ESCOLA LINUX

TREINAMENTOS

A Escola Linux preparou para você esse sensacional guia rápido do GIT para que você inicie e tenha poucas dificuldades com essa plataforma de controle de versões.
Uma das coisas mais interessantes sobre o GIT é que ele foi criado pelo próprio Linus Torvalds.
Aproveite o nosso guia e mergulhe de cabeça nesse software maravilhoso.

Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo.

DOWNLOAD

Windows

Linux

OSX

Criar um novo repositório

Crie um novo diretório, abra-o e execute

```
git init
```

Para criar um novo repositório

Criar um novo repositório

Crie uma cópia de trabalho de um repositório local executando o comando:

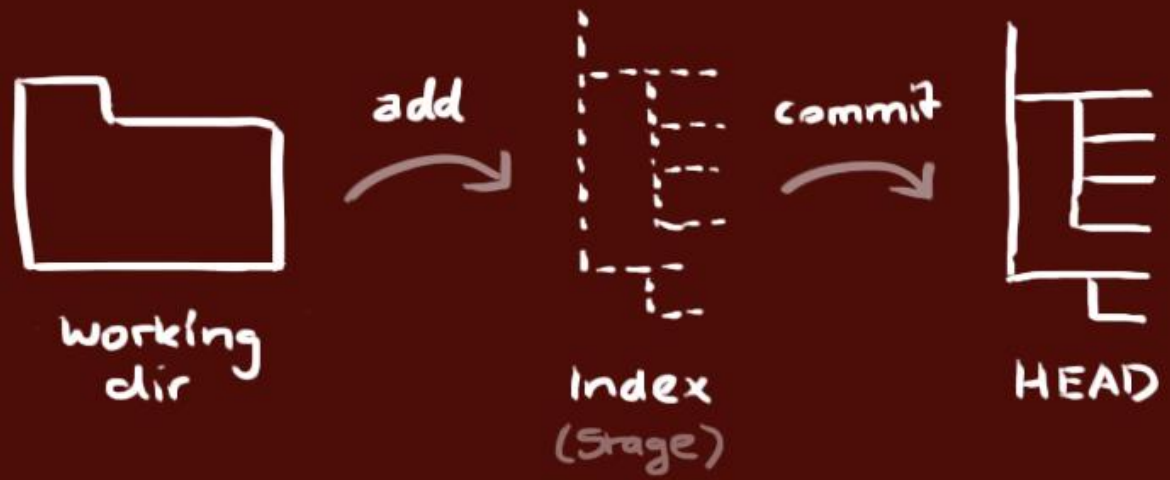
```
git clone /path/to/repository
```

Ao usar um servidor remoto, seu comando será:

```
git clone username@host:/path/to/repository
```

Fluxo de trabalho

Seu repositório local consiste em três "árvores" mantidas pelo git. o primeiro é o seu diretório de trabalho, que contém os arquivos reais. o segundo é o Index que atua como uma área de teste e finalmente o HEAD que aponta para o último commit que você fez.



add e commit

Você pode propor mudanças (adicioná-lo ao Índice) usando:

```
git add <filename>
```

```
git add *
```

Esta é a primeira etapa no fluxo de trabalho básico do git.
Para realmente fazer essas mudanças, use:

```
git commit -m "Commit message"
```

Agora o arquivo está vinculado com o HEAD.

Empurrando mudanças

Suas alterações estão agora no HEAD de sua cópia de trabalho local. Para enviar essas alterações para o seu repositório remoto, execute:

```
git push origin master
```

Mude o mestre para qualquer branch para o qual deseja enviar suas alterações.

Empurrando mudanças

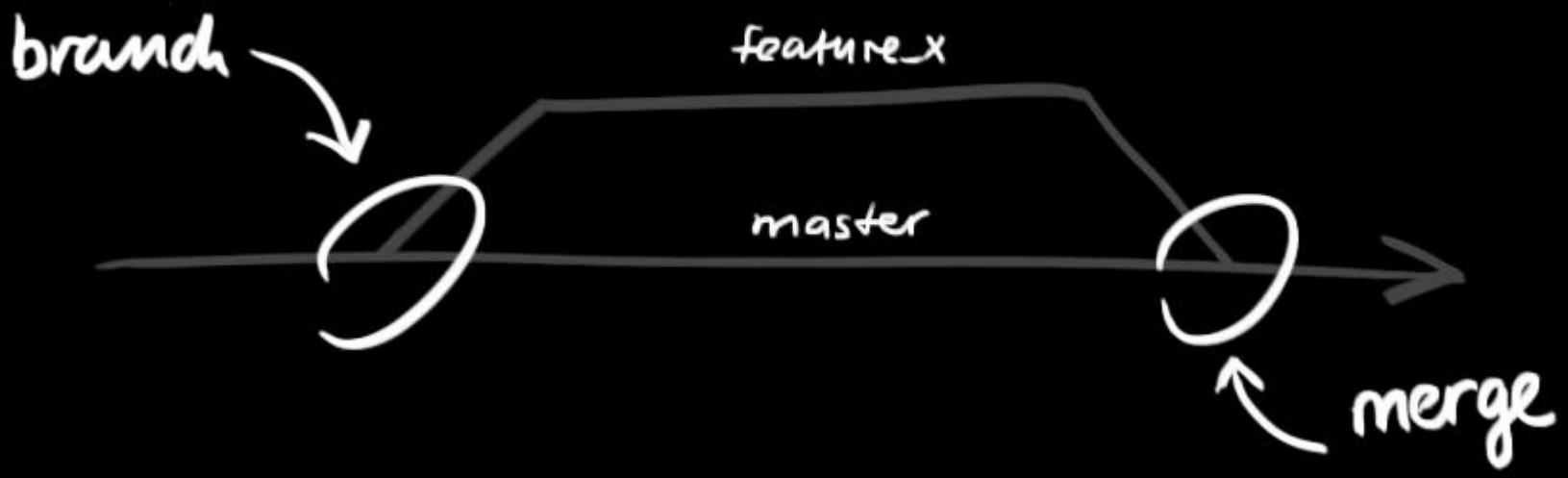
Se você não clonou um repositório existente e deseja conectar seu repositório a um servidor remoto, você precisa adicioná-lo com:

git remote add origin <server>

Agora você pode enviar suas alterações para o servidor remoto

Ramificação

Desenvolva recursos isolados uns dos outros com ramificações.
O branch master é o branch "padrão" quando você cria um repositório.
Crie branches para desenvolvimento e mescle-os de volta ao branch master após a conclusão.



Ramificação

Crie um novo branch chamado "feature_x" e mude para ele usando:

```
git checkout -b feature_x
```

voltar ao mestre

```
git checkout master
```

e exclua o ramo novamente

```
git branch -d feature_x
```

Ramificação

Um branch não está disponível para outros a menos que você envie o branch para seu repositório remoto:

```
git push origin <branch>
```

update & merge

Atualize seu repositório local para o commit mais recente, execute:

```
git pull
```

Em seu diretório de trabalho busque e mescle alterações remotas.
Para fundir outro branch em seu branch ativo use:

```
git merge <branch>
```

update & merge

O git tenta mesclar automaticamente as alterações.
Infelizmente, nem sempre isso é possível e resulta em conflitos.
Você precisa mesclar esses conflitos manualmente, editando
os arquivos mostrados pelo git. Marque eles como mesclados com:

```
git add <filename>
```

Antes de mesclar as alterações, você pode visualizá-las usando:

```
git diff <source_branch> <target_branch>
```

Etiquetagem

E recomendado criar tags para versões de software.
Este é um conceito conhecido, que também existe no SVN.
Você pode criar uma nova tag chamada 1.0.0 executando:

```
git tag 1.0.0 1b2e1d63ff
```

O 1b2e1d63ff representa os primeiros 10 caracteres do ID do commit que você deseja referenciar com sua tag.

log

Você pode estudar a história do repositório usando:

```
git log
```

Você pode adicionar vários parâmetros para customizar o log.
Para ver apenas os commits de um determinado autor:

```
git log --author=bob
```

log

Para ver um registro muito compactado em que cada confirmação é uma linha:

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

Ou talvez você queira ver uma árvore de arte ASCII de todas as ramificações, decorada com os nomes das tags e ramificações:

```
git log --graph --oneline --decorate --all
```

log

Veja apenas quais arquivos foram alterados:

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

Esses são apenas alguns dos parâmetros possíveis que você pode usar.
Para mais, veja:

```
git log --help
```

Substituir as mudanças locais

Caso você tenha feito algo errado, você pode substituir as alterações locais usando o comando:

```
git checkout -- <filename>
```

Isso substitui as mudanças em sua árvore de trabalho pelo último conteúdo do HEAD. As alterações já adicionadas ao índice, assim como os novos arquivos, serão mantidas.

```
git log --help
```

Substituir as mudanças locais

Se você deseja descartar todas as suas alterações e commits locais, pegue o histórico mais recente do servidor e aponte seu branch master local assim:

```
git fetch origin
```

```
git reset --hard origin/master
```

Dicas úteis

GUI git integrado

```
gitk
```

Usar saída git colorida

```
git config color.ui true
```

Mostrar log em apenas uma linha por confirmação

```
git config format.pretty oneline
```

ESCOLA LINUX

TREINAMENTOS

Visite nosso site: www.escolalinux.com.br