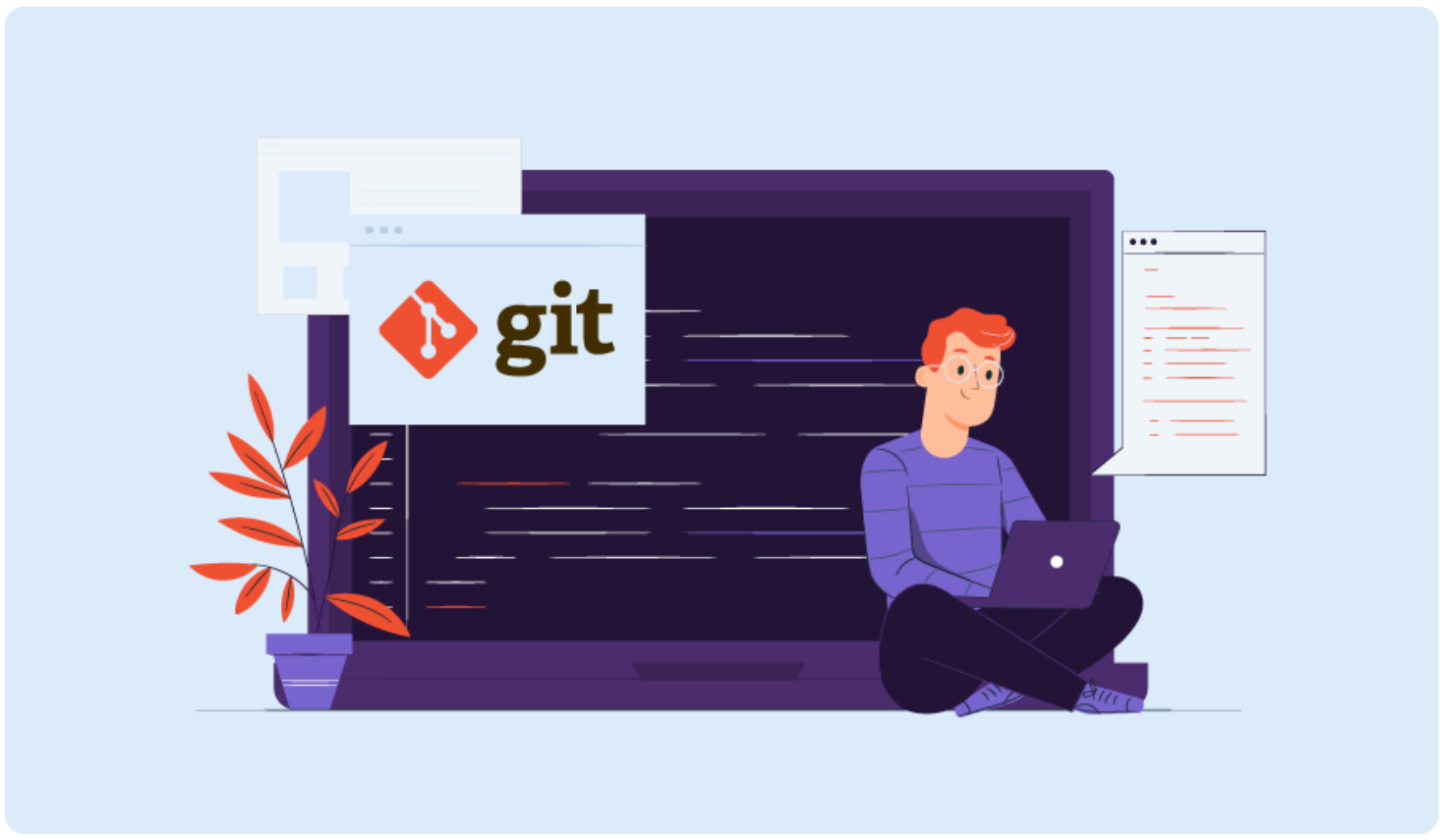
GIT jul 21, 2023 Ariane G. 7min de leitura

GIT Tutorial Para Iniciantes



O objetivo desse GIT tutorial é apresentar tudo o que você precisa para começar a usá-lo.

Primeiro de tudo, entenda que o GIT é um sistema de controle de versão, também conhecido pela sigla VCS – *version control system.*

Os VCS são uma ótima forma de otimizar o trabalho ao analisar as alterações feitas em um código de projeto compartilhado, além de ser um componente fundamental do sistema de gerenciamento de configuração de software que cuida das mudanças que precisam ser feitas em um projeto.

As alterações/revisões/ atualizações feitas são identificáveis através de códigos de letra ou números. Informações como data e a identidade do autor da alteração também são mantidas.

Neste tutorial, vamos focar no GIT, um dos sistemas de controle de versão mais usados no mundo.

Conteúdo

GIT Tutorial Para Iniciantes – O que é GIT?

Em 2005, Linus Torvalds (o homem conhecido por criar o núcleo, ou kernel, do SO Linux) desenvolveu o GIT, que desde então tem sido ativamente mantido por Junio Hamano, um engenheiro de software japonês.

Atualmente, o GIT é um dos mais famosos sistemas de controle de versão de código aberto e milhões de projetos no mundo inteiro o utilizam para seu controle de versão (incluindo projetos comerciais e de código aberto).

O Git é totalmente grátis e pode ser instalado em Mac, Linux, Windows e Solaris diretamente **do site oficial**. Algumas das características essenciais do GIT são:

1. Um sistema de controle de versão distribuído, o GIT segue uma abordagem **peer to peer,** contrário de outros como o Subversion (SVN) que segue um modelo baseado em **cliente-servidor**.
2. GIT permite aos desenvolvedores ter uma infinidade de ramos de código completamente independente. Criação, exclusão e fusão desses ramos é simples e não leva tempo.
3. No GIT, todas as operações são atômicas. Isso significa que uma ação pode ter sucesso ou falhar (sem fazer nenhuma alteração). Isso é importante porque em alguns sistemas de controle de versão (como o CVS) onde as operações não são atômicas, se uma operação de repositório é suspensa, ela pode deixar o repositório em um estado instável.
4. No GIT, tudo é armazenado dentro da pasta .git. Isso não é o mesmo em outros VCS como SVN e CVS onde os metadadados de arquivos são armazenados em pastas ocultas (por exemplo, .cvs, .svn, etc.)
5. GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um *commit* é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.
6. Outra característica presente no GIT é sua **área de teste** ou **índice**. Na área de preparação, os desenvolvedores podem formatar *commits* e receber feedback antes de aplicá-los.

O GIT é consideravelmente simples de usar. Para começar, você pode criar um repositório ou conferir um já existente. Após a instalação, um simples `git-init` irá deixar tudo pronto. Da mesma maneira, o comando `git clone` pode criar uma cópia de um repositório local para um usuário.

1º Passo – Instalar o GIT em Sistemas Diferentes

Nos parágrafos abaixo vamos ensinar a maneira mais simples de instalar o Git em diversos sistemas operacionais:

Instalar o GIT no Windows:

Instalar o GIT no Windows é tão simples como baixar um instalador e executá-lo. Execute os seguintes passos para instalar o GIT no Windows:

1. Acesse **o site oficial** e faça o download do instalador do GIT para Windows.

3. Abra o **prompt de comando** e digite os seguintes comandos no terminal:

```
git config --global user.name "João Silva"
git config --global user.email "exemplo@seuemail.com.br"
```

Importante! Nota: Lembre de substituir **João Silva** e **exemplo@seuemail.com.br** com seus dados. Qualquer *commit* criado posteriormente será associado à esses dados.

GIT instalado no Windows com sucesso!

Instalar o GIT no MacOS:

Existem muitas maneiras de instalar o GIT em um Mac. Inclusive, há uma chance de que o GIT já esteja no seu computador se você tiver o XCode instalado. Execute o seguinte comando no terminal para verificar:

```
git --version
```

Se você obtiver uma resposta como `git version 2.7.0 (Apple Git-66)`, então você está com sorte. Caso você não tenha resposta, execute os seguintes passos:

1. Visite **este site** e faça o download do instalador mais recente para Mac.

3. Tente novamente o comando `git --version` para confirmar se a instalação foi bem sucedida.
4. Execute os seguintes comandos no terminal para configurar seu e-mail e nome de usuário que serão associados à sua conta GIT:

```
git config --global user.name "João Silva"
git config --global user.email "exemplo@seuemail.com.br"
```

Importante! Nota: Lembre-se de substituir **João Silva** e **exemplo@seuemail.com.br** com seus dados. Qualquer *commit* criado posteriormente será associado à esses dados.

Instalar o GIT no Linux:

Se você usa Linux em qualquer sistema (ou num **servidor VPS**), então você deve estar acostumado com instalar programas e pacotes em seu computador usando comandos de instalação `apt-get` ou `yum`. Instalar o GIT não é diferente:

Para usuários Debian/Ubuntu (apt-get):

1. Abra o terminal e execute os seguintes comandos:

```
sudo apt-get update
sudo apt-get install git
```

2. Verifique se a instalação ocorreu com sucesso usando `git --version`.
3. Execute os seguintes comandos no terminal para configurar seu e-mail e nome de usuário que serão associados à sua conta GIT:

```
git config --global user.name "João Silva"
git config --global user.email "exemplo@seuemail.com.br"
```

Fedora (yum/dnf):

Você pode baixar pacotes do GIT usando yum e dnf.

1. Abra o terminal e execute os seguintes comandos:

```
sudo dnf install git
sudo yum install git
```

2. Verifique se a instalação ocorreu com sucesso usando `git --version`.
3. Execute os seguintes comandos no terminal para configurar seu e-mail e nome de usuário que serão associados à sua conta GIT:

```
git config --global user.name "João Silva" git config --global user.email
"exemplo@seuemail.com.br"
```

Importante! Nota: Lembre-se de substituir **João Silva** e **exemplo@seuemail.com.br** com seus dados. Qualquer *commit* criado posteriormente será associado à esses dados.

2º Passo – Como Usar o GIT

Agora que o GIT está instalado e configurado no seu dispositivo, vamos explorar os conceitos básicos do GIT e como começar a usar o GIT.

Criar/configurar/verificar um repositório

Um repositório é o maior bem de qualquer projeto controlado por versão. Para transformar qualquer diretório em um repositório GIT, o simples comando `git init <directory>` pode ser utilizado. Uma pasta chamada .git também deve começar a existir no diretório em que o comando foi executado.

Por outro lado, se você já tem um diretório e deseja verificar (clone-lo), você pode usar o comando git clone. Se você estiver tentando verificar um repositório local, use o seguinte comando:

```
git clone /path/to/local/repository
```

Se você pretende verificar um repositório armazenado remotamente, use:

```
git clone user.name@host:/path/to/remote/repository
```

Se você tem uma conta na Hostinger, você pode facilmente clonar e gerenciar repositórios via **hPanel** → **Hospedagem** → **Avançado** → **GIT**. Por exemplo, se você quer clonar um repositório GIT, basta digitar seu endereço, escolher um ramo e instalar o caminho, e clicar no botão criar.

Após a conclusão da criação, você poderá gerenciar seu repositório na mesma seção.

Fluxo de Trabalho – Tutorial Git

Agora que o repositório está pronto, vamos falar sobre a estrutura que é mantida pelo GIT.

Então, é assim que o fluxo de trabalho pode ser explicado: o usuário adiciona um arquivo ou alterações do diretório de trabalho para o índice (a área de teste) e uma vez revistos, o arquivo ou as alterações são finalmente comprometidos com o **HEAD**.

Os comandos *Add* e *Commit:*

Alterações ou adições de arquivos propostas são adicionadas ao índice usando o comando *add*. Para adicionar qualquer arquivo, o comando é:

```
git add <nome_do_arquivo>
```

Se você está realmente confiante o suficiente para fazer essas mudanças no ***HEAD***, então você pode usar o comando *commit*:

```
git commit –m “Adicionar qualquer mensagem sobre o commit aqui”
```

Importante! Nota: Uma vez que o comando *commit* é executado (a partir do diretório de trabalho), o arquivo fica comprometido com o ***HEAD***, mas ainda não é enviado para o repositório remoto.

Dando Continuidade às Mudanças

Depois de confirmar as alterações (e acreditar que elas estão prontas para serem enviadas para o repositório original), você pode usar o comando push.

Uma vez que o `git push origin master` é executado de dentro do diretório de trabalho, as mudanças presentes no **HEAD** são enviadas para o repositório remoto. No comando acima mencionado, o **master** pode ser alterado para o nome do *branch* ao qual você deseja que as alterações sejam comprometidas.

Se, no entanto, um repositório existente ainda não tiver sido clonado e você pretende estabelecer uma ligação entre o seu repositório e um servidor remoto, execute o seguinte comando:

```
git remote add origin <servidor>
```

Importante! Nota: Substitua pelo endereço do servidor remoto.

Uma vez clonado, quaisquer alterações feitas serão aplicadas para o servidor pertinente.

Branches

Outra característica brilhante (mas avançada) do GIT é sua capacidade de permitir que desenvolvedores e gerentes de projeto criem vários ramos (*branches*) independentes dentro de um único projeto.

O objetivo principal de um *branch* é desenvolver novas funcionalidades, mantendo-os isolados uns dos outros. O *branch* padrão em qualquer projeto é sempre o ***master branch***. Tantos *branches* quanto necessários podem ser criados e eventualmente mesclados ao *master branch*.

Um novo *branch* pode ser criado usando o seguinte comando:

```
git checkout -b feature_n *
```

feature_n é o nome do *branch.*

Se você deseja retornar ao *master* branch, o seguinte comando pode ser usado:

```
git checkout master
```

Para tornar o *branch* disponível para outros usuários, você terá que movê-lo para o repositório remoto. Para fazer isso, use o seguinte comando:

```
git push origin feature_n
```

Atualizando e Dando *Merge*

Caso você queira atualizar seu diretório de trabalho local para o mais recente do repositório remoto, o simples comando `git pull` pode ser usado.

Para mesclar outro branch (dar um *merge*) no atualmente ativo, use: `git merge feature_n`.

Se você der um *merge* ou *pull*, o GIT sempre tenta lidar com os conflitos por conta própria, mas as vezes não consegue. Em caso de falha devido a conflitos, o usuário tem que resolver os conflitos manualmente. Depois de editar os arquivos (para erradicar conflitos), marque-os como *merged* usando:

```
git add <nome.arquivo>
```

Se antes do *merge* você desejar visualizar as alterações, o seguinte comando pode ser executado:

```
git diff <nome_branch_origem> <nome_branch_alvo>
```

Tagging

Antes de lançar atualizações/alterações de software, é sempre recomendado criar tags. Para fazer isso no GIT, use o seguinte comando:

```
git tag 1.1.0 1c2d2d56fa
```

O **1c2d2d56fa** no comando acima refere-se aos primeiros 10 caracteres do **commit-id** que é referenciado com a tag. O ID de *commit* pode ser encontrado no log.

Para recuperar os *commits* feitos por um único usuário, você pode usar:

```
git log --author =Smith
```

Uma versão compactada do log (um *commit* por linha) pode ser visualizada usando:

```
git log --pretty=oneline
```

Para exibir somente os arquivos que foram alterados:

```
git log --name-status
```

Substituindo Alterações Locais

Se você acabou fazendo bagunca e precisa reverter as alterações feitas em qualquer arquivo, faça isso

Tutoriais relacionados

01 mar • GIT

O Que é Bitbucket e Como Usá-lo na Gestão dos seus Projetos Git

Você conhece o Bitbucket? Ele é uma das principais soluções para hospedagem e colaboração em projetos de programação, e conhecer suas...

Por Bruno Santana

05 ago • GIT

Como Renomear um Branch Local ou Remoto do Git – Um Guia Rápido

Quando você está trabalhando no Git, às vezes acaba nomeando acidentalmente um branch do jeito errado, ou simplesmente quer que seu projeto fique...

18 jun • GIT

Os Melhores Clientes Git GUI de 2023 para Windows, Linux e Mac

Quase todos os projetos de desenvolvimento de software, sejam eles comerciais ou pessoais, estão usando o Git para gerenciamento. Neste artigo, vamos...

Por Carlos E.

Comentários

Faça um comentário

Jose Jesus
maio 28 2019

RESPONDER

Bom tutorial. Tentaria fazer o pedido de outro tutorial acerca do rebase vantagens e perigos. Obrigado

André
abril 11 2021

RESPONDER

Carlos E.
abril 16 2021

Olá, André! O comando Git Checkout é usado para trocar a branch em que estamos trabalhando, enquanto o "git branch -d" é o comando para deletar mesmo! :D

Flávio Lucas
dezembro 14 2021

RESPONDER

O comando sudo está escrito com captularização

Carlos E.
janeiro 07 2022

Opa, obrigado pela dica Flávio! Já arrumei aqui as strings!?

Osvaldo Vargas Jaques
abril 27 2022

RESPONDER

Talvez eu ajude com algumas observações que sofri para aprender: (1) Em git remote add origin, que associa o local ao . Ok, mas acho que temos que deixar claro que 'origin' é um nome padrão que damos ao . Isto é , poderia ser git remote add qualquerNome (2) Em git pull origin master, que baixa as atualizações de origin (ou qualquerNome) do ramo principal NOMEADO POR PADRÃO "master". Atualmente, o github está usando o nome padrão "main". Portanto, atualmente no github a coisa ficaria git pull origin main. Ou, no caso (1), poderia ser git pull qualquerNome main

Carlos E.
maio 09 2022

Obrigado pelas observações, Osvaldo! É muito legal ver a comunidade contribuindo com nossos artigos e compartilhando conhecimento ?

Deixe uma resposta

Comentário*

Nome*

Email*

Enviar

HOSTINGER

Somos uma provedora de hospedagem de sites e nossa missão é promover o sucesso a todos que querem começar sua jornada online. Para isso, buscamos sempre melhorar a tecnologia dos nossos servidores, oferecer um suporte profissional e tornar a experiência dos nossos clientes com a hospedagem de sites a melhor de todas.

E Mais

HOSPEDAGEM

Hospedagem de Site

Hospedagem de Sites Profissional

Servidor VPS

Host Minecraft

VPS CyberPanel

Hospedagem Cloud

Cheap WordPress Hosting

Email Profissional

Hospedagem CMS

Hospedagem de Loja Virtual

Hospedagem Grátis

Loja Virtual

Criador de Sites

Criador de Logo

Gerador de Nomes para Empresas

DOMÍNIOS

Registro de Domínio

Transferir Domínio

Domínio Grátis

Domínio .xyz

Domínio Barato

Extensões de Domínios

WHOIS

Certificado SSL Gratuito

AJUDA

Tutoriais

Base de Conhecimento

Denunciar Abuso

Migração de Site

Status do Sistema

Programa de Afiliados

Formas de Pagamento

Prêmio

Opiniões de Clientes

Preços

Mapa do Site

SOBRE NÓS

Sobre a Hostinger

Tecnologias Hostinger

Carreiras

Roadmap (em inglês)

Fale Conosco

Blog

LEGAL

Política de Privacidade

Termos de Serviço