

GIT



SUMÁRIO

01

O QUE É

02

INSTALAÇÃO

03

CONFIGURAÇÕES INICIAIS

04

COMANDOS
BÁSICOS

05

BRANCHES

06

CONFLITOS



01

O QUE É

O que seria o Git? Para que serve?





Git

- É um controle de versão de arquivos
- Otimização
- Remoto



02

INSTALAÇÃO

Vamos instalar o git na sua máquina?



INSTALAÇÃO



TERMINAL

apt-get install git



WINDOWS

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



03

CONFIGURAÇÕES INICIAIS

Vamos instalar o git na sua máquina?

Configure seu git

Nome

```
git config --global  
user.name <seu_nome>
```

Email

```
git config --global  
user.email <seu_email>
```

Editor

```
git config --global  
core.editor <seu_editor>
```





04

COMANDOS BÁSICOS

Vamos instalar o git na sua máquina?



- `git clone "url"` → Clona um repositório
- `git init` → Inicia um repositório git
- `git remote add url` → Adiciona a url do repositório
- `git fetch` → Busca todas as branches da url para o repositório local

- `git status` → Mostra as alterações feitas no repositório atual
- `git add "arquivo"` → Mostra as alterações feitas no repositório atual
- `git log` → Mostra a lista de commits

- git commit

Commit é uma forma de agrupar as várias alterações

→ git commit -m “o que foi feito”

→ git commit -s

→ git commit -m “o que foi feito

>

>

co-authored-by: Fulano <fulano@email.com>

Mostra a lista de commits



05

BRANCHES





CRIAR

- `git checkout -b nome_da_branch`
- `git branch nome_da_branch`

LISTAR

- `git branch`

MUDAR

- `git checkout nome_da_branch`
- 



JUNTANDO BRANCHES

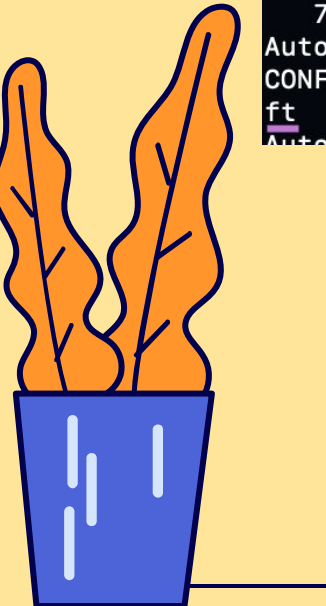
- `git merge nome_da_branch`
- `git pull origin nome_da_branch *`

ATUALIZAR COM REMOTO

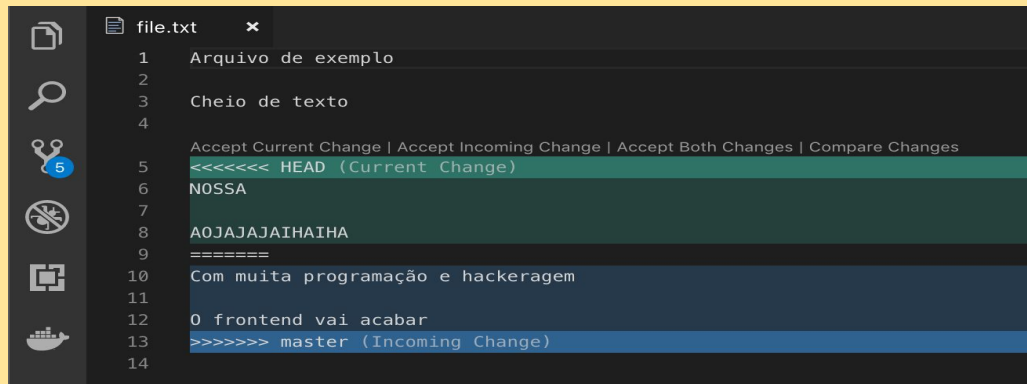

**SEMPRE dar o pull para atualizar sua
branch local com a branch remota**

- `git push origin nome_da_branch *`
- 

RESOLVENDO CONFLITOS



```
→ ProjetoFloresEmConflito git:(master) git pull
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 12 (delta 9), reused 12 (delta 9), pack-reused 0
Unpacking objects: 100% (12/12), done.
From https://github.com/juvigato/ProjetoFloresEmConflito
  7ece12b..48b5db5  master    -> origin/master
Auto-merging ProjetoFloresEmConflito/ViewController.swift
CONFLICT (content): Merge conflict in ProjetoFloresEmConflito/ViewControll
er.swift
Automatic merge failed; fix conflicts and then commit the result
```



```
file.txt
1  Arquivo de exemplo
2
3  Cheio de texto
4
5  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
6  <<<<<<< HEAD (Current Change)
7  NOSSA
8  AOJAJAJAIHAIHA
9  =====
10 Com muita programação e hackeragem
11
12 O frontend vai acabar
13 >>>>>> master (Incoming Change)
14
```

!! ATENÇÃO !!

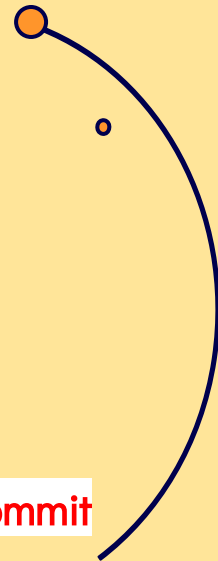


● git add .

● -f



● **git reset --HARD hash_do_commit**



RESUMO

Checar alterações

git status

Juntar e salvar
alterações

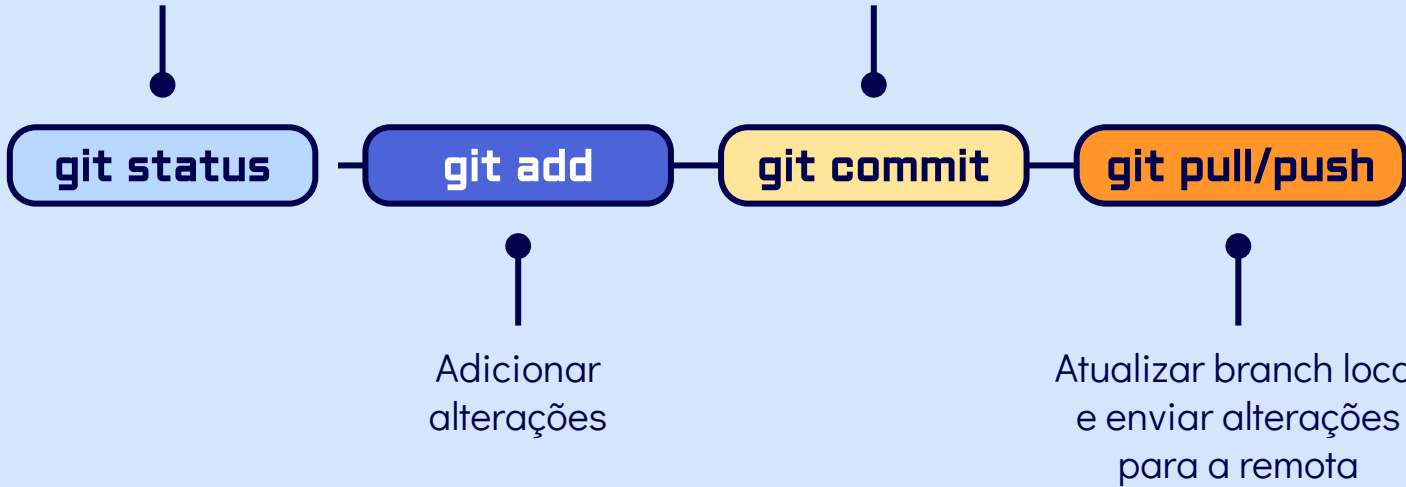
git add

git commit

git pull/push

Adicionar
alterações

Atualizar branch local
e enviar alterações
para a remota



INFORMAÇÕES ADICIONAIS E BOAS PRÁTICAS

**Padronização
de commits**

Voltar commits

- `git revert hash_do_commit`
- `git checkout hash_do_commit`
- `git stash`

Wiki

Markdown





GITFLOW



SUMÁRIO

01
O QUE É

02
COMO FUNCIONA

03
RAMIFICAÇÕES



01

O QUE É

O que é o GitFlow? Para que serve?

O **GitFlow** é uma idéia abstrata do fluxo do Git. Ele dita um fluxo de trabalho que é usado para gerenciar projetos maiores.





02

COMO FUNCIONA

Como funciona o GitFlow?



Repositório Git

Para utilizar o **GitFlow** é necessário ter uma ramificação de desenvolvimento ou a branch development na sua máquina local.

Para verificar basta digitar o comando ao lado no terminal;

Comando

Entrada

```
$ git branch
```

Saída

```
$ git branch  
* develop  
master
```


Repositório Git

Caso a branch não exista:

- faça a sincronização do seu repositório remoto;
- faça o checkout criando sua branch develop;
- envie para seu repositório remoto:

Comando

Entrada

```
$ git fetch origin
```

```
$ git checkout -b develop
```

```
$ git push origin develop
```



Instalação

Após ter criado a development, onde irá acontecer todo desenvolvimento, crie a branch respectiva a sua implementação, lembre-se de manter um padrão de nomenclatura para facilitar o entendimento como é sugerido no git flow:

Instalação

GitFlow é uma ferramenta, portanto requer um processo de instalação. Mas não se preocupe! É bem simples, basta digitar os comando no terminal.

O comando ao lado é uma extensão do comando padrão `$ git init` e não altera nada no repositório, além de criar ramificações.

Comando

Entrada

```
$ git flow init
```

Saída

```
Initialized empty Git repository in ~/project/.git/  
No branches exist yet. Base branches must be created now.  
Branch name for production releases: [master]  
Branch name for "next release" development: [develop]  
  
How to name your supporting branch prefixes?  
Feature branches? [feature/]  
Release branches? [release/]  
Hotfix branches? [hotfix/]  
Support branches? [support/]  
Version tag prefix? []
```



03

RAMIFICAÇÕES

Para que servem as branches?



Repositório Principal/master

O branch master é o branch "padrão" quando você cria um repositório.

Comumente utilizado para conter apenas o pedaço do projeto mais estável, testado e versionado que será entregue ao cliente.

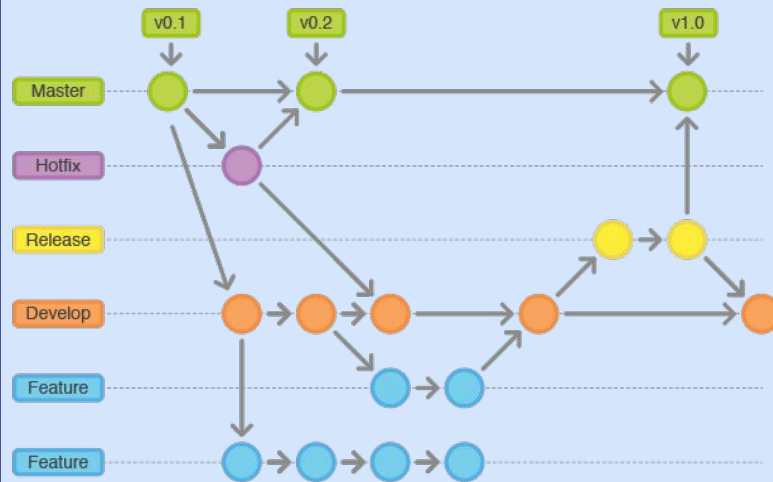


Iniciar

AUTO

Repositório de desenvolvimento/develop

A **develop** deve sempre conter o **código mais atual**, onde as branches de features serão ramificadas tendo ela como base.



Iniciar

AUTO

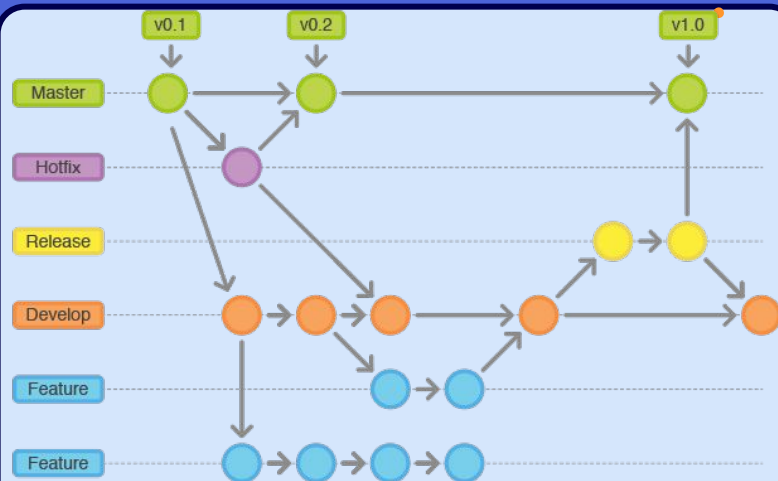
```
$ git checkout -b develop
```

Repositório recurso/feature

Cada nova feature deve possuir sua própria branch, a qual pode ser enviada por push para a develop para backup/colaboração.

As branches de feature utilizam a branch de develop como ramificação pai.

Quando concluído, ele é mesclado de volta na development.



Iniciar

```
$ git flow feature start feature_branch
```

Finalizar

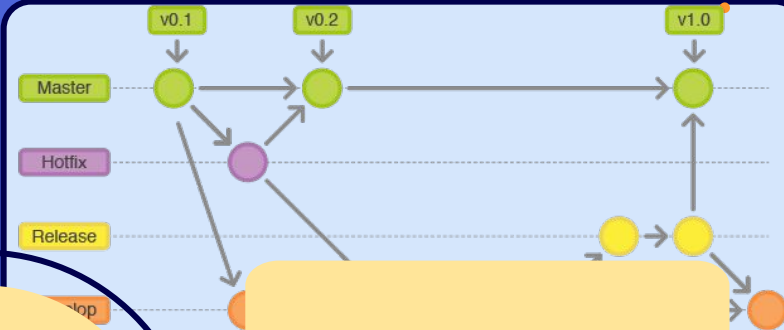
```
$ git flow feature finish  
feature_branch
```

Repositório recurso/feature

Cada nova feature deve possuir sua própria branch, a qual pode ser enviada por push para a develop por backup/colaboração.

As branches de feature utilizam a branch de develop como ramificação pai.

Quando concluído, ele é mesclado de volta na develop.



As features não
devem **NUNCA**
interagir com a
branch master

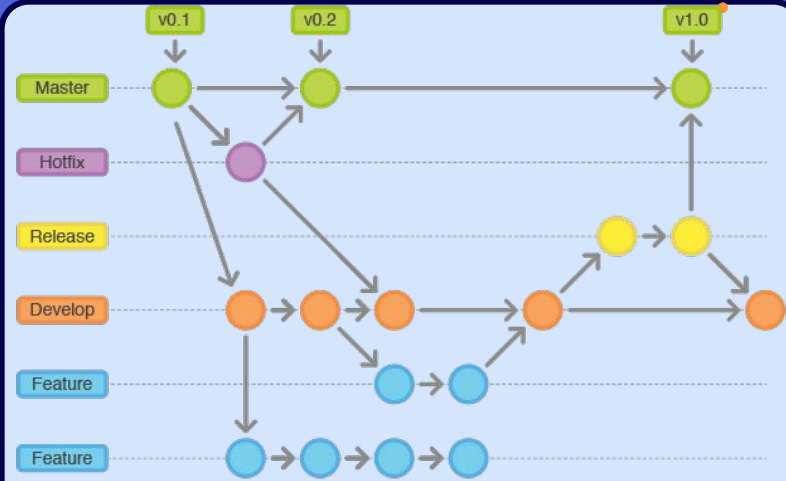
Finalizar

```
$ git flow feature finish  
feature_branch
```


Repositório lançamento/release

Uma vez que a develop adquiriu **features o suficiente** para um lançamento, uma branch de **release** é criada.

Quando estiver pronta para ser lançada, a branch de release é mergeada com a master e marcada com um **número de versão**.



Iniciar

```
$ git flow release start 0.1.0  
Switched to a new branch 'release/0.1.0'
```

Finalizar

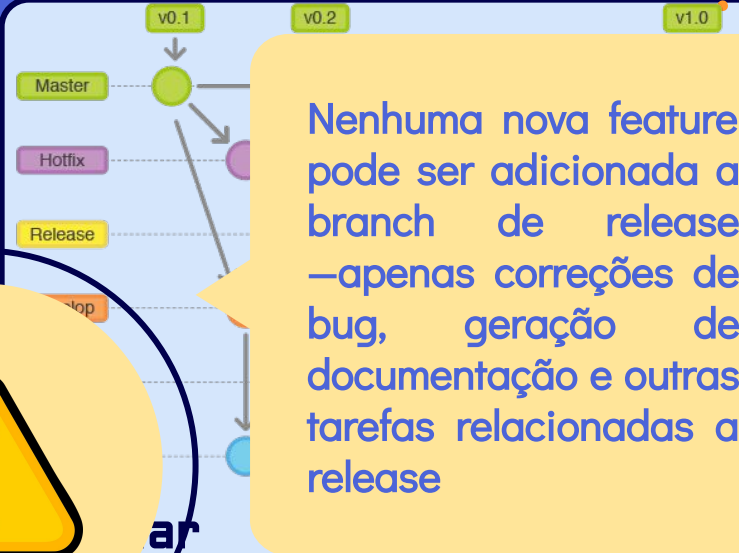
```
$ git flow release finish '0.1.0'
```

Repositório lançamento/release

Uma vez que a develop
adquiriu recursos e bastante

A branch deve ser
mesclada de volta
com a ramificação
de desenvolvimento

master e marcada com um
número de versão.



```
git flow release start 0.1.0
Switched to a new branch 'release/0.1.0'
```

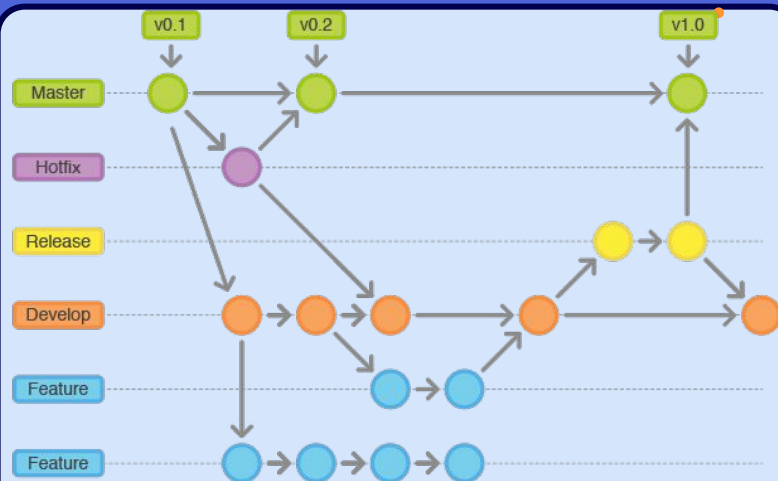
Finalizar

```
$ git flow release finish '0.1.0'
```

Repositório correção/hotfix

Hotfix são usadas para corrigir com rapidez releases de produção e são baseadas na **main**.

Assim que a correção é concluída, ela deve ser **mesclada** tanto na **main** quanto na **develop** (ou na ramificação de lançamento atual) e a main deve ser marcada com um **número de versão atualizado**.



Iniciar

```
$ git flow hotfix start hotfix_branch
```

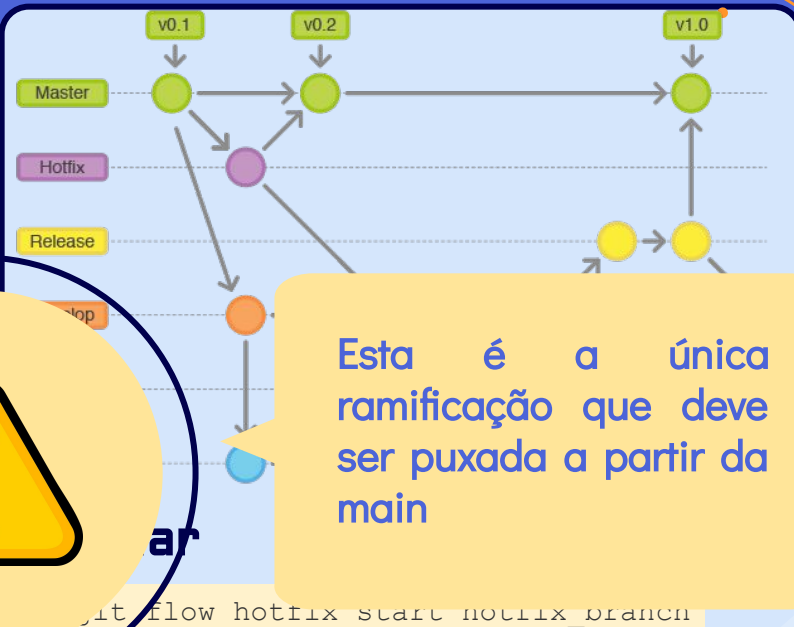
Finalizar

```
$ git flow hotfix finish hotfix_branch
```

Repositório correção/hotfix

Hotfix são usadas para corrigir com rapidez releases de produção e são baseadas na **main**.

Assim que a correção concluída, ela deve **mesclada** tanto na **main** quanto na **develop** (ou na ramificação de lançamento atual) e a **main** deve ser marcada com um **número de versão atualizado**.



Finalizar

```
$ git flow hotfix finish hotfix_branch
```

MATERIAIS DE CONSULTA



- <https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=Gitflow%20Workflow%20C3%A9%20um%20design,base%20no%20lan%20C3%A7amento%20do%20projeto.>
- <https://medium.com/trainingcenter/utilizando-o-fluxo-git-flow-e63d5e0d5e04>
- <https://readthedocs.org/projects/git-flow/downloads/pdf/latest/>
- https://rogerdudler.github.io/git-guide/index.pt_BR.html

Obrigada!

Perguntas?

Dores?

Reclamações da vida?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**

