**1. Git Branch**

Um Git branch (ou ramificação) é um ponteiro móvel para um commit, que funciona como uma linha de desenvolvimento separada dentro de um repositório, permitindo trabalhar em novas funcionalidades ou corrigir bugs sem afetar o código principal do projeto. Criar e usar branches é fundamental para a organização do desenvolvimento, facilitando a colaboração entre equipes, o isolamento de mudanças e a integração segura do código no ramo principal, como *main* ou *master*.

**O que um Git branch faz:**

* **Cria linhas de desenvolvimento paralelas:** permite que desenvolvedores trabalhem em recursos isolados sem interferir no código já estável.
* **Isola o trabalho:** as alterações feitas em um branch só afetam aquele branch específico, garantindo que o código principal permaneça intacto.
* **Facilita a colaboração:** várias pessoas podem trabalhar em branches diferentes simultaneamente, sem conflitos, tornando o desenvolvimento mais eficiente.

**Por que usar Git branches:**

* **Desenvolvimento de novas funcionalidades:** possibilita testar e implementar novas ideias de forma organizada e segura.
* **Correção de bugs:** facilita a identificação e correção de erros em um ambiente isolado, sem impactar a versão em produção.
* **Experimentação:** oferece um espaço seguro para experimentar e testar ideias que podem ou não ser incorporadas ao projeto.

**Como um Git branch é criado e gerenciado:**

* **Ponteiro móvel:** um branch é essencialmente um arquivo que armazena o ID (SHA-1) do último commit feito nele.
* **Criação fácil:** criar um novo branch no Git é um processo rápido, pois é apenas um novo arquivo com esse ponteiro, não uma cópia de todo o projeto.
* **Integração com outros comandos:** o comando git branch é usado em conjunto com outros, como git checkout (para mudar de branch) e git merge (para combinar o trabalho de um branch no principal).

**2. Git Merge**

O comando git merge é utilizado para combinar as alterações de uma ramificação (branch) em outra, criando um histórico de desenvolvimento unificado. Ele integra linhas de desenvolvimento independentes, sendo fundamental para mesclar um trabalho em uma ramificação de funcionalidade de volta para a linha principal (como a ramificação *main*), preservando o histórico de ambos os branches.

**Como funciona o Git merge:**

1. **Identifica a base comum:** o Git encontra o commit base em comum entre as duas ramificações que serão mescladas.
2. **Cria uma nova confirmação de merge:** em seguida, cria uma nova confirmação (commit) de merge, com duas confirmações mães, combinando as mudanças das ramificações de origem e destino.
3. **Mescla os históricos:** a ramificação atual é atualizada para refletir a mesclagem, unificando as linhas de desenvolvimento.

**Tipos de merge:**

* **Fast Forward (Rápido):** se não houver alterações divergentes, o Git apenas move o ponteiro da ramificação principal para o último commit da ramificação que está sendo mesclada.
* **Merge Commit (Confirmação de Merge):** se as ramificações tiveram desenvolvimento paralelo, o Git cria uma confirmação de merge dedicada para unir os dois históricos.

**Quando usar o Git merge:**

* **Integração de funcionalidades:** maneira padrão de incorporar o trabalho concluído em uma ramificação de feature para a ramificação principal.
* **Consolidação de trabalho em equipe:** permite que diferentes desenvolvedores unifiquem suas alterações, mesmo quando trabalham em paralelo em diferentes ramificações.

**Por que usar Git merge:**

* **Preserva o histórico:** mantém o histórico completo dos commits de ambos os branches, oferecendo uma visão transparente do progresso.
* **Não destrutivo:** é uma operação segura que não altera os branches existentes, apenas os atualiza com as novas informações, ao contrário de operações como o *rebase*.

1. **Git Push**

O comando git push envia as alterações do repositório local para um repositório remoto, como o GitHub. Esse processo transfere os commits para o repositório remoto, permitindo que outros membros da equipe vejam as modificações ou que você tenha um backup do seu código em um servidor.

**Como funciona:**

1. **Alterações locais:** após realizar modificações no código e gravá-las com git commit, elas ficam salvas no repositório local.
2. **Envio para o repositório remoto:** o comando git push é utilizado para enviar esses commits locais para um repositório remoto, hospedado em serviços como GitHub, GitLab ou Bitbucket.
3. **Compartilhamento e backup:** ao usar git push, você compartilha seu trabalho com outros desenvolvedores ou garante que suas alterações estejam salvas em um local centralizado.