Introdução ao Git e ao GitHub - (Material Curso Dio)

**#GitHub** **#Git**

**[[](javascript:void();)](javascript:void();)**

[**Anderson Froes**](javascript:void();)**15/02/2021 21:09**

Professor:

Otávio Reis

Link para Download do Artigo:

<https://drive.google.com/file/d/1Slgz6lPrYhlDNok2E7yTeaOu5cJd5yJn/view?usp=sharing>

Capitulo 1 - Introdução ao Git

O que é 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.

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.

Desempenho

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. 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.

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.

Flexibilidade

O Git foi projetado para tratar os branches e tags como cidadãos de primeira classe (ao contrário do SVN) e operações que afetam branches e tags (como mesclagem ou reversão) também são armazenadas como parte do histórico de alterações. Nem todos os sistemas de controle de versão apresentam esse nível de rastreamento.

Git é um projeto de código aberto de qualidade

O código aberto reduz o custo para desenvolvedores amadores, pois eles podem usar o Git sem pagar uma taxa. Para uso em projetos de código aberto, o Git é sem dúvida o sucessor das gerações anteriores de sistemas bem-sucedidos de controle de versão de código aberto, SVN e CVS.

Capitulo 2 - Navegação via command line interface e instalação

Instalando o GIT - (Windows)

Link para Download - <https://git-scm.com/>

Marcar as opções Git Bash Here e Git Gui Here

Escolher configuração para windows ou unix

Instalando o GIT - (Unix)

Link para Download - <https://git-scm.com/>

Escolher o código conforme sua distribuição de sistema (Ubuntu, Fedora)

Usar o código no Terminal

git --version (para verificar a versão instalada)

Instalando o GIT - (macOS)

Link para Download - <https://git-scm.com/>

Link para Homebrew - <https://brew.sh/>

Buscar o código Homebrew no site do Git e rodar no seu Terminal

Comandos básicos para um bom desempenho no terminal

O que vamos aprender ?

Mudar de pastas, Listar as pastas, Criar pastas/arquivos, Deletar pastas/arquivos

Sistema Windows

cd = (mudar pastas)

dir = (listar pastas)

mkdir = (criar pastas/arquivos)

del = (deletar arquivos)

rmdir = (deletar a pasta com todos seus arquivos)

cls = (limpar a tela)

Tecla Tab = (alto completa um nome na hora de digitar)

Sistema Unix

cd = (mudar pastas)

ls = (listar pastas)

mkdir = (criar pastas/arquivos)

rm - rf = (deletar pastas/arquivos)

clear = (limpar a tela) ou atalho (CTRL+L)

Tecla Tab = (alto completa um nome na hora de digitar)

Capitulo 3 - Entendendo como o Git funciona por baixo dos panos

Tópicos fundamentais para entender o funcionamento do Git

SHA1, Objetos Fundamentais, Sistema Distribuído, Segurança

SHA1

A sigla SHA significa Secure Hash Algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA). A encriptação gera conjunto de characteres identificador de 40 dígitos. É uma forma curta de representar um arquivo.

Comando que gera o SHA1 = openssl sha1 (nome do arquivo.txt)

Chave gerada = SHA1(dio.txt) = 476ad3f360ced25383c25d13eae57f59e30cecb9

Quando você mudar a informação no arquivo e executa o comando ele muda a chave SHA1 = SHA1(dio.txt) = bf3bb2f68cdabbf6a27b73f876b350af80af6953

Se você alterar a informação no arquivo para a que estava anteriormente e executar o comando SHA1 novamente, ele retorna para a chave anterior = SHA1(dio.txt)= 476ad3f360ced25383c25d13eae57f59e30cecb9

Resultados Final Arquivo Versao1 = 476ad3f360ced25383c25d13eae57f59e30cecb9

Arquivo Modificado = bf3bb2f68cdabbf6a27b73f876b350af80af6953

Arquivo Modificado Versao1 = 476ad3f360ced25383c25d13eae57f59e30cecb9

Objetos internos no Git

Blobs, Trees, Commits

Blobs - Git gera SHA1 e armazena Meta-Dados (bloco básico de composição) Possuem o SHA1 do arquivo echo 'conteudo' | git hash-object --stdin = gera um SHA1 echo -e 'blob 9\0conteudo' | openssl sha1 = gera mesmo SHA1

Trees - Armazenam Blobs (armazena e aponta para um tipo de blobs) E responsável por toda a estrutura por montar a estrutura de um arquivo Possuem o SHA1 dos meta-dados

Commit - É o objeto mais importante, que junta tudo! Ele aponta para uma arvore, parente, autor, mensagem. O SHA1 desse commit é o hash de toda essa informação.

Sistema Distribuido Seguro

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.

Capitulo 4 - Primeiros comandos com Git

Iniciando o Git e Criando um commit

O que vamos aprender ?

Iniciar o GIT , Iniciar o versionamento , Criar um commit

Comandos

git init (iniciar repositorio no git)

git add (mover arquivos e dar inicio ao versionamento)

git commit (criar commites)

Criando um Repositório

Acessar via terminal a pasta do seu projeto

git init = Efetuar esse comando para criar uma pasta oculta .git

ls -a = (mostrar arquivos e pastas ocultas)

Adicionando um Arquivo

Adicionar um arquivo na pasta do seu projeto (Markdown) (md)

git add \* = Comando para adicionar as alterações feitas em todos os arquivos do projeto

git commit -m "" = Comando para fazer um commit do seu projeto (colocar uma mensagem dentro das strings "")

Capitulo 5 - Ciclo de vida dos arquivos no Git

Passo a passo no ciclo de vida

Untracked, Unmodified, Modifies, Staged

Tracked são os arquivos que o GIT já tem ciência deles

Untracked são arquivos que o GIT ainda não tem ciência deles

Modified são os arquivos que foram modificados

Staged são os arquivos que já foram adicionados no projetos e estão esperando o commit

Arquivos dentro do Tracked são Unmodified, Modified e Staged

Unmodified é o arquivo que não sofreu nenhuma alteração

Modified é o arquivo que sofreu alguma alteração

Staged é o conceito chave de onde ficam os arquivos que estão esperando o commit para serem finalizados no projeto

Arquivo Untracked é o arquivo que você acabou de criar e o git ainda não tem ciência dele

O Comando git add adiciona o arquivo para o modo Staged

Quando faz alterações no arquivo, o git confere a chave SHA1 desse arquivo, e caso tenha diferença nessa chave, ela muda de Unmodified para Modified, sendo necessário usar o git add para mudar esse arquivo para o modo Staged

Quando se deleta um arquivo, ele volta para o modo Untracked

Quando o arquivo esta no modo Stage, ele está aguardando o Commit para finalizar o processo virando um arquivo commit

O Comando commit, finaliza todo o processo, transformando os arquivos novamente em Unmodified

Servidor

Remote Repository (Quando você empurra um commit para um repositório)

Ambiente de desenvolvimento

Working Directory (Seu Computador)

Staging Area (Quando criado, modificado ou adicionado um arquivo no Git)

Local Repository (Quando é efetuado um Commit)

Comando git status = Mostra as informações sobre a situação atual do seu repositório

Capitulo 6 - Introdução ao GitHub

Comando para configurar seu perfil no GIT

git config --global --unset user.email (comando para limpar o seu email no perfil)

git config --global --unset user.nickname (comando para limpar o seu user no perfil)

git config --list (comando para ver a lista de configurações)

git config --global user.email "seu email"(comando para adicionar o seu email no perfil)

git config --global user.nickname "seu nome"(comando para adicionar o seu user no perfil)

Criando uma conta no GitHub

Criar uma conta no GitHub - <https://github.com/>

Criar um repositório na Parte Repository / News

Clicar em Creater Repository

Criar ou não um arquivo README

Marcar seu repositório como público ou privado

Colocar uma descrição para o seu repositório

Colocar um nome no seu repositório

Configuração do seu Repositório no GIT

Copiar o caminho html do seu repositório, que o GitHub vai dar e colocar no GIT

git remote add origin (link do seu repositório GitHub) = comando para adicionar seu repositório GitHub a sua maquina local

origin = apelido do seu link repositório

git remote -v = Comando para listar seus repositórios cadastrados

git push origin master = Comando para empurrar seu repositório local para o remoto

Comando para configurar seu perfil no GIT

git config --global --unset user.email (comando para limpar o seu email no perfil)

git config --global --unset user.nickname (comando para limpar o seu user no perfil)

git config --list (comando para ver a lista de configurações)

git config --global user.email "seu email"(comando para adicionar o seu email no perfil)

git config --global user.nickname "seu nome"(comando para adicionar o seu user no perfil)

Criando uma conta no GitHub

Criar uma conta no GitHub - <https://github.com/>

Criar um repositório na Parte Repository / News

Clicar em Creater Repository

Criar ou não um arquivo README

Marcar seu repositório como público ou privado

Colocar uma descrição para o seu repositório

Colocar um nome no seu repositório

Configuração do seu Repositório no GIT

Copiar o caminho html do seu repositório, que o GitHub vai dar e colocar no GIT

git remote add origin (link do seu repositório GitHub) = comando para adicionar seu repositório GitHub a sua maquina local

origin = apelido do seu link repositório

git remote -v = Comando para listar seus repositórios cadastrados

git push origin master = Comando para empurrar seu repositório local para o remoto

Capitulo 7 - Resolvendo conflitos

Como os conflitos acontecem no GitHub e como resolvê-los

git pull origin master = Comando para puxar um conteúdo do seu repositório remoto

Para resolver , assim que você puxar o conteúdo do seu repositório remoto, verifique as modificações que devem ser feitas, faças as devidas alterações e depois faça o passo de subir as suas alterações para o GitHub novamente.

Como clonar um repositório público do GitHub na sua maquina

Entre em algum repositório público e clique no botão CODE

Copie esse link e vá no seu terminal Git e faça o seguintes passos

git clone (link)