



# Git

Professor: Luciano Kelvin



## O que é GIT ?

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.



## Quem criou ?

Em 2005, Linus Torvalds (o homem conhecido por criar o kernel Linux OS) desenvolveu o GIT e, desde então, tem sido ativamente mantido por Junio Hamano, um engenheiro de software japonês;

# Modos de utilizar

## Interface Gráfica

The screenshot displays the GitHub Desktop application interface. At the top, the 'Current Repository' is 'desktop', and the 'Current Branch' is 'esc-pr' with a commit hash of '#3972'. A 'Fetch origin' button indicates the last fetch was 3 minutes ago. The main area is split into two panes. The left pane shows a 'History' tab with a list of commits, including 'Add event handler to dropdown component' by iAmWillShepherd and Markus Olsson, and 'Move escape behavior to correct co...' by the same authors. The right pane shows a code diff for the file 'app/src/ui/t.../dropdown.tsx'. The diff highlights changes in the 'ToolbarDropdown' component, including the addition of a 'private get isOpen()' method and a 'private dropdownIcon' method. The code is shown with line numbers and color-coded changes (green for additions, red for deletions).

Current Repository: desktop

Current Branch: esc-pr #3972

Fetch origin: Last fetched 3 minutes ago

Changes: History

Appense linter

iAmWillShepherd committed a day ago

Add event handler to dropdown component

iAmWillShepherd and Markus Olsson committed a day ago

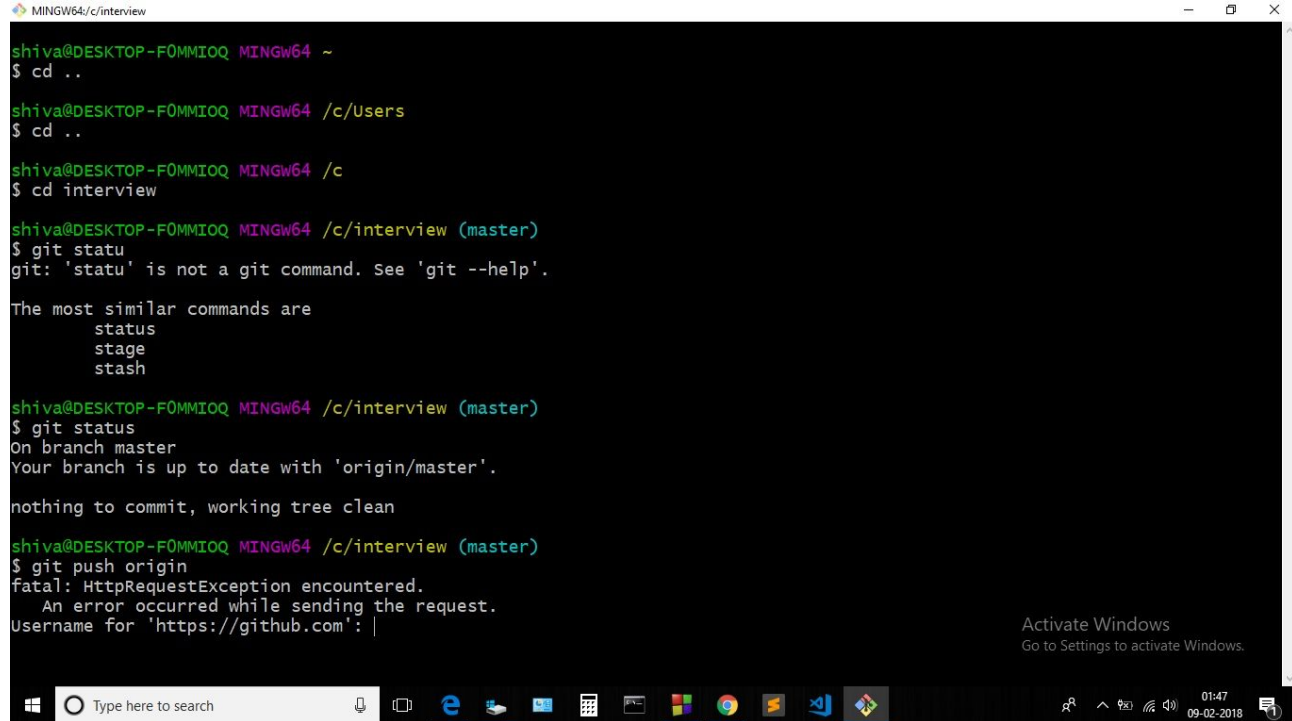
Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>

app/src/ui/t.../dropdown.tsx

```
@@ -145,6 +145,10 @@ export class ToolbarDropdown extends
  React.Component<
    145         this.state = { clientRect: null }
    146     }
    147
    148 +     private get isOpen() {
    149 +         return this.props.dropdownState === 'open'
    150 +     }
    151 +
    148     private dropdownIcon(state: DropdownState): Octicon
    149     bol {
    150         // @TODO: Remake triangle octicon in a 12px version
    151         // right now it's scaled badly on normal dpi monitors.
    152
    153         @@ -249,6 +253,13 @@ export class ToolbarDropdown extends
    154         React.Component<
    155             }
    156         }
    157
    158 +     private onFoldoutKeyDown = (event:
    159 +         React.KeyboardEvent<HTMLElement>) => {
    160 +         if (!event.defaultPrevented && this.isOpen &&
    161 +             event.key === 'Escape') {
    162 +             event.preventDefault()
    163         }
```

# Modos de utilizar

## Linha de comando



```
MINGW64/c:/interview

shiva@DESKTOP-F0MMIOQ MINGW64 ~
$ cd ..

shiva@DESKTOP-F0MMIOQ MINGW64 /c/Users
$ cd ..

shiva@DESKTOP-F0MMIOQ MINGW64 /c
$ cd interview

shiva@DESKTOP-F0MMIOQ MINGW64 /c/interview (master)
$ git statu
git: 'statu' is not a git command. See 'git --help'.

The most similar commands are
    status
    stage
    stash

shiva@DESKTOP-F0MMIOQ MINGW64 /c/interview (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

shiva@DESKTOP-F0MMIOQ MINGW64 /c/interview (master)
$ git push origin
fatal: HttpRequestException encountered.
An error occurred while sending the request.
Username for 'https://github.com': |
```

Activate Windows  
Go to Settings to activate Windows.

Type here to search

01:47  
09-02-2018



## Criando conta no github

- Criar uma conta no site <https://github.com/>
- Criar um repositório



## Faça download do github

- Download github GUI - <https://desktop.github.com/>
- Download github Bash - <https://gitforwindows.org/>



## Como iniciar ?

GIT é consideravelmente simples de usar. Para começar, você pode criar um repositório ou fazer check-out de um existente. Pós-instalação, um simples **git-init** irá levá-lo a todos os set up; Pelo contrário, o **git clone** pode configurar uma cópia de trabalho do repositório local para um usuário.





## Para configurar o usuário no git


- `git config --global user.name "John Smith"`
- `git config --global user.email "example@email.com"`



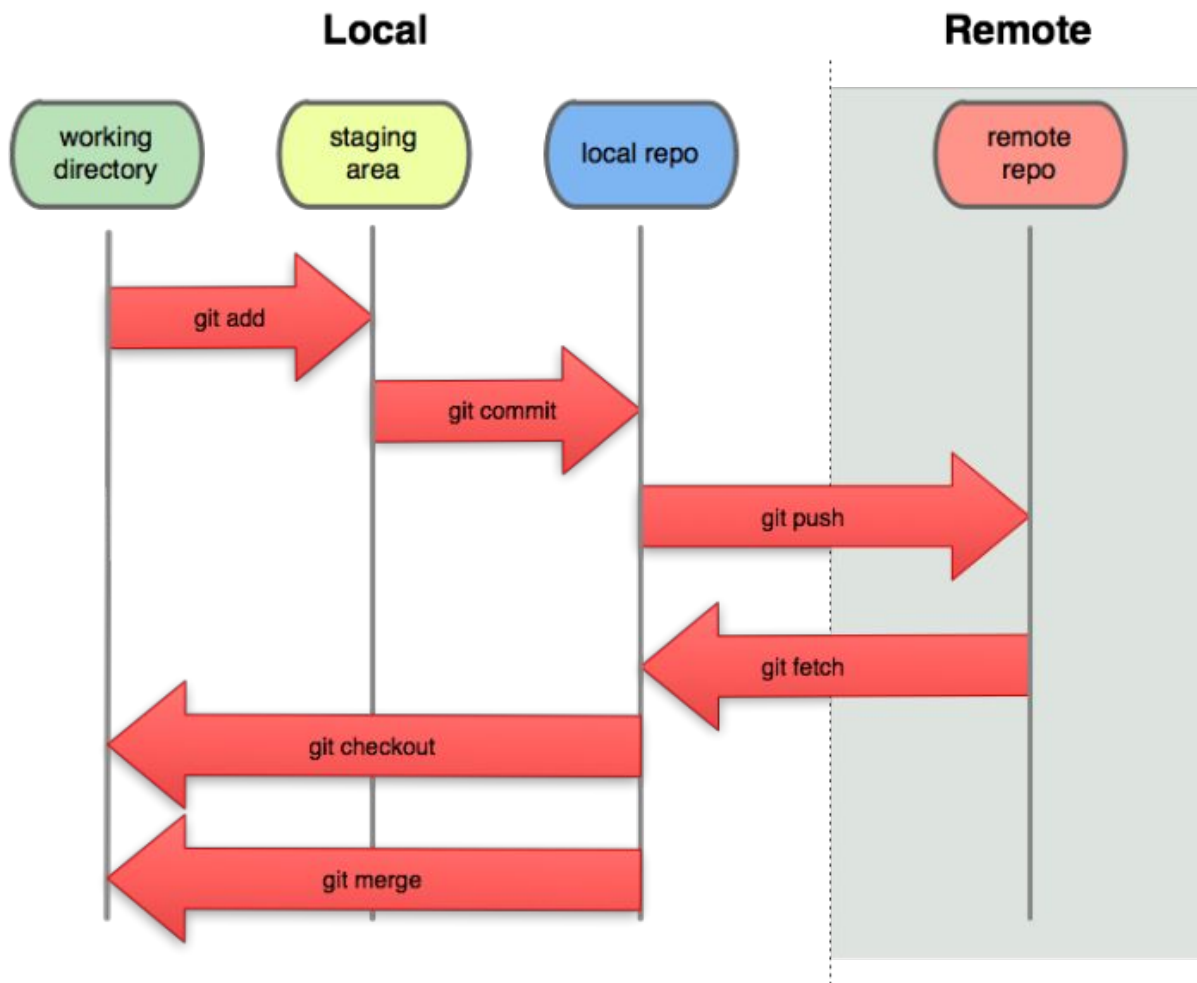
**Para criar seu repositório.**

Se ele já existe:

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



Cada repositório local consiste em três árvores: o **diretório de trabalho** que contém os arquivos reais; O **índice** que desempenha o papel de uma área de teste e o **HEAD** que é um ponteiro para o último comentário feito pelo usuário. 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**.






## Comandos

Para enviar as alterações e novos arquivos:

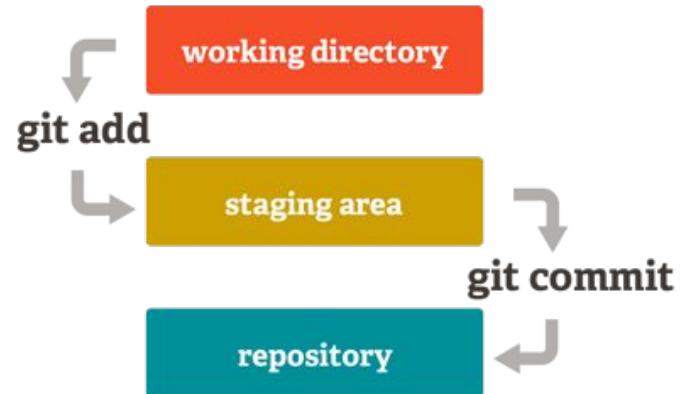
*git add <file\_name>*

Exemplo:

- `git add teste.java`
- `git add .`



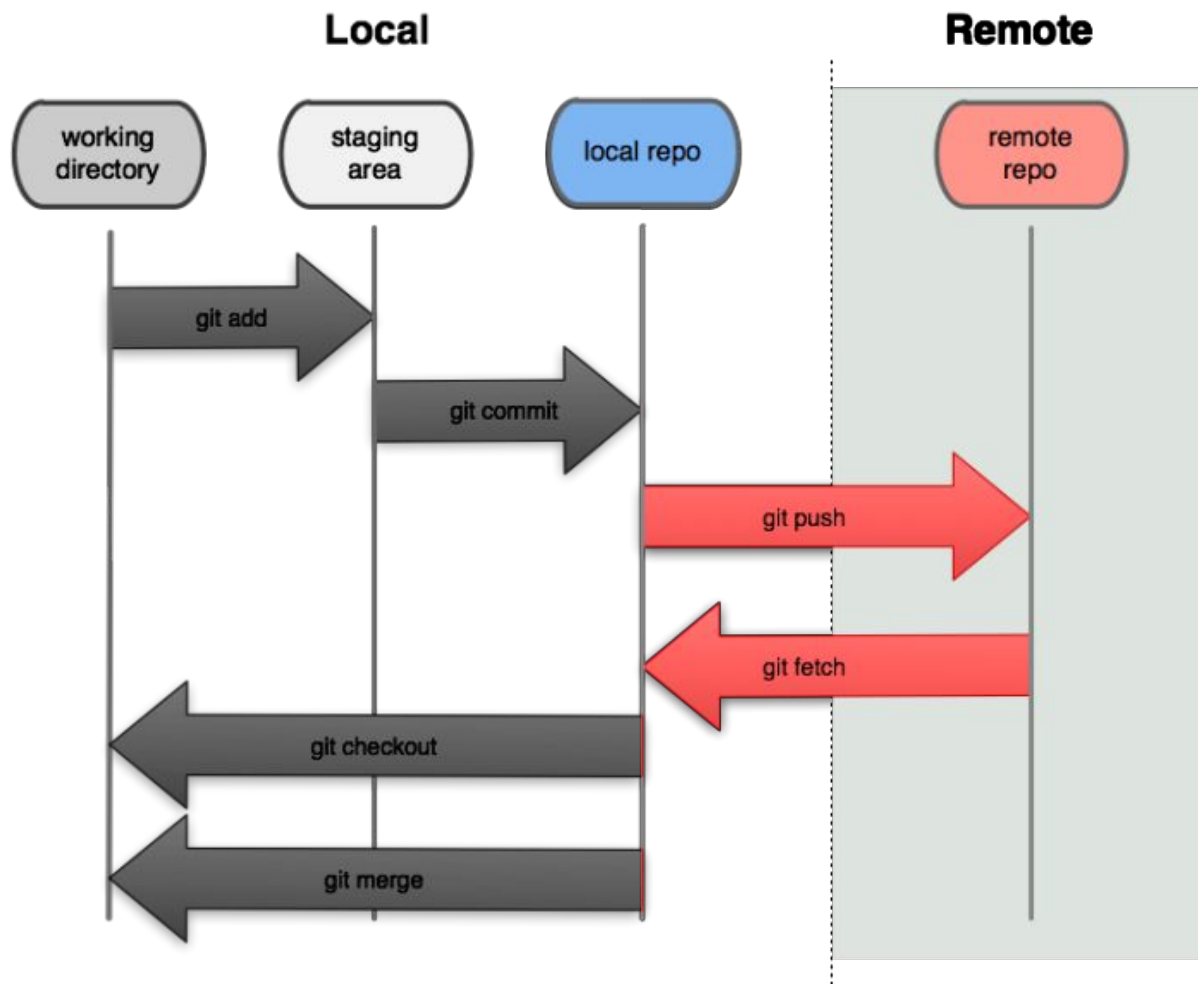
```
git commit -m "Adicionar qualquer comentário aqui"
```






# git push

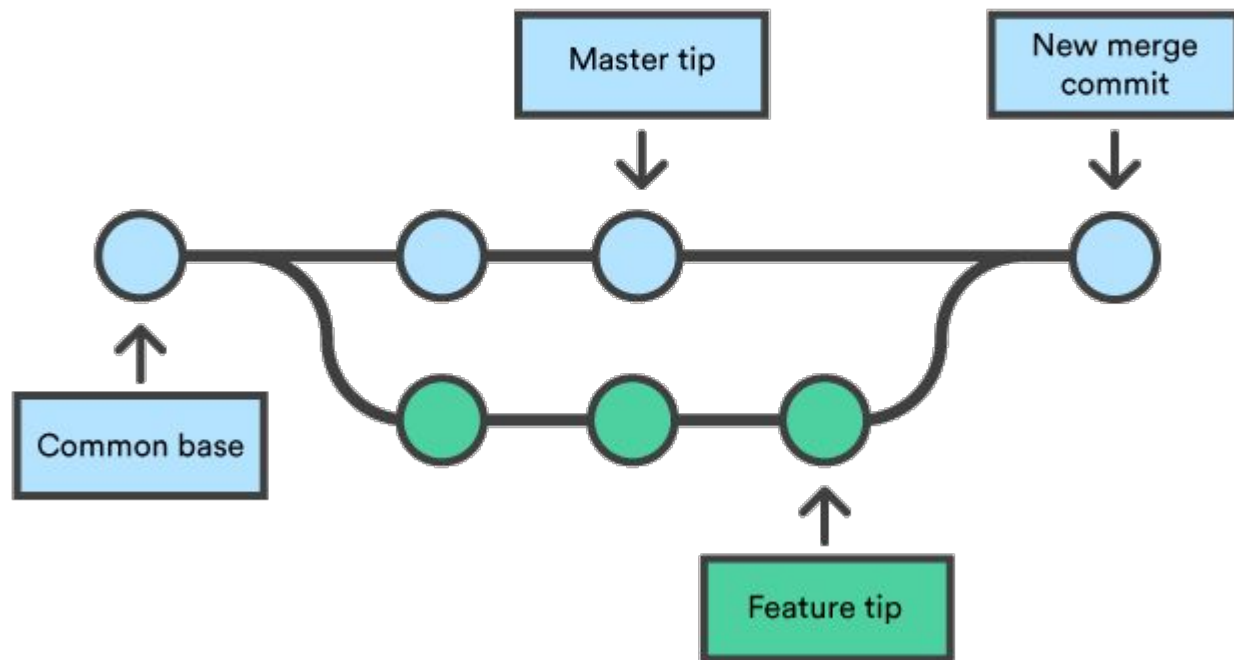
git push origin master







Outra característica brilhante (mas avançada) do GIT é sua capacidade de permitir que desenvolvedores e gerentes de projeto criem vários ramos independentes dentro de um único projeto. O objetivo principal de um ramo é desenvolver recursos, mantendo-os isolados uns dos outros. O ramo padrão em qualquer projeto é sempre o ramo mestre.





Um novo ramo pode ser criado usando o seguinte comando:

```
git checkout -b akdbfsdf
```

**feature** é o nome do novo ramo.



## Enviando para o repositório

- `git push origin feature`
- `git push origin master`



## Fazer o merge da Branch na Master

```
git merge branchName
```

# Tarefa



- Crie um repositório
- Clone na sua máquina
- Adicione dois arquivos de texto
- Suba esses arquivos para o repositório
- Crie uma nova branch
- Mude o repositório para a branch
- Faça modificações nesses arquivos na nova branch
- Suba as modificações para a nova branch