





# Está em nosso DNA

## Missão

Promover transformações no mundo digital reconhecendo as necessidades das pessoas e das organizações

## Visão

Múltiplas soluções digitais que impactam negócios, simplificam a vida e integram pessoas

## Valores

Agimos potencializados pela **Ética**;

**Diversidade, Liberdade e Confiança**;

A **pessoa** primeiro, a **competência técnica** e o potencial, todos juntos;

Não dá só para se envolver, há que se **comprometer**;

**Paixão** por tecnologia, por aprender, ensinar e dar soluções inovadoras;

**Respeito** a si próprio e aos demais;

**Divirta-se** com seus dons e competências.

# Transformação de ponta a ponta



## Ideia / Problema

Entender o desafio a fundo, propor questões, pesquisar e começar a debater os primeiros caminhos.



## Protótipo / Solução

Transformar as ideias em uma solução viável. Formatar, testar, viabilizar.



## Design

Preparar a solução com a melhor usabilidade para o usuário, criando um produto funcional e atrativo.



## Desenvolvimento

Muita técnica e conhecimento envolvidos para rodar bem, sem "bugs" e trazendo o resultado esperado.



## Lançamento

Acompanhamos todo o processo de colocar a solução disponível e funcionando perfeitamente.



## Manutenção

Acompanhamento permanente e cuidados para eventuais ajustes e adaptações.

A diagram illustrating a Git branching strategy. It features a vertical grey line representing the main branch, with three grey circular commit markers. A blue line branches off from the first marker, moves up, and then branches off again. A yellow line branches off from the second marker, moves up, and then branches off again. The lines end in colored circles (blue and yellow) representing the current state of the branches.

# Introdução do git

\$ git init .

\$ echo \$(whoami)

- Paulo Fernandes
- Formado em Sistemas de Informação pela PUC Minas
- Desenvolvedor na Raro Labs
- <https://github.com/pauloFernandes>

```
$ git config --global user.name "Paulo Fernandes"
```

```
$ git config --global user.email phfernandespereira@gmail.com
```



git is...

"a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."

<https://git-scm.com/>

git is...

"a free and **open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency."



<https://git-scm.com/>



git is...

"a free and **open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency."

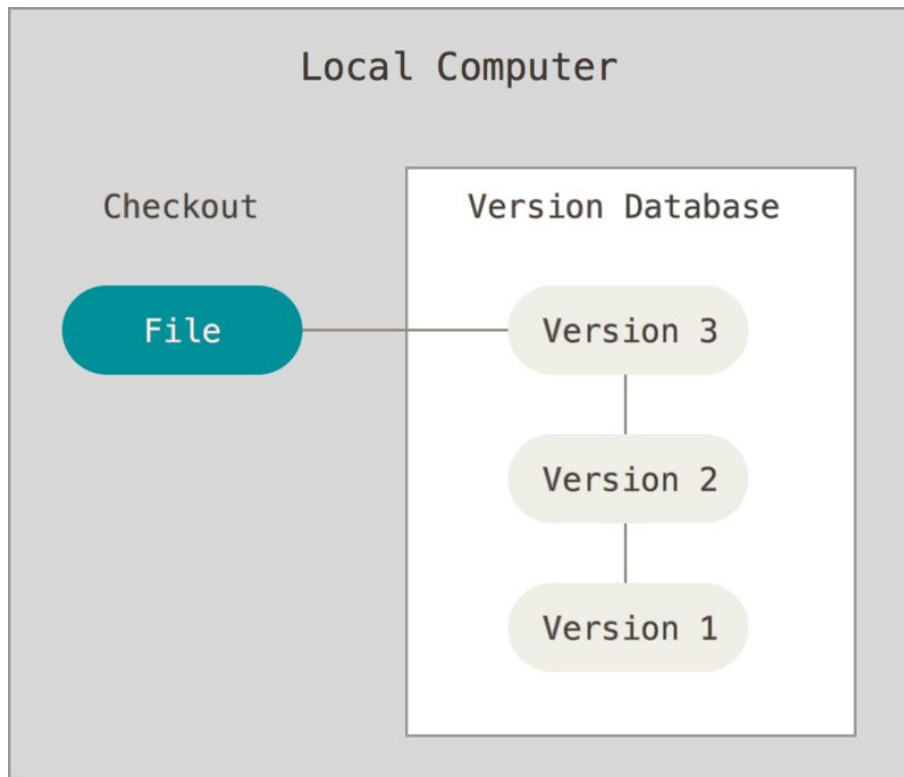


<https://git-scm.com/>

git is...

"a free and open source distributed **version control system** designed to handle everything from small to very large projects with speed and efficiency."

<https://git-scm.com/>



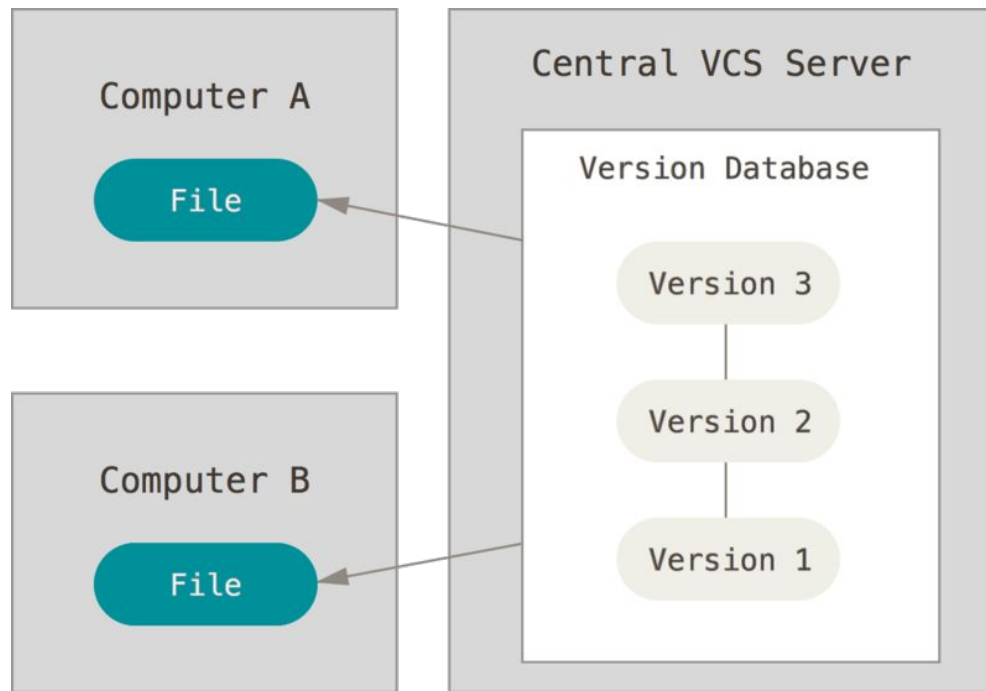
<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

git is...

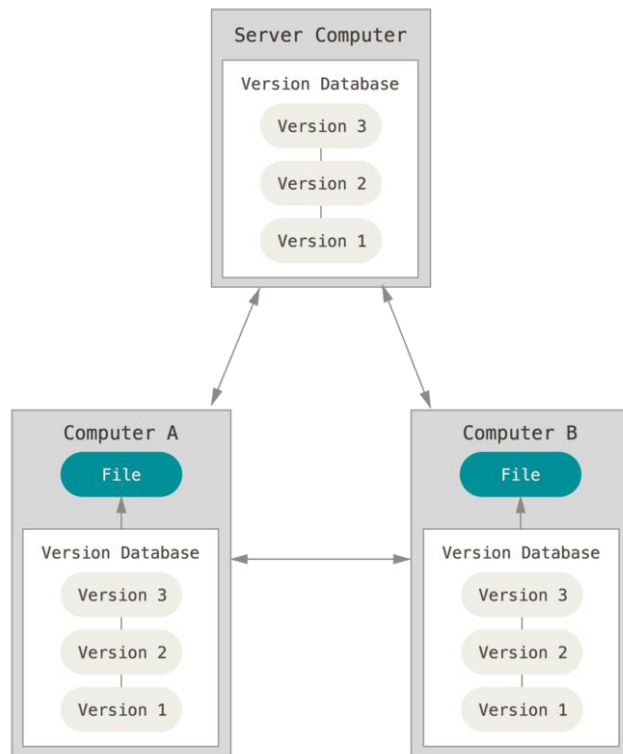
"a free and open source **distributed** version control system designed to handle everything from small to very large projects with speed and efficiency."

<https://git-scm.com/>

# Controle de versão centralizado



# Controle de versão distribuído



# \$ git --help

These are common Git commands used in various situations:

start a working area

clone

init

work on the current change

add

checkout

examine the history and state

status

diff

log

show

grow, mark and tweak your common history

branch

commit

merge

reset

rebase

collaborate

fetch

pull

push

# Resumo

## Inicialização

git init  
git clone

## Comparação

git status  
git diff  
git log  
git show

## Versionamento

git add  
git commit

## Branching

git checkout  
git merge  
git log  
git reset  
git cherry-pick

## Colaboração

git fetch  
git pull  
git push



# Iniciando um repositório

# Criando um repositório localmente

```
$ git init .
```

# Clonando um repositório já existente de uma réplica remota

```
$ git clone git@github.com:pauloFernandes/try_git.git
```

# Estados dos arquivos

- untracked
- staged
- modified
- committed

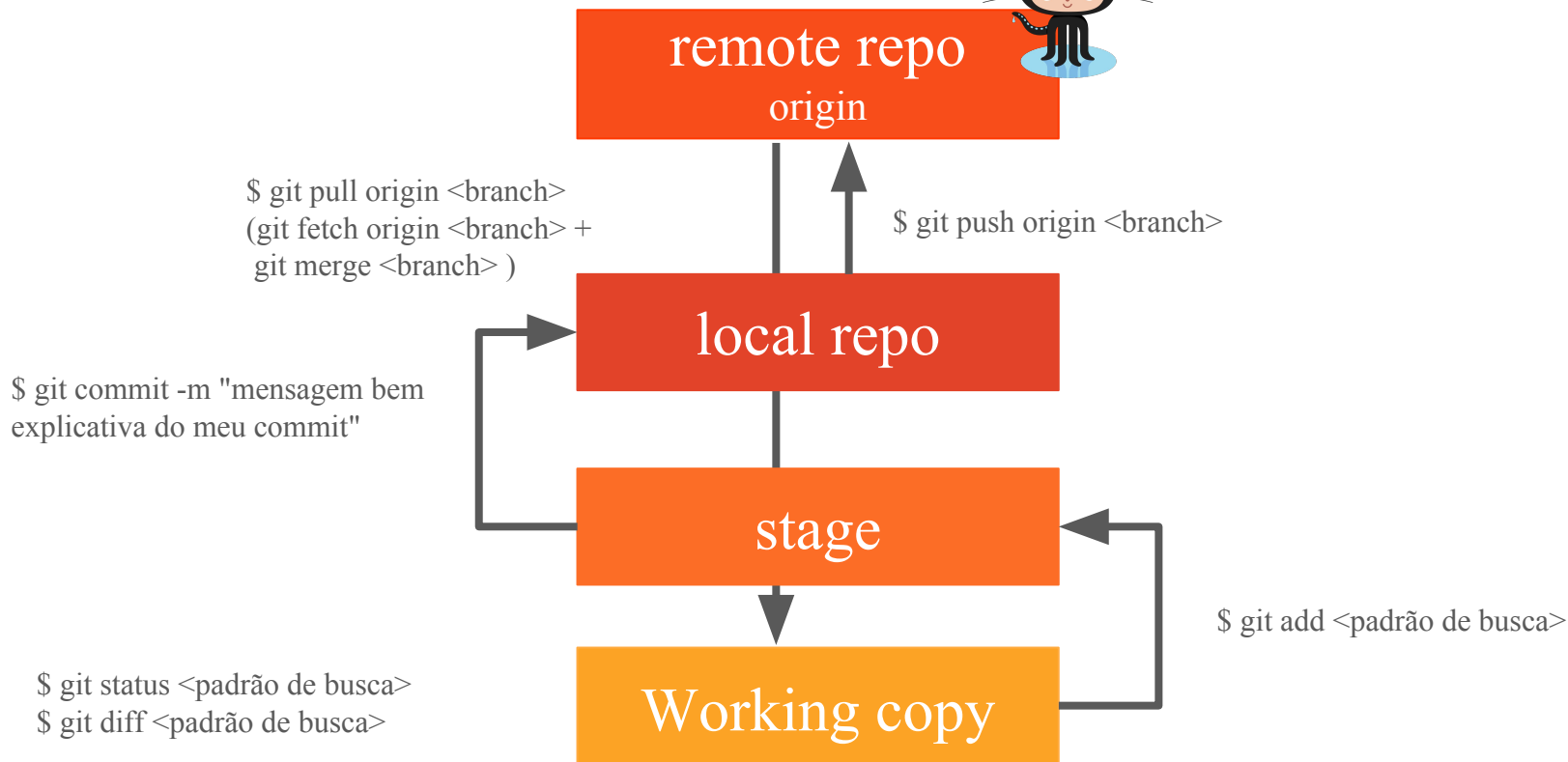
```
→ ip-manager git:(master) X git status .
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   data/iplist.json
        modified:   package-lock.json
        modified:   package.json

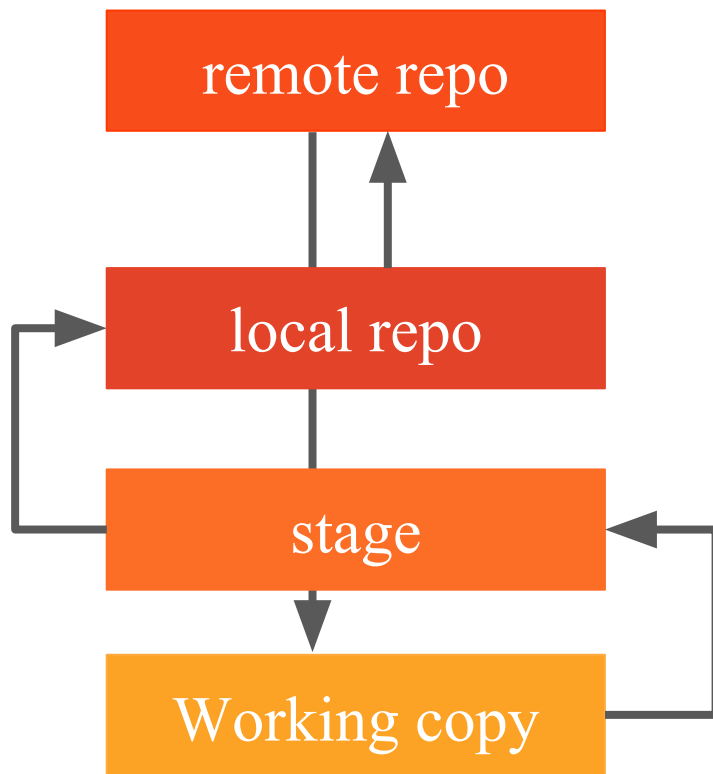
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ola.txt
```

# git workflow básico

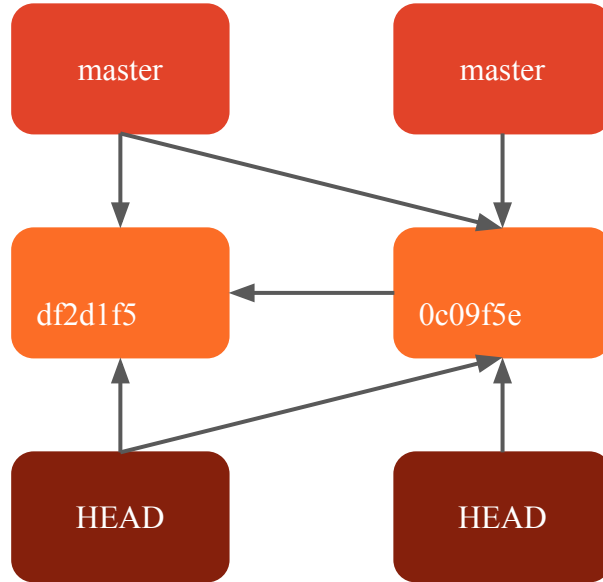


# git workflow básico

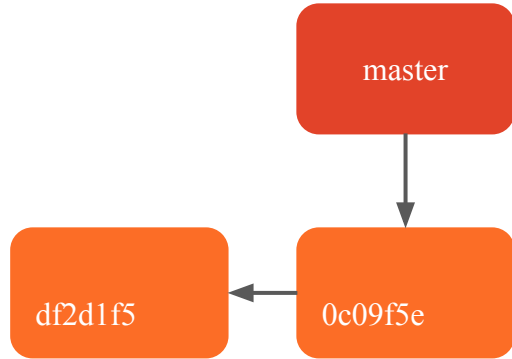
- git status
- git diff
- git add
- git commit
- git push
- git pull



\$ git commit -m 'registrando mudanças no repo'

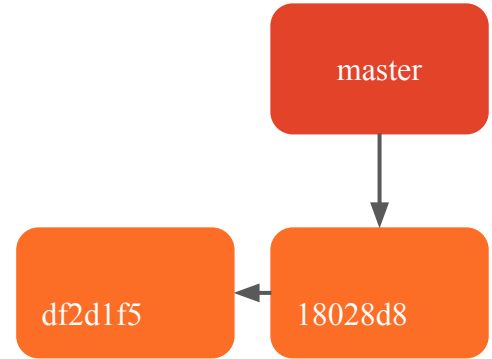
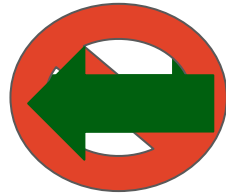


# git push origin master



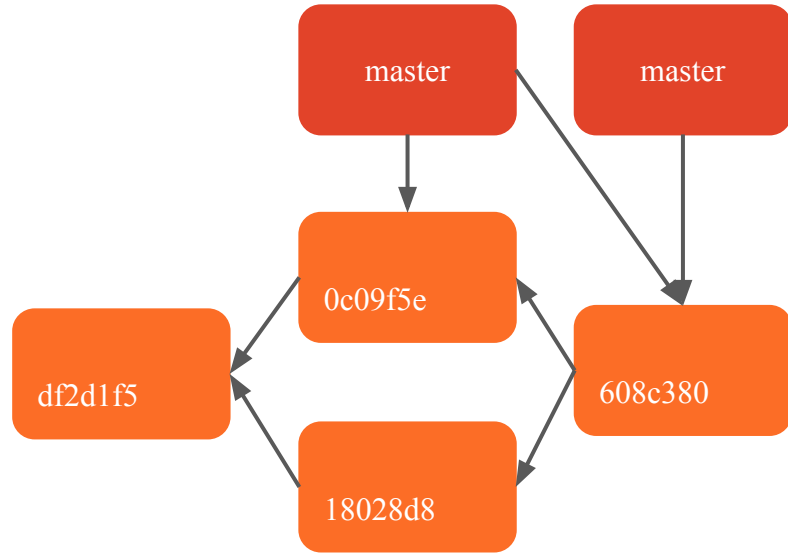
Repositório Local

\$ git push origin master



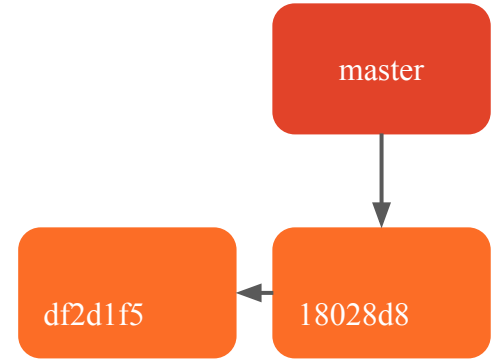
Repositório Remoto

# git push origin master



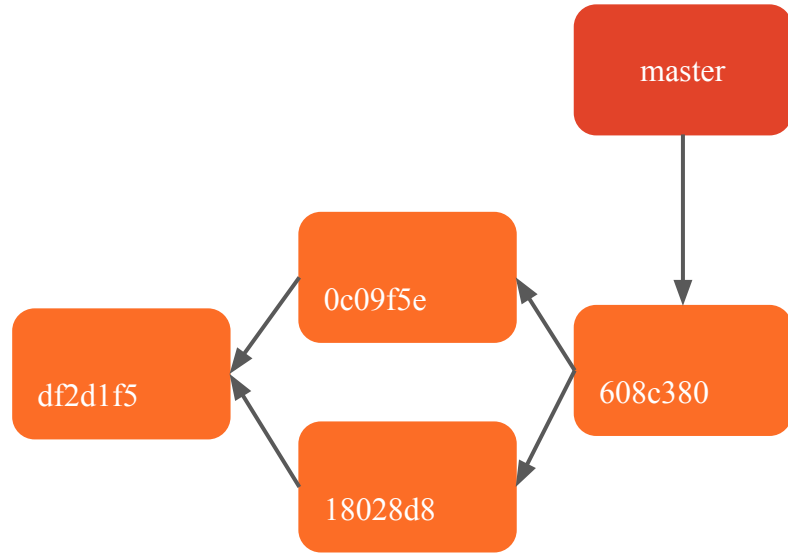
Repositório Local

\$ git push origin master

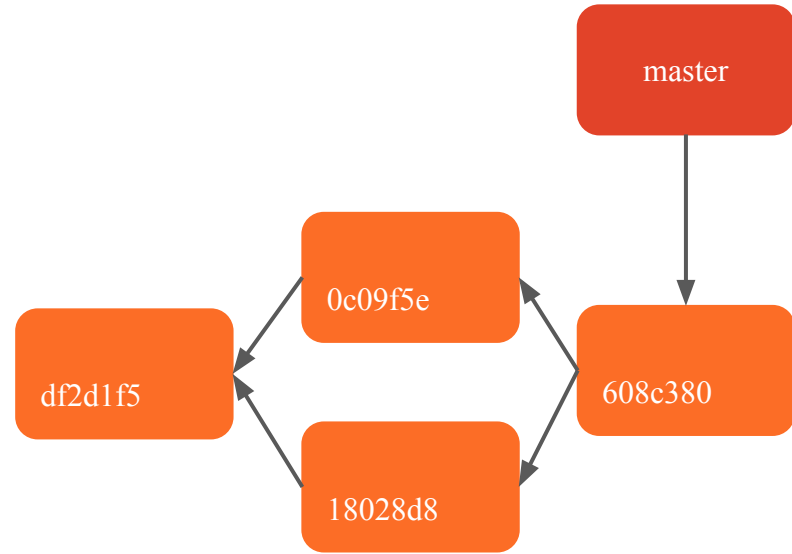
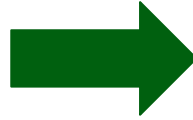


Repositório Remoto

git push origin master



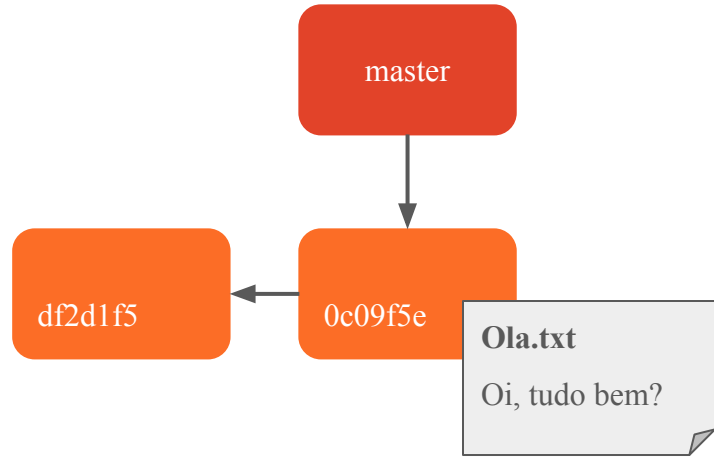
Repositório Local



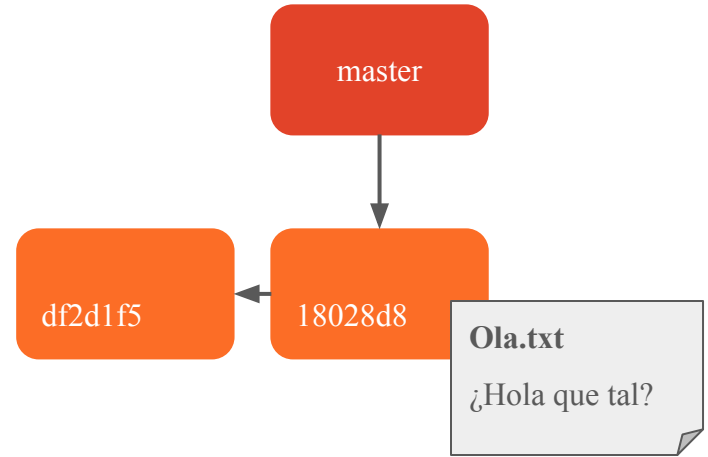
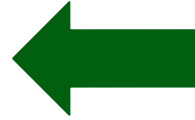
Repositório Remoto



# git pull origin master

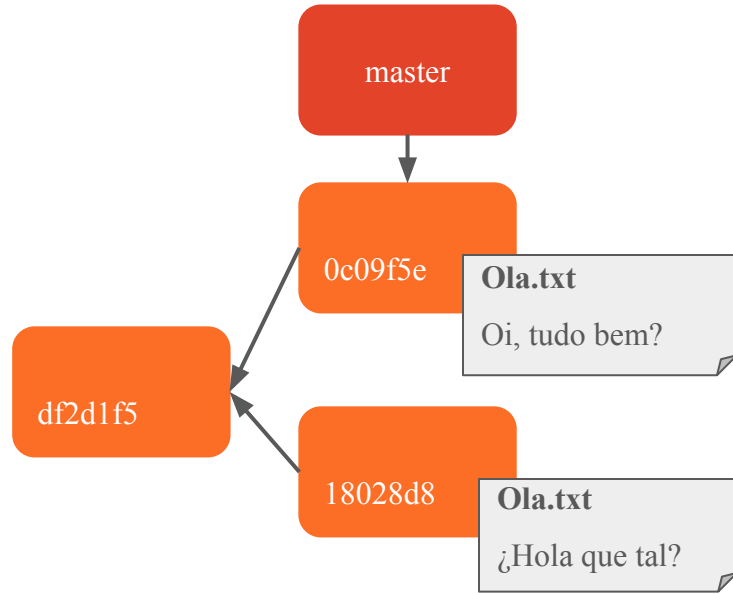


Repositório Local

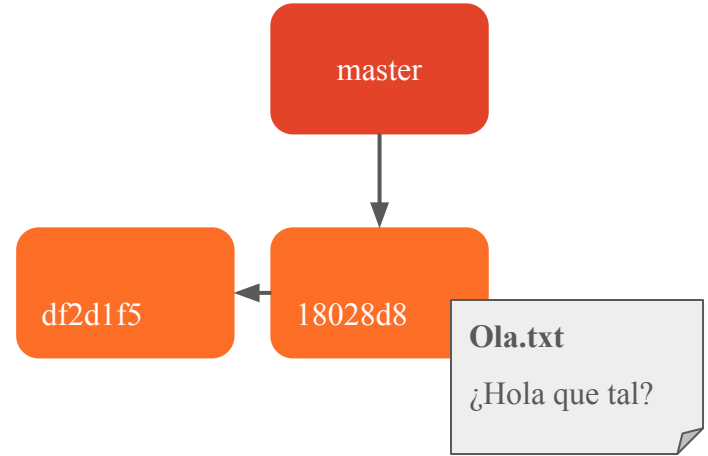
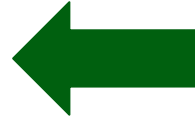


Repositório Remoto

# git pull origin master



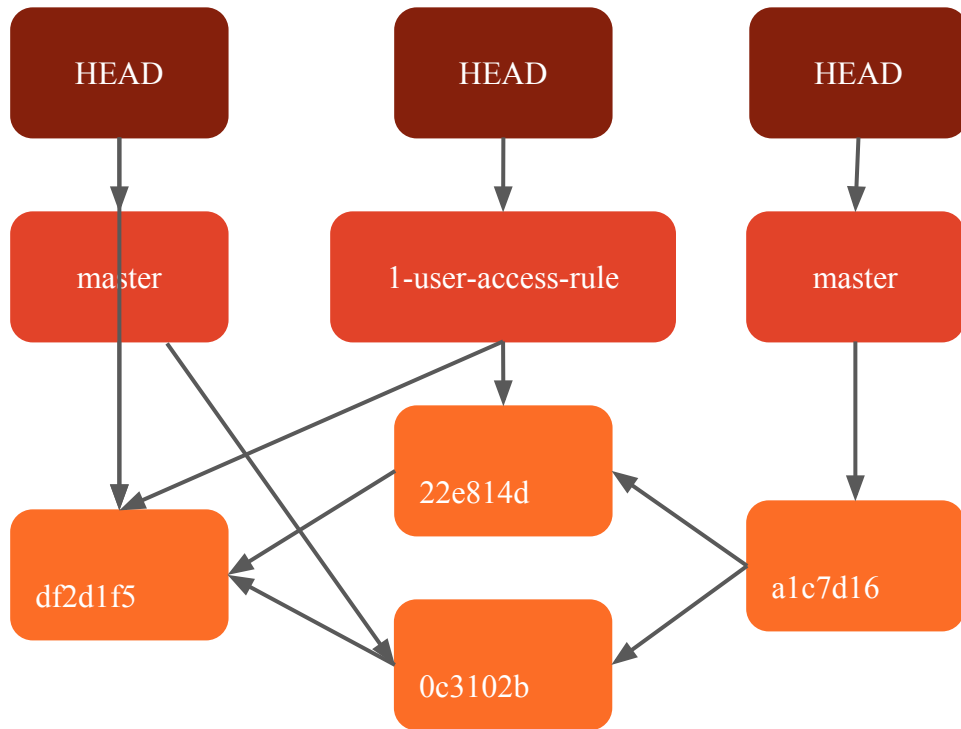
Repositório Local



Repositório Remoto



# Branching, Merging and checking out



```
$ git checkout -b 1-user-access-rule
```

```
# Executar as devidas alterações
```

```
$ git add .
```

```
$ git commit -m 'closes #1'
```

```
$ git checkout master
```

```
# Executar as devidas alterações
```

```
$ git add .
```

```
$ git commit -m 'modifica algo em master'
```

```
$ git merge 1-user-access-rule
```

```
$ git checkout df2d1f5
```

# Branching, Merging and checking out

- git checkout
- git merge
- git branch
- git log
- git show

```
commit 1f30b9cdf022eaa9132e85ba3be64f9d9959886 (HEAD -> master, origin/master)
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Tue Jun 29 23:52:10 2021 -0300

    fixes boolean extension

commit 5b8b6870b452bfdd34d2c4838545593139b6e7f8
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Tue Jun 29 23:45:34 2021 -0300

    small refactor to serializer

commit 68dc676cf6617984264e2fbacae9e54974046249
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Tue Jun 29 23:43:23 2021 -0300

    refactors deserializers

commit f067e4abf641f22a8e2b731763e354701a89932a
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Tue Jun 29 22:32:54 2021 -0300

    adds node serializer/deserializer

commit 3f780c0a2a1e02b0373f53b82bab62cdaf7984d7
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Mon Jun 28 21:58:47 2021 -0300

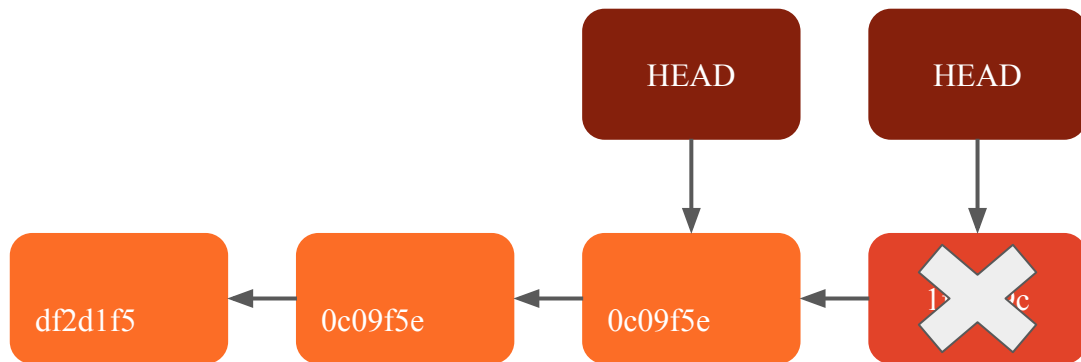
    adds operations and tests

commit ed4251a5723a52141a3e6b2186885eca88ce8d9
Author: Paulo Henrique Fernandes Pereira <phfernandespereira@gmail.com>
Date: Sun Jun 27 21:00:30 2021 -0300

    creates a real example
```

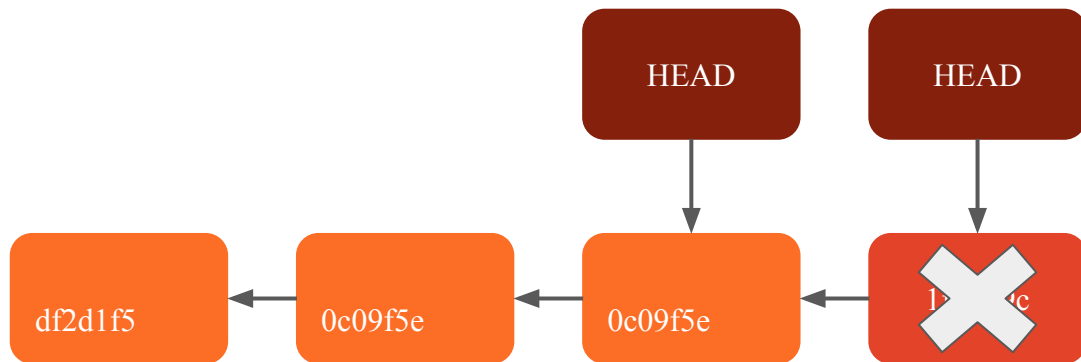
\$ git reset

- git reset
- git reset --hard
- git reset HEAD~1
- git reset HEAD~1 --hard

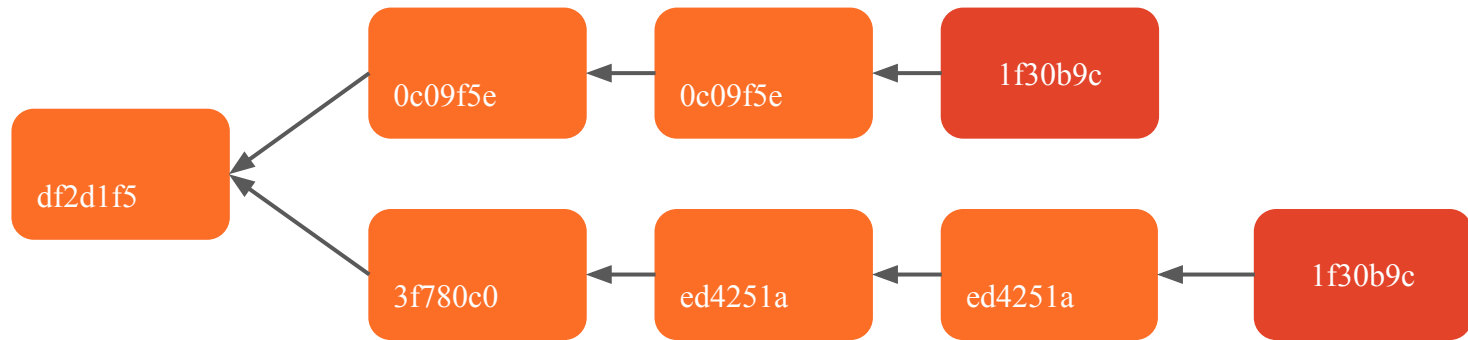


\$ git reset

- git reset
- git reset --hard
- git reset HEAD~1
- git reset HEAD~1 --hard



\$ git cherry-pick



\$ git cherry-pick 1f30b9c



# Resumo

## Inicialização

git init  
git clone

## Comparação

git status  
git diff  
git log  
git show

## Versionamento

git add  
git commit

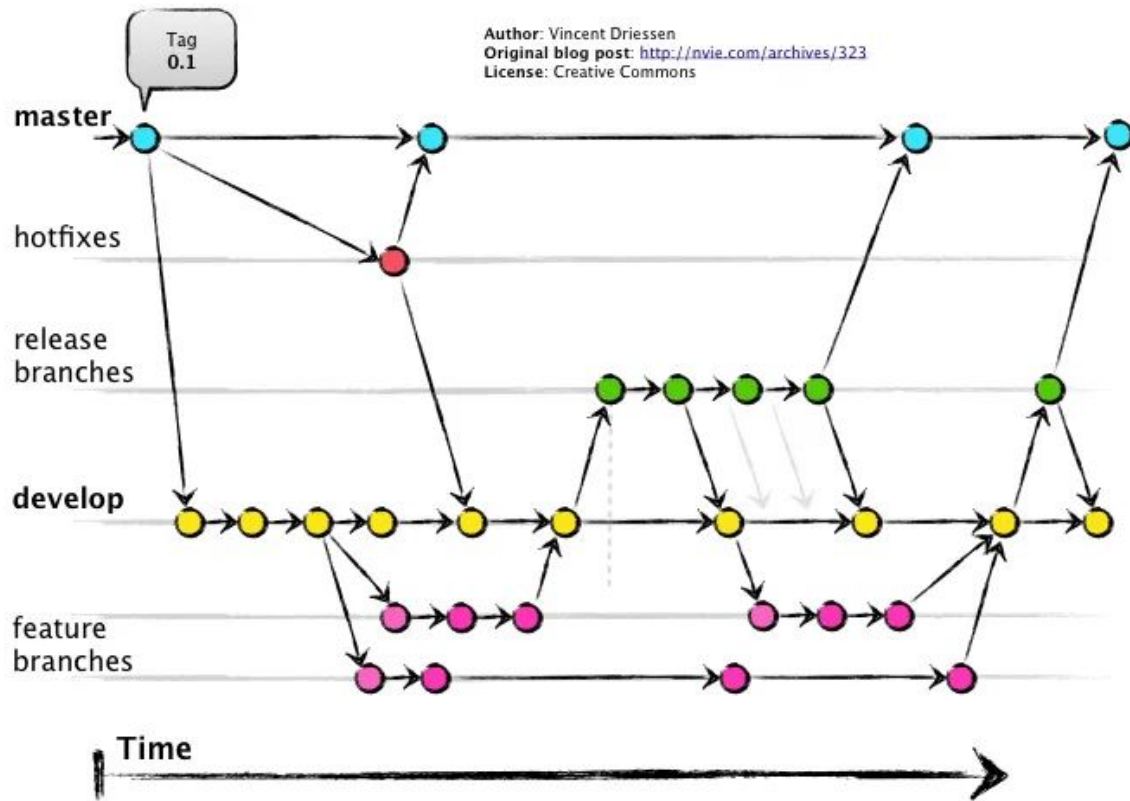
## Branching

git checkout  
git merge  
git log  
git reset  
git cherry-pick

## Colaboração

git fetch  
git pull  
git push

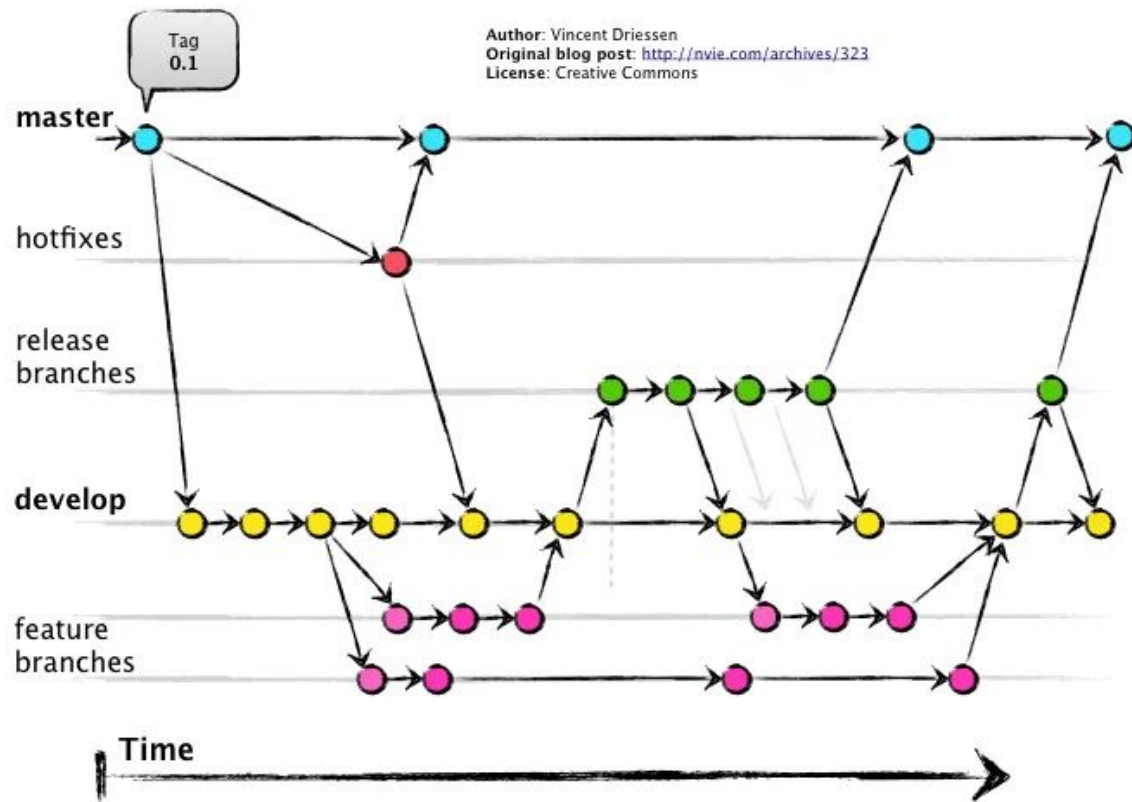
# gitflow



gitflow

"Gitflow Workflow is a Git workflow that helps with continuous software development and implementing DevOps practices."

# gitflow



# Bônus - Customizações

```
$ git short # git status --short --branch
```

```
$ git loggy # log --oneline
```

```
$ git tree # log --oneline --graph --decorate
```

```
$ git wannadd # add --intent-to-add
```

```
$ git padd # add -p
```

```
$ git review # commit -m \"code review\"
```

```
$ git wip # commit -m \"wip\"
```

```
$ git closes # "!f() { msg=#${1}; git commit -m \"closes $msg\"; }; f"
```

# Dicas pessoais

- Saiba exatamente o que está commitando. Evite o `git add .`;
  - Use git `git diff` e `git status` em todos os arquivos
- Saiba exatamente o que está fazendo! Evite perder um dia de trabalho por um comando incorreto
  - Dica altamente pessoal mas... Use o bash, não interfaces gráficas
- Seja claro nas mensagens de commit e nos nomes dos branches;
- Crie padrões de nomenclatura e mensagens: utilize isso a seu favor;

# Dicas pessoais

- Faça commits de pequenos blocos lógicos. Não espere finalizar todo o trabalho antes de versiona-lo: use os commits como uma lista das atividades que foram executadas;
- Não termine o dia antes de fazer um commit e push: garanta que nada do seu trabalho será perdido
  - Se necessário, crie outro branch para guardar o trabalho incompleto. Quando estiver pronto, faça o merge ao seu branch de trabalho

# Dicas pessoais

- Use branches para tudo
  - Nova funcionalidade;
  - Experimentos;
  - Backup de código;
  - Suporte para merges arriscados;



# Dicas pessoais

- Se for necessário trocar de branch/funcionalidade, não tenha medo de commitar, inclusive trabalho não concluído. Isso pode ser posteriormente revertido

```
$ git add .
```

```
$ git commit -m 'work in progress'
```

```
$ git checkout 'branch-com-outra-demanda'
```

```
$ git checkout 'minha-branch-de-trabalho'
```

```
# Remove o texto commitado em "wip" e coloca de volta em "not staged"
```

```
$ git reset HEAD~1
```

# Dicas pessoais

- Use git para tudo. Torne-o divertido

