**Git stash é lindo**

O **Git Stash** é uma funcionalidade do Git que permite salvar temporariamente as alterações no seu ambiente de trabalho (working directory) e no índice (staging area) sem precisar fazer um commit. Ele é útil quando você precisa alternar de branch ou realizar outra tarefa antes de concluir as alterações atuais, mas quer "guardar" essas mudanças para retomá-las posteriormente.

**Por Que Usar o Git Stash?**

Imagine que você está no meio de uma implementação, mas precisa:

1. Mudar para outro branch para corrigir um bug urgente.
2. Atualizar seu branch local para sincronizar com o repositório remoto.
3. Realizar algum outro trabalho que não pode esperar.

Nesse caso, fazer um commit pode não ser a melhor opção, especialmente se as mudanças ainda não estão prontas ou são temporárias. O Git Stash permite que você armazene essas mudanças "fora do caminho" e retome quando estiver pronto.

**Comandos Básicos do Git Stash**

**1. Salvar alterações no stash:**

git stash

* Este comando salva todas as alterações não commitadas (arquivos modificados e rastreados) e restaura seu ambiente de trabalho ao estado do último commit.
* Se você quiser incluir arquivos novos (não rastreados), adicione a flag -u:
* git stash -u
* Para incluir todos os arquivos ignorados (untracked e ignored), use --all:
* git stash --all

**2. Visualizar o stash:**

git stash list

* Mostra uma lista de todas as entradas armazenadas no stash, cada uma com um identificador único, como stash@{0} (o stash mais recente).

**3. Aplicar alterações do stash:**

git stash apply

* Este comando aplica as alterações mais recentes do stash ao seu ambiente de trabalho, mas **não remove** o stash da lista.
* Para aplicar uma entrada específica:
* git stash apply stash@{2}

**4. Remover e aplicar alterações (pop):**

git stash pop

* Aplica as alterações mais recentes do stash ao ambiente de trabalho e **remove** a entrada do stash.

**5. Excluir uma entrada do stash:**

git stash drop stash@{0}

* Remove uma entrada específica do stash.
* Para remover a entrada mais recente:
* git stash drop

**6. Limpar todo o stash:**

git stash clear

* Exclui todas as entradas armazenadas no stash.

**Cenários Comuns de Uso do Git Stash**

1. **Interromper o trabalho atual para corrigir um bug**:
   * Você está desenvolvendo uma nova funcionalidade no branch feature e, de repente, precisa corrigir um bug no branch main. Você pode usar o stash para salvar as alterações atuais:
   * git stash
   * git checkout main
   * Após corrigir o bug, volte ao branch e recupere suas alterações:
   * git checkout feature
   * git stash pop
2. **Atualizar o branch sem perder progresso**:
   * Você quer atualizar seu branch local (por exemplo, main) com as últimas mudanças do repositório remoto, mas tem alterações locais:
   * git stash
   * git pull origin main
   * git stash pop
3. **Prototipagem rápida**:
   * Você começou a implementar uma ideia, mas decidiu abandoná-la temporariamente. Use o stash para guardar seu trabalho sem comprometer o histórico do branch:
   * git stash

**Dicas e Funcionalidades Avançadas do Stash**

1. **Nomear um stash**: Ao salvar alterações no stash, você pode adicionar uma mensagem descritiva para facilitar a identificação:
2. git stash push -m "Alterações na tela de login"
3. **Visualizar o conteúdo do stash**: Use o comando git stash show para visualizar um resumo das alterações no stash mais recente:
4. git stash show

Para ver detalhes completos:

git stash show -p stash@{0}

1. **Criar um branch a partir do stash**: Se as alterações armazenadas no stash se transformarem em algo significativo, você pode criar um novo branch diretamente:
2. git stash branch nova-feature stash@{0}

**Vantagens do Git Stash**

* **Fluxo de trabalho mais flexível**: Permite pausar o trabalho sem a necessidade de fazer commits temporários.
* **Preserva o histórico limpo**: Evita commits intermediários que podem poluir o histórico do projeto.
* **Rápido e fácil**: Salvar e restaurar alterações é simples, com poucos comandos.

**Limitações do Git Stash**

* **Não salva arquivos não rastreados por padrão**: É necessário usar a flag -u ou --all para incluir arquivos não rastreados ou ignorados.
* **Pode ser confuso com múltiplos stashes**: Quando há muitos stashes, gerenciar entradas específicas pode se tornar complexo.
* **Histórico não versionado**: As alterações no stash não são parte do histórico do Git, então, se o stash for perdido (por exemplo, em um novo clone do repositório), as alterações também serão perdidas.

**Conclusão**

O **Git Stash** é uma ferramenta poderosa para gerenciar alterações temporárias no Git, oferecendo flexibilidade ao permitir que você interrompa seu trabalho atual e retorne a ele mais tarde. Ele é especialmente útil em fluxos de trabalho dinâmicos, onde mudanças de contexto são frequentes. Entretanto, deve ser usado com atenção, especialmente ao lidar com múltiplos stashes ou arquivos não rastreados.