

Git

Versionamento

Pra que serve?

- Controlar histórico
- Trabalhar em equipe
- Marcação de versões
- Resgate de versões
- Ramificação de projeto
- Rastreabilidade



Instalação

Git

○ <https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Instalando-o-Git>



Faremos uso também do **Sourcetree** para visualização porém não é obrigatório

Primeiras Configurações

Verificando

- `git config --list --global # ~/.gitconfig`
- `git config --list --local # .git/config`
- `git config --list --system # /etc/gitconfig`

Nome

- `git config --global user.name "[NomeSobrenome]"`

Email

- `git config --global user.email "[nome.sobrenome@email.com]"`



Criação de Projeto

Iniciar versionamento a partir da pasta atual

☐ git init



Perceba a pasta .git criada

Nome	Data de modificação	Tipo	Tamanho
 .git	26/05/2021 18:22	Pasta de arquivos	
 Git.pptx	26/05/2021 18:21	Microsoft PowerP...	691 KB

Status

Verificar Status

○ git status

```
$ git status
On branch master

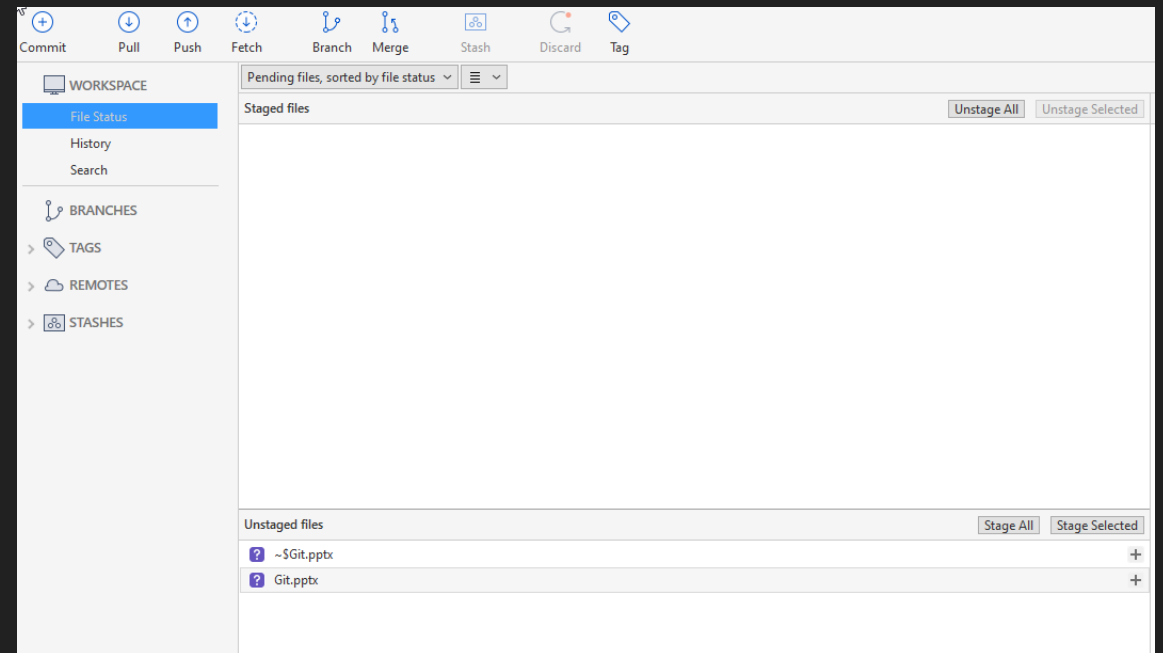
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  Git.pptx
  ~$Git.pptx

nothing added to commit but untracked files present (use "git add" to track)
```



Perceba o status “Untracked files”



Ciclo de Vida (Part I)

staging area: área onde ficam os arquivos prontos para receberem uma descrição da alteração e essa descrição ser “commitada”

Primeiro tracking de arquivo

ou

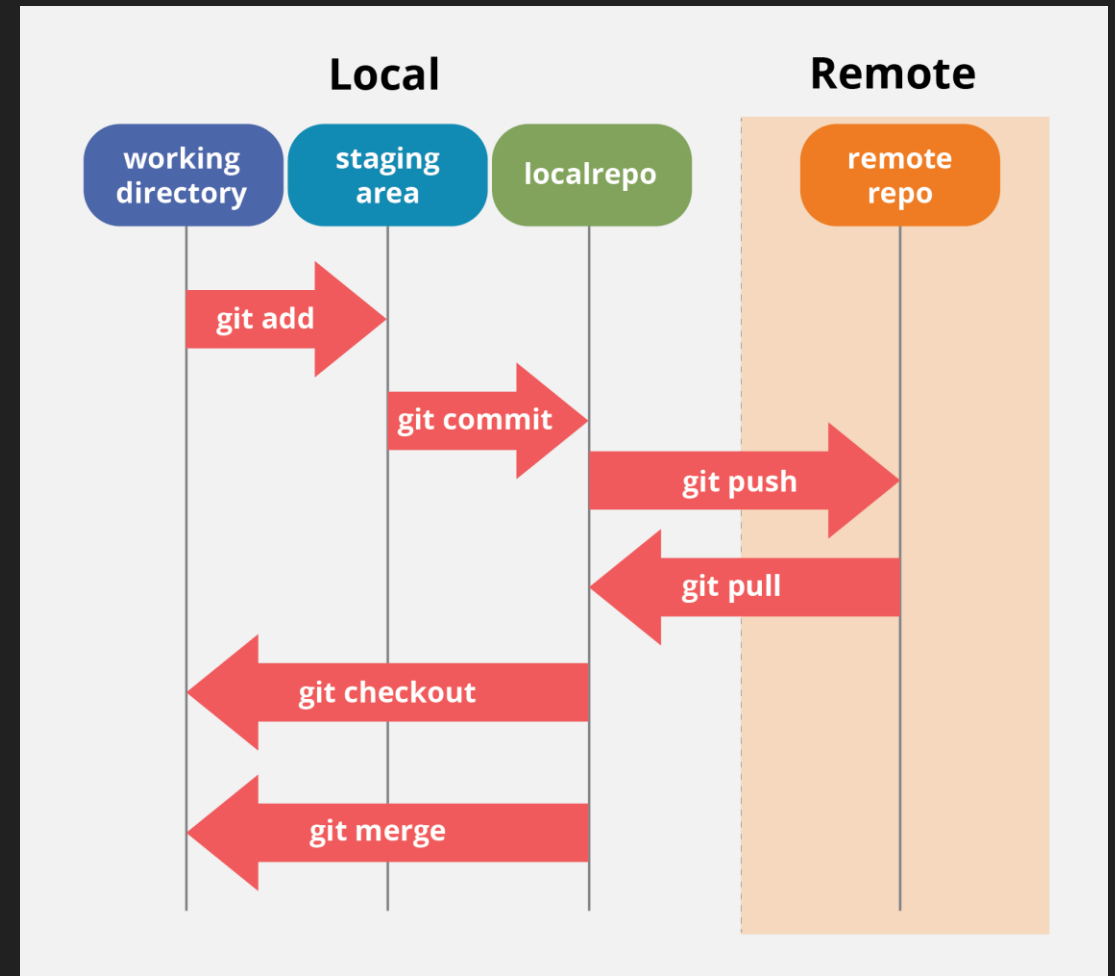
Adição a staging area

○ `git add [nomearquivo]`

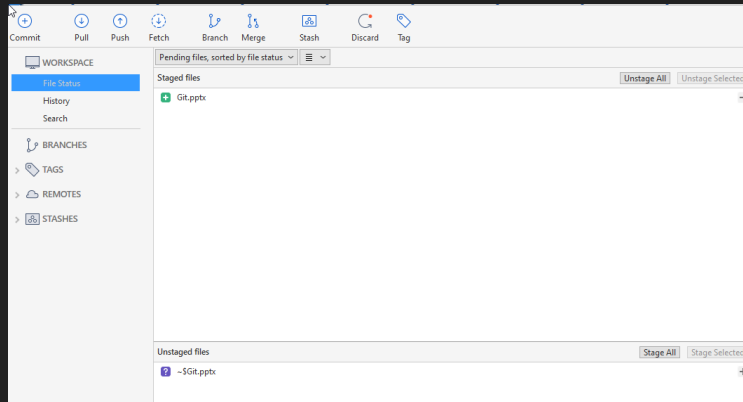
ou

○ `git add .`

para todos os arquivos

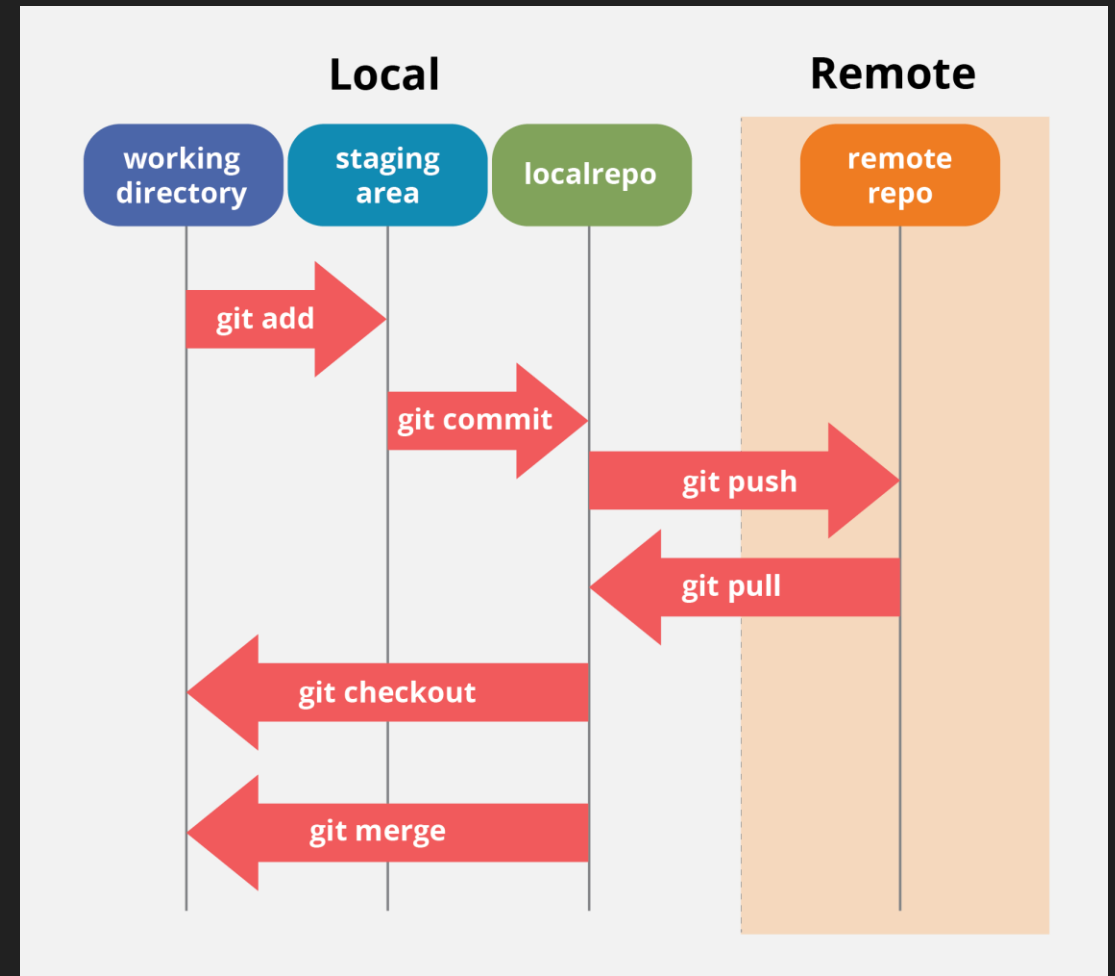


Ciclo de Vida (Part I)



```
$ git add Git.pptx  
  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   Git.pptx  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    ~$Git.pptx
```

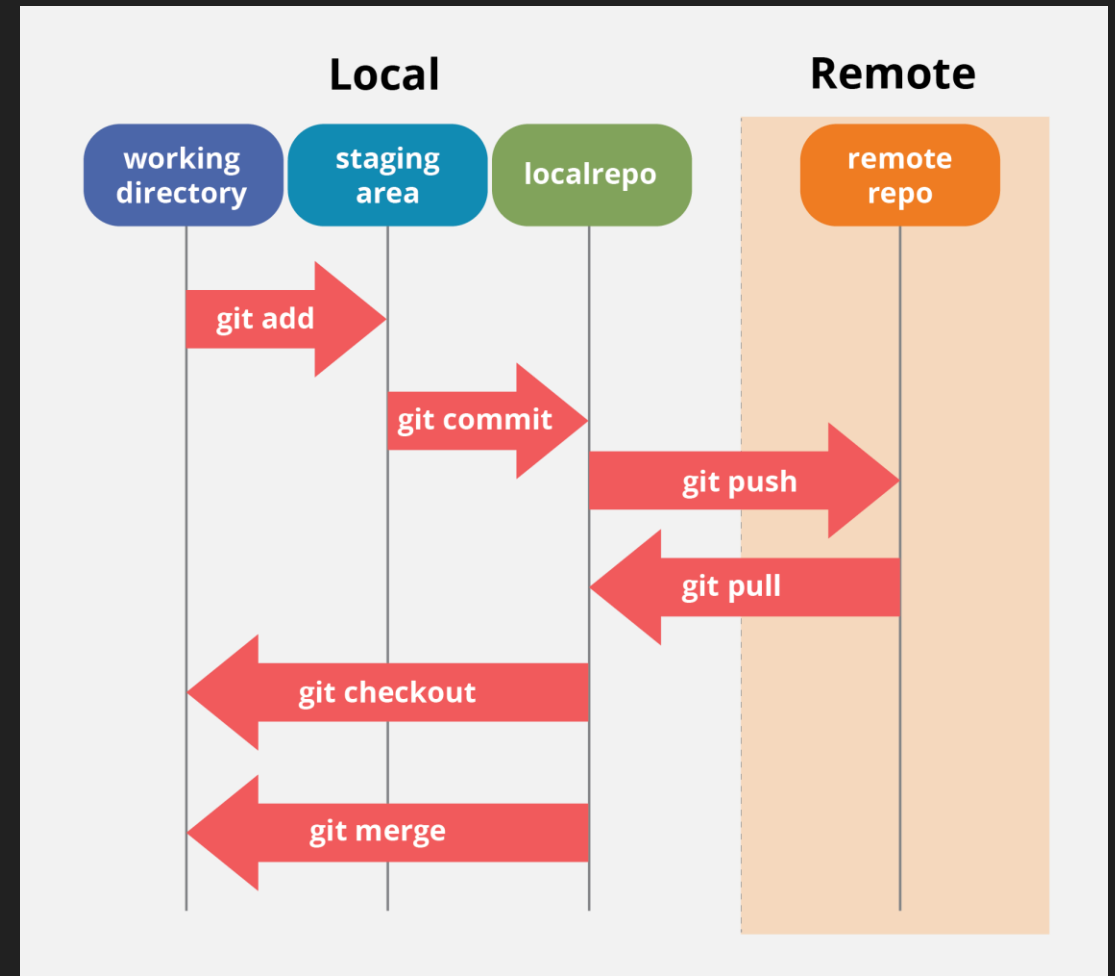
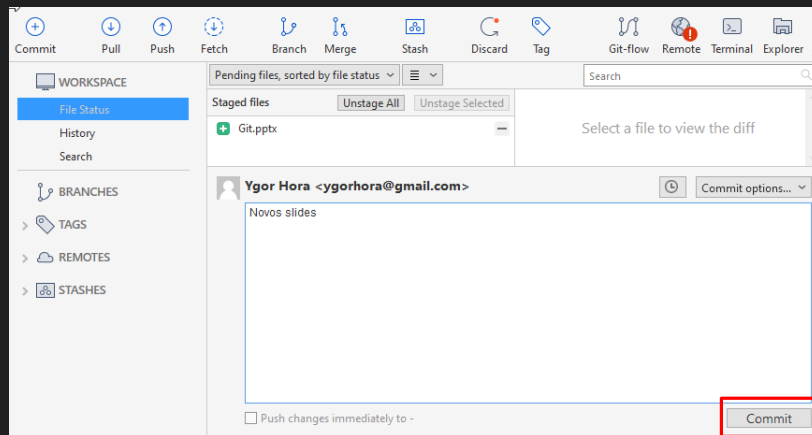
(master)
(master)



Ciclo de Vida (Part II)

Informando a modificação e persistindo no repositório local (commit)

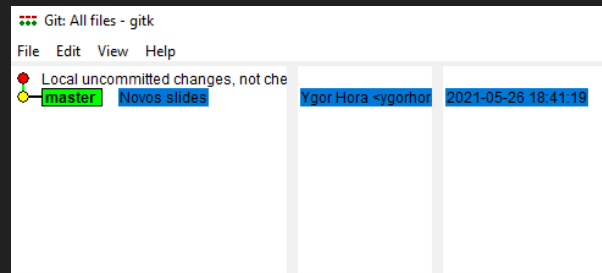
○ `git commit -m "[O que foi alterado?]"`



Ciclo de Vida (Part II)

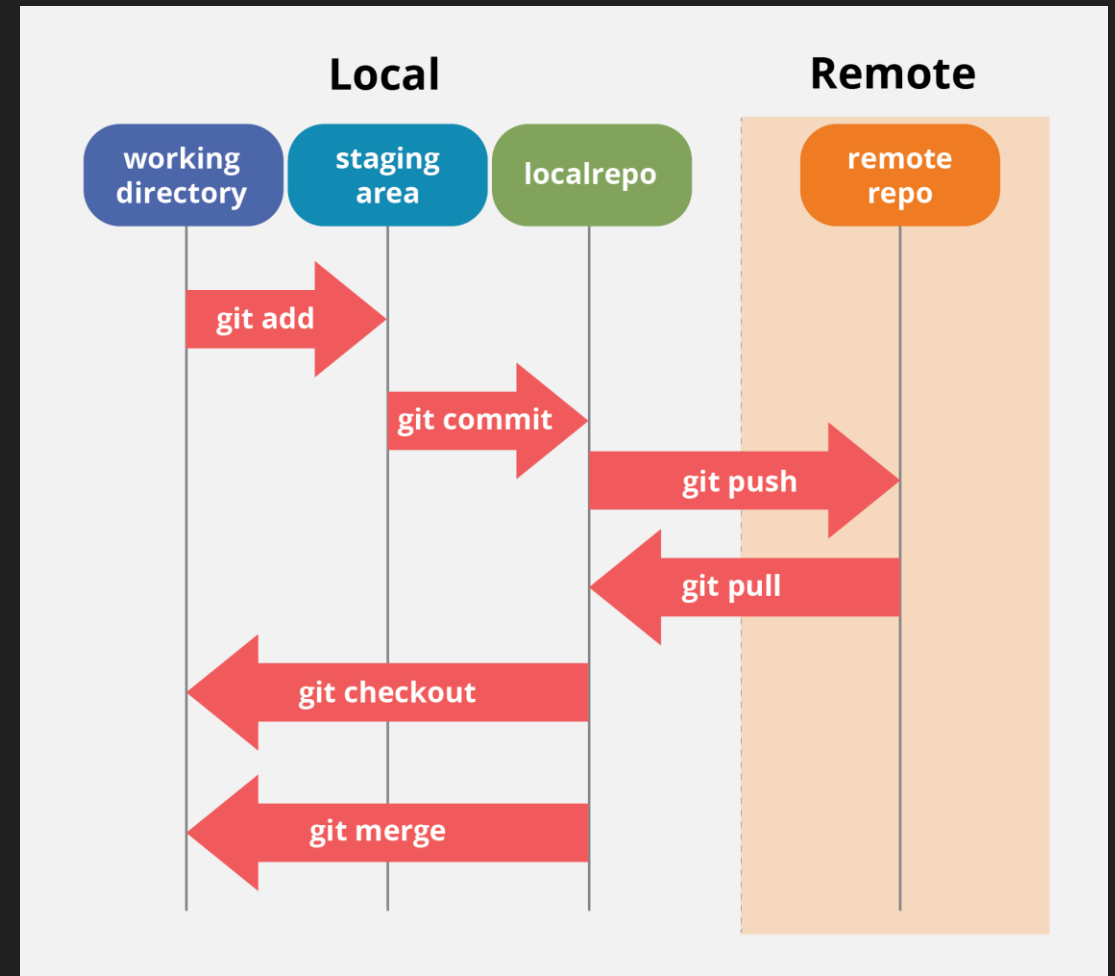
Verificando histórico de commits

- gitk
- git log



```
git log
commit fee19be9118986f67e97db67e948b203af81571c (HEAD -> master)
Author: Ygor Hora <ygorhora@gmail.com>
Date: Wed May 26 18:41:19 2021 -0300
```

Novos slides



Ciclo de Vida (Part III)

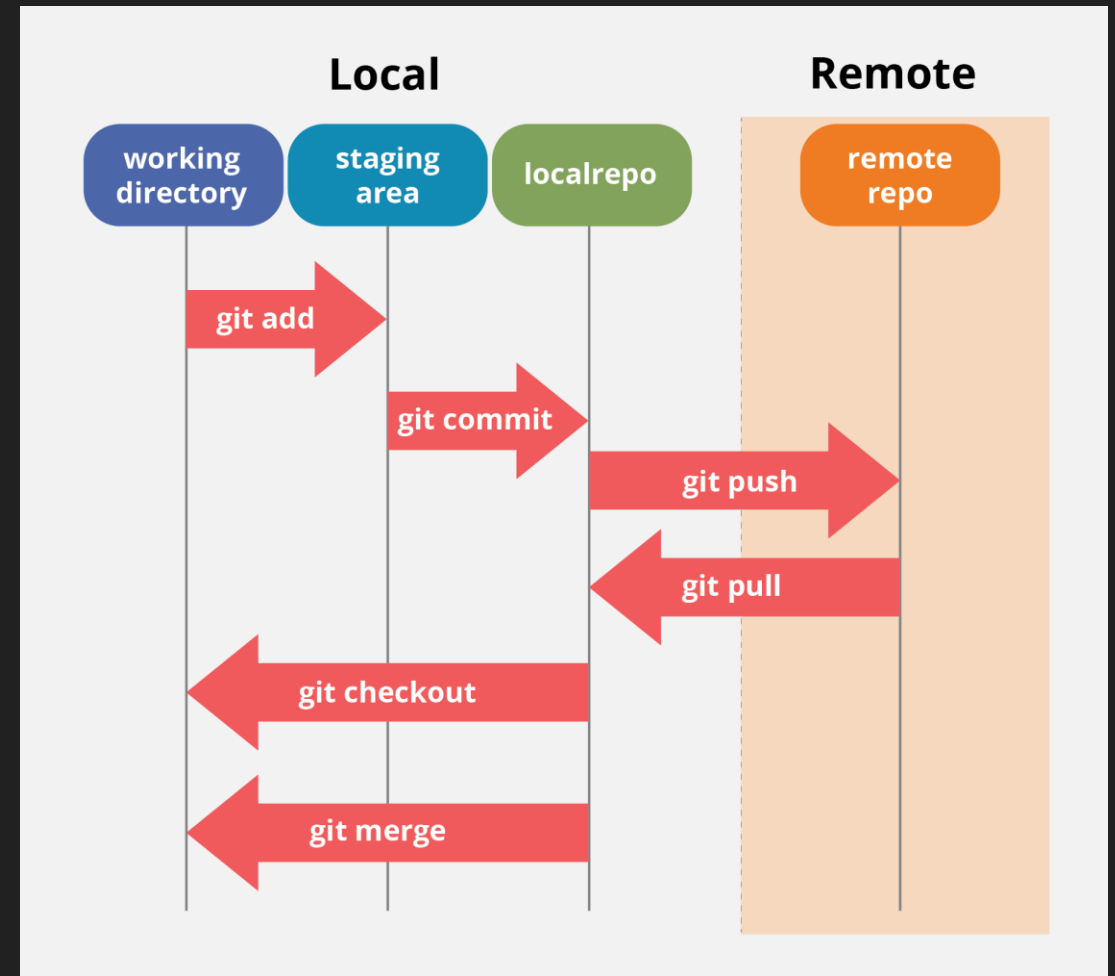
Da staging area para o working directory

- `git add algumarquivo.txt`

```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
   new file:   algumarquivo.txt
```



Perceba a dica



Ciclo de Vida (Part III)

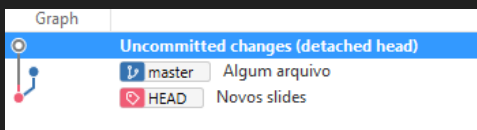
- `git commit -m "Algum arquivo"`

Visualizando hashes

- `git log`

Indo para estados anteriores

- `git checkout [hash_do_estado]`

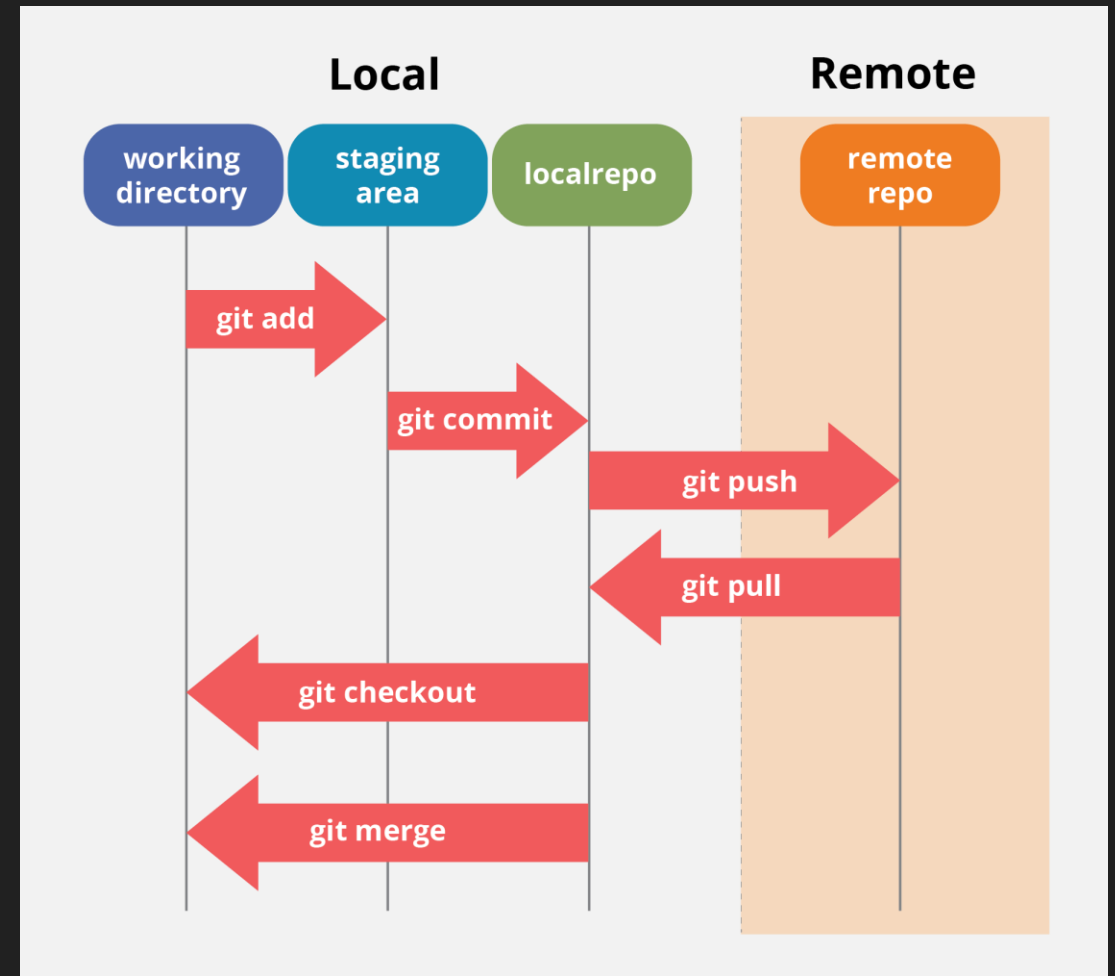


Voltando...

- `git checkout master`

ou

`git checkout -`

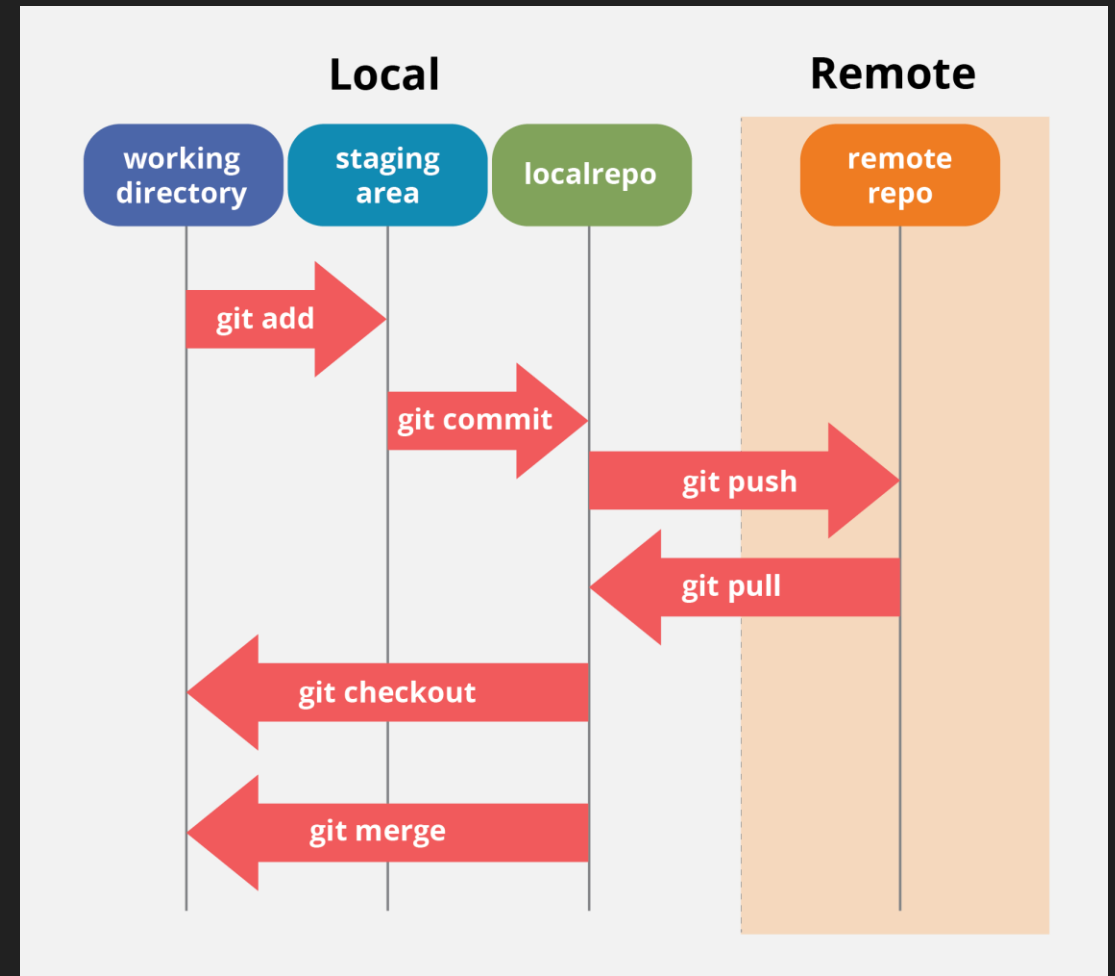


Ciclo de Vida (Part IV)

Criar chave SSH

- ssh-keygen
- eval \$(ssh-agent -s)
- ssh-add ~/.ssh/id_rsa

The screenshot shows the 'SSH keys / Add new' page in a web application. On the left is a sidebar with navigation links: 'Account settings', 'Profile', 'Account', 'Appearance' (marked with a 'New' badge), 'Account security', 'Billing & plans', 'Security log', 'Security & analysis', 'Emails', 'Notifications', and 'SSH and GPG keys' (which is highlighted). The main content area has the title 'SSH keys / Add new'. It contains a 'Title' input field, a 'Key' text area with a placeholder text: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'', and a green 'Add SSH key' button at the bottom right.



Ciclo de Vida (Part IV)

- git push



Ops! Perceba a dica

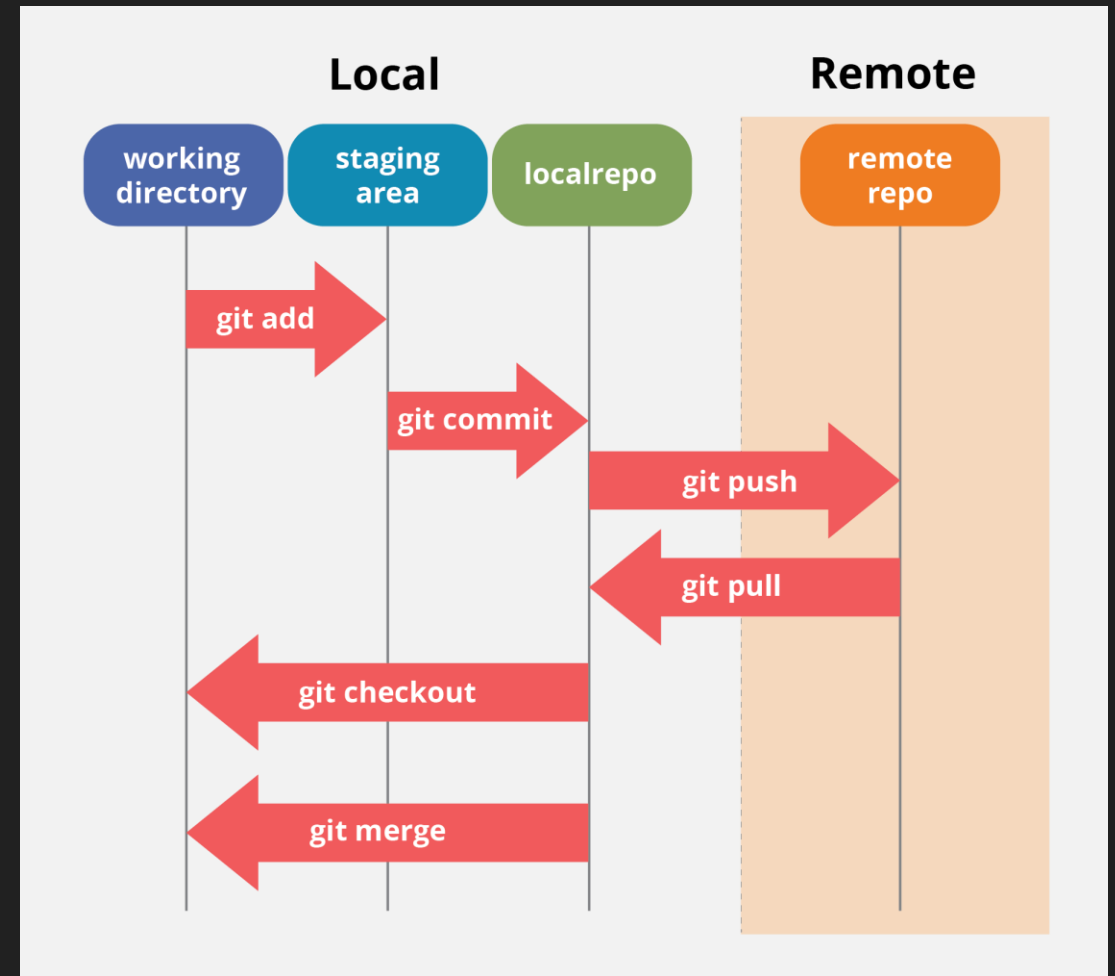
```
$ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>
```

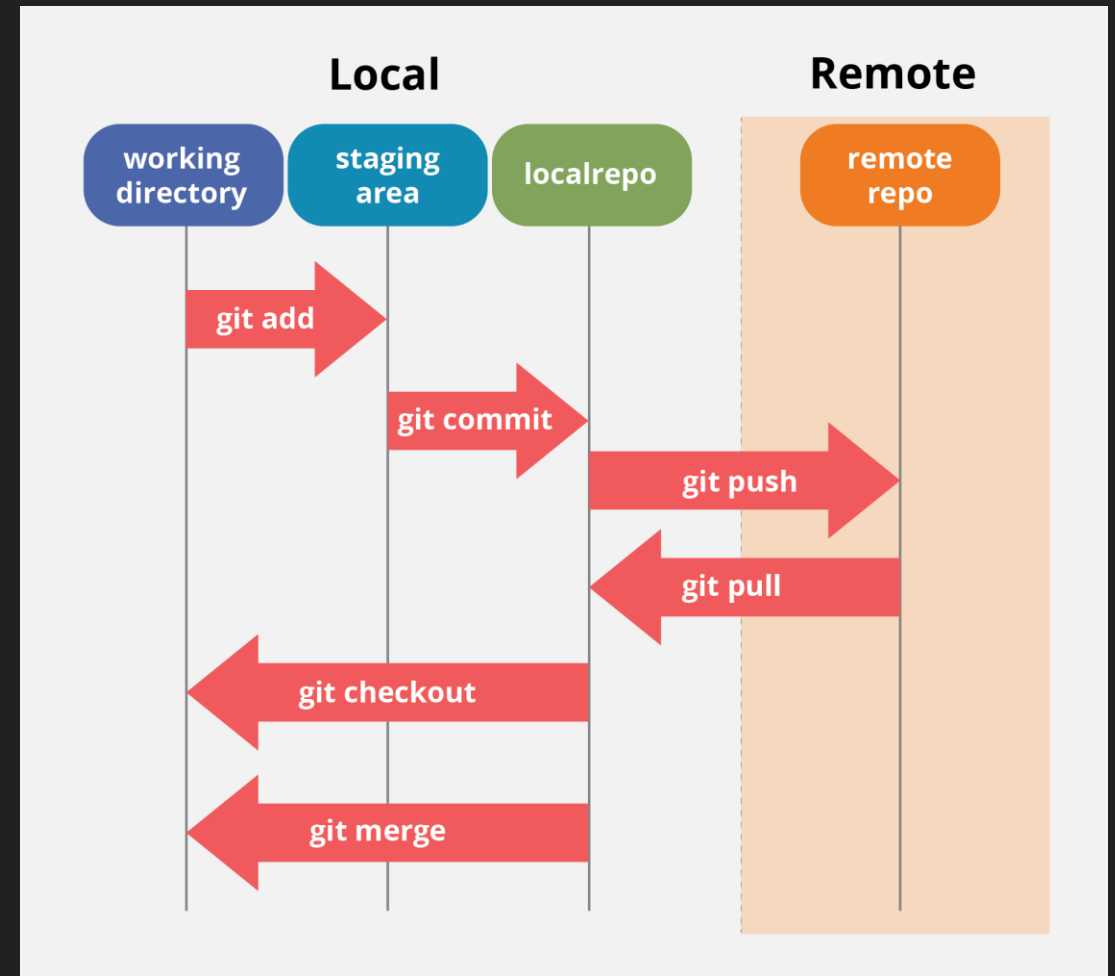
- git remote add origin
git@github.com:xxxxx/yyyyyyyyyyy.git
- git remote -v
- git push [origin] [branch]



Ciclo de Vida (Part V)

Obter dados da branch

- `git pull [origin] [branch]`



Clone

Clone do projeto

○ `git clone git@github.com:xxxxxxxxx/yyyyyyyyyyy.git`



Isso criará uma pasta com o nome padrão do projeto sendo clonado no diretório atual

Criação de Branches

Criar branch

- `git branch [nome_nova_branch]`

Listar branches acessíveis

- `git branch -a`

Mudar de branch

- `git checkout [nome_branch]`



Para criar e mudar rapidamente para a branch:

- `git checkout -b [nome_nova_branch]`

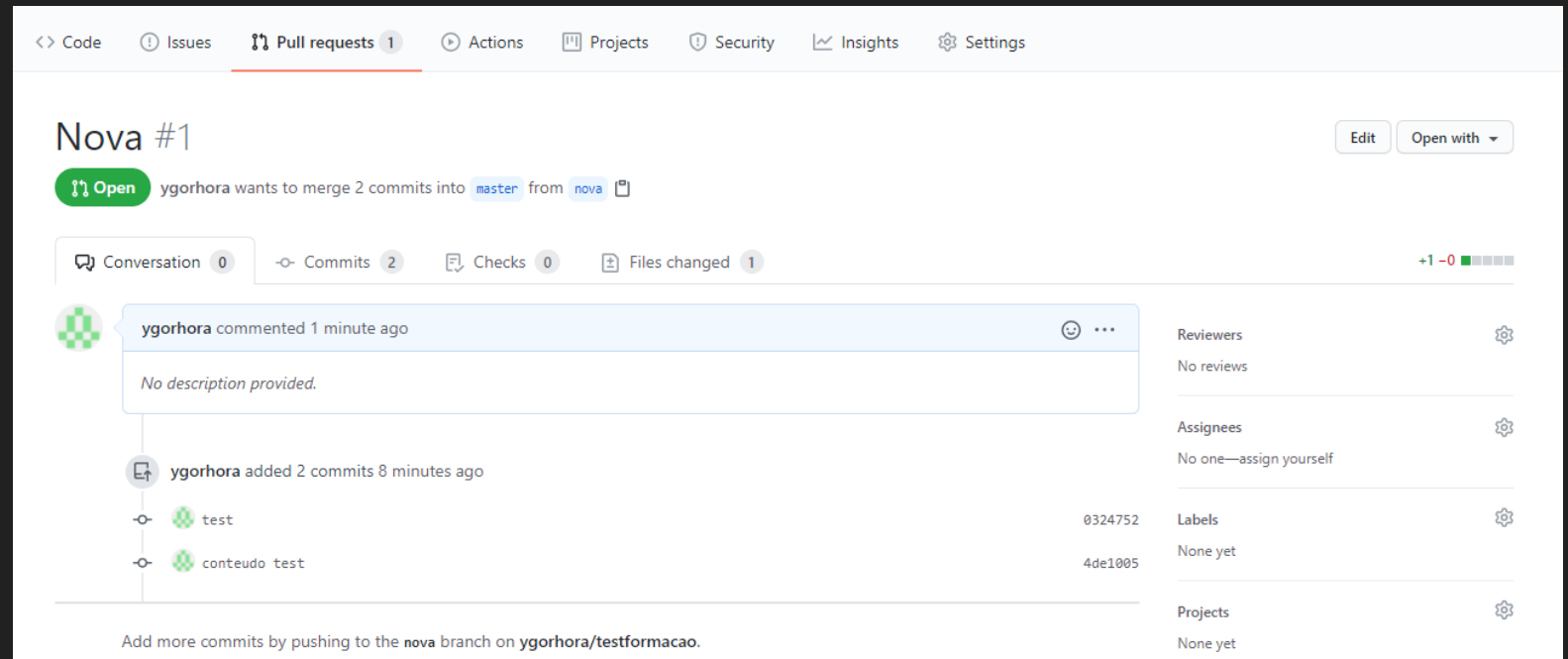


Para sincronizar as branches que foram criadas no repositório localmente:

- `git fetch`

Criação de Pull Request

1. Procurar a opção Pull requests
2. Branch origem será a que possui a nova funcionalidade
3. Branch destino geralmente será a com versões finais



Merge

Fazer checkout para branch destino

- `git checkout [branch_destino]`

Trazer as novas funcionalidades sobre o branch destino

- `git merge [branch_feature]`

Conflitos

1. Averiguar conflitos (MERGING)
 - git status
2. Resolver conflitos
3. Adicionar arquivos conflitantes na staging area
 - git add [arquivo]
4. Commitar o merge
5. Status sairá de MERGING

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session
1 <<<<<< HEAD (Current Change)
2 1
3 =====
4 2
5 >>>>>> ad77dfc10dd876bf37db899eb833079b31c767ba (Incoming Change)
6
```

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   conflituoso
```

Stash

Trata-se de uma pilha compartilhada entre todas as branches onde podemos guardar temporariamente o nosso estado atual e reaplicá-lo futuramente.



Agora é com vocês! Vamos ao Projeto!

Projeto

Em dupla:

- Criar um projeto Git público (no GitHub) com as seguintes características:
 - Com pelo menos 2 branches
 - Cada pessoa ter commits feitos dentro da branch
 - Causar obrigatoriamente pelo menos um conflito
 - Preparar apresentação (pode ser um ou dois slides) justificando o conflito causado
 - Pesquisar sobre tags e utilizá-la no projeto
 - Criar cenário onde seja necessário utilizar a stash
 - Descrever cenário em apresentação (pode ser um ou dois slides)
 - Pesquisar sobre diferença de merge e rebase
 - Descrever cenário em apresentação (pode ser um ou dois slides)
 - Pesquisar sobre como criar repositório local com o git, ou seja, um repositório em que outra pasta é que servirá como servidor para guardar as mudanças feitas no código
 - Descrever cenário em apresentação (pode ser um ou dois slides)
 - Pesquisar como excluir branches
 - Descrever cenário em apresentação (pode ser um ou dois slides)
 - Pesquisar como mover últimos commits feitos acidentalmente na master por exemplo para uma nova branch
 - Descrever cenário em apresentação (pode ser um ou dois slides)
- Entregar apresentação com o nome da dupla