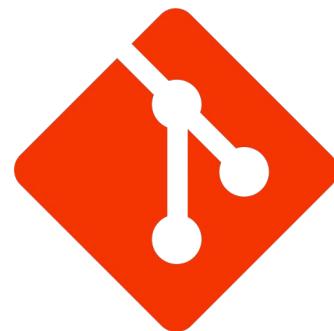


# Workshop



# git

Git: Sistema de controle de  
versão



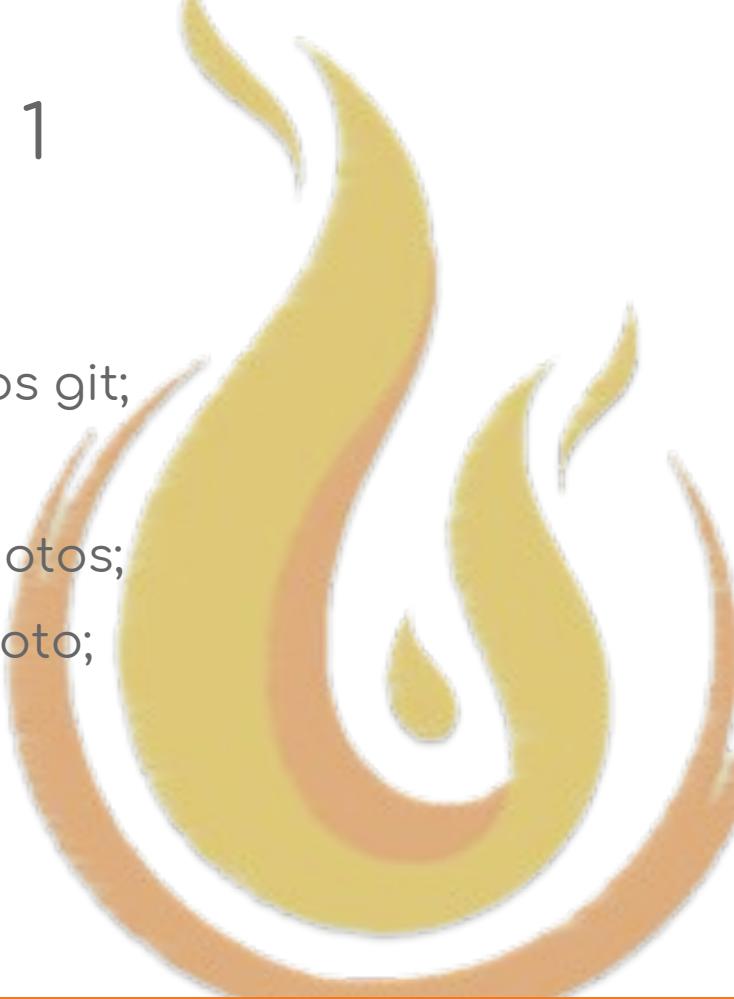
# Iniciativa para a oficina

- Capacitar membros internos do grupo efestus para desenvolvimento de projetos em grupo;
- Difundir conhecimentos técnicos para a comunidade acadêmica e não acadêmica;
- Consequentemente difundir sobre o papel produtor de tecnologia por parte das universidades enfatizando o ensino, pesquisa e extensão dessas;



# Conteúdo da oficina - Parte 1

- 1 - DevOps e controles de versão;
- 2 - GitHub e servidores para repositórios git;
- 3 - Configuração básica do git;
- 4 - Criação de repositórios locais e remotos;
- 5 - Commit para repositório local e remoto;
- 6 - Referências no git;
- 7 - Merging;



# Conteúdo da oficina - Parte 2

- 8 - Resolvendo conflitos de merge;
- 9 - Fetch, Pull e Push;
- 10 - Rebase
- 11 - Submódulos;
- 12 - Git Workflows (trabalhando em grupo);

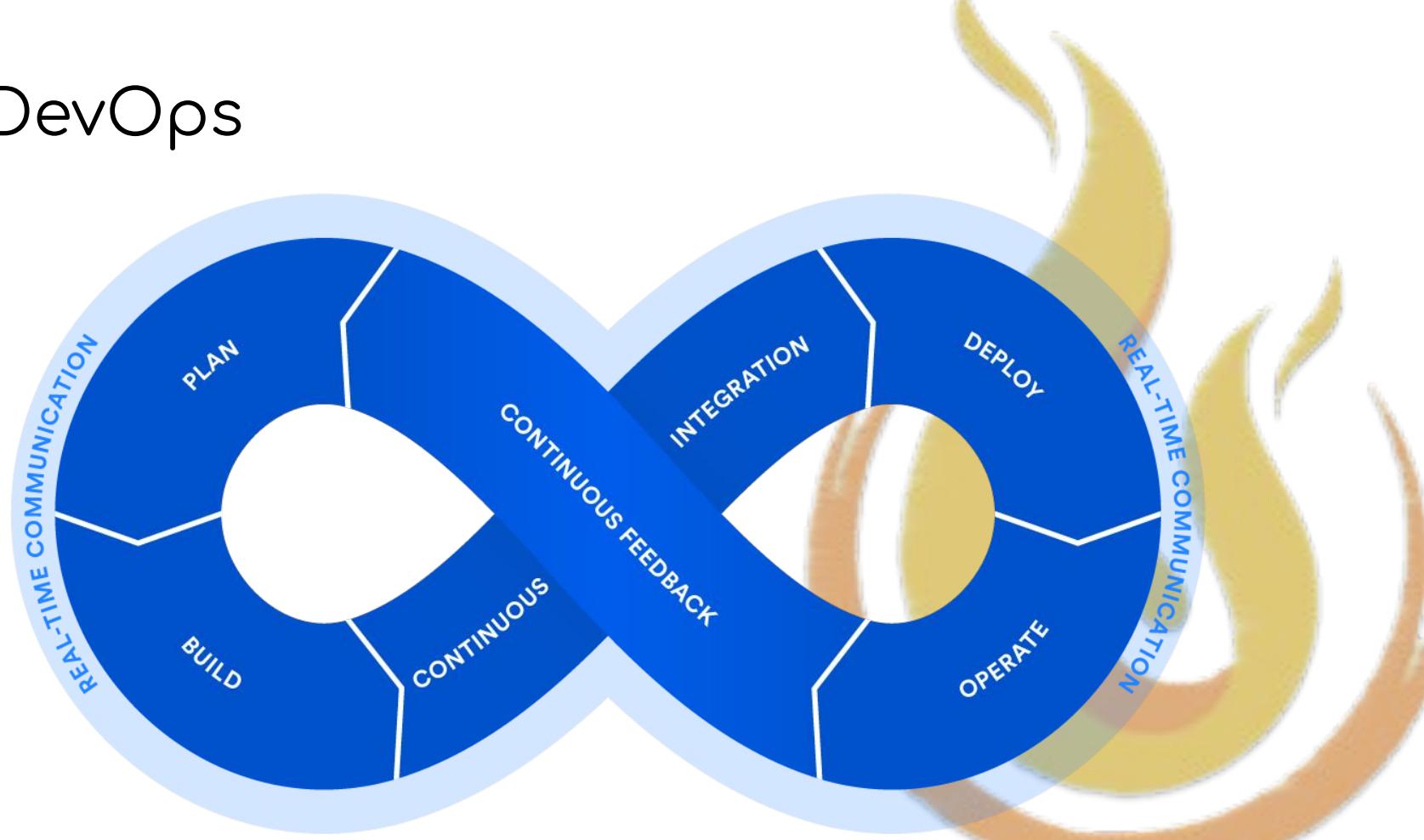


# Comandos de linux úteis

- ls - vê os arquivos da pasta;
- cd - entra em um diretório;
- mkdir - cria um diretório;
- cd .. - volta para o diretório pai
- TAB - autocompleta com arquivos existentes no diretório atual que começa com as mesmas letras

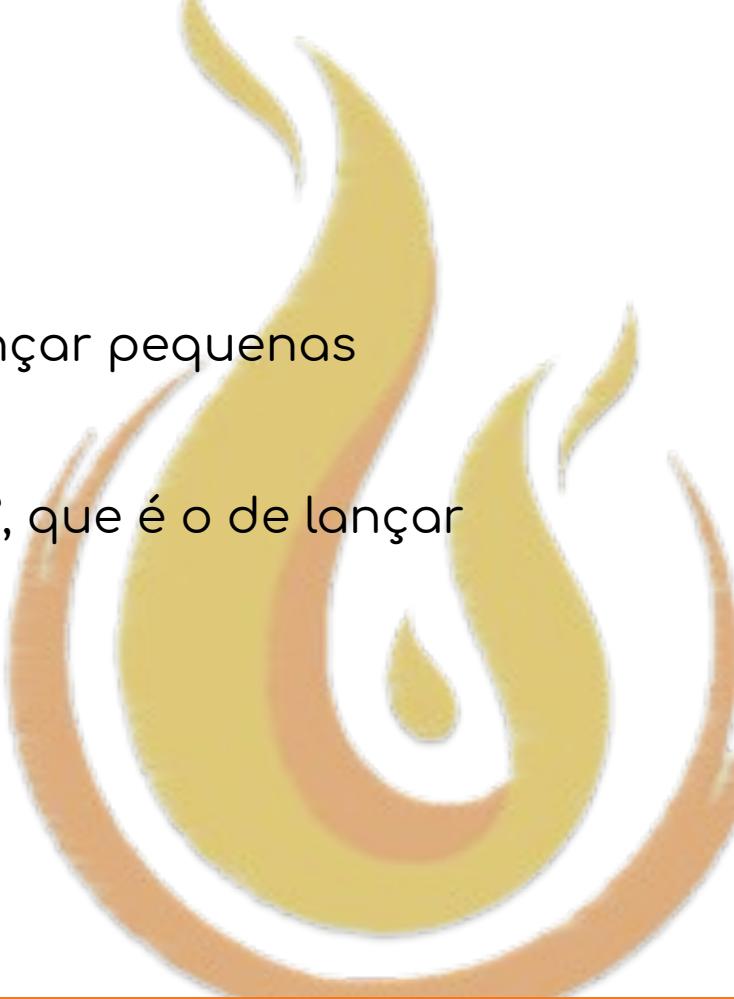


# 1 - DevOps



# 1 - Princípio do DevOps

- Continuamente planejar, construir e lançar pequenas melhorias ao projeto;
- Ao contrário do “Método da Cachoeira”, que é o de lançar diversas modificações de uma vez só;



# 1 - DevOps e sistemas de controle de versão

- DevOps e sistemas de controle de versão se relacionam pela motivação de lançar melhorias para um projeto ao decorrer do tempo, ou seja, versões do projeto;
- Esses sistemas permitem uma maior agilidade e organização com os projetos, o que a prática DevOps almeja;



# 1 - DevOps e sistemas de controle de versão

## Companies & Projects Using Git

Google

facebook

Microsoft

twitter

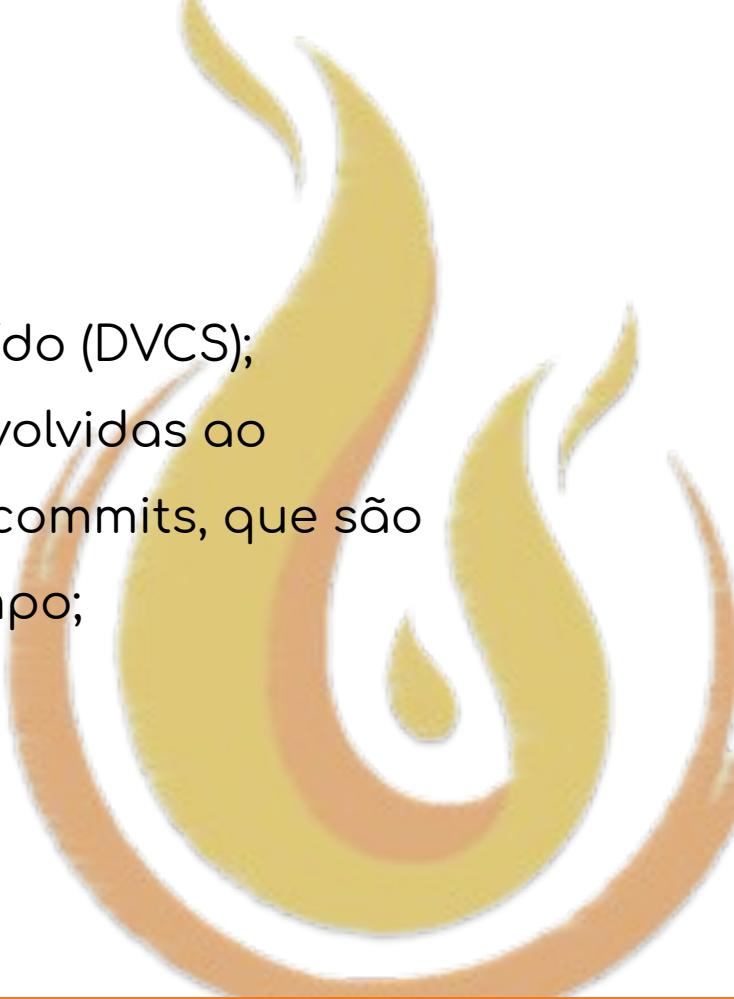
LinkedIn

NETFLIX



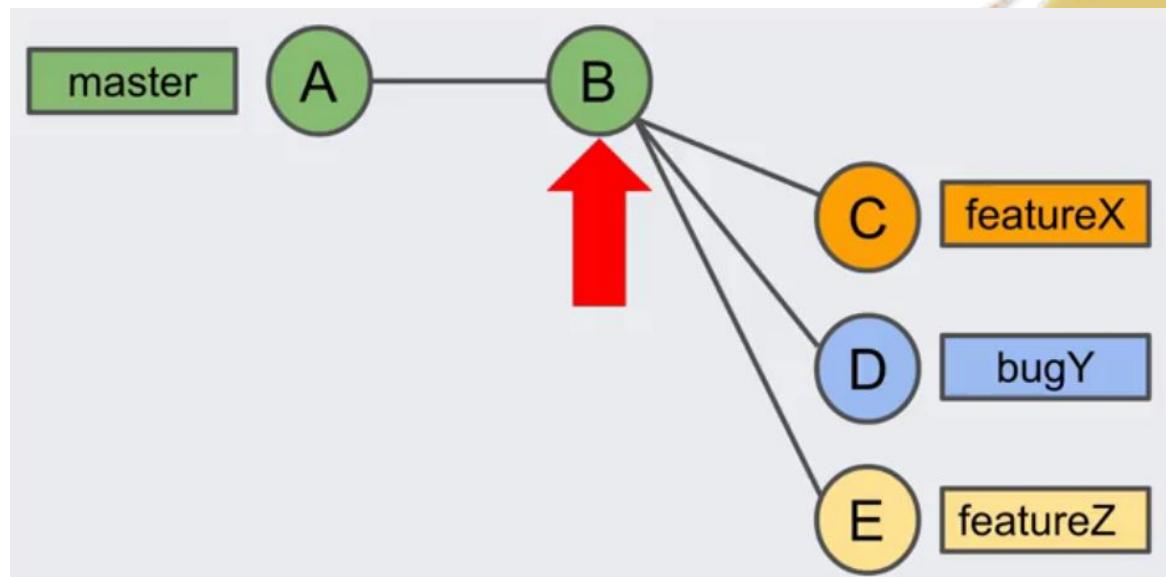
# 1 - Git

- Sistema de controle de versão distribuído (DVCS);
- As pequenas melhorias que são desenvolvidas ao decorrer do projeto são chamados de commits, que são “snapshots” do projeto ao longo do tempo;



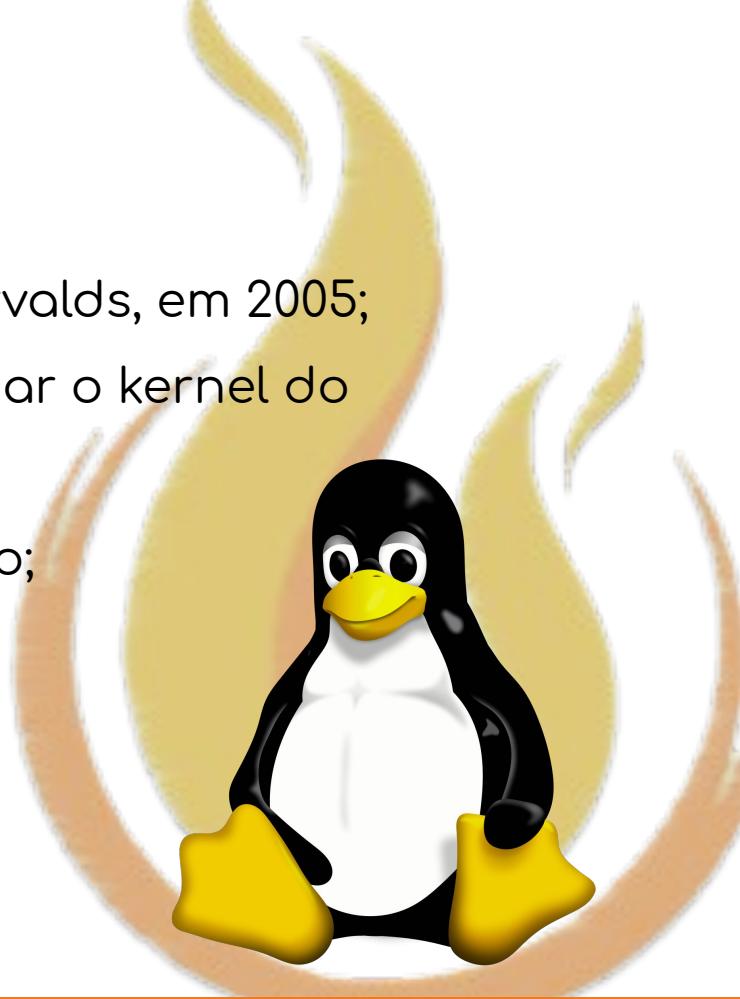
# 1 - Git

- Sistema de desenvolvimento não linear;



# 1 - Git: Contextualização

- Criado pelo criador do Linux, Linus Torvalds, em 2005;
- Desenvolvido com a intenção de guardar o kernel do Linux;
- Intenção de ser Open-Source e gratuito;



## 2 - GitHub

- Apenas um servidor para armazenar repositórios git;

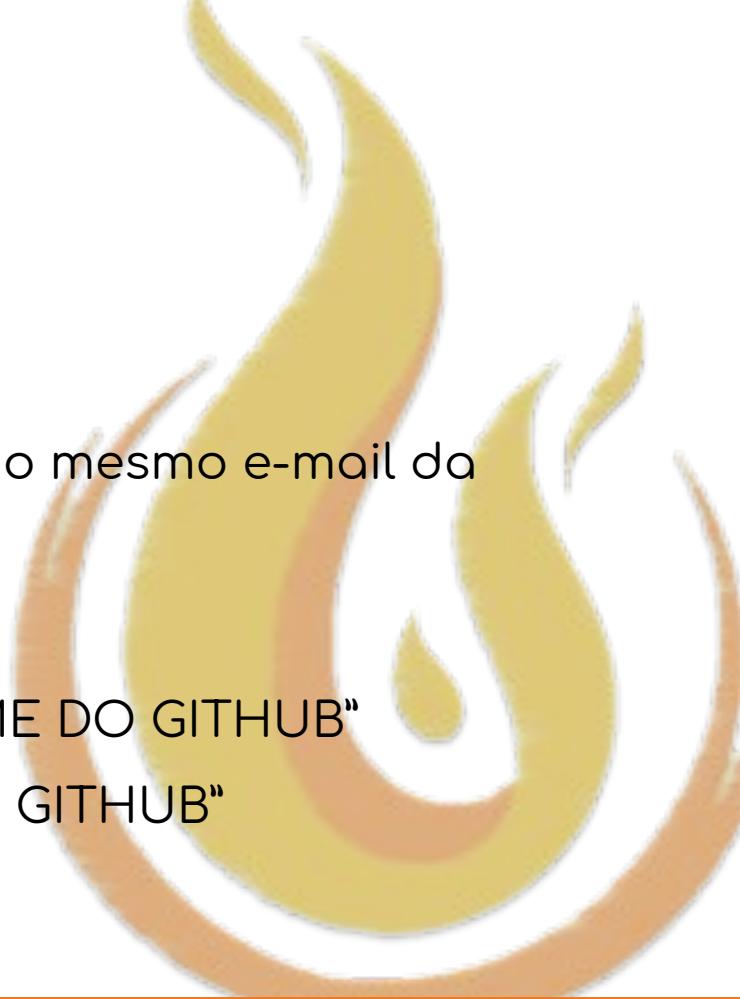


"GitHub is to Git  
what Facebook is to  
your actual face."

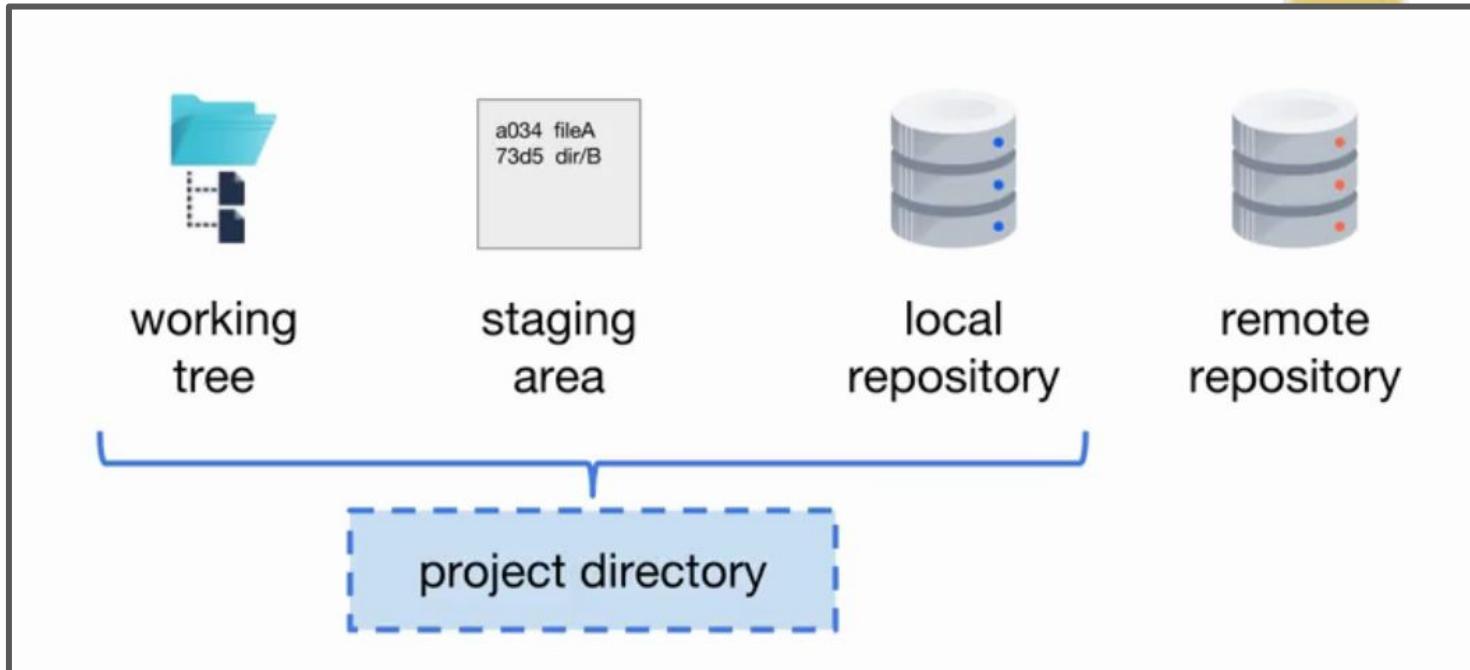
# 3 - Configuração do Git

## Tarefas:

- Vamos criar uma conta no GitHub com o mesmo e-mail da inscrição;
- No seu terminal use os comandos:  
`git config --global user.name "USERNAME DO GITHUB"`  
`git config --global user.email "EMAIL DO GITHUB"`  
`git config --list`



### 3 - Configuração do Git



## 4 - Criação de repositórios

- Podemos criar repositórios de forma local ou remota;
- Iremos fazer os dois métodos:

### Tarefas (Repositório local):

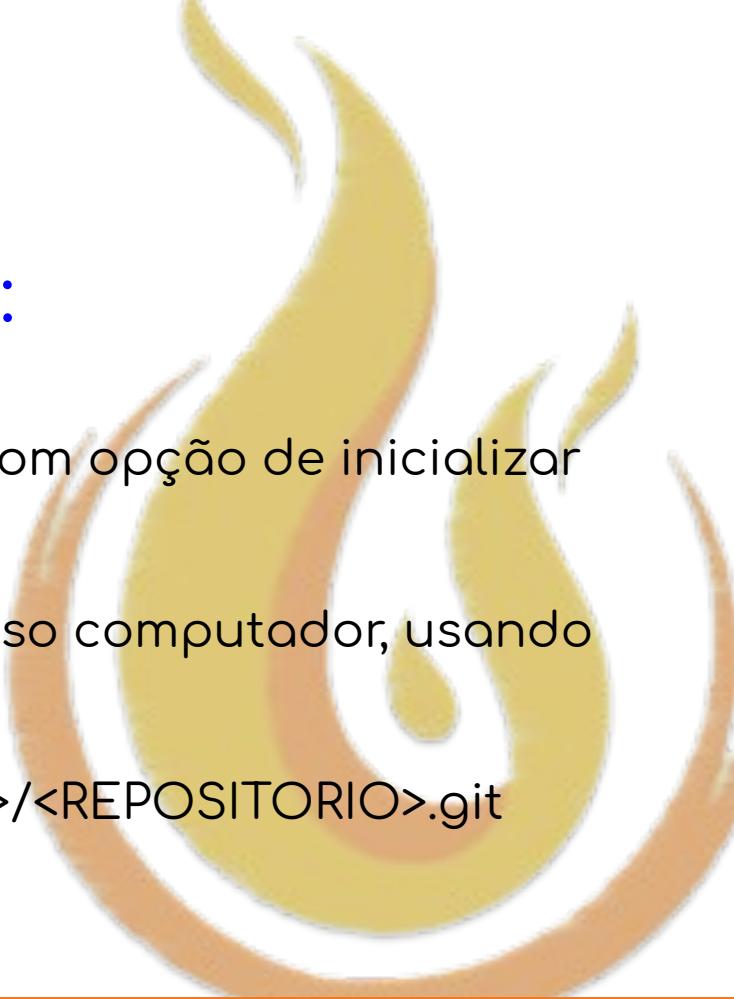
- Crie um diretório no seu computador;
- Acesse a pasta pelo terminal e digite o comando:  
`git init`



# 4 - Criação de repositórios

## Tarefas (Repositório remoto):

- Acesse o GitHub e crie um repositório com opção de inicializar com um arquivo README;
- Vamos baixar esse repositório para nosso computador, usando o comando:  
`git clone https://github.com/<USUARIO>/<REPOSITORIO>.git`

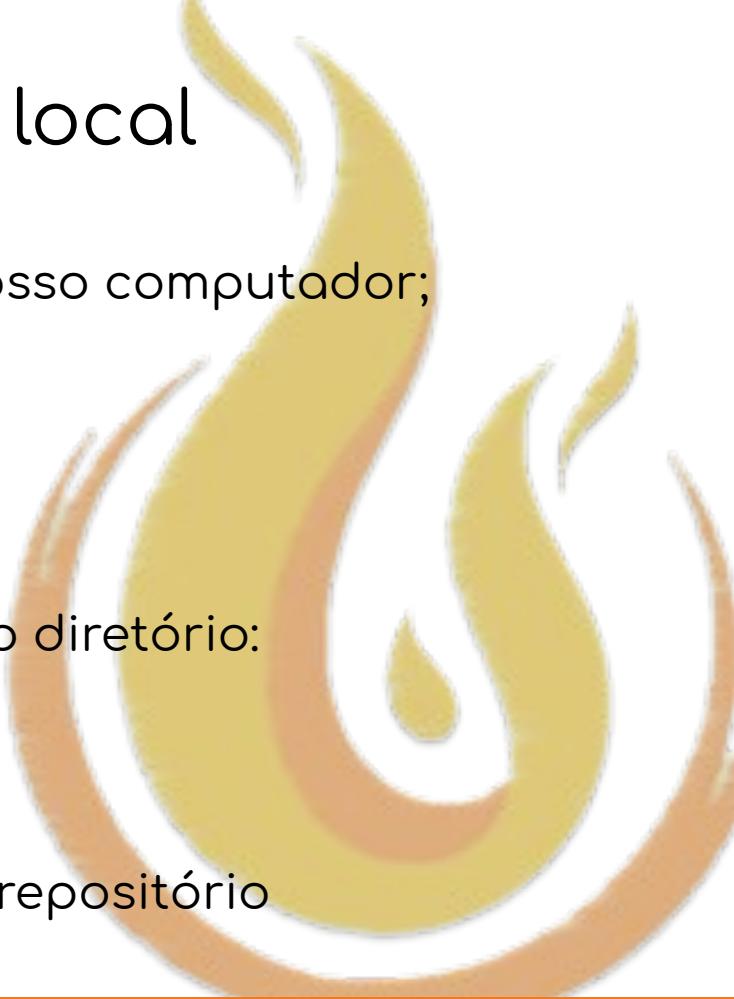


# 5 - Commit para repositório local

- Com isso temos dois repositórios em nosso computador;

## Tarefas:

- Crie um arquivo em seu diretório;
- Use os comandos no terminal dentro do diretório:  
`git status`  
`git add .`
- Repita os mesmos passos para o outro repositório



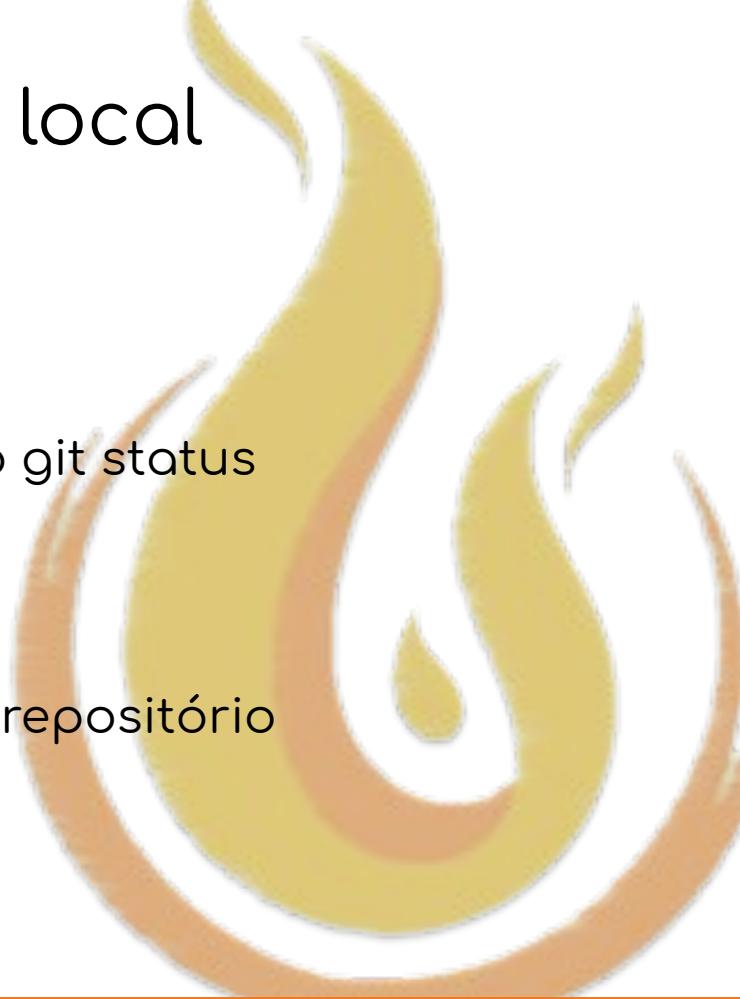
# 5 - Commit para repositório local



# 5 - Commit para repositório local

## Tarefas:

- Cheque o staging area com o comando `git status`
- Use o seguinte comando:  
`git commit -m "MENSAGEM"`
- Repita os mesmos passos para o outro repositório



# 5 - Commit para repositório local



# 5 - Commit para repositório remoto

- Um de nossos repositórios não tem um repositório remoto;
- Iremos criar um para ele;

## Tarefas:

- Crie no GitHub um repositório sem README;

- Vamos usar então o seguinte comando:

```
git remote add origin https://github.com/<USUARIO>/<REPOSITORIO>.git
```

Have a local repository?	Task
no	<i>clone the remote</i>
yes	<i>add the remote</i>

## 5 - Commit para repositório remoto

- Os dois repositórios locais têm agora repositórios remotos;
- Você pode checar com o comando:  
git remote --v

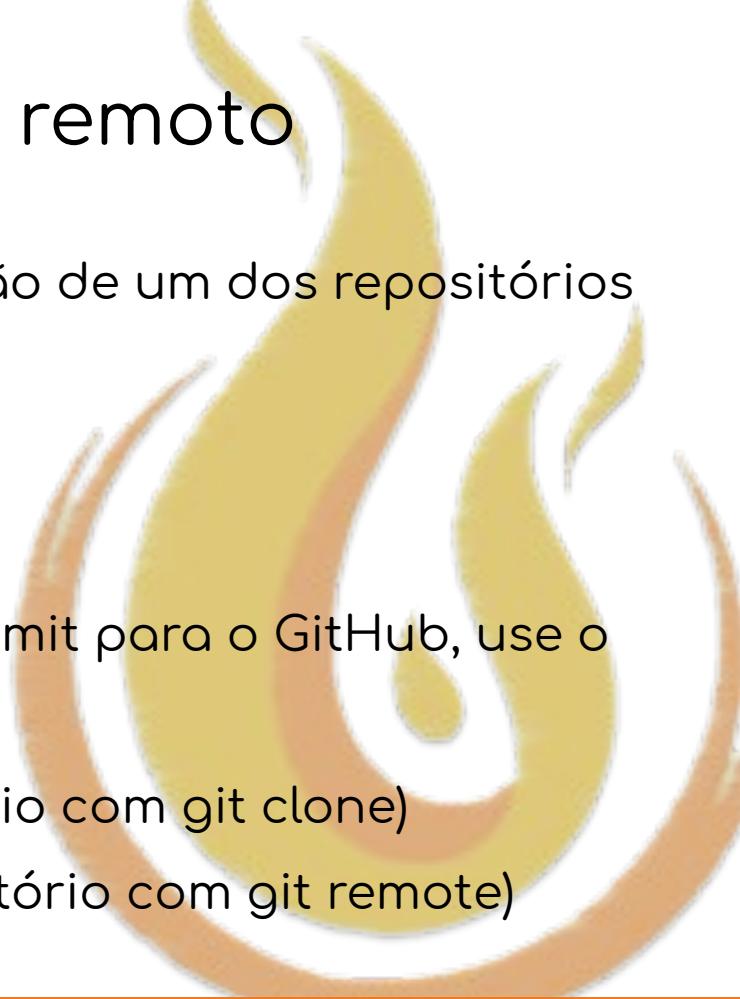
```
[TarzanUSP@localhost git]$ git remote -v
origin  https://github.com/BrenoUSP/test.git (fetch)
origin  https://github.com/BrenoUSP/test.git (push)
[TarzanUSP@localhost git]$ █
```

# 5 - Commit para repositório remoto

- Vamos agora subir o commit em questão de um dos repositórios que estamos trabalhando;

## Tarefas:

- No diretório que você quer subir o commit para o GitHub, use o comando:  
`git push origin master` (Para o repositório com `git clone`)  
`git push -u origin master` (Para o repositório com `git remote`)



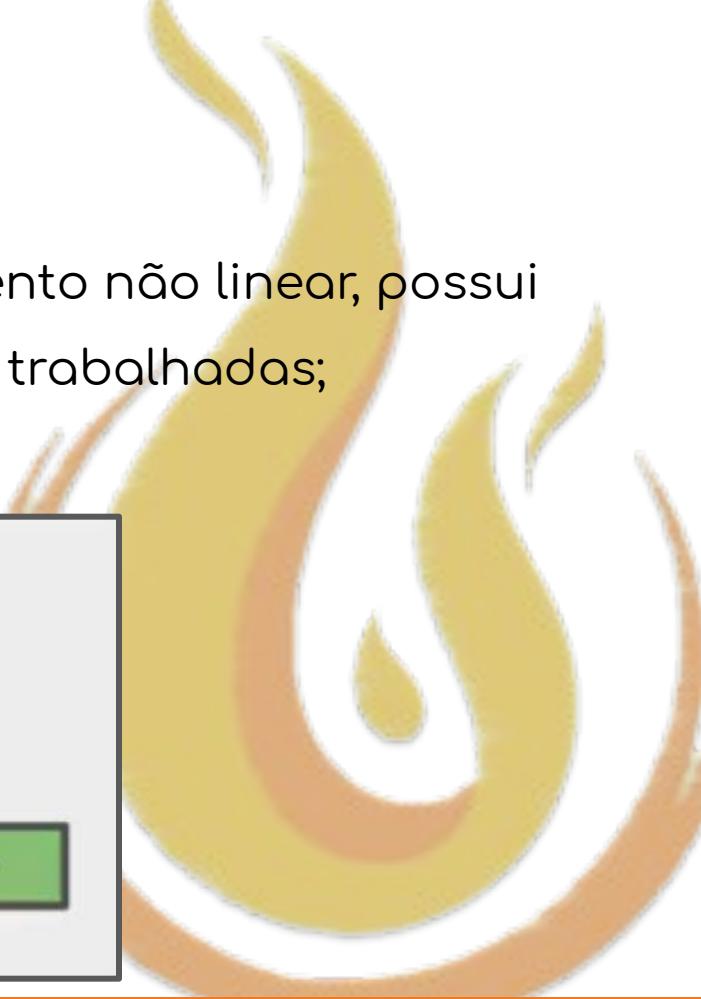
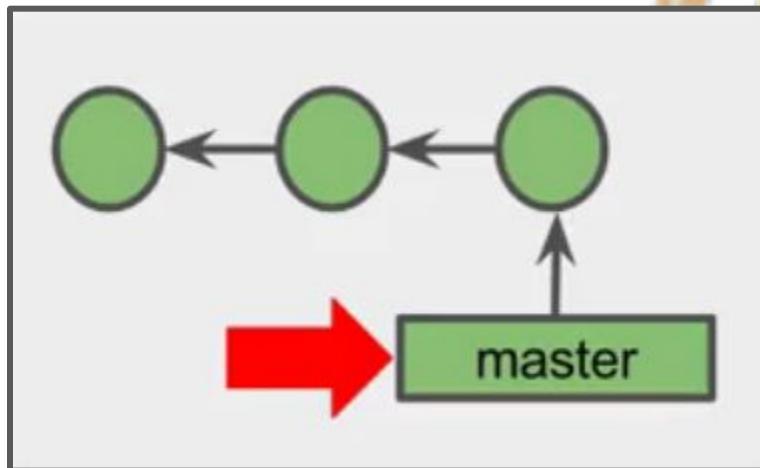
# 5 - Commit para repositório remoto

- Conseguimos upar o commit do repositório local para o remoto!

```
[TarzanUSP@localhost git]$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 211 bytes | 211.00 KiB/s, done
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/BrenoUSP/test.git
 * [new branch]      master -> master
```

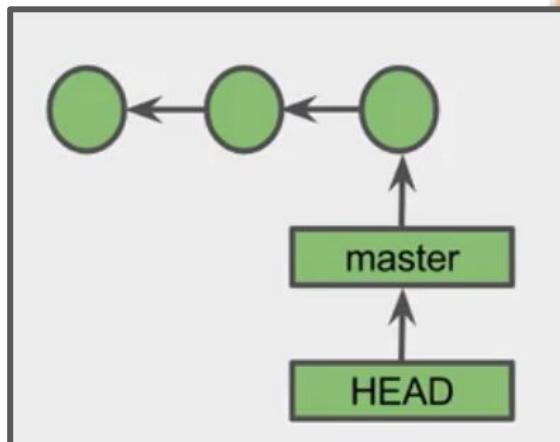
## 6 - Referências no git

- O git, sendo um sistema de desenvolvimento não linear, possui diversos tipos de referências para serem trabalhadas;
- Incluem branches e tags, por exemplo;



## 6 - HEAD

- O termo HEAD se refere ao commit atual;
- Aponta para a branch atual;
- Apenas um HEAD por repositório;



## 6 - Tags

- Tags são referências a commits específicos;
- Extremamente útil para indicar versões de programas, por exemplo;

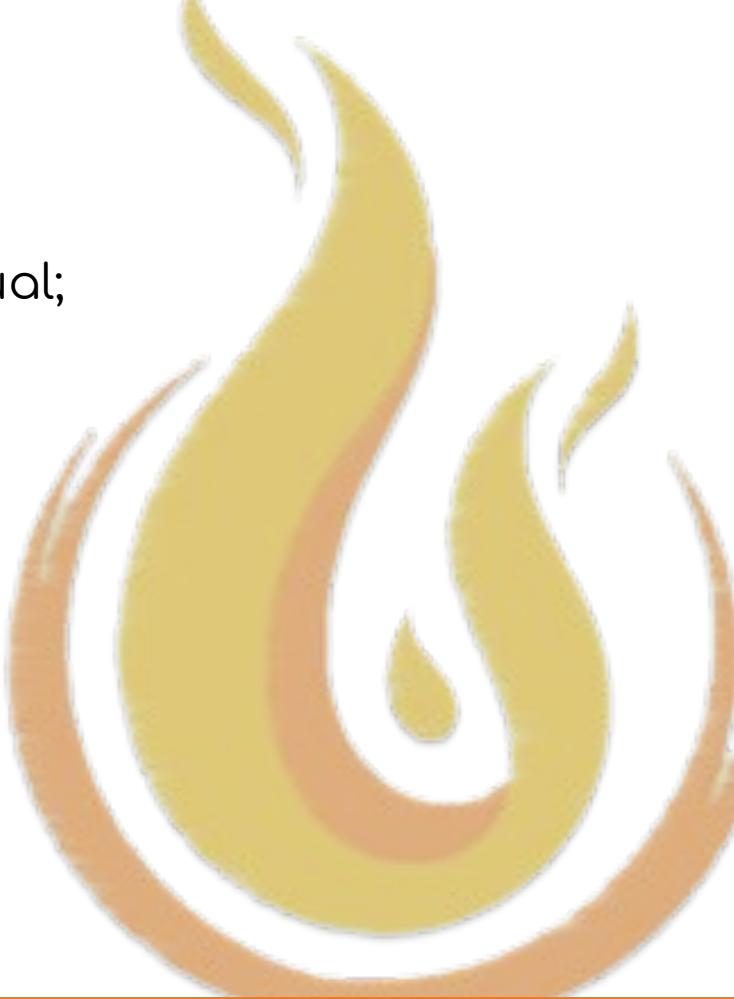
```
$ git tag
v0.1
v1.0
$ git show v0.1
commit e8d41c0322e629aaf7d7aa8a9a1335bff2f76f72 (tag: v0.1)
(commit info)
```

## 6 - Tags

- Vamos criar uma tag para o commit atual;

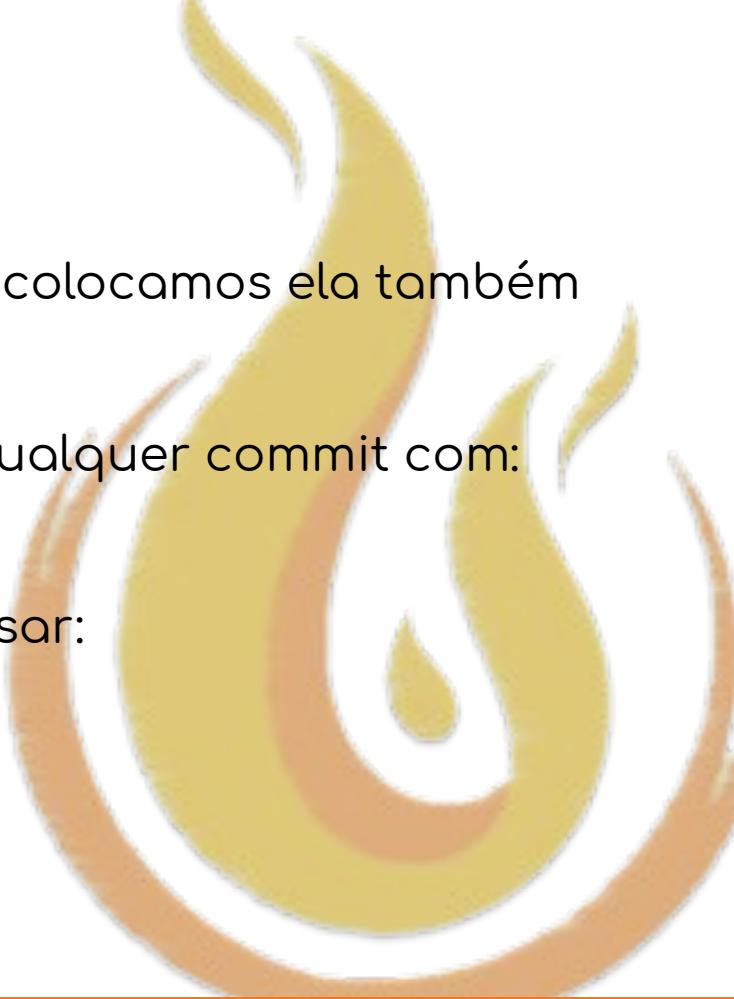
### Tarefas:

- No diretório use os comandos:  
`git tag "NOME DA TAG"`  
`git push origin --tags`



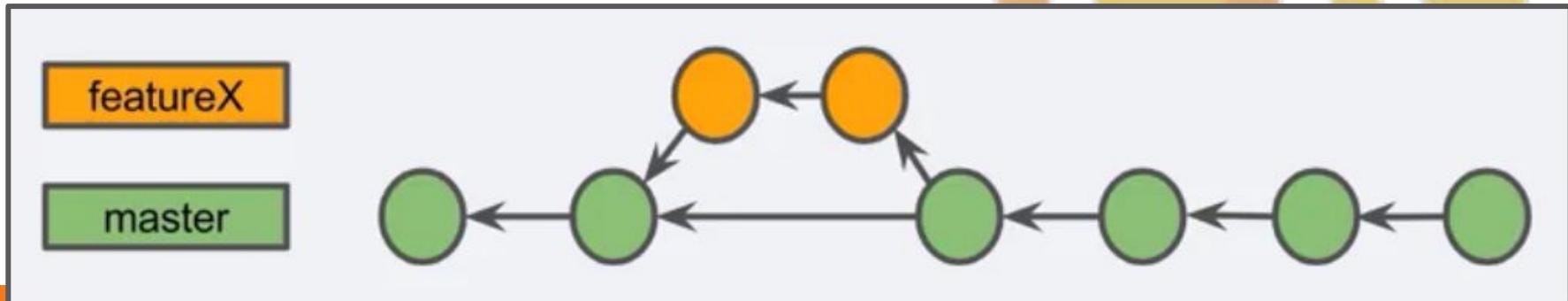
## 6 - Tags

- Criamos uma tag no repositório local e colocamos ela também no remoto;
- Você pode usar tags para referenciar qualquer commit com:  
`git tag "NOME DA TAG" "COMMIT"`
- Para upar apenas uma tag específica usar:  
`git push origin "NOME DA TAG"`



# 6 - Branches

- Branches são ramificações que ocorrem no desenvolvimento de um projeto, estrutura em grafos orientados;
- Ramificações podem ser criadas para implementar uma “feature” especial preservando o projeto original branch master, por exemplo;

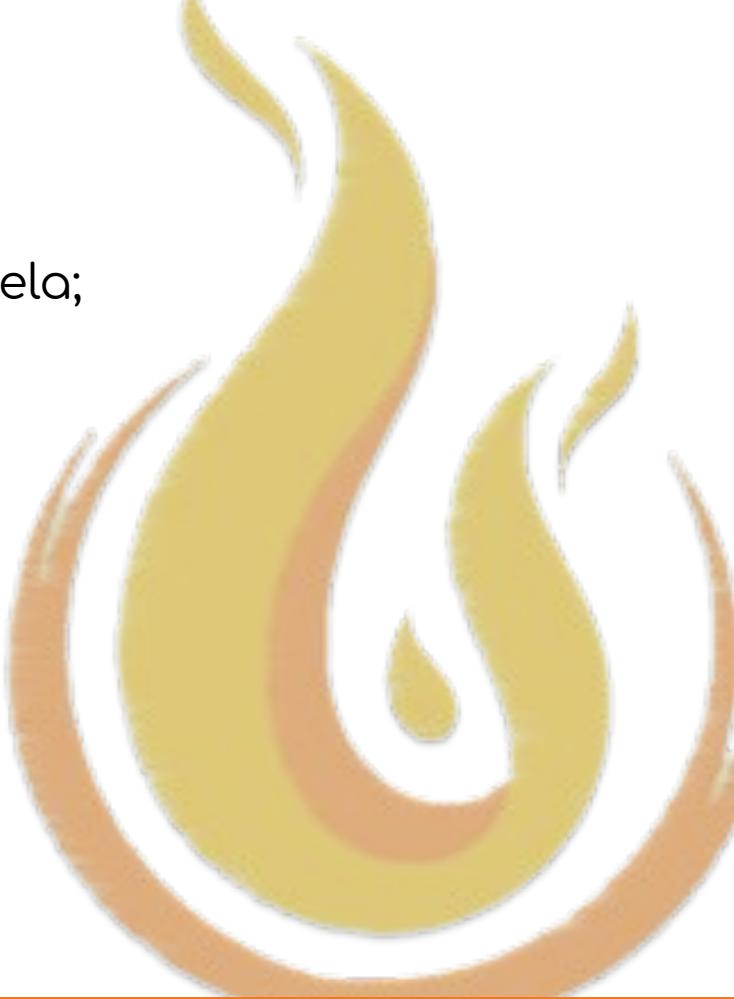


# 6 - Branches

- Vamos criar uma branch e mudar para ela;

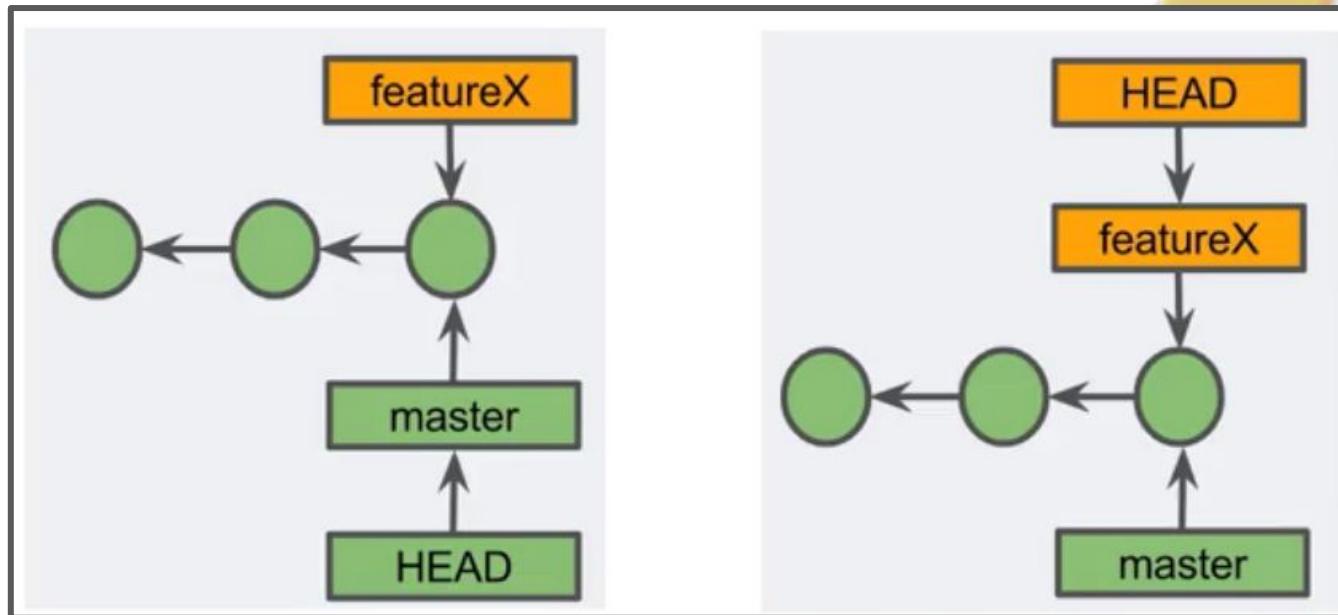
## Tarefas:

- No diretório use os comandos:  
`git branch "NOME DA BRANCH"`  
`git checkout "NOME DA BRANCH"`



## 6 - Branches

- Quando mudamos de branch ocorre o seguinte:



## 6 - Branches

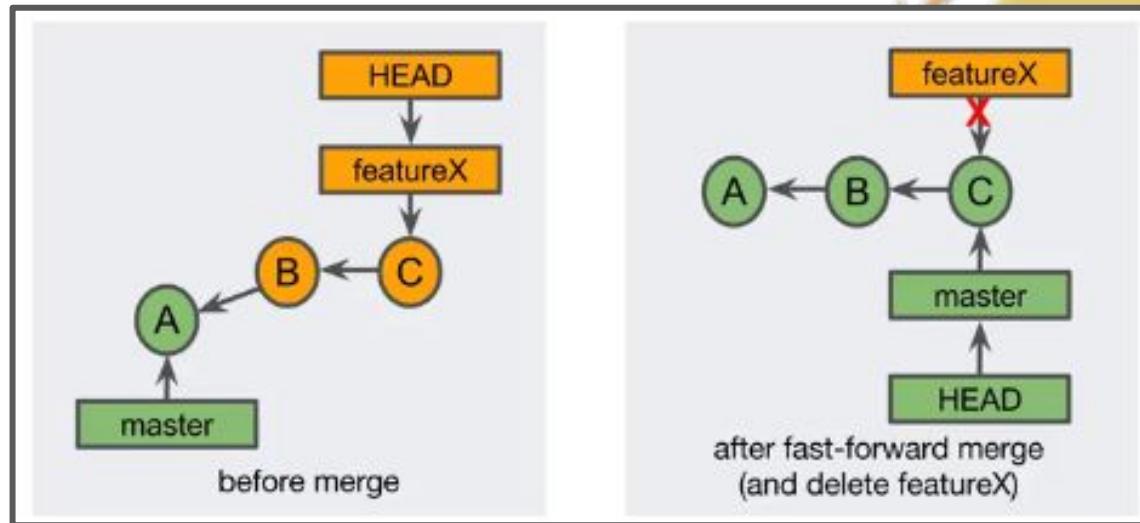
- Para criar uma branch e já mudar para ela:  
`git checkout -b "NOME DA BRANCH"`
- Para deletar a branch basta usar o comando:  
`git branch -D "NOME DA BRANCH"`





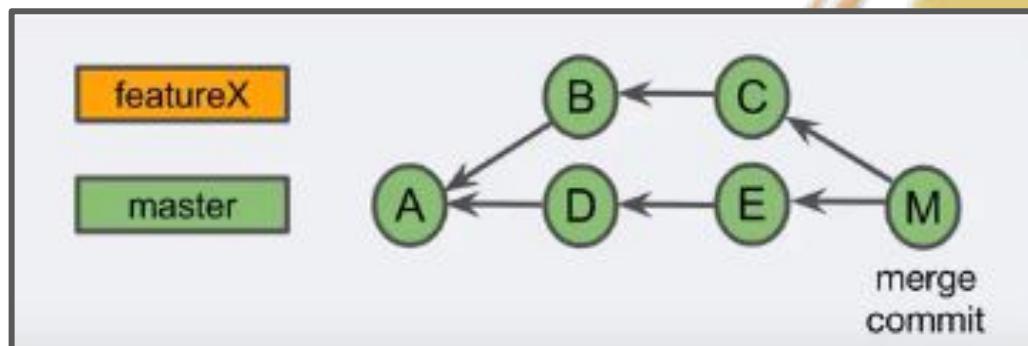
# 7 - Merging

- Grosso modo, método para juntar commits;
- Existem tipos de merge, como o fast-forward;



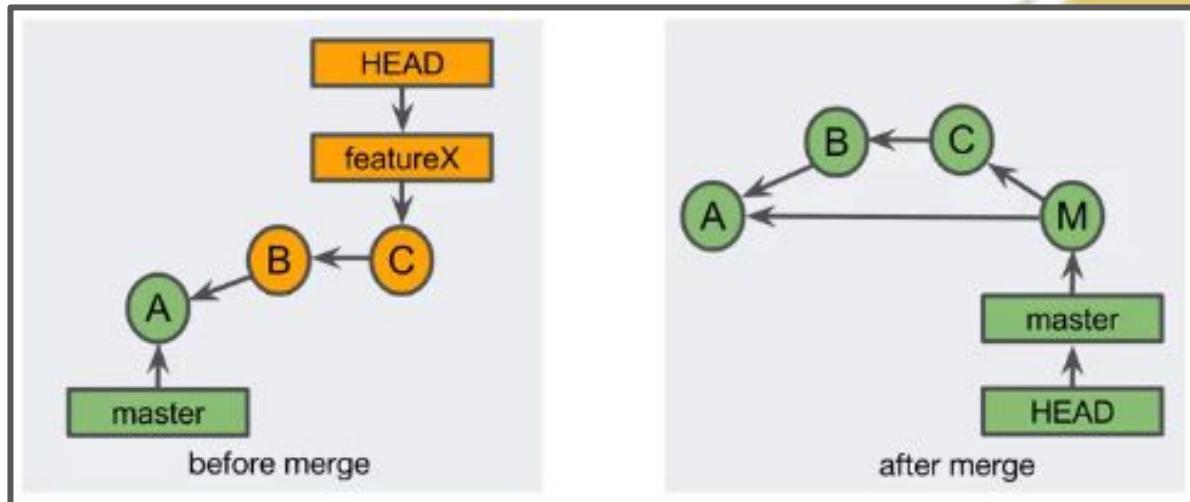
## 7 - Merging

- Se houver commits na master após a criação da branch ocorrerá o seguinte:



# 7 - Merging

- Se não for um fast-forward merge:

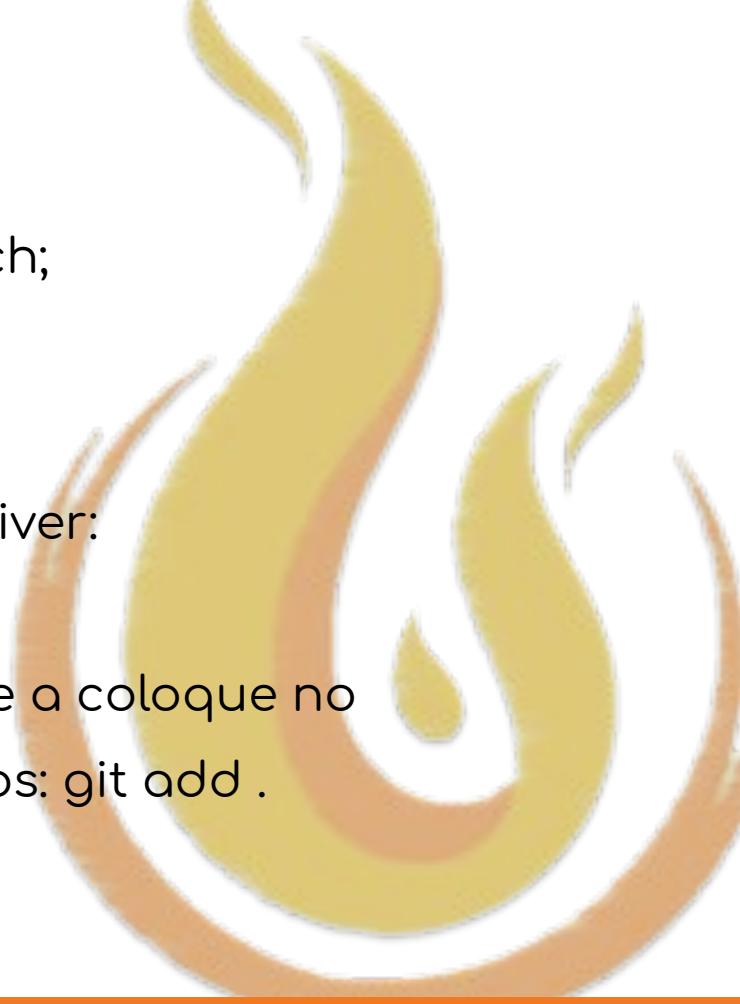


# 7 - Merging

- Vamos fazer merging com a nova branch;

## Tarefas:

- Mude para a nova branch se já não estiver:  
`git checkout "NOME DA BRANCH"`
- Crie um arquivo qualquer no diretório e a coloque no repositório remoto usando os comandos: `git add .`  
`git commit -m "MENSAGEM"`  
`git push origin "NOME DA BRANCH"`



# 7 - Merging

## Tarefas:

- Mude para a branch master:  
`git checkout master`
- Fazendo o merge:  
`git merge "NOME DA BRANCH"`
- Foi gerado um novo commit, basta colocar no repositório remoto, com os três comandos básicos;



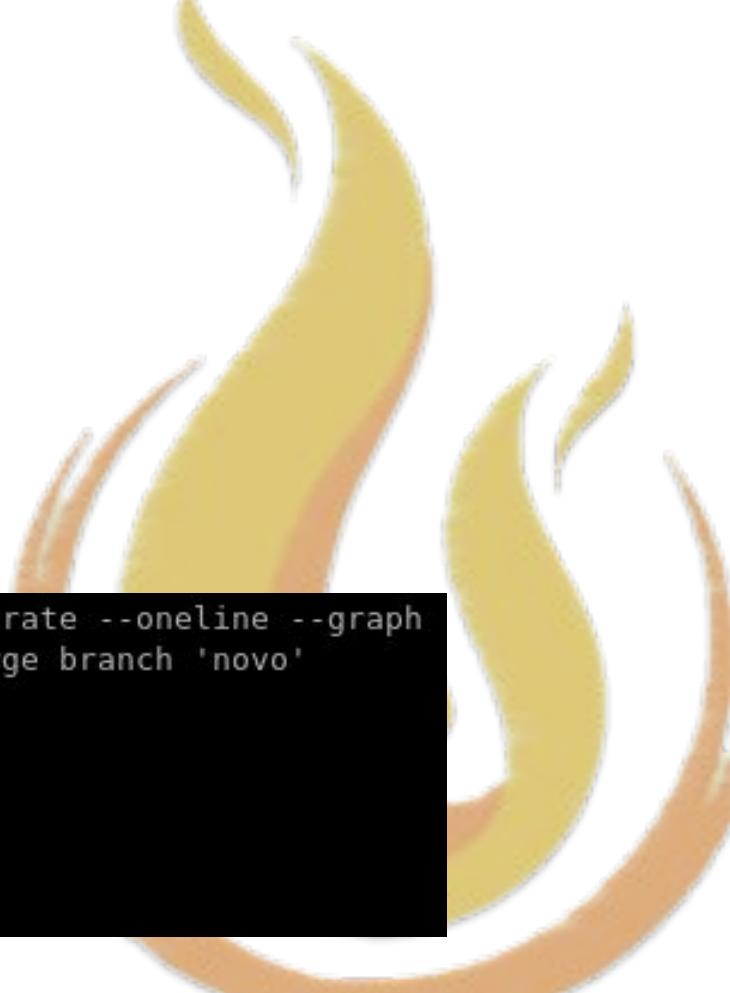
# 7 - Merging

## Tarefas:

- Use o comando:

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

```
[TarzanUSP@localhost git]$ git log --all --decorate --oneline --graph
*   b188f2a (HEAD -> master, origin/master) Merge branch 'novo'
|\ 
* | 9ec6bec (origin/novo, novo) novo
* |  df13cb6 test
|\ 
* a2668fb (tag: v1.0) primeiro commit
[TarzanUSP@localhost git]$
```



## 7 - Merging

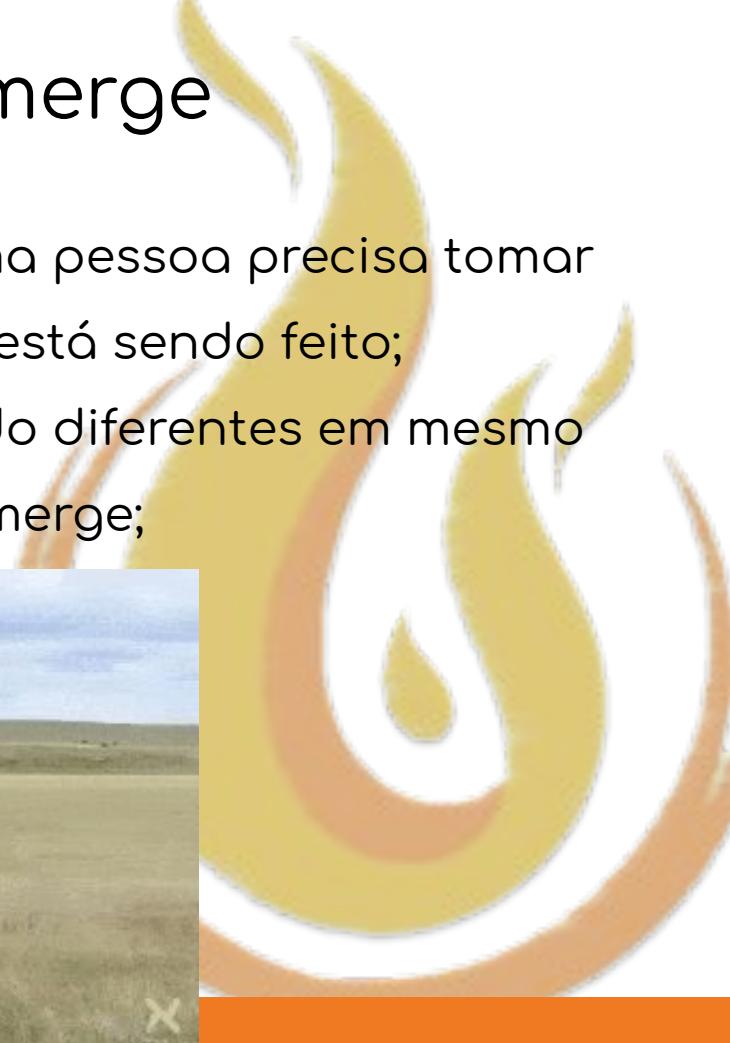




# Intervalo!

## 8 - Resolvendo conflitos de merge

- Conflitos de merge ocorrem quando uma pessoa precisa tomar uma decisão em relação ao merge que está sendo feito;
- Geralmente ocorrem por existir conteúdo diferentes em mesmo ponto de arquivos que serão parte do merge;

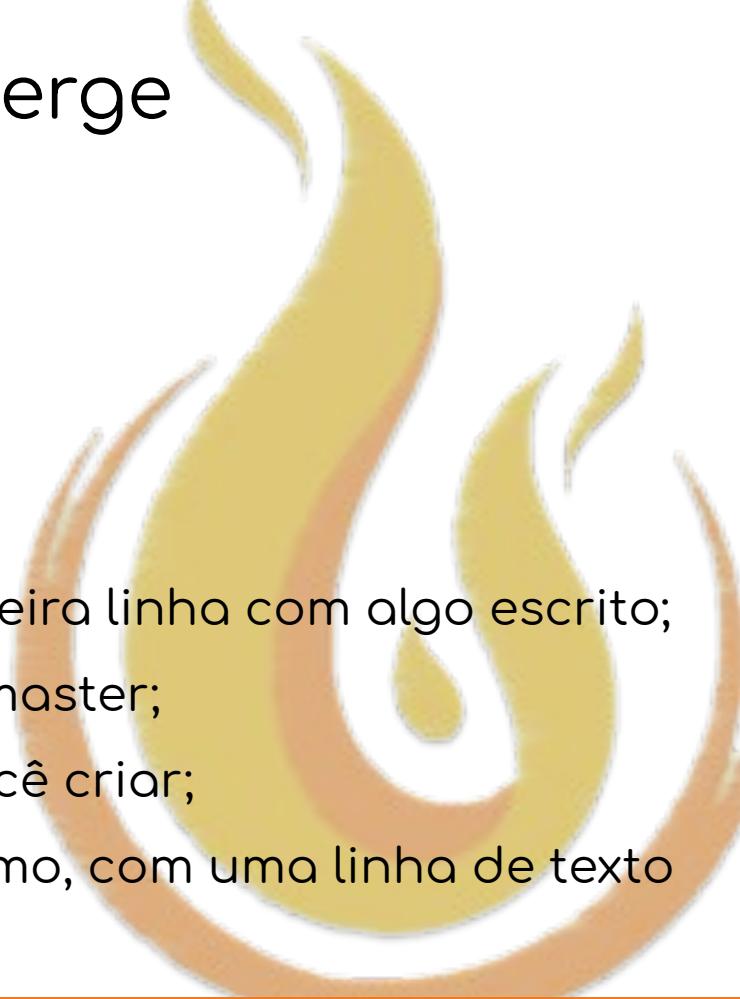


# 8 - Resolvendo conflitos de merge

- Vamos gerar conflitos!

## Tarefas:

- Crie um repositório do zero com git init;
- Adicione um arquivo de texto com a primeira linha com algo escrito;
- Dê o commit para o repositório local na master;
- Faça o mesmo só que em uma branch você criar;
- Volte para a branch master, e faça o mesmo, com uma linha de texto diferente;



# 8 - Resolvendo conflitos de merge

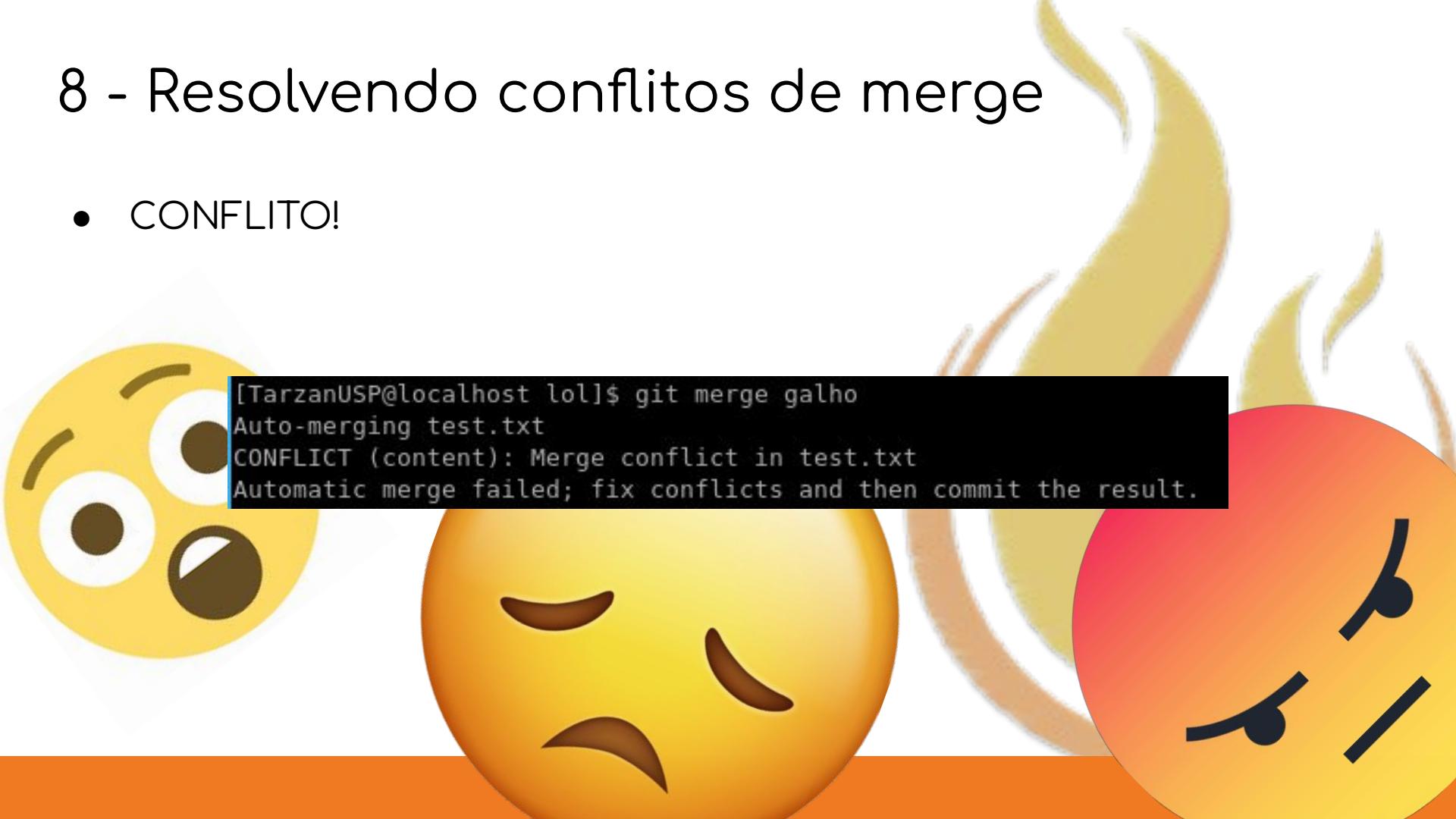
## Tarefas:

- Estando na branch master, use o git merge;

```
[TarzanUSP@localhost lol]$ git log --all --decorate --oneline --graph
* 002848e (HEAD -> master) master
| * e479ca5 (galho) galho
|/
* fd321d3 oi
```

# 8 - Resolvendo conflitos de merge

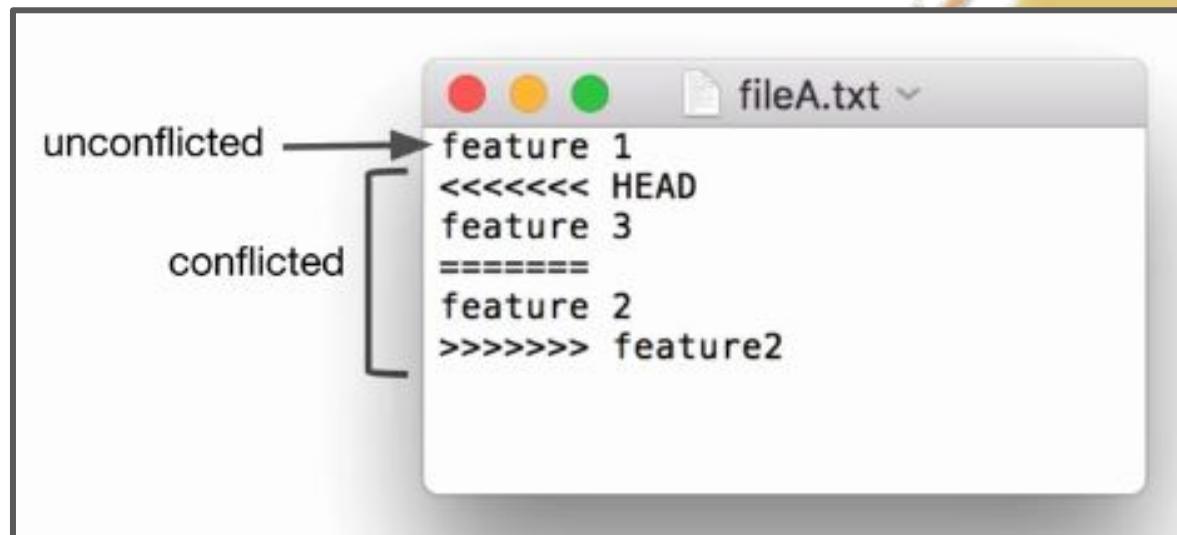
- CONFLITO!



```
[TarzanUSP@localhost lol]$ git merge galho
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

# 8 - Resolvendo conflitos de merge

- Basta apenas modificar o arquivo em questão com o que você realmente quer esteja no merge;

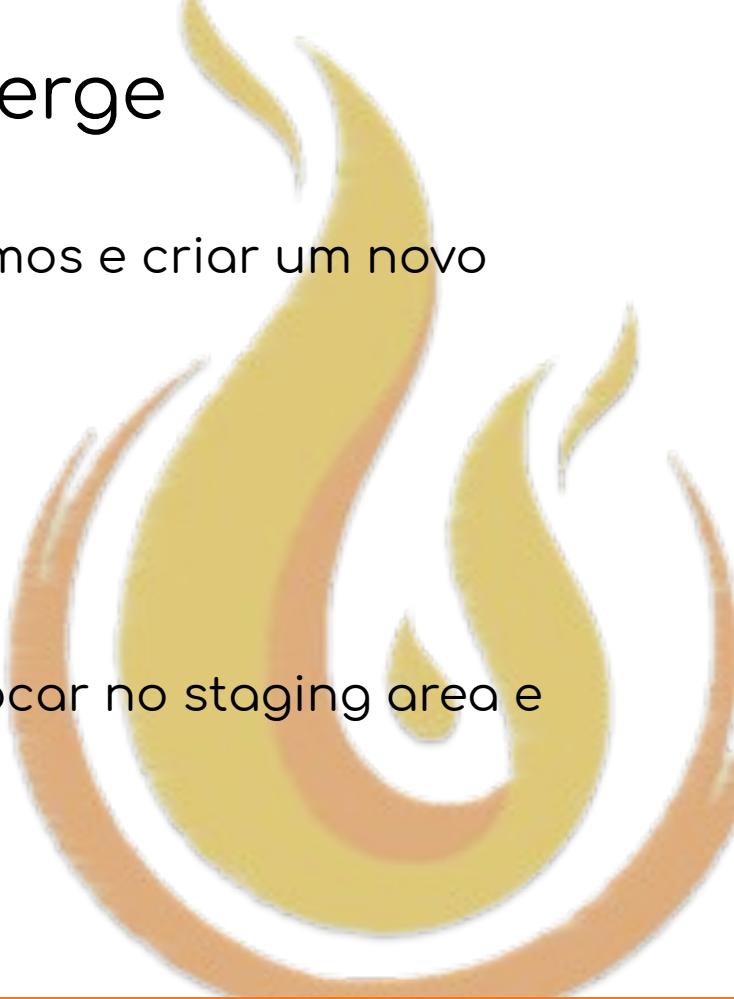


## 8 - Resolvendo conflitos de merge

- Vamos editar o arquivo com o que queremos e criar um novo commit!

### Tarefas:

- Edite o arquivo com o você quer;
- Faça os procedimentos básicos para colocar no staging area e consequentemente no repositório local;



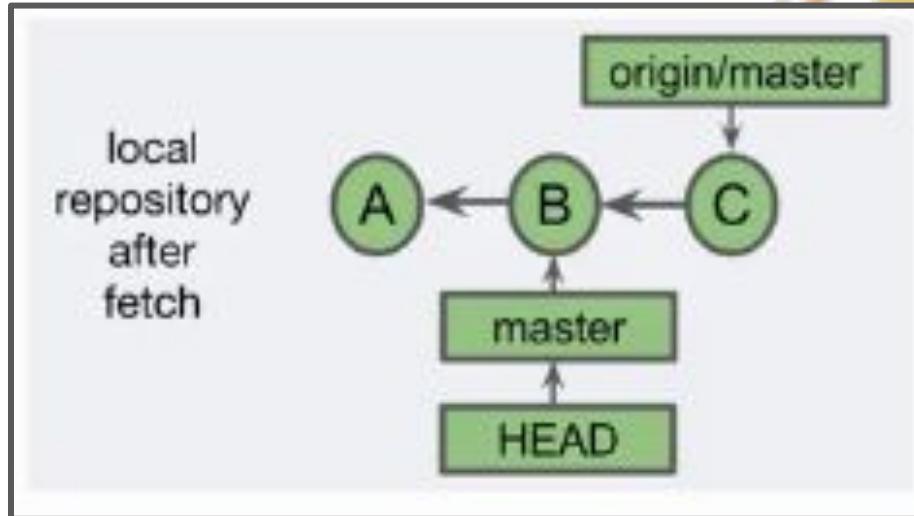
# 8 - Resolvendo conflitos de merge

- Devemos ter commits da seguinte forma:

```
[TarzanUSP@localhost lol]$ git log --all --decorate --oneline --graph
* ac5d150 (HEAD -> master) novo commit
|\ \
* | e479ca5 (galho) galho
* | 002848e master
|\ /
* fd321d3 oi
```

## 9 - Fetch

- Fetch é um comando do git para adquirir novos objetos e referências de um repositório;



## 9 - Pull

- Pull é praticamente um comando dois em um;
- git pull equivale a usar git fetch e git merge;



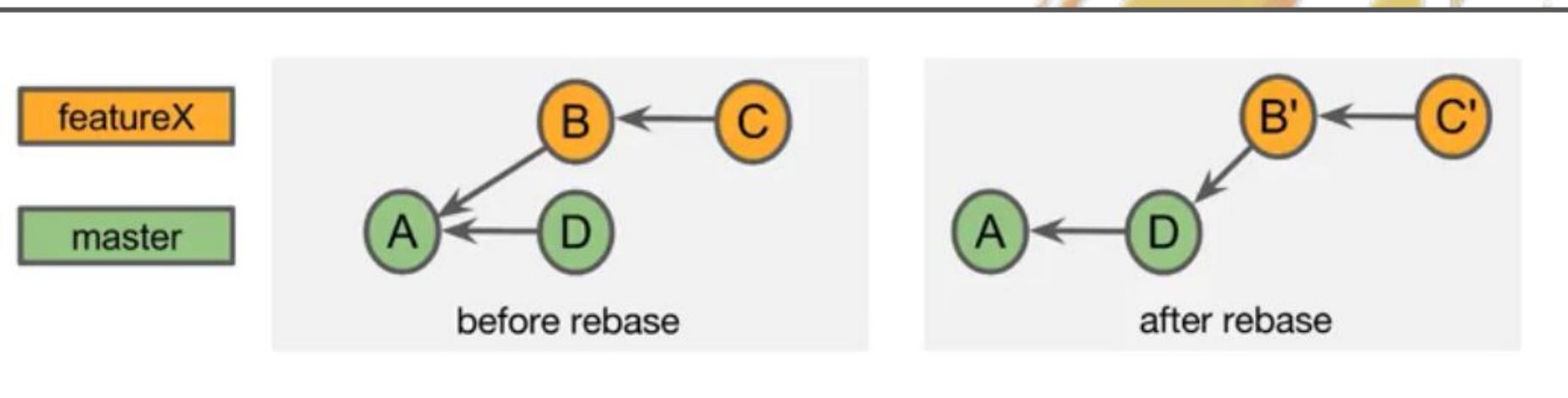
## 9 - Push

- Se tiver alterações no repositório sempre use um git pull antes de um git push;

```
(create a commit on the remote repository)
$ touch fileB.txt
$ git add fileB.txt
$ git commit -m "add fileB.txt"
[master 07c5e2a] add fileB.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 fileB.txt
$ git push
To https://bitbucket.org/me/projectf.git
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to 'https://me@bitbucket.org/me/projectf.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about Fast-forwards' in 'git push --help' for details.
```

# 10 - Rebase

- Como o nome diz, muda base, no caso, para outro commit;
- É também uma forma de merge;



## 10 - Rebase

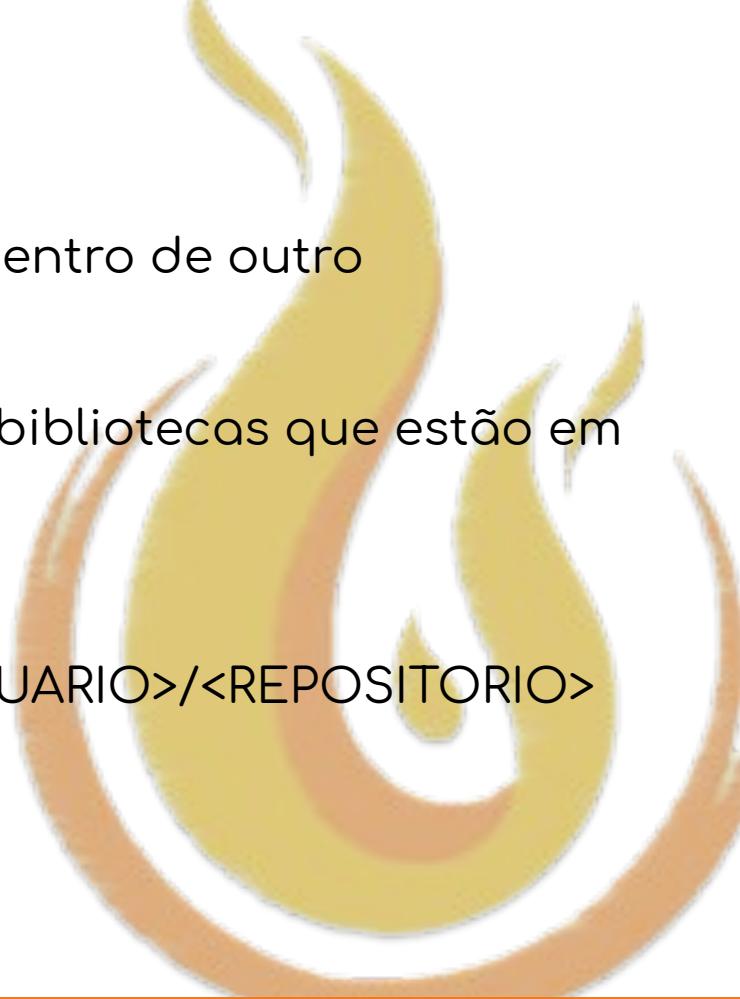
- Prós:
  - Você pode incorporar mudanças da branch pai;
  - Evita commits desnecessários;
- Contras:
  - Conflitos merges a serem resolvidos;
  - Você não preserva o histórico de commits;



# 11 - Submódulos

- Permite armazenar um repositório git dentro de outro repositório git;
- Útil para projetos que for implementar bibliotecas que estão em um repositório git;

```
git submodule add https://github.com/<USUARIO>/<REPOSITORIO>
```



## 12 - Git Workflows

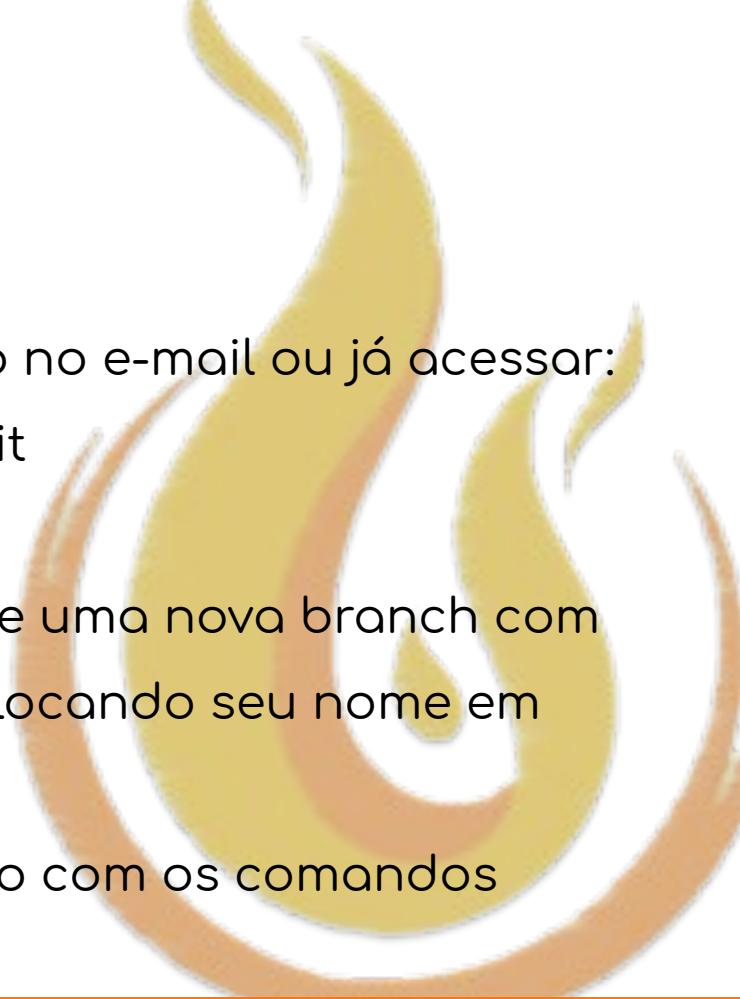
- É o conceito do trabalho em grupo usando git;
- Existem vários métodos:
  - Workflow centralizado;
  - Workflow com branches;
  - Workflow com fork;



# 12 - Git Workflows

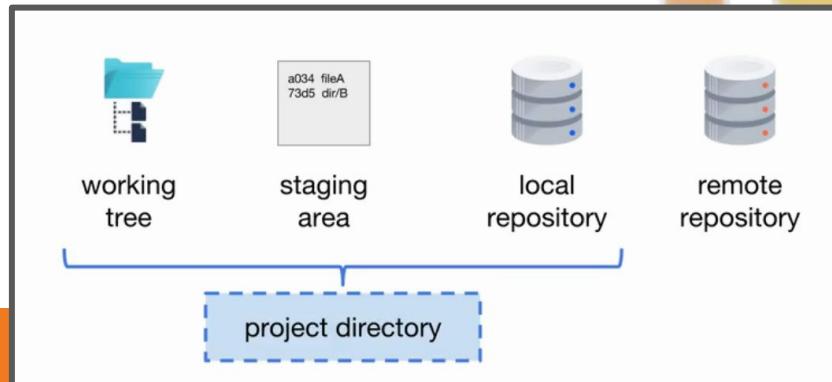
## Tarefas:

- Aceitar pedido de colaboração enviado no e-mail ou já acessar:  
<https://github.com/BrenoUSP/OficinaGit>
- Deem git clone para esse repositório;
- No terminal e dentro do repositório, crie uma nova branch com seu nome, edite o arquivo nomes.txt colocando seu nome em qualquer lugar;
- Coloque a branch no repositório remoto com os comandos básicos;



# Revisão

- Git se encaixa no conceito de DevOps;
- GitHub é um servidor para repositórios git;
- É possível criar repositórios local (git init) ou remotamente (sites como GitHub);
- O funcionamento de um repositório é assim:



# Revisão

- Para vários casos, basta git add ., git commit -m “MENSAGEM” e git push origin “NOME DA BRANCH”;
- Branches são práticas interessantes quando lidar com desenvolvedores diferentes;
- Conflitos de merges para projetos muitos grandes talvez terá que usar ferramentas como mergetool;





Dúvidas?



Obrigado pela  
presença!