Alisson Martins | N8498-35

Cristhian Eduardo | N8939-C0

Thiago Santiago | G54DE-A2

Pedro Augusto Roseiro Ipolito | N9140-B4

**O que é Git**

De longe, o sistema de controle de versão moderno mais usado no mundo hoje é o Git. O Git é um projeto de código aberto maduro e com manutenção ativa desenvolvido em 2005 por Linus Torvalds, o famoso criador do kernel do sistema operacional Linux. Um número impressionante de projetos de software depende do Git para controle de versão, incluindo projetos comerciais e de código-fonte aberto. Os desenvolvedores que trabalharam com o Git estão bem representados no pool de talentos de desenvolvimento de software disponíveis e funcionam bem em uma ampla variedade de sistemas operacionais e IDEs (Ambientes de Desenvolvimento Integrado).

Tendo uma arquitetura distribuída, o Git é um exemplo de DVCS (portanto, Sistema de Controle de Versão Distribuído). Em vez de ter apenas um único local para o histórico completo da versão do software, como é comum em sistemas de controle de versão outrora populares como CVS ou Subversion (também conhecido como SVN), no Git, a cópia de trabalho de todo desenvolvedor do código também é um repositório que pode conter o histórico completo de todas as alterações.

Além de ser distribuído, o Git foi projetado com desempenho, segurança e flexibilidade em mente.

# **Desempenho**

As características brutas de desempenho do Git são muito fortes quando comparadas a muitas alternativas. Fazer o commit de novas alterações, branches, mesclagem e comparação de versões anteriores – tudo é otimizado para desempenho. Os algoritmos implementados no Git aproveitam o conhecimento profundo sobre atributos comuns de árvores de arquivos de código-fonte reais, como costumam ser modificados ao longo do tempo e quais são os padrões de acesso.

Diferente de alguns softwares de controle de versão, o Git não se deixa enganar pelos nomes dos arquivos ao determinar qual deve ser o armazenamento e o histórico de versões da árvore de arquivos. Em vez disso, o Git se concentra no conteúdo do arquivo. Afinal, os arquivos de código fonte são renomeados, divididos e reorganizados com frequência. O formato do objeto dos arquivos de repositório do Git usa uma combinação de codificação delta (armazenamento de diferenças de conteúdo) e compactação e armazena com clareza o conteúdo do diretório e os objetos de metadados da versão.

A distribuição também oferece benefícios significativos de desempenho.

Por exemplo, digamos que uma desenvolvedora, Alice, faça alterações no código-fonte, adicionando um recurso para a próxima versão, 2.0, e faça o commit dessas alterações com mensagens descritivas. Ela então trabalha em um segundo recurso e faz o commit dessas alterações também. Como esperado, eles são armazenados como peças de trabalho separadas no histórico de versões. Alice então muda para o branch da versão 1.3 do mesmo software para corrigir um erro que afeta apenas a versão mais antiga. O objetivo disso é permitir que a equipe de Alice lance uma versão de correção de bug, a versão 1.3.1, antes que a versão 2.0 esteja pronta. Alice pode retornar ao branch 2.0 para continuar trabalhando nos novos recursos da versão 2.0. Tudo isso pode ocorrer sem nenhum acesso à rede e, portanto, é um processo rápido e confiável. Ela poderia até fazer isso em um avião. Quando estiver pronta para fazer o commit de todas as alterações como itens individuais no repositório remoto, Alice vai poder enviar todas elas por push em um único comando.

# **Segurança**

O Git foi projetado com a integridade do código-fonte gerenciado como uma prioridade. O conteúdo dos arquivos, bem como os verdadeiros relacionamentos entre arquivos e diretórios, versões, tags e commits, todos esses objetos no repositório do Git são protegidos com um algoritmo de hash de criptografia seguro chamado SHA1. Isso protege o código e o histórico de alterações contra alterações acidentais e maliciosas e garante que o histórico tenha rastreabilidade total.

Com o Git, você pode ter certeza de ter um histórico de conteúdo autêntico do código-fonte.

Alguns outros sistemas de controle de versão não têm proteções contra alterações secretas posteriores. Isso pode ser uma vulnerabilidade séria de segurança das informações para qualquer empresa que depende do desenvolvimento de software.

# **Repositório Online**

Os repositórios de software permitem que os usuários localizem e obtenham dados, códigos ou pacotes de software específicos em um ambiente seguro usando sistemas de controle de acesso e autenticação.

# **Alguns dos repositórios de software hospedados mais famosos são GitHub, BitBucket e SourceForge.**

**Git e GitHub na prática:**

1 - Vá até seu perfil, na aba **Repositorios** e clique em **NEW**

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

2 - Preencha os campos e clique em **Create repository**

Texto

Descrição gerada automaticamente

3 - Na aba repositorios clique no repositorio criado

Tela de celular com foto de homem

Descrição gerada automaticamente

4 – Agora copie o link

Tela de computador com texto preto sobre fundo escuro

Descrição gerada automaticamente

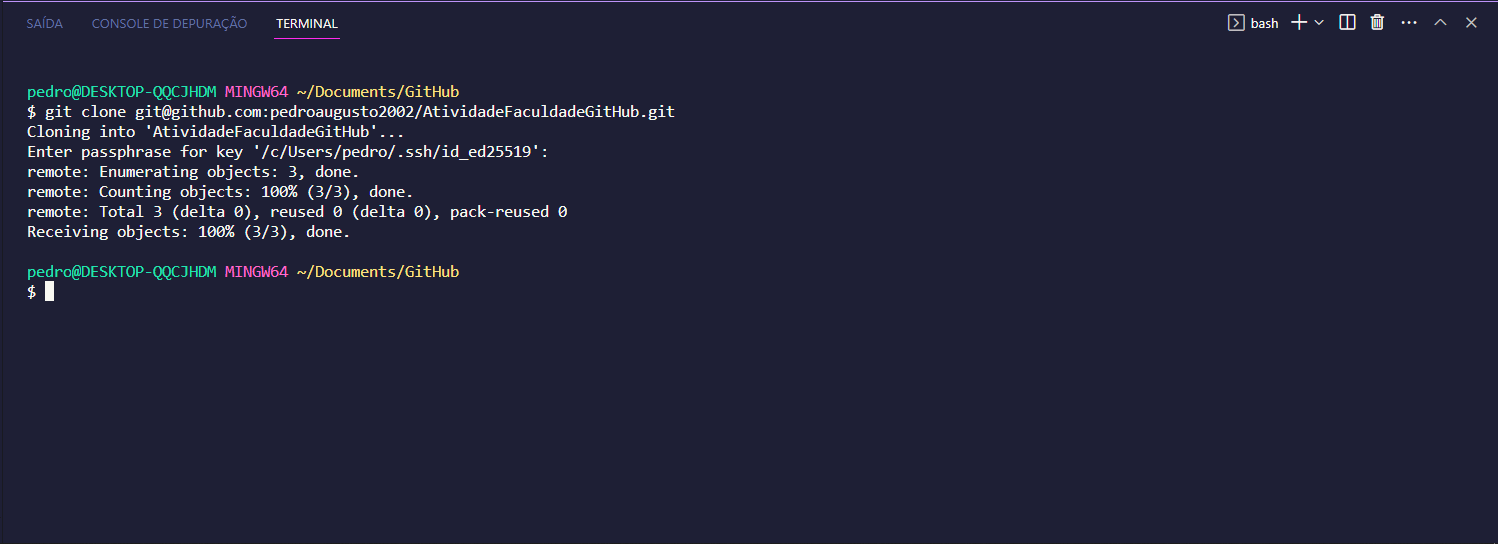
5 – Execute o comando:

**$ git clone “link do seu repositorio”**

Texto

Descrição gerada automaticamente

6 – Resultado esperado:



7 – Após adicionar fazer as alterações desejadas, verifique os status do repositorio com o comando:

**$ git status**

Texto

Descrição gerada automaticamente

8 – Resultado:

Texto

Descrição gerada automaticamente

9 – Adicione as alterações com o comando:

**$ git add .**

Texto

Descrição gerada automaticamente

10 – Resultado:

Texto

Descrição gerada automaticamente

11 – Após adicionar os arquivos faça o **commit** junto a messagem para registrar as alterações com o comando:

**$ git commit -am “Messagem do commit”**

Texto

Descrição gerada automaticamente

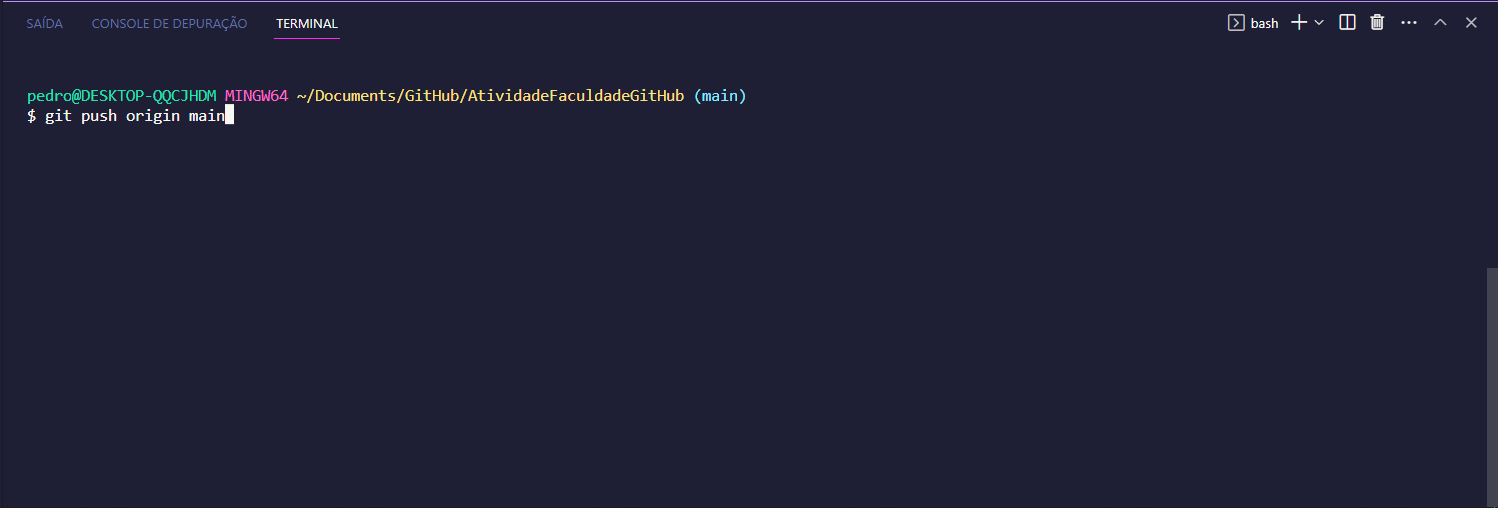
12 – Resultado:

Texto

Descrição gerada automaticamente

13 – Agora basta fazer o upload para o repositório remoto com o comando:

**$ git push origin “nome da branch”**



14 – Insira a passphrase (caso haja)

Texto

Descrição gerada automaticamente

15 -Resultado:

Texto

Descrição gerada automaticamente

16 - Resultado no GitHub:

Tela de celular com aplicativo aberto

Descrição gerada automaticamente