Git Cheat Sheet

1. Configuração Inicial (Faça apenas uma vez)

- **git config user.name "Seu Nome"**
    - **O que faz:** Define o nome que aparecerá nos seus commits.
    - **Se você utilizar o parâmetro "–global" após a palavra config, irá mudar esta configuração para todos os repositórios git nesta máquina.**
- **git config user.email "seu@email.com"**
    - **O que faz:** Define o email que aparecerá nos seus commits.
    - **Se você utilizar o parâmetro "–global" após a palavra config, irá mudar esta configuração para todos os repositórios git nesta máquina.**

2. Iniciando um Projeto

- **git init**
    - **O que faz:** Inicia um novo repositório na pasta atual.
- **git clone url_do_repositorio**
    - **O que faz:** Baixa (clona) um projeto existente do GitHub para o seu computador. Não é necessário fazer mais de uma vez.

3. Configurando um repositório remoto

- **git remote add origin url_do_seu_repositorio_remoto.git**
    - **O que faz:** adiciona ao repositório local o link do repositório na nuvem para que as informações sejam passadas com git push.
- **git branch -M main**
    - **O que faz:** renomeia a sua Branch principal para main.
- **git push -u origin main**

- **O que faz:** executa seu primeiro push com o parâmetro -u, criando um link do repositório local com o remoto, depois não será necessário o parâmetro -u. Basta usar git push.

4. O Ciclo Básico

- **git status**
  - **O que faz:** Mostra o estado atual do seu projeto. Quais arquivos foram modificados e o que está pronto para ser "fotografado". **Use o tempo todo!**
- **git add nome_do_arquivo**
  - **O que faz:** Adiciona um arquivo específico à "área de preparação" (*staging area*).
- **git add .**
  - **O que faz:** Adiciona **todas** as alterações atuais à área de preparação.
- **git commit -m "Sua mensagem clara e descritiva"**
  - **O que faz:** Tira a "foto" (salva o *snapshot*) das alterações que estavam na área de preparação e a guarda no histórico local.
  - **Cuidado para não esquecer o " -**m ".
  - **Lembre-se** a mensagem do commit deve iniciar com um verbo e falar exatamente sobre o que está sendo feito. Exemplo de commit bem escrito: "Adiciona o campo CPF para cadastro de usuário."
  - **Não espere muitas alterações ocorrerem para fazer o commit, isso tornará a manutenção do código difícil.**
- **git push**
  - **O que faz:** envia suas alterações para o repositório remoto.

5. Sincronizando com o Repositório Remoto (GitHub)

Nesta etapa tenha sempre muita atenção ao caminho do repositório e se você está no local correto e com o usuário correto!

- **git pull**
    - **O que faz:** Busca e integra as atualizações do GitHub no seu repositório local.
- **git remote -v**
    - **O que faz:** Lista os repositórios remotos configurados.
- **git fetch**
    - **O que faz**: verifica os links de repositórios remotos.

6. Trabalhando com Branches

- **git branch**
    - **O que faz:** Lista todos os branches do seu projeto. **O branch atual é marcado com um *.**
- **git branch nome_do_novo_branch**
    - **O que faz:** Cria um novo branch a partir do seu branch atual.
- **git checkout nome_do_branch ou git switch nome_do_branch**
    - **O que faz:** Muda para o branch especificado. (switch é a sintaxe mais moderna e recomendada).
- **git checkout -b nome_do_novo_branch ou git switch -c nome_do_novo_branch**
    - **O que faz:** Cria um novo branch e já muda para ele em um único comando. **(Super útil!)**

7. Juntando Histórias (Merge)

- **git merge nome_do_branch_a_ser_juntado**

- **O que faz:** Traz as alterações de outro branch para o seu branch atual.
- **Fluxo Típico:**
    1. git switch main (Vá para o branch que vai receber as alterações)
    2. git pull (Garanta que ele está atualizado)
    3. git merge nome_do_seu_branch (Traga as novidades)

8. Viajando no Tempo (Desfazendo Alterações)

- **git log**
    - **O que faz:** Mostra o histórico de commits do seu branch.
    - **Dica de Ouro:** Use git log --oneline para uma visão mais limpa e compacta!
- **git revert hash_do_commit**
    - **O que faz:** Cria um **novo commit** que desfaz as alterações de um commit antigo.
    - **Por que usar?** É a forma **mais segura** de reverter algo, pois não apaga o histórico. Ideal para commits que já foram enviados ao GitHub.
- **git reset hash_do_commit**
    - **O que faz:** Volta o projeto para um estado anterior, **apagando** os commits que vieram depois.
    - **ATENÇÃO:** Use com cautela e principalmente em commits que ainda não foram enviados ao GitHub (push). reset reescreve a história.
- **git checkout hash_do_commit -- [caminho/do/arquivo]**

- **O que faz:** Restaura um **único arquivo** para a versão em que ele estava em um commit específico, sem afetar o resto do projeto. **(Extremamente útil para recuperar um arquivo específico!)**