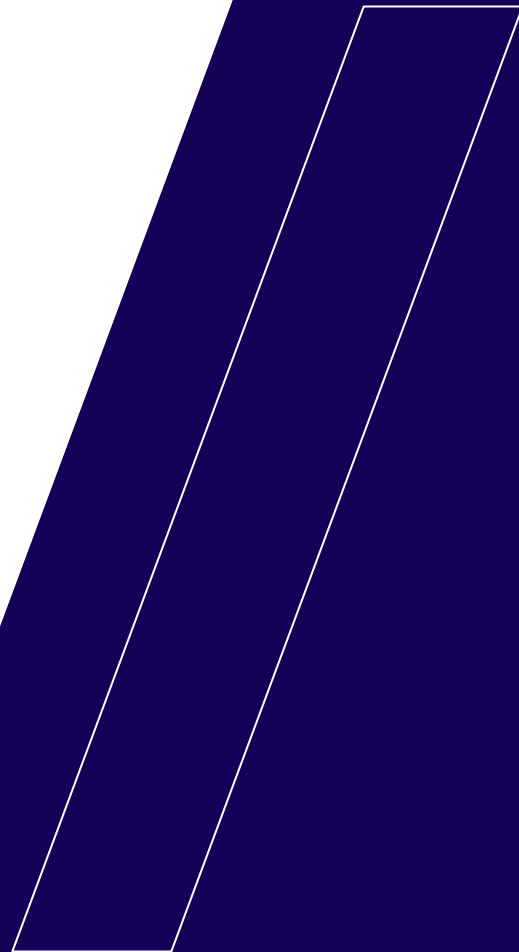




Formação Desenvolvedor Full Stack Júnior



Versionamento de Código com Git e Github



O que é versionamento de código?

Histórico de mudanças → O versionamento cria um histórico de todas as alterações realizadas no código, o que facilita a compreensão de quando e por que uma alteração foi feita.

Colaboração → Vários desenvolvedores podem trabalhar no mesmo projeto simultaneamente. O versionamento ajuda a coordenar o trabalho de todos, evitando conflitos entre alterações feitas por diferentes desenvolvedores. E como ele faz isso?

Ramos (branches) e fusões (merges) → O versionamento permite que os desenvolvedores criem ramificações (branches) para desenvolver funcionalidades ou corrigir erros isoladamente do código principal. Posteriormente, essas ramificações podem ser fundidas (merged) de volta ao ramo principal.

Rastreabilidade → Facilita o rastreamento de bugs e a identificação de alterações que causaram problemas, permitindo que os desenvolvedores voltem a uma versão anterior do código que funcionava corretamente.

Backup → O controle de versão funciona também como um sistema de backup, pois todas as versões do código estão armazenadas e podem ser recuperadas a qualquer momento.

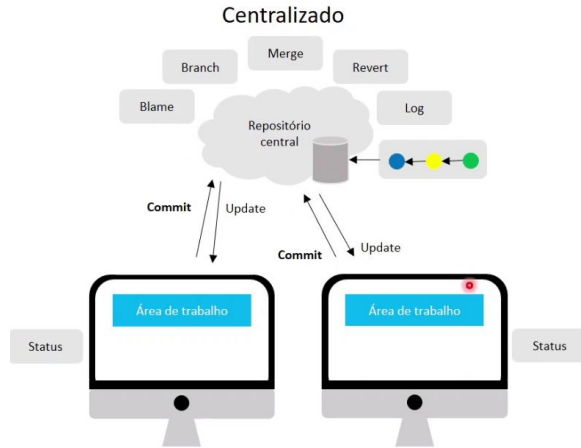
Sistemas de Controle de Versão

- Os VCS (Sistemas de Controle de Versão) são softwares com o objetivo de realizar o gerenciamento de versão de um documento qualquer.
- O sistema de controle de versão de código surgiu em 1972 e foi chamado de **Source Code Control System** (SCCS).

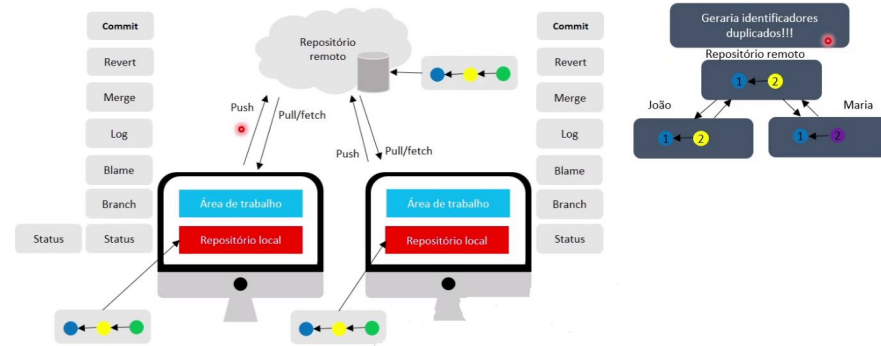


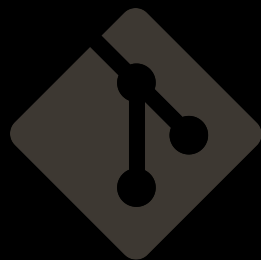
Tipos de Sistemas de Controle de Versão

VCS Centralizado (CVCS)



VCS Distribuído (DVCS)





Git

O que é Git?



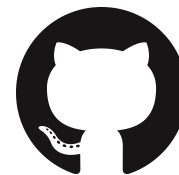
Sistema de controle de versão distribuído

- *Global Information Tracker.*
- Desenvolvido por Linus Torvalds em **10 dias**.
- Gratuito e *open source*.



GitHub

O que é Github?



Plataforma de hospedagem de código para controle de versão com Git, e colaboração.

- Comunidade ativa.
- Mascote - Octocat.
- Desenvolvido por **Chris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon.**
- Vítima de um dos maiores ataques de **DDoS** (ataque distribuído de negação de serviço).
- Comprado pela Microsoft Corporation por **US \$ 7,5 bilhões.**

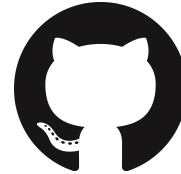
Git != Github



Software de Controle de Versão



Repositório remoto e plataforma de rede social para desenvolvedores



Instalação, Configuração e Autenticação

Criação de repositórios e Configuração de Usuário

Configurando determinada pasta como um repositório Git

```
$ git init
```

Configurando seu nome de usuário

```
$ git config --global user.email "<seu e-mail>"
```

Configurando seu email de usuário

```
$ git config --global user.name "<seu nome>"
```

Você pode armazenar suas credenciais para reduzir o número de vezes que você deve digitar seu nome de usuário ou senha:



Salvando no cache:

```
$ git config --global credential.helper cache
```



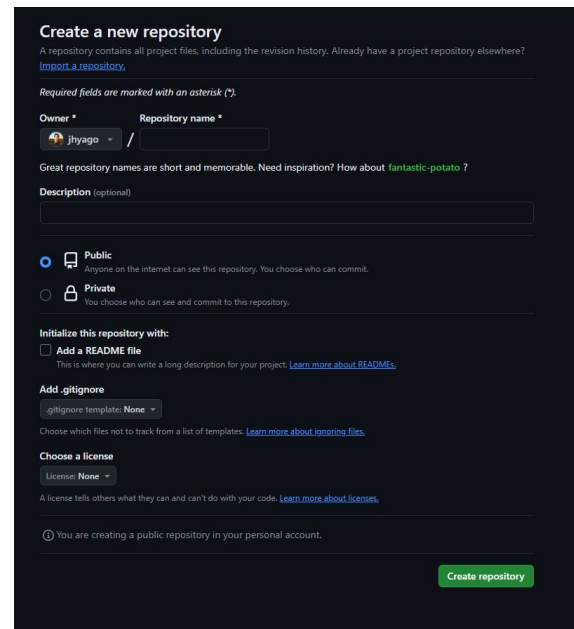
Ou permanentemente:

```
$ git config --global credential.helper store
```

Criando Repositório Remoto

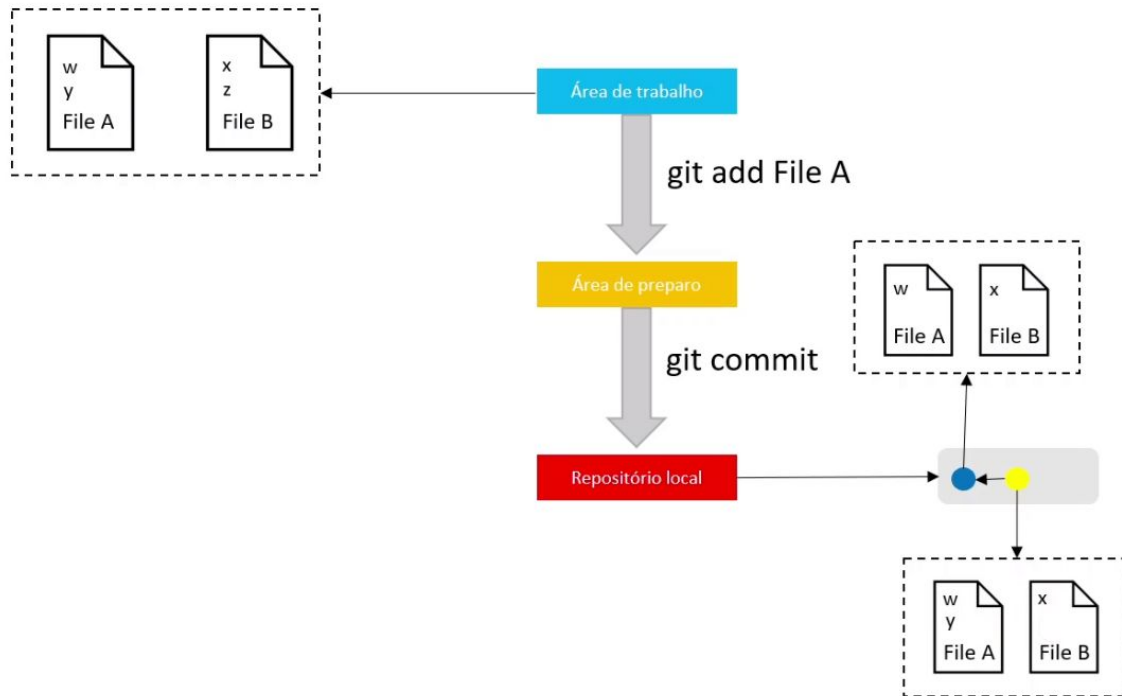
Acesse a sua conta do GitHub, clique no “+” no canto superior direito, e em “New”:

- 1 – Insira um nome (obrigatório), e a descrição (opcional);
- 2 – Coloque uma breve descrição sobre o projeto;
- 3 – Defina se o acesso será público ou privado;
- 4 – Escolha como deseja inicializar seu repositório (se quiser vazio, deixe as opções desmarcadas);
- 5 – Clique em “New Repository” e pronto!



The screenshot shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are fields for 'Owner' (set to 'jhyago') and 'Repository name'. A note states that repository names should be short and memorable, with an example 'fantastic-potato'. There is an optional 'Description' field. Under the 'Visibility' section, 'Public' is selected, with a note that anyone can see and commit to the repository. 'Private' is also an option, where only chosen users can see and commit. The 'Initialize this repository with' section has an unchecked checkbox for 'Add a README file'. The 'Add .gitignore' section shows a dropdown menu set to 'None'. Below this is a link to 'Learn more about ignoring files'. The 'Choose a license' section also has a dropdown menu set to 'None' and a link to 'Learn more about licenses'. At the bottom right, there is a green 'Create repository' button.

Commit, histórico e áreas do Git



Gerando o primeiro commit

Verificar o estado atual do repositório

```
$ git status
```

Criar um novo arquivo README.md

```
$ git add README.md
```

Nota: o arquivo é adicionado a área de preparo e será considerado para entrar no próximo commit

Realizar o commit com a mensagem "first commit"

```
$ git commit -m "first commit"
```

Verificar o histórico de commits

```
$ git log
```

Nota: o comando git log mostrará strings de 40 caracteres como o que vemos aqui. Essa string é o nome de um objeto de commit (contido em `.git\objects`)

Vinculando repositório local e repositório remoto e enviando modificações para o repositório remoto

Comando que irá criar um novo remote com o nome *origin* referenciando a URL do repositório remoto. O nome tradicional "*origin*" é apenas uma convenção para o repositório primário central.

```
git remote add <nome_do_repositório_remoto> <URL_do_repositório_remoto>
```

Comando utilizado para enviar as modificações locais ao repositório remoto

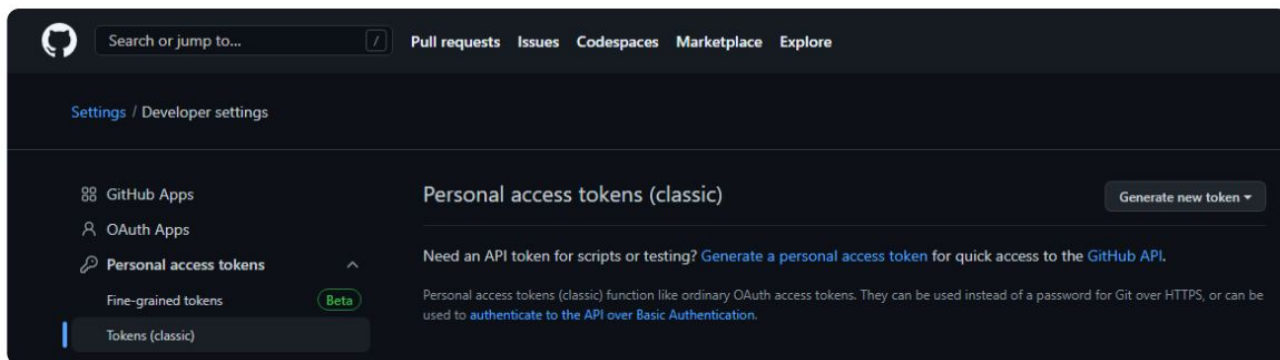
```
$ git push
```

Executar a sugestão do Git

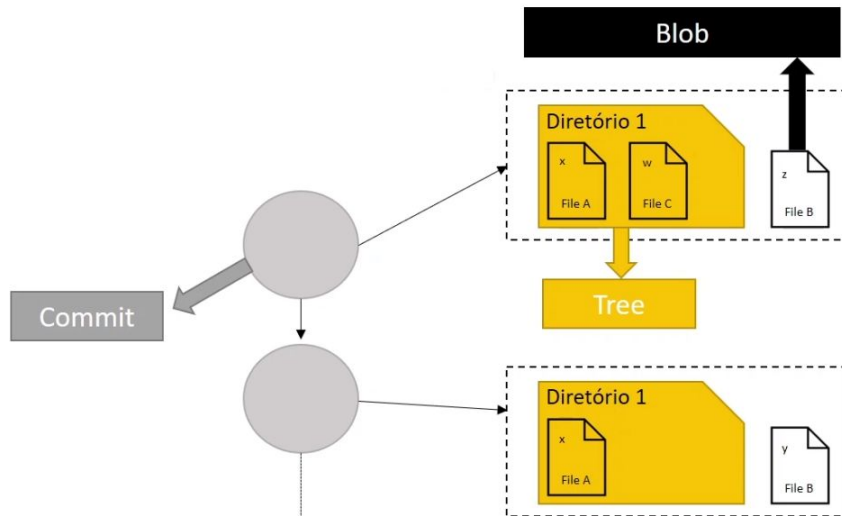
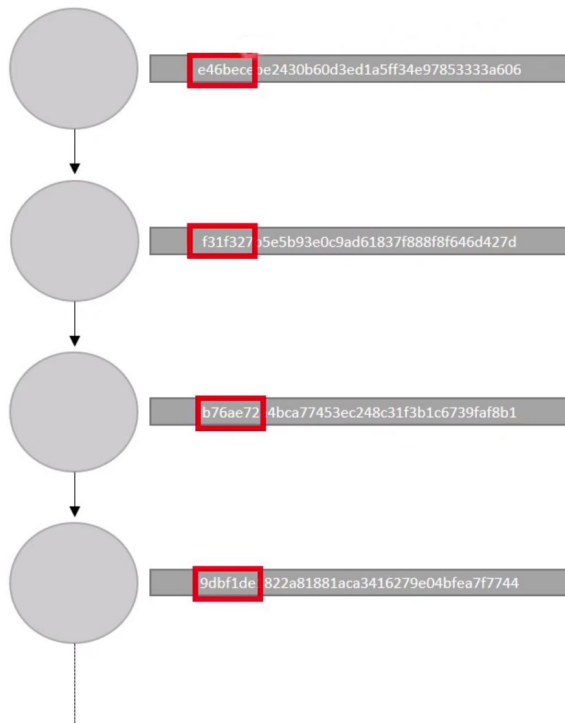
```
$ git push --set-upstream origin master
```


Autenticação via Token

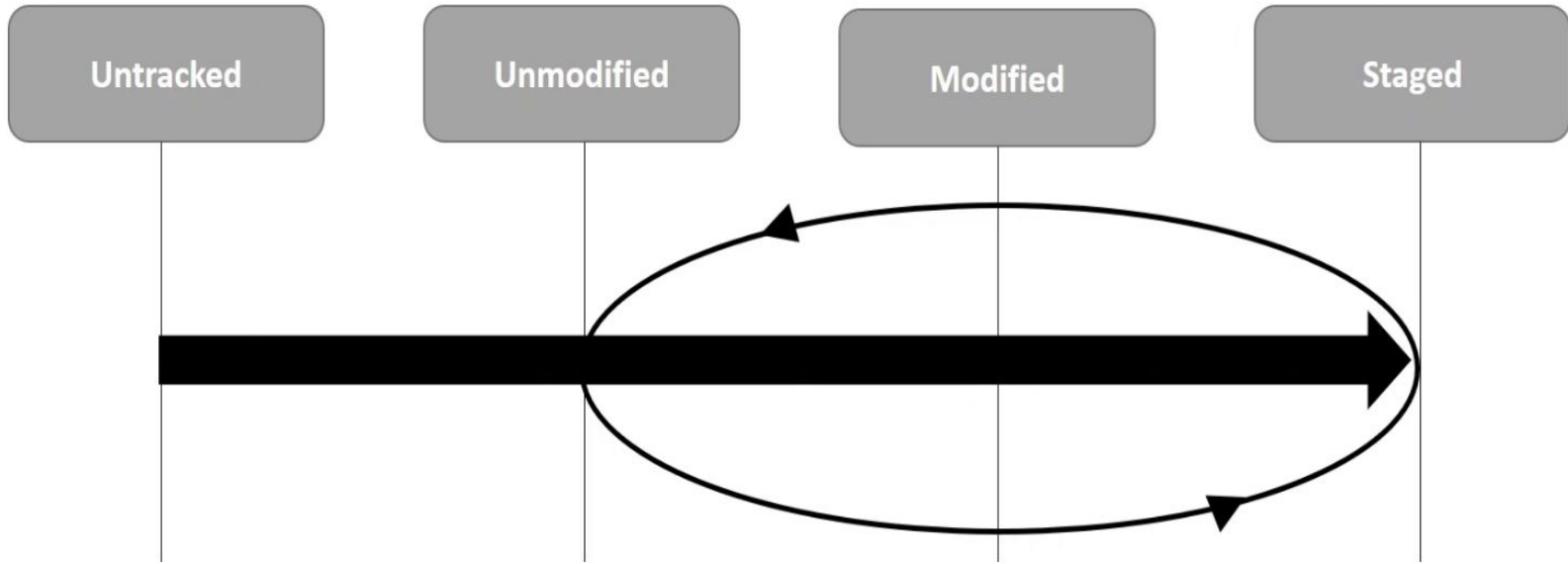
Para gerar um Token, acesse sua conta no GitHub, e no menu superior direito clique em **Settings > Developer settings > Tokens (classic) > Generate new token**.



Git Id e Git Objects

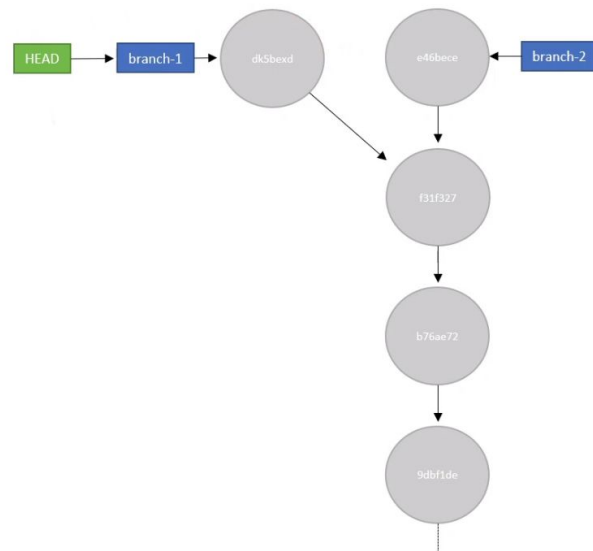
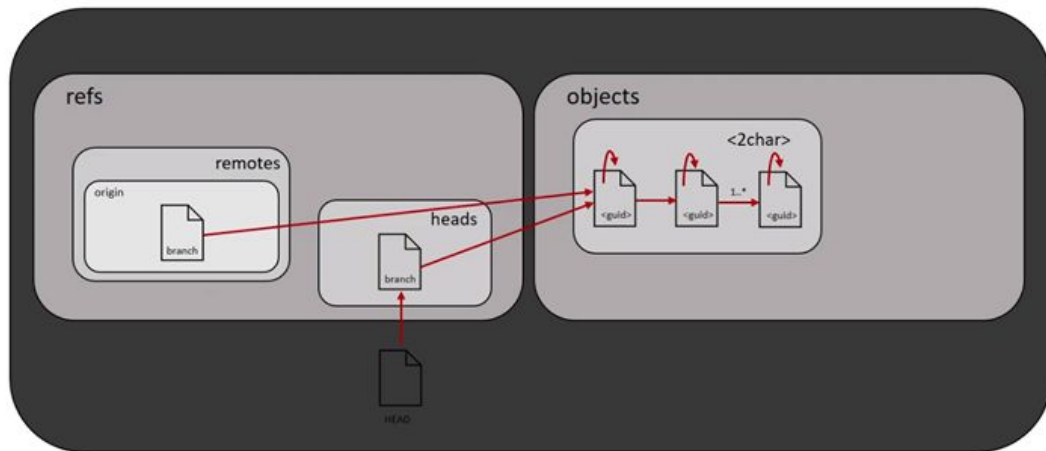


Estado dos arquivos

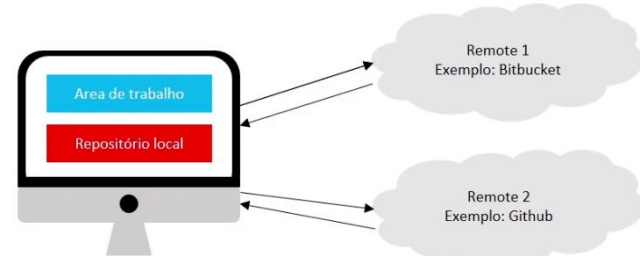
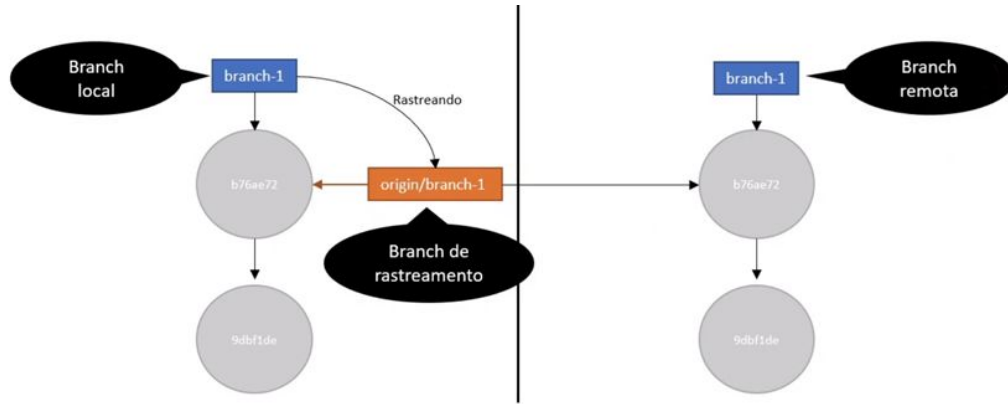


Branches

Branches e HEAD



Branche de Rastreamento



Criando e manipulando branches

Criar uma nova branch local para 'funcionalidade de cadastro'

```
$ git branch registerr
```

Alterna para a branch registerr

```
$ git switch registerr
```

Comando utilizado para enviar as modificações locais ao repositório remoto

```
$ git push
```

Comando utilizado definir o rastreamento com um branch remota, no exemplo, a master

```
$ git push --set-upstream origin master
```

Manipulando branches

Listar branches de todos os tipos

```
$ git branch -a
```

Renomear branch local

```
$ git branch -m register
```

Removendo uma branch

```
$ git branch -d register
```

Remover branch no repositório remoto

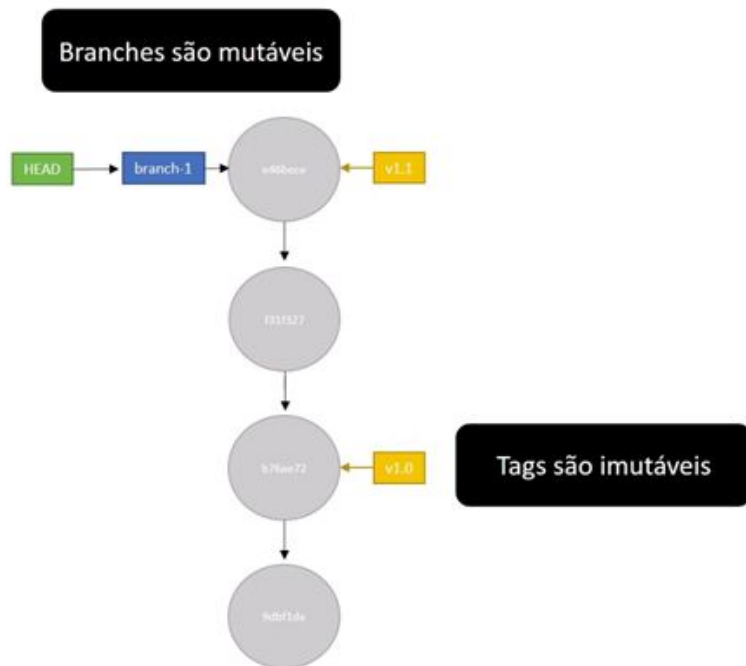
```
$ git push origin :register
```


Tags

Tags Leves e Anotadas

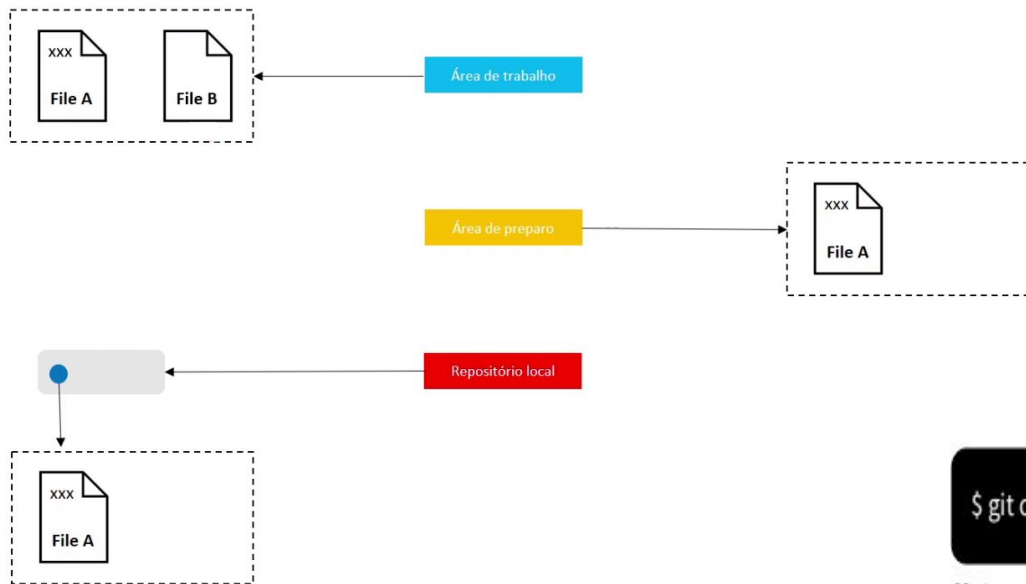
```
$ git tag -a "v1.0" -m "first version"
```

- -a: indica que é uma tag anotada
- -m: indica que o próximo parâmetro será a descrição da tag
- Por padrão a tag será adicionado no commit corrente. Mas você pode adicionar tags a qualquer outro commit do passado, basta adicionar a hash do commit



Descartando Alterações

Alterações na Área de Trabalho

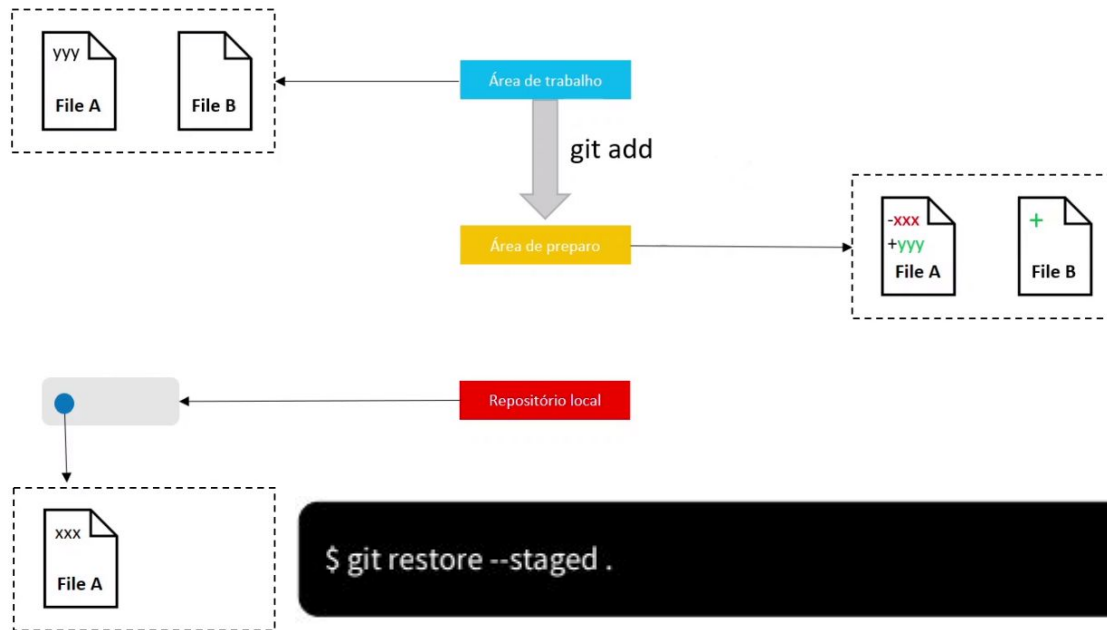


```
$ git clean logout.js -f
```

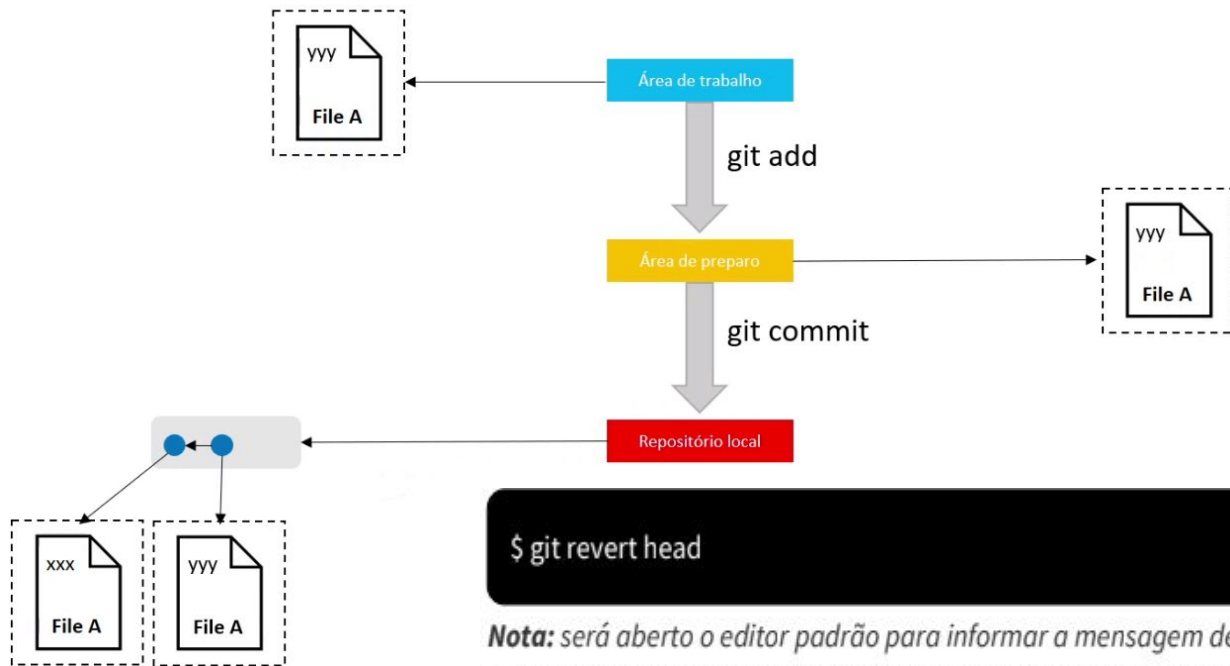
Nota: cuidado ao realizar esse comando! Lembre-se que não há histórico de alterações desse arquivo

Nota: neste comando, se é desejável remover todos os arquivos não rastreados basta usar `$git clean -f` (não é necessário trocar o nome do arquivo por '.' para remover todos)

Alterações na Área de Preparo



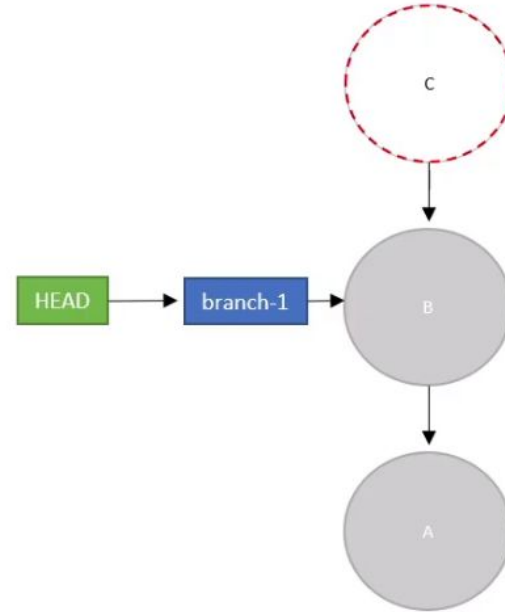
Revert de Commit



Nota: será aberto o editor padrão para informar a mensagem de reversão. É interessante que o novo commit de uma reversão utilize a mensagem padrão "revert <mensagem do commit original>"

Reset de Commit

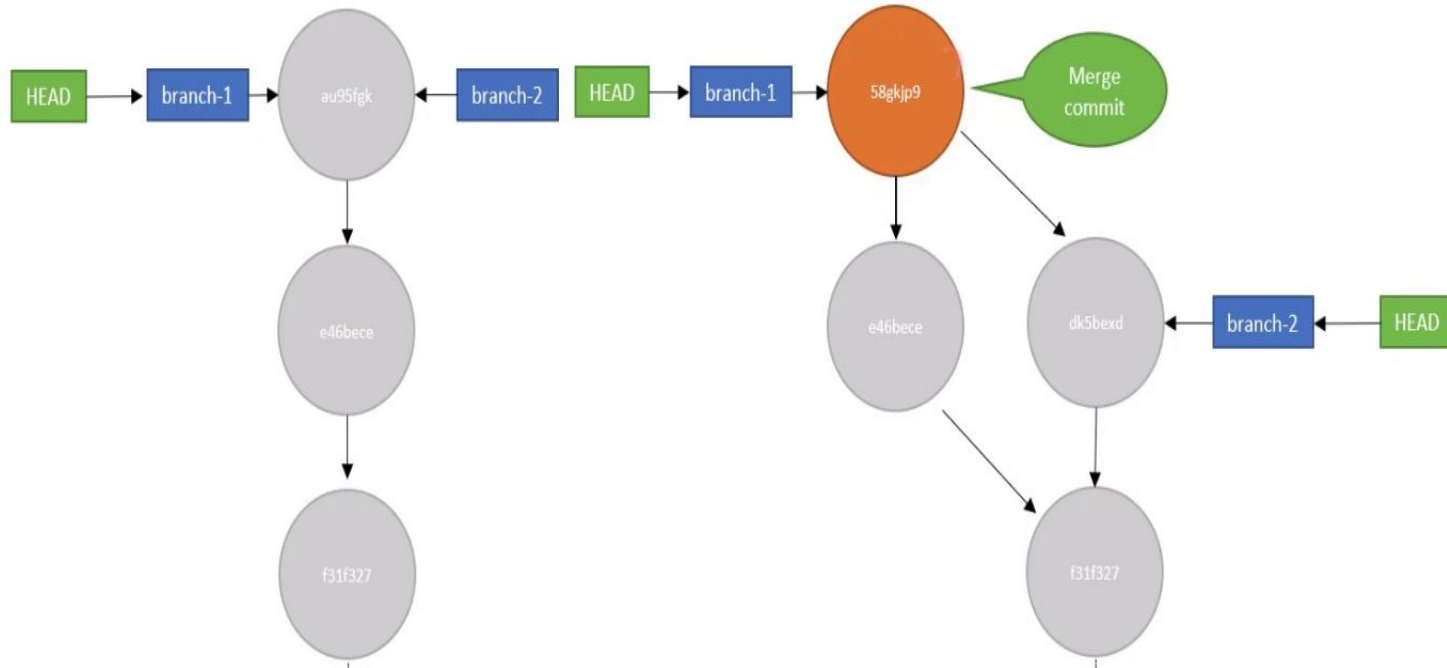
- Há três tipos: soft, mixed e hard
- Quando não definido parâmetro o Git assume como padrão o Mixed
- Os três tipos irão mudar o **rótulo da branch atual** para o commit referenciado. A diferença entre eles será nas duas áreas lógicas que poderão ser afetadas pelo comando a **área de preparo** e a **área de trabalho**



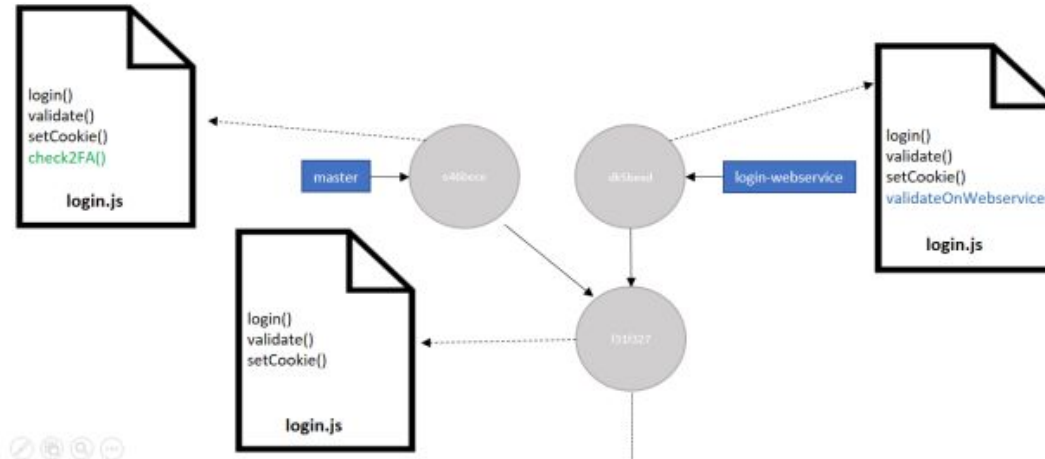
```
$ git reset --soft head~
```

Mesclando Alterações

Exemplos de merge fast-forward e merge three way



Conflitos



Realizar o merge com a branch login-webservice

```
$ git merge login-webservice
```

Nota: o merge automático não foi possível pois houve conflito

/codifica