



OCEAN

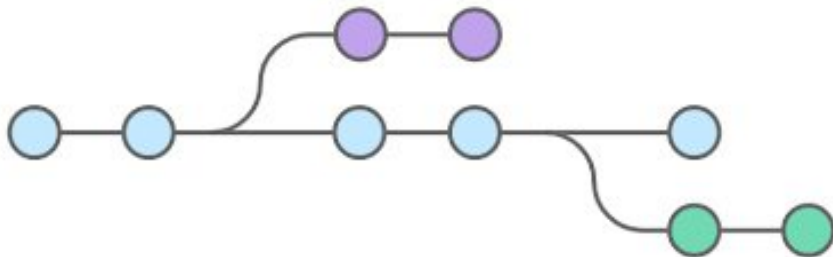


git



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





Modos de utilizar

- Interface gráfica
- Linha de comandos

Interface Gráfica

Current Repository
desktop

Current Branch
esc-pr #3972

Fetch origin
Last fetched 3 minutes ago

Changes

History

Appease linter
iAmWillShepherd committed a day ago

Add event handler to dropdown component
iAmWillShepherd and Markus Olsson...
Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>

Move escape behavior to correct co...
iAmWillShepherd and Markus Olsson...

Remove event handler from the bran...
iAmWillShepherd and Markus Olsson...

Merge branch 'master' into esc-pr
iAmWillShepherd committed a day ago

Merge pull request #4044 from des...
Neha Batra committed a day ago

Merge pull request #4070 from desk..
Brendan Forster committed 2 days ago

bump to beta3
Brendan Forster committed 2 days ago

Merge pull request #4057 from desk..
Brendan Forster committed 2 days ago

Merge pull request #4067 from desk..
Brendan Forster committed 2 days ago

Release to 1.1.0-beta2
Neha Batra committed 2 days ago

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

145	145	@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
146	146	this.state = { clientRect: null }
147	147	}
	148	+ private get isOpen() {
	149	+ return this.props.dropdownState === 'open'
	150	+ }
	151	+ }
148	152	private dropdownIcon(state: DropdownState): OcticonSym
149	153	bol {
150	154	// @TODO: Remake triangle octicon in a 12px version,
		// right now it's scaled badly on normal dpi monitor
		s.
		@@ -249,6 +253,13 @@ export class ToolbarDropdown extends
		React.Component<
249	253	}
250	254	}
251	255	}
	256	+ private onFoldoutKeyDown = (event:
		React.KeyboardEvent<HTMLInputElement>) => {
	257	+ if (!event.defaultPrevented && this.isOpen &&
		event.key === 'Escape') {
	258	+ event.preventDefault()
	259	+ this.props.onDropdownStateChanged('closed', 'keybo

Interface Gráfica

The screenshot displays the NodeGit GUI interface, which is used for managing Git repositories. The interface is divided into several sections:

- Left Panel:** Shows the repository structure. It includes a search bar, a list of branches (LOCAL and REMOTE), and a list of pull requests and tags. The current branch is 'feature-a'.
- Commit History:** A central vertical timeline showing the sequence of commits. Each commit is represented by a colored circle (green for master, purple for feature-a) and a commit message. The history is filtered by the selected branch.
- Diff View:** A table on the right side of the commit history showing the changes made in each commit. The table has columns for the commit message, the author, and the date. The changes are categorized by type: added (green), deleted (red), and modified (yellow).
- Right Panel:** Contains a search bar for commits, a section for adding a README, and a section for viewing the diff of the selected commit. The diff shows the changes to the file 'test.txt'.

The interface is designed to be user-friendly and provides a visual representation of the repository's state and history.

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': |
```

**Vamos utilizar linha de
comando!**





Faça download do git bash do github

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



Comandos básicos

- **cd** - abrir uma pasta
- **git init** - transforma a pasta em um projeto git
- **git add** - rastreia os arquivos que serão comitados
- **git commit** - salva todas as alterações feitas até aquele momento
- **git push** - envia as modificações para o servidor



Comando “cd”

- `cd /pasta_do_projeto`
- `cd ..`
- `cd diretorio/subdoretorio/projeto`



Para configurar o usuário do git

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



Git init

Todo comando git deve ser executado dentro de um repositório git.

Para transformar um diretório comum em um repositório git, temos que utilizar o comando “**git init**”



Git add

Rastreia os arquivos que deverão ser comitados

git add <file_name>

- git add teste.java
- git add *.java
- git add .



Local

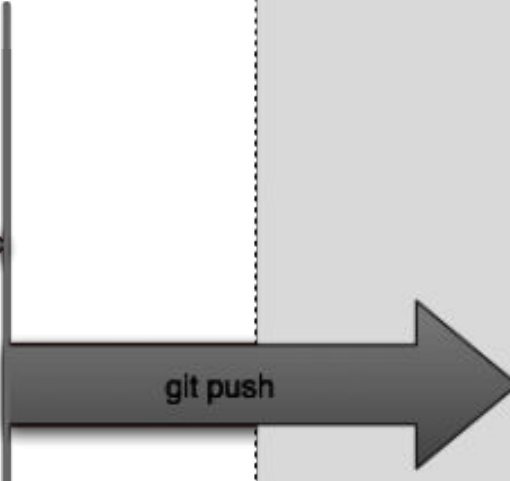
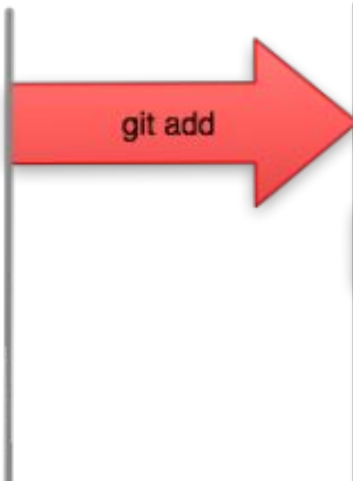
Remote

working
directory

staging
area

local repo

remote
repo



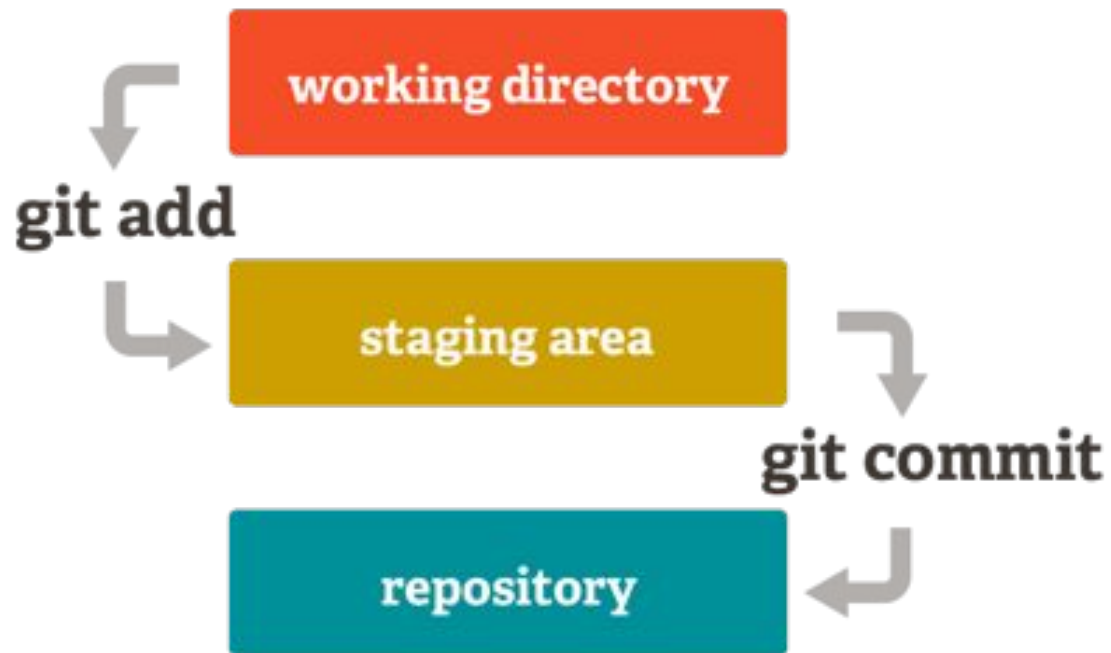


Git commit

Salva todos os arquivos mapeados até aquele momento.

Exemplos

- `git commit -m "Primeiro commit"`
- `git commit -m "Criação da classe calculadora"`
- `git commit -m "Implementação dos testes para calculadora"`





Git push

Salva todas as modificações no repositório remoto

Exemplos

- `git push origin master`



Local

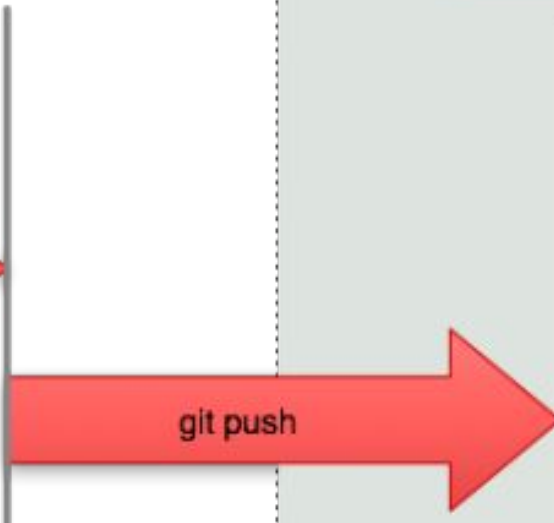
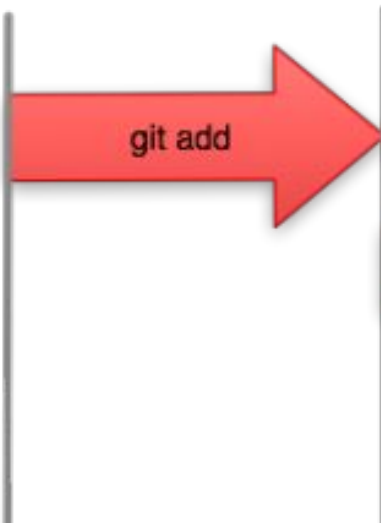
Remote

working
directory

staging
area

local repo

remote
repo



Repositório remoto



GitLab



Criando conta no github

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



Repositório remoto

Para clonar o repositório remoto para o seu pc:

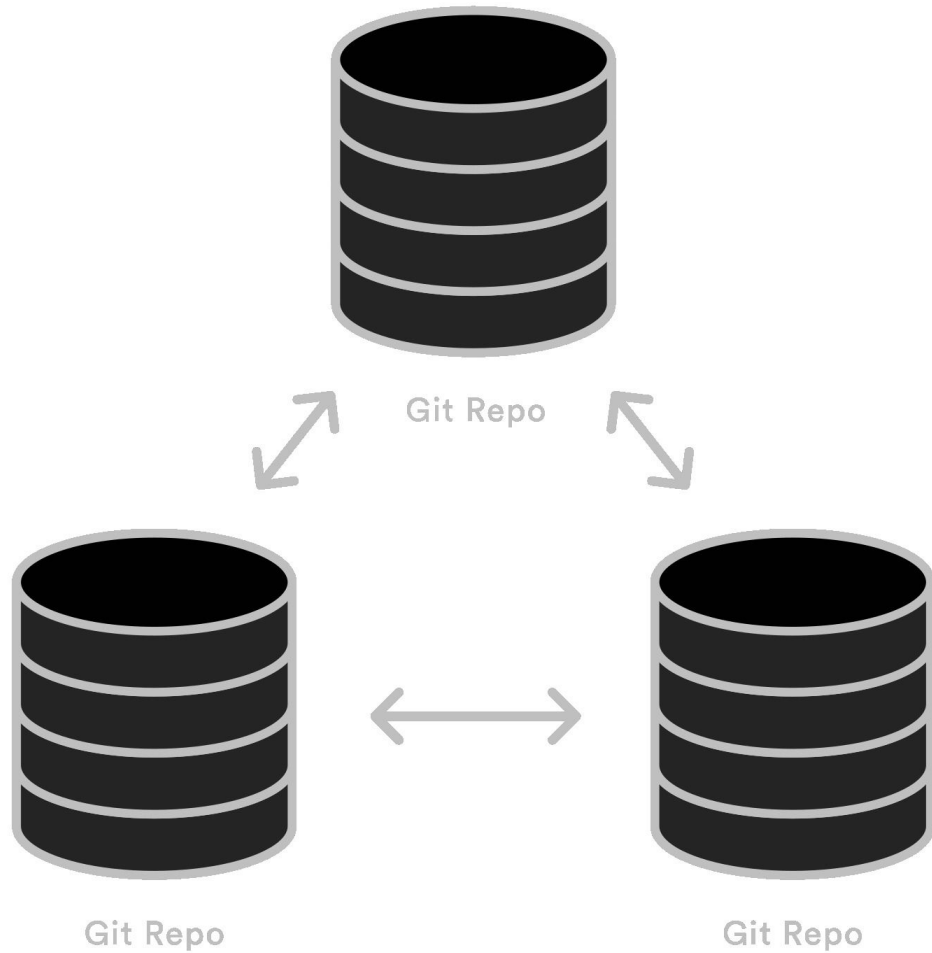
- `git clone user.name@host:/path/to/remote/repository`

Para vincular um projeto git já existente com o repositório remoto:

- `git remote add origin https://github.com/projeto.git`

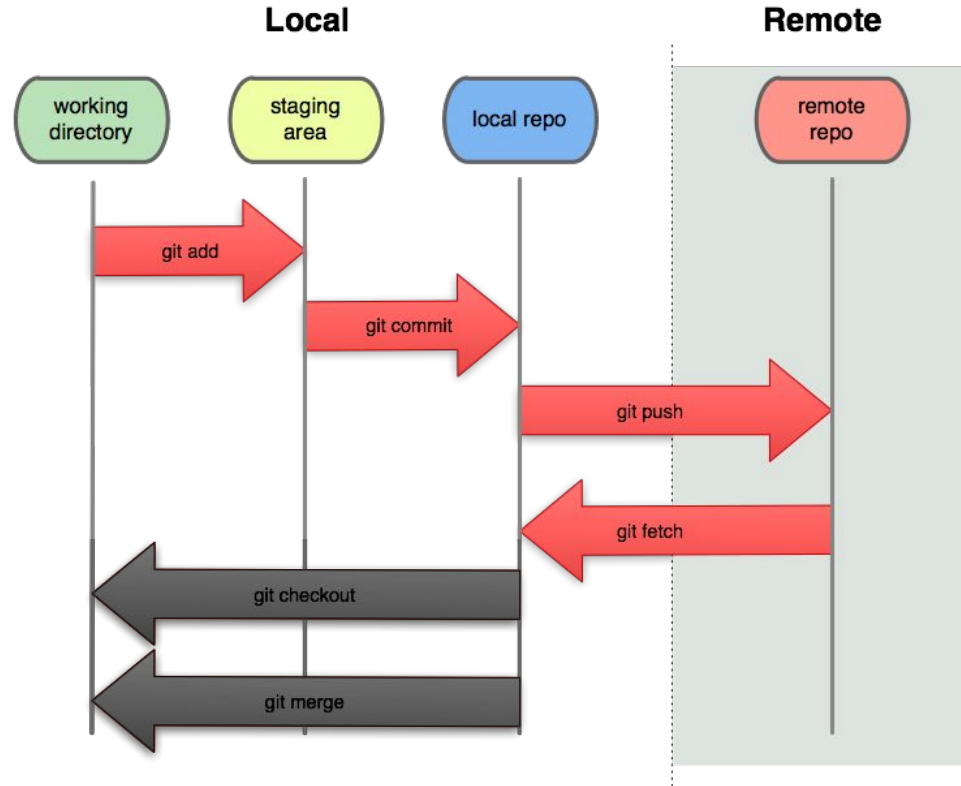


Git clone



Git fetch

Busque ramificações de um ou mais outros repositórios, juntamente com os objetos necessários para completar seus históricos. As ramificações de rastreamento remoto são atualizadas.





Git checkout

Para alternar entre as versões existentes

git checkout <commit>

- **git checkout** 0ee5959be455eddbd90f5217be75ac79a3e60954



Git pull

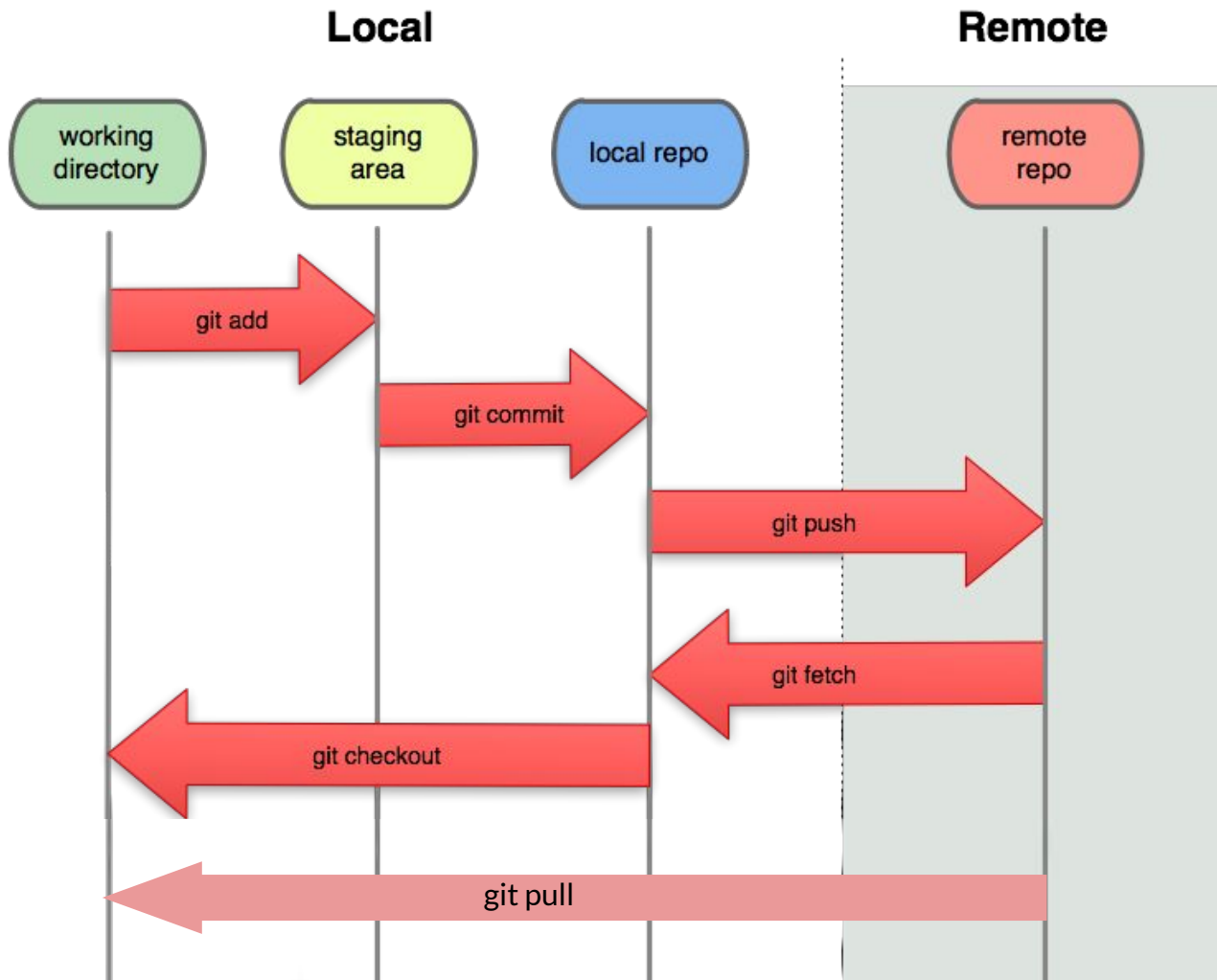


Traz as mudanças do repositório remoto para o local.

Exemplos:

- `git pull`

Git pull

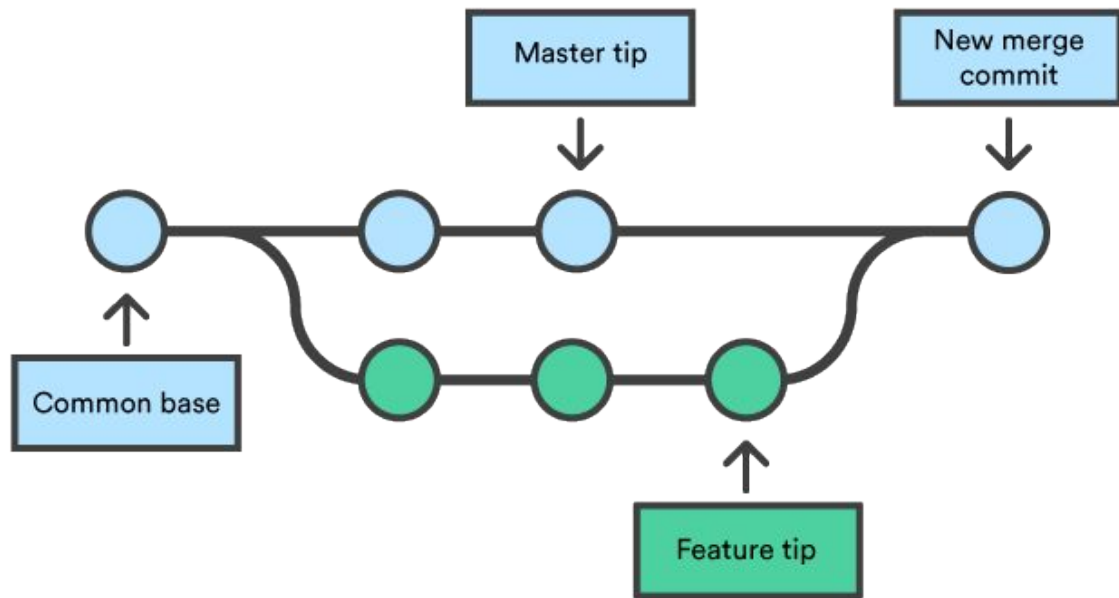




Branches

Outra característica 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.





Criando uma nova branch

Um novo ramo pode ser criado usando o seguinte comando:

```
git checkout -b nome_da_branch
```



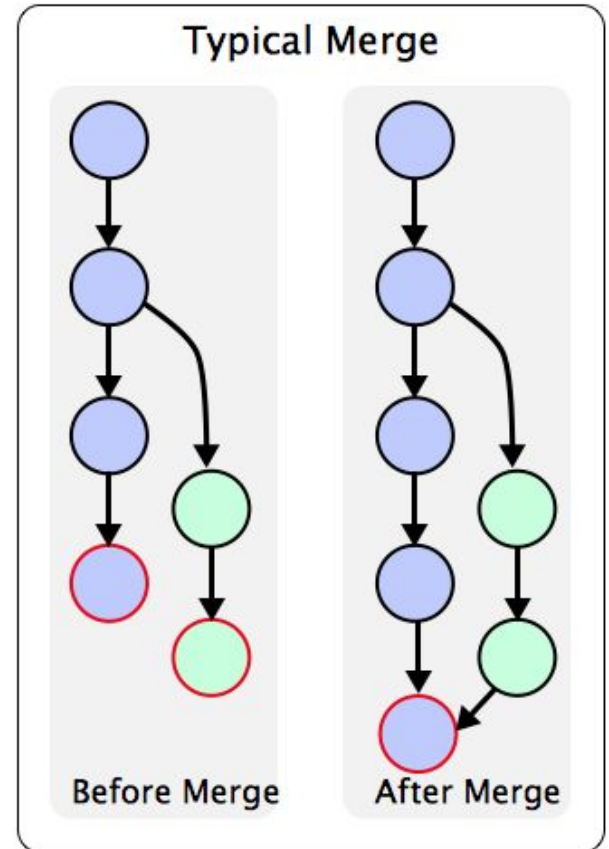
Enviando branch para o repositório remoto

- `git push origin featureName`
- `git push origin master`

Merge da Branch na Master

O merge serve para juntar duas branches

`git merge <nome_da_branch>`





Git checkout

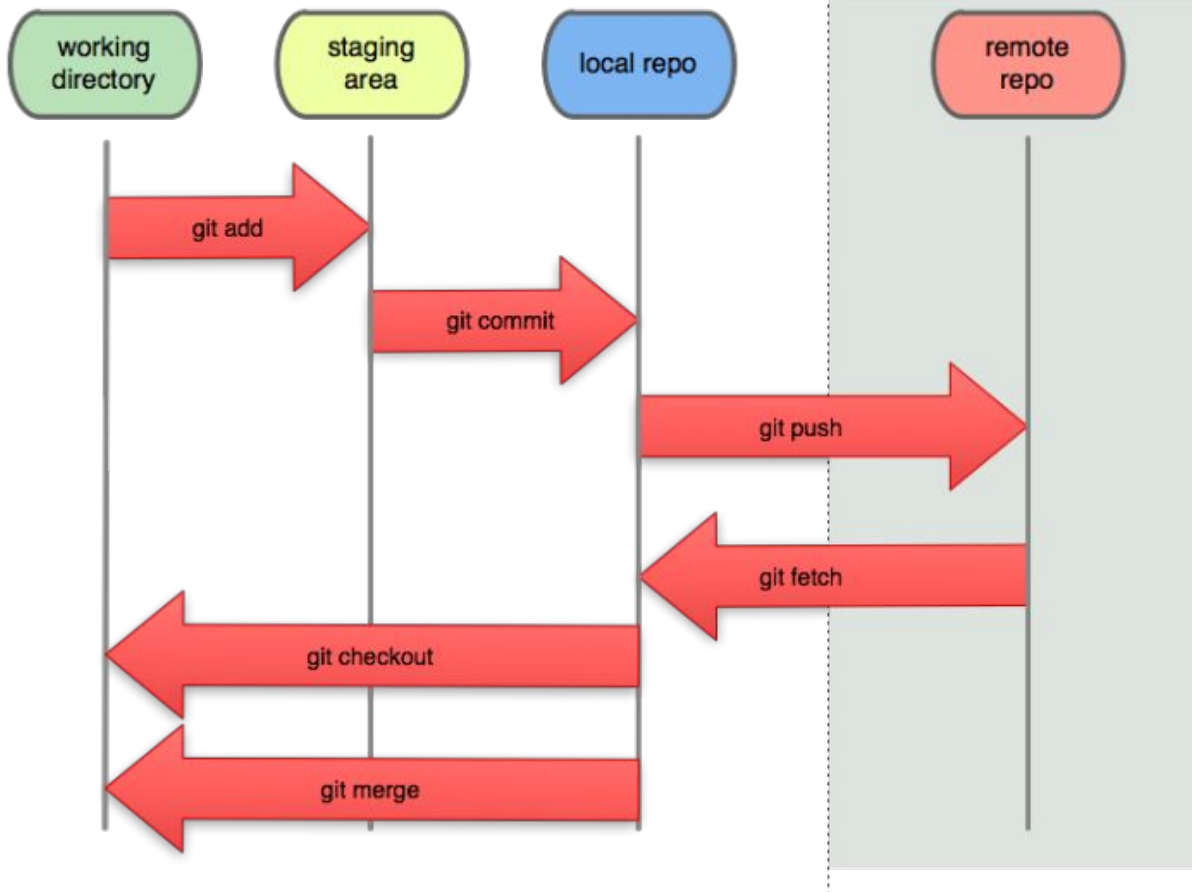
Para alternar entre os commits ou entre as branches utilizamos o comando “git checkout”

- `git checkout nome_da_branch`



Local

Remote





Tarefa 1

- Crie um repositório online no github
- Clone esse repositório na sua máquina
- Adicione um arquivo de texto
- Faça um commit
- Adicione um arquivo de imagem nesse diretório
- Faça outro commit
- Faça um push para salvar as modificações no repositório.



Tarefa 2

- Crie uma nova branch nesse repositório
- Adicione três imagens no diretório
- Faça um commit
- Suba as modificações para a nova branch
- Faça o merge da master com essa nova branch



That's all Folks!