RARO labs

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

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

Agimos potencializados pela **Ética**;

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

Diversidade, **Liberdade** e **Confiança**;

Respeito a si próprio e aos demais;

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

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.

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...

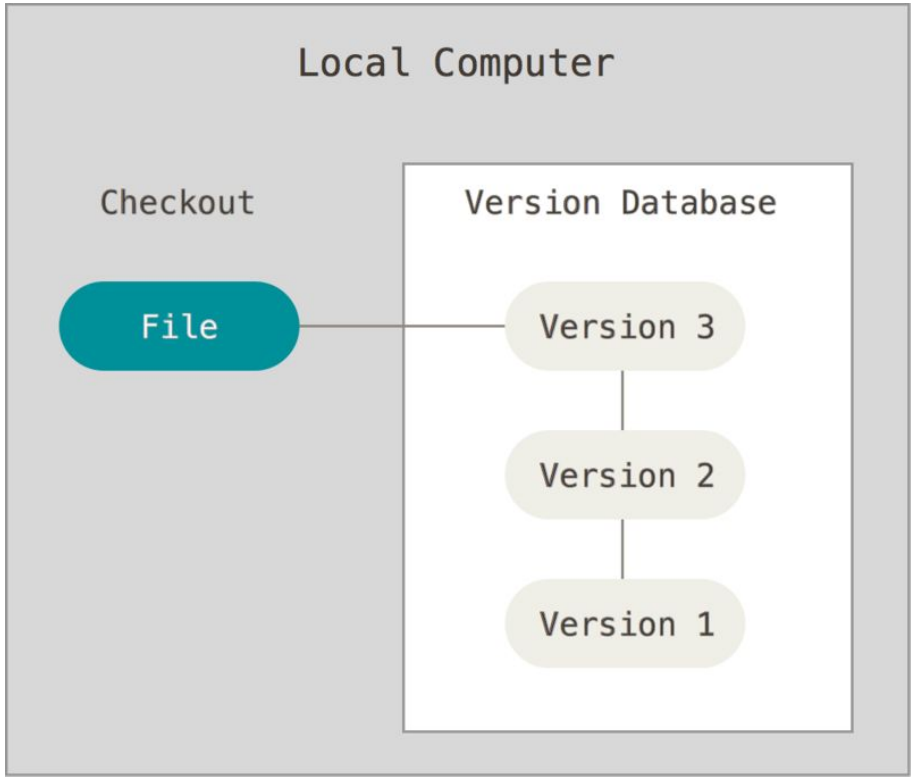
"a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."

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/



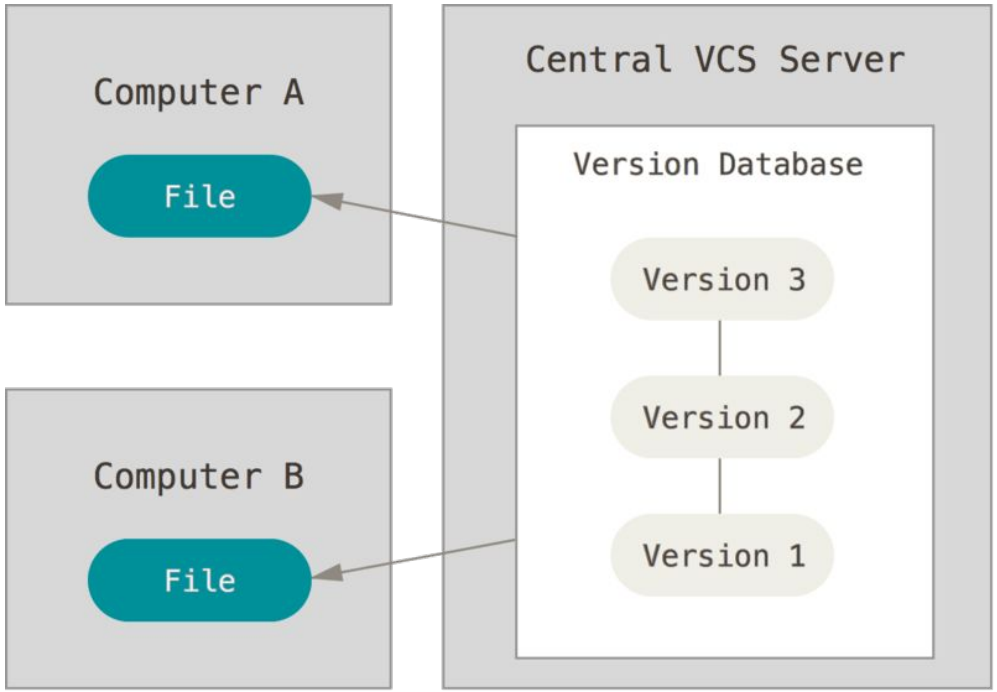
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

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."

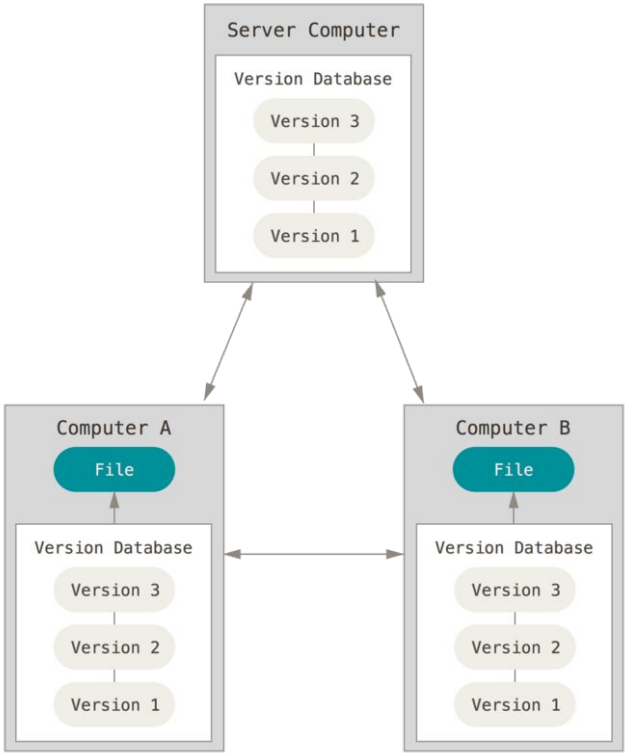
https://git-scm.com/

Controle de versão centralizado



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

Controle de versão distribuído



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

$ 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

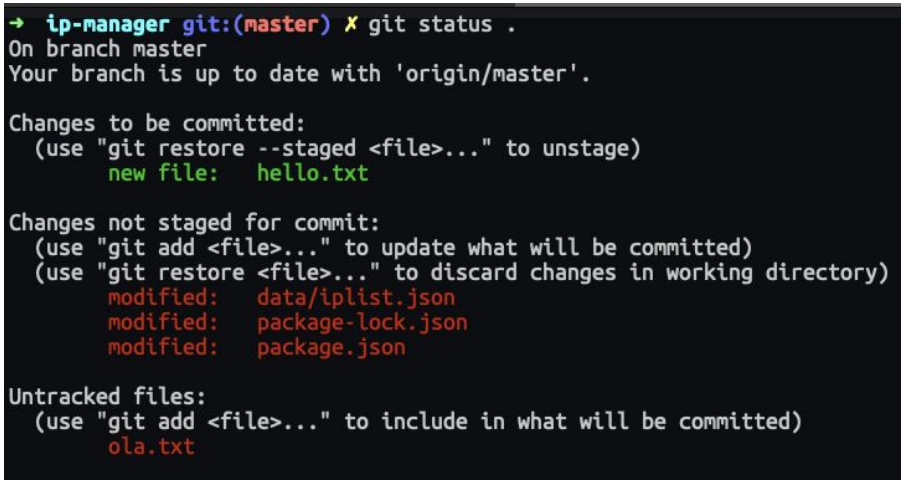
git fetch
git pull
git push

Iniciando um repositório

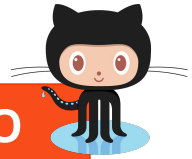
```
# Criando um repositório localmente
$ git init .

# 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

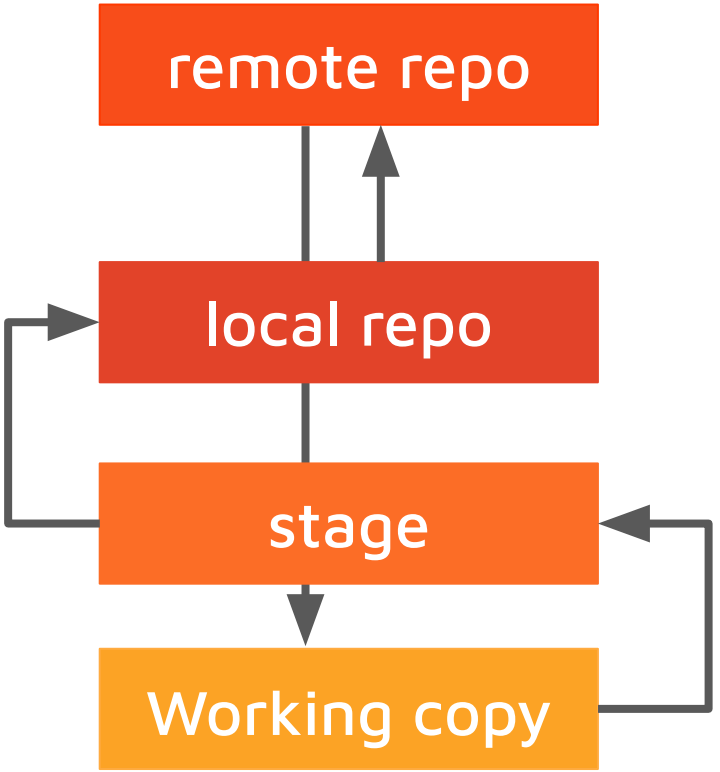


git workflow básico

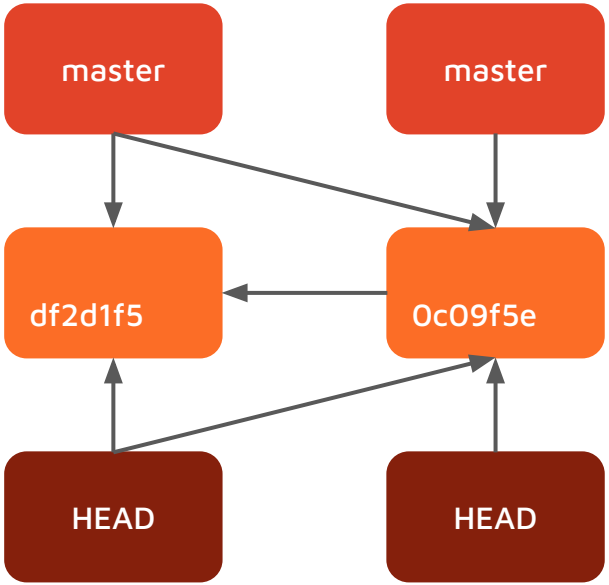


git workflow básico

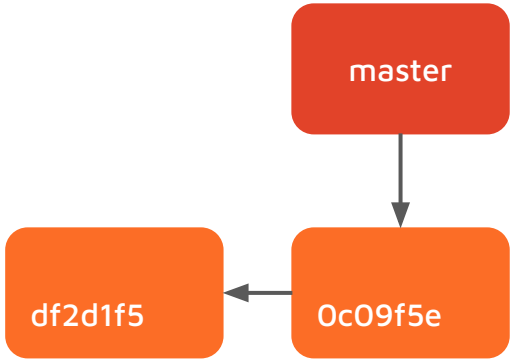
- git status
- git diff
- git add
- git commit
- git push
- git pull

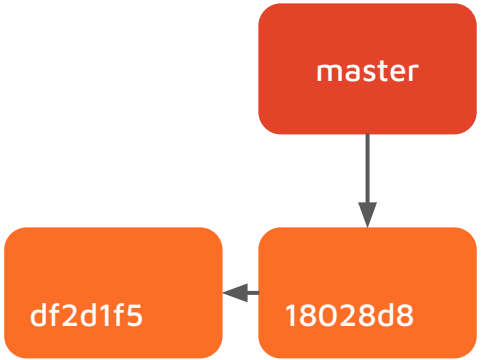


$ git commit -m 'registrando mudanças no repo'



git push origin master

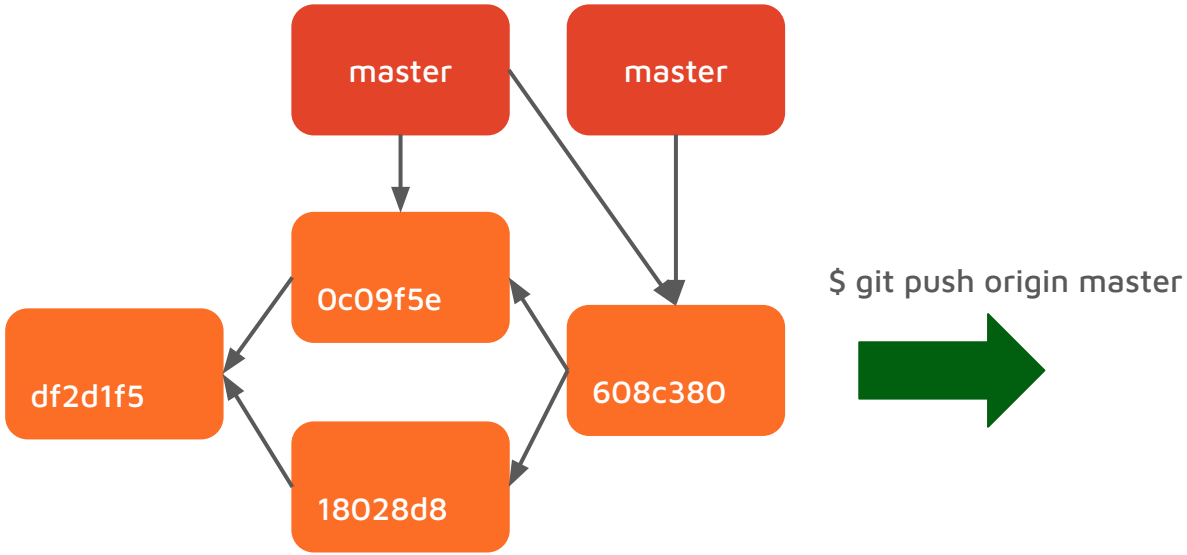




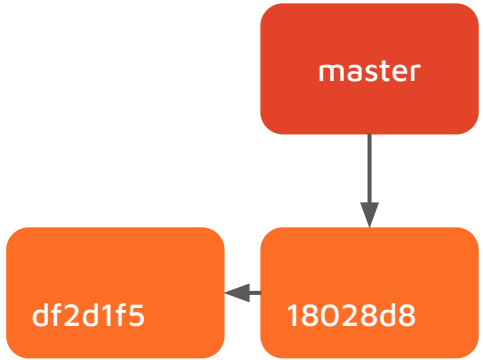
Repositório Local

Repositório Remoto

git push origin master



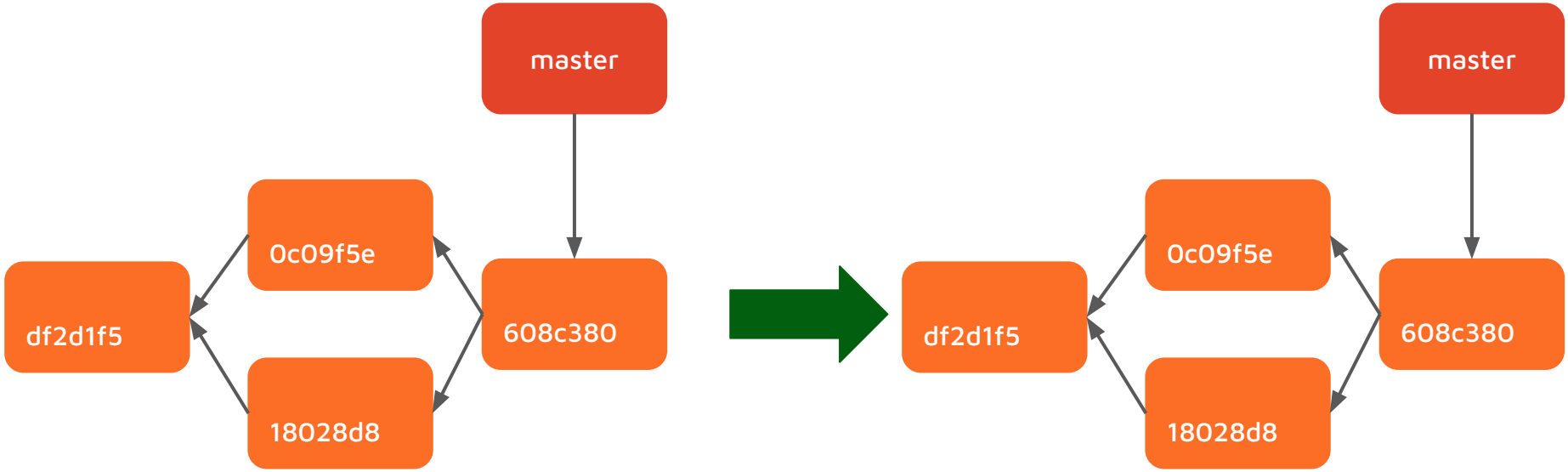
$ git push origin master



Repositório Local

Repositório Remoto

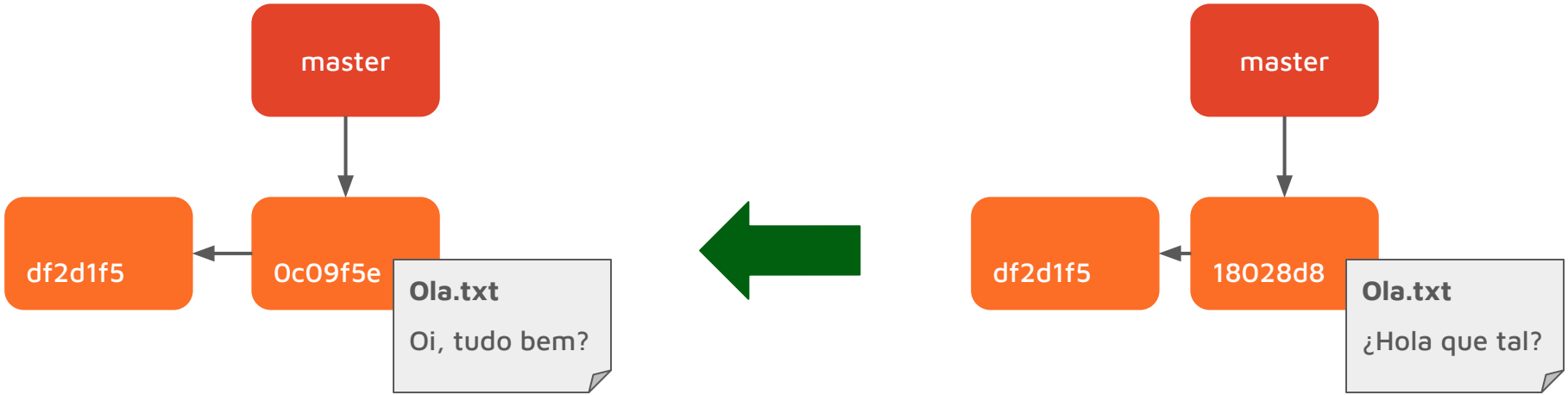
git push origin master



Repositório Local

Repositório Remoto

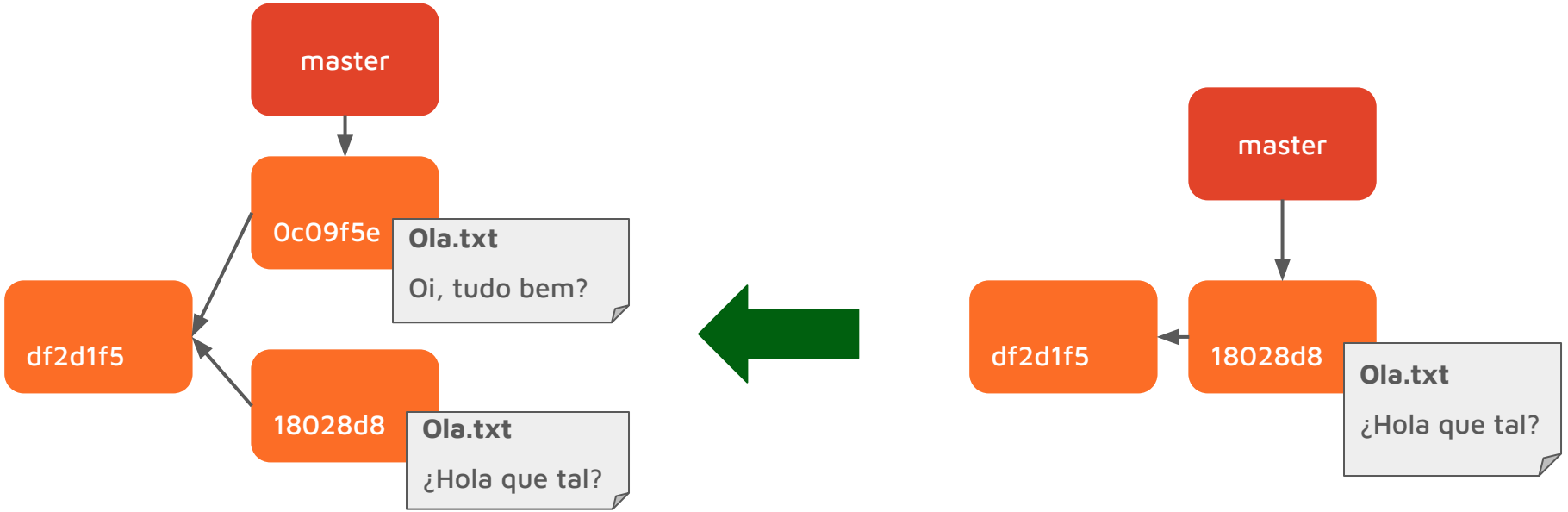
git pull origin master



Repositório Local

Repositório Remoto

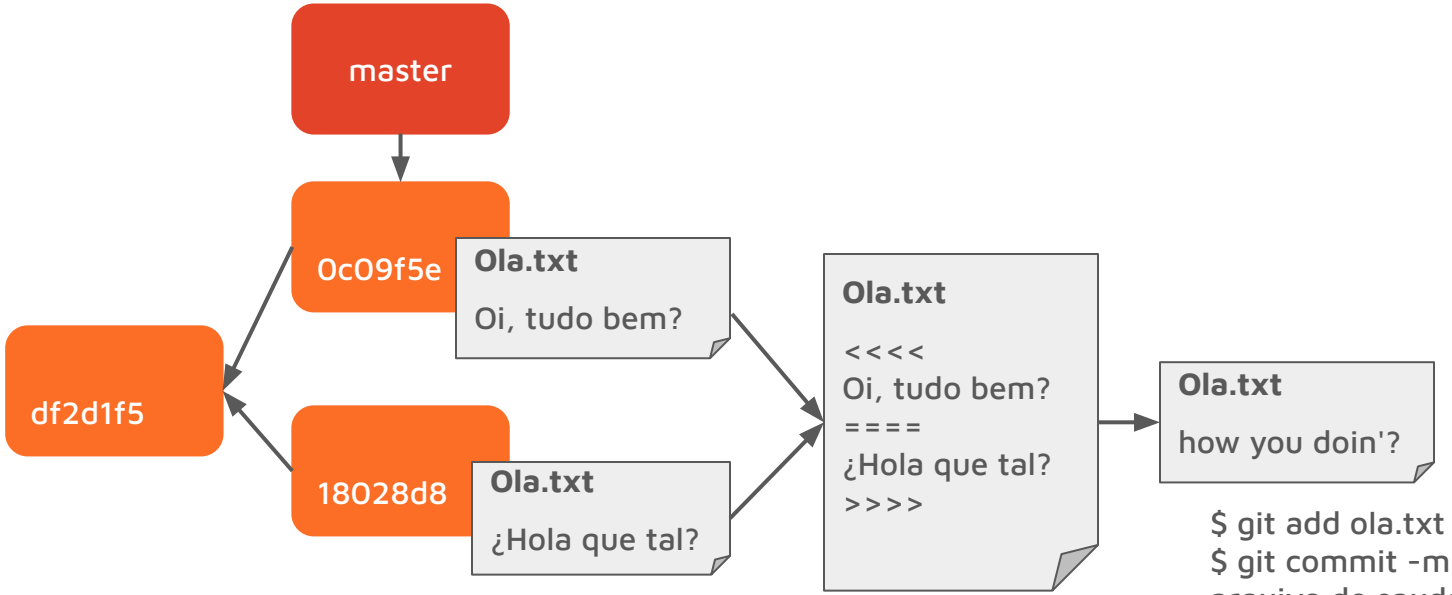
git pull origin master



Repositório Local

Repositório Remoto

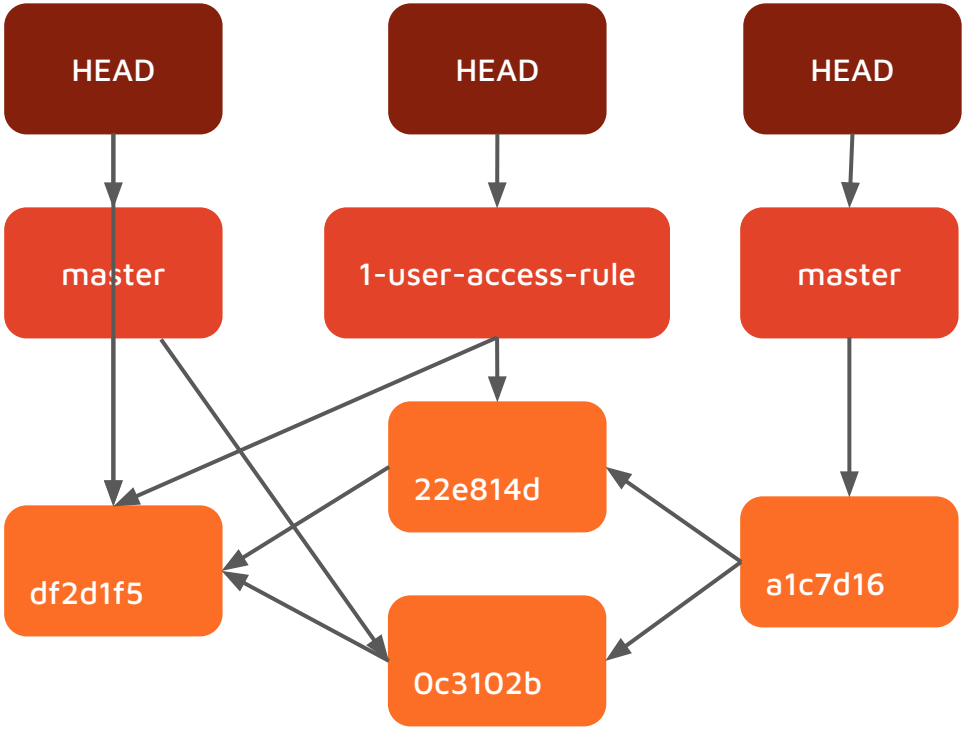
Conflitos!!!




```
$ git add ola.txt
$ git commit -m "resolvendo o conflito do
arquivo de saudação"
# git commit neste caso é válido!
$ git push origin master
```

Repositório Local

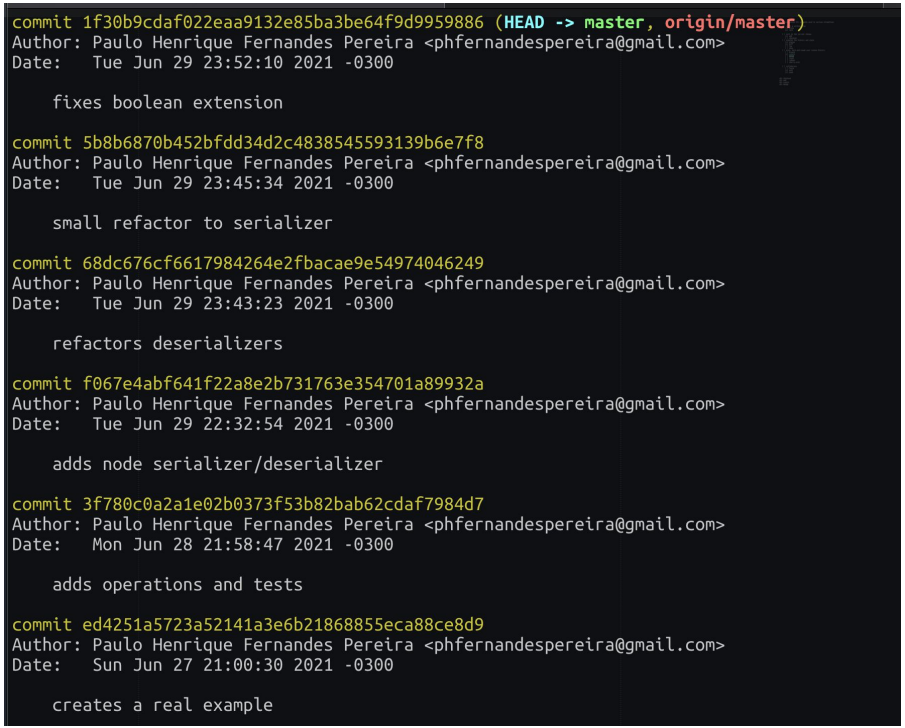
Branching, Merging and checking out



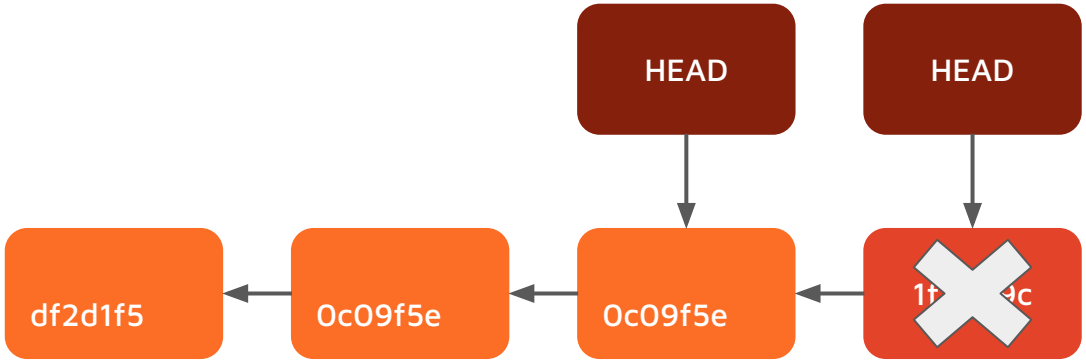

```
$ git checkout -b 1-user-access-rule

# 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

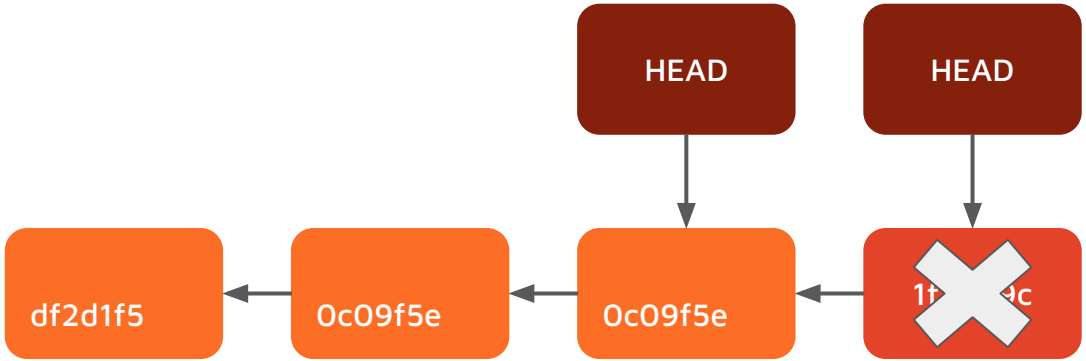


$ git reset



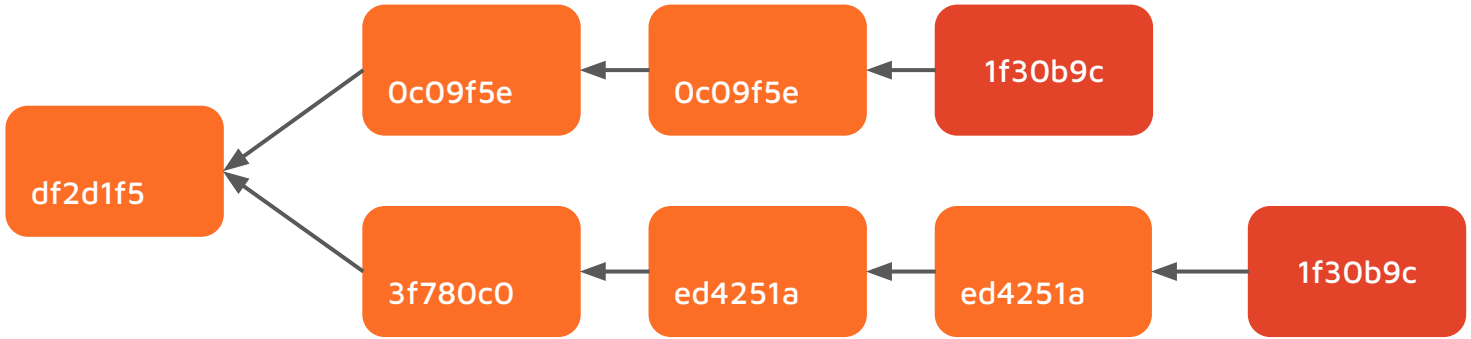
- git reset
- git reset --hard
- git reset HEAD~1
- git reset HEAD~1 --hard

$ 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

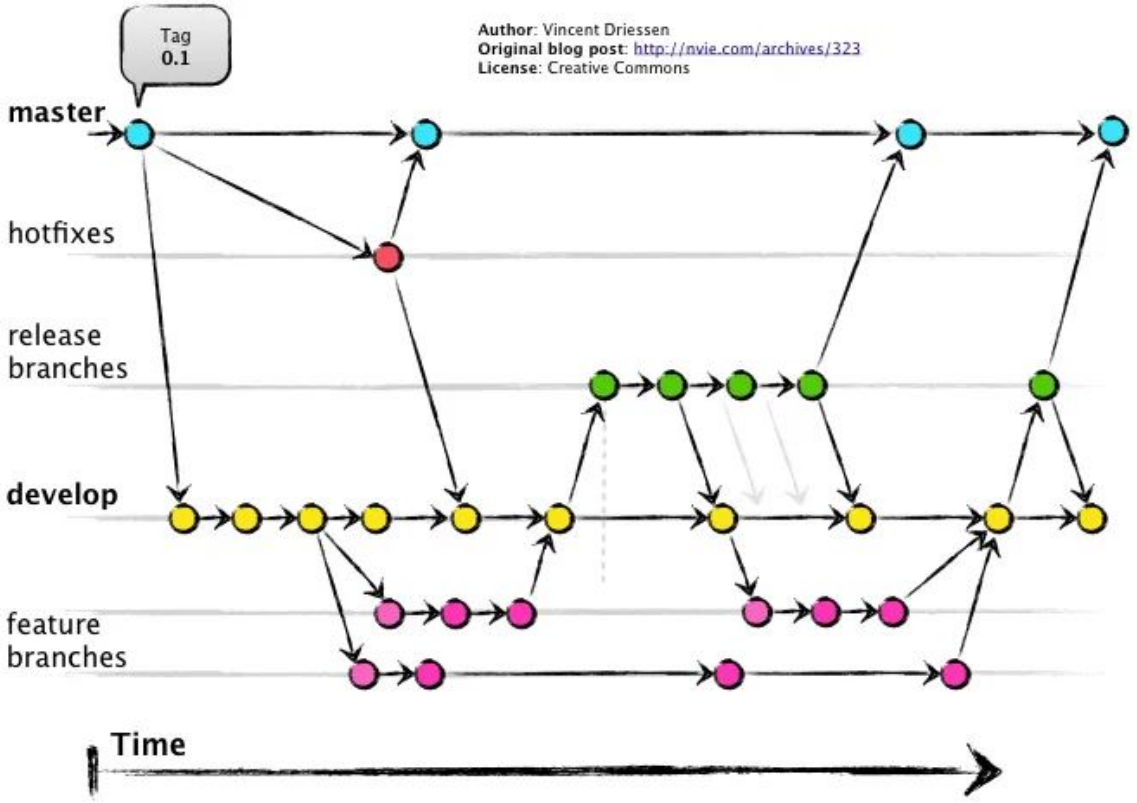
Branching

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

Colaboração

git fetch
git pull
git push

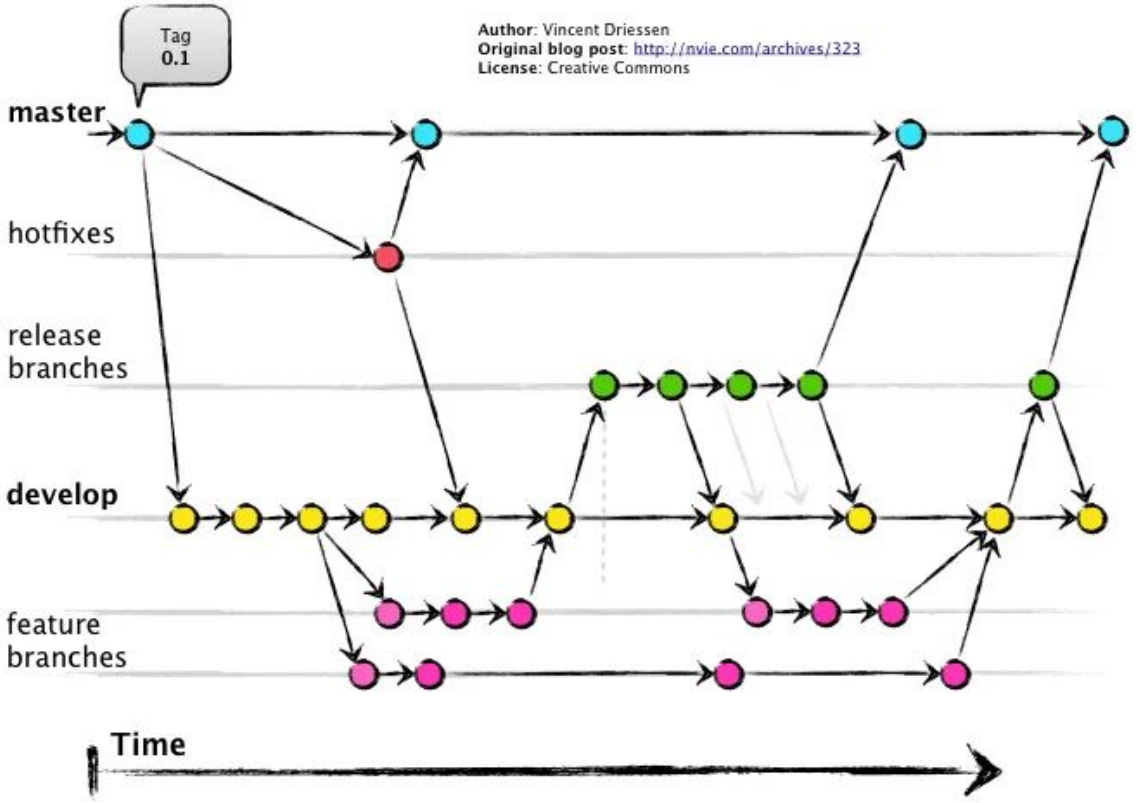
gitflow


Tag
0.1
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons
master
hotfixes
release
branches
develop
feature
branches
Time

gitflow

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

gitflow


Tag
0.1
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons
master
hotfixes
release
branches
develop
feature
branches
Time

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

RARO labs