**Versionando com Tags**

No Git, as **tags** são usadas para marcar pontos específicos no histórico de commits que são importantes, como o lançamento de uma versão de software. Elas são frequentemente usadas para indicar versões de lançamento, como v1.0, v2.1.3, etc., e tornam mais fácil localizar e referenciar esses commits no futuro.

**Tipos de Tags no Git**

Existem dois tipos principais de tags no Git:

1. **Tags Leves (Lightweight Tags)**:
   * São essencialmente um ponteiro para um commit específico.
   * Não possuem metadados adicionais como uma mensagem ou um autor.
   * Usadas para marcações simples e temporárias.
2. **Tags Anotadas (Annotated Tags)**:
   * Armazenam metadados adicionais, como:
     + Mensagem do criador da tag.
     + Autor.
     + Data.
   * São armazenadas como objetos completos no banco de dados do Git.
   * Recomendadas para uso em lançamentos oficiais.

**Criando Tags no Git**

**1. Criar uma Tag Leve:**

git tag <nome-da-tag>

Exemplo:

git tag v1.0

Isso cria uma tag leve chamada v1.0 no commit atual.

**2. Criar uma Tag Anotada:**

git tag -a <nome-da-tag> -m "<mensagem>"

Exemplo:

git tag -a v1.0 -m "Primeiro lançamento oficial"

Isso cria uma tag anotada chamada v1.0 com a mensagem "Primeiro lançamento oficial".

**3. Tag para um Commit Específico:**

Você pode criar uma tag para um commit específico fornecendo seu hash:

git tag -a <nome-da-tag> <hash-do-commit> -m "<mensagem>"

Exemplo:

git tag -a v1.0 1a2b3c4d -m "Lançamento para commit específico"

**Listar Tags**

Para ver todas as tags no repositório:

git tag

Para procurar tags que seguem um padrão:

git tag -l "v1.\*"

**Compartilhando Tags com o Repositório Remoto**

As tags não são enviadas automaticamente quando você faz um git push. Para enviar uma tag específica:

git push origin <nome-da-tag>

Exemplo:

git push origin v1.0

Para enviar todas as tags de uma vez:

git push origin --tags

**Usando Tags**

**1. Checkout em uma Tag:**

Para alternar para o estado do repositório em um ponto marcado por uma tag:

git checkout <nome-da-tag>

Isso coloca o repositório em um estado "desanexado" (detached HEAD), ou seja, você não está em nenhum branch, mas no estado exato daquele commit.

**2. Comparar Tags:**

Para comparar diferenças entre duas tags:

git diff <tag1> <tag2>

Exemplo:

git diff v1.0 v2.0

**3. Verificar Detalhes de uma Tag:**

Se for uma tag anotada, você pode ver os detalhes:

git show <nome-da-tag>

**Excluindo Tags**

**1. Excluir Tag Localmente:**

git tag -d <nome-da-tag>

Exemplo:

git tag -d v1.0

**2. Excluir Tag no Repositório Remoto:**

Primeiro, exclua localmente:

git tag -d v1.0

Em seguida, exclua no remoto:

git push origin --delete <nome-da-tag>

Exemplo:

git push origin --delete v1.0

**Cenários Comuns para Uso de Tags**

1. **Versionamento Semântico (SemVer)**:
   * Tags são amplamente usadas para marcar versões de software seguindo a convenção de versionamento semântico:
   * v<major>.<minor>.<patch>
     + **Major**: Mudanças significativas ou incompatíveis.
     + **Minor**: Novas funcionalidades, mas compatíveis com versões anteriores.
     + **Patch**: Correções de bugs.

Exemplo:

git tag -a v2.1.0 -m "Adicionado recurso de login"

1. **Lançamentos de Produção**:
   * Antes de enviar o código para produção, marque a versão com uma tag para referência futura.
2. **Marcar Marcos Importantes**:
   * Use tags para identificar commits importantes, como o início de um projeto ou o fechamento de uma grande funcionalidade.

**Tags no GitHub**

Tags são integradas ao GitHub e podem ser usadas para criar **releases**. Um **release** no GitHub é uma tag com metadados adicionais, como notas de lançamento, que facilitam a entrega de software para usuários.

**Criar um Release no GitHub:**

1. Vá até a página do repositório.
2. Clique em **Releases** > **Draft a new release**.
3. Escolha uma tag existente ou crie uma nova.
4. Adicione uma descrição e publique.

**Conclusão**

O versionamento com tags no Git é uma forma eficaz de marcar e rastrear pontos importantes no histórico de commits. Ele é amplamente usado para gerenciar versões de software, identificar marcos importantes e facilitar a integração com plataformas como GitHub. Com o uso correto das tags, você pode melhorar a organização e a clareza do seu projeto.