tersebut. Setara dengan git branch [nama\_branch] diikuti git checkout [nama\_branch]. |
| git merge [nama\_branch] | Menggabungkan riwayat dari [nama\_branch] ke branch Anda saat ini. Biasanya, Anda akan berada di branch main atau develop lalu me-merge branch fitur. |
| git branch -d [nama\_branch] | Menghapus branch lokal. Git akan mencegah penghapusan jika branch tersebut belum di-merge. Gunakan -D (kapital) untuk memaksa hapus. |

### 5. Memeriksa Riwayat & Perubahan (Inspecting History)

Perintah untuk melihat apa yang sudah terjadi di dalam repositori.

| Perintah | Penjelasan |
| --- | --- |
| git log | Menampilkan riwayat commit untuk branch saat ini, dari yang terbaru hingga yang terlama. Tekan q untuk keluar. |
| git log --oneline | Menampilkan log dalam format yang lebih ringkas, satu commit per baris. |
| git log --graph --oneline --decorate --all | **Versi "power user":** Menampilkan log dalam bentuk grafis yang visual, ringkas, dan menunjukkan semua branch. Sangat direkomendasikan untuk membuat alias untuk perintah ini. |
| git diff | Menampilkan perbedaan antara perubahan di *working directory* Anda dengan yang ada di *staging area*. |
| git diff --staged atau git diff --cached | Menampilkan perbedaan antara perubahan di *staging area* dengan commit terakhir. |
| git diff [branch1] [branch2] | Menampilkan perbedaan konten antara dua branch. |

### 6. Membatalkan Perubahan (Undoing Things)

Setiap orang membuat kesalahan. Git menyediakan beberapa cara untuk memperbaikinya.

| Perintah | Penjelasan |
| --- | --- |
| git checkout -- [nama\_file] | **Hati-hati:** Membatalkan semua perubahan pada file di *working directory* dan mengembalikannya ke versi terakhir yang di-commit. Perubahan yang dibatalkan akan hilang selamanya. |
| git reset [nama\_file] | Mengeluarkan file dari *staging area* (unstage), tetapi perubahannya tetap ada di *working directory*. Ini adalah kebalikan dari git add. |
| git reset --hard [hash\_commit] | **SANGAT HATI-HATI:** Mengembalikan kondisi seluruh proyek (file dan riwayat) ke sebuah commit spesifik. Semua commit dan perubahan setelah titik itu akan hilang. **Jangan gunakan pada branch yang sudah di-push ke remote.** |
| git revert [hash\_commit] | Membuat sebuah **commit baru** yang isinya adalah kebalikan dari commit yang ingin dibatalkan. Ini cara yang lebih aman untuk membatalkan perubahan pada branch publik karena tidak mengubah riwayat yang sudah ada. |

### 7. Bekerja dengan Repositori Remote (Remotes)

Mengelola koneksi ke repositori lain.

| Perintah | Penjelasan |
| --- | --- |
| git remote -v | Menampilkan daftar semua koneksi remote yang sudah terkonfigurasi. Biasanya akan ada origin. |
| git remote add [nama\_remote] [url] | Menambahkan koneksi ke repositori remote baru. Misalnya, git remote add origin https://... |
| git fetch [nama\_remote] | Mengambil semua data terbaru (commit, branch, tag) dari remote, tetapi **tidak** menggabungkannya ke branch lokal Anda. Ini berguna untuk melihat apa yang telah dikerjakan orang lain sebelum mengintegrasikannya ke pekerjaan Anda. |

### 8. Perintah Lanjutan yang Berguna (Advanced & Useful)

Perintah ini mungkin tidak digunakan setiap hari, tetapi sangat powerful dalam situasi tertentu.

| Perintah | Penjelasan |
| --- | --- |
| git stash | Menyimpan sementara semua perubahan yang belum di-commit, sehingga *working directory* Anda menjadi bersih. Berguna jika Anda harus berpindah branch mendadak tapi pekerjaan Anda belum siap di-commit. |
| git stash pop | Menerapkan kembali perubahan yang terakhir Anda simpan dengan git stash dan menghapusnya dari daftar stash. |
| git rebase -i [hash\_commit\_sebelumnya] | Mode *interactive rebase*. Memungkinkan Anda untuk mengedit, menggabungkan (squash), menyusun ulang, atau menghapus commit-commit terakhir sebelum Anda mem-pushnya. Sangat berguna untuk merapikan riwayat commit lokal. **Jangan rebase branch yang sudah di-push!** |
| git cherry-pick [hash\_commit] | Mengambil satu commit spesifik dari branch lain dan menerapkannya pada branch Anda saat ini. Berguna untuk mengambil perbaikan bug kecil tanpa harus me-merge seluruh branch. |
| git tag -a v1.0.0 -m "Versi 1.0" | Memberi tanda (tag) pada sebuah commit, biasanya digunakan untuk menandai rilis versi software. |

### Contoh Alur Kerja Sederhana

1. Ambil proyek dari GitHub:  
   git clone https://github.com/user/proyek.git
2. Masuk ke folder proyek:  
   cd proyek
3. Buat branch baru untuk fitur login:  
   git checkout -b fitur-login
4. **Mulai coding...** (buat file login.html, login.css)
5. Lihat status:  
   git status
6. Tambahkan file baru ke staging:  
   git add .
7. Simpan pekerjaan Anda:  
   git commit -m "feat: Menambahkan halaman dasar untuk login"
8. Pindah kembali ke branch utama:  
   git checkout main
9. Pastikan branch utama up-to-date:  
   git pull
10. Gabungkan branch fitur:  
    git merge fitur-login
11. Kirim hasilnya ke GitHub:  
    git push origin main

Semoga daftar ini membantu! Kunci untuk menguasai Git adalah **praktik**. Jangan takut untuk membuat repositori coba-coba dan bereksperimen dengan perintah-perintah ini. Selamat coding!