Git e GitHub

Controle de versão

O versionamento de código permite dentre outras:

- Controle do desenvolvimento e do que deve estar no código principal.
- Salvar e controlar diferentes estágios de desenvolvimento ou versões do código.
- Permite que diferentes desenvolvedores trabalhem em diferentes partes do código.

Ferramentas para controle de versão

Existem algumas ferramentas que possibilitam o versionamento de arquivos.

Dentre elas podemos destacar:

- SVN - Sistema de versionamento mais antigo.
- GIT - Sistema de versionamento mais moderno e mais utilizado atualmente.

Serviços de versionamento

O GIT é uma ferramenta de código aberto que pode ser implantada como serviço local ou remoto de modo privado (uso restrito a funcionários de uma empresa).

Porém, também existem serviços de versionamento públicos em nuvem baseados no GIT como:

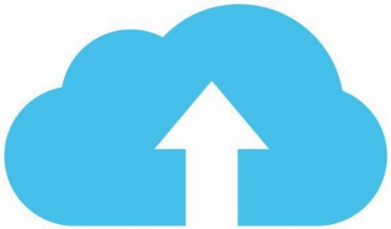
- GitHub;
- GitLab;
- Bitbuket.

São exemplos de serviços de versionamento que utilizam o GIT como ferramenta principal e também adicionam algumas funcionalidades extras.

Em nossas aulas e trabalhos vamos utilizar o GitHub para versionar e hospedar nossos códigos.

https://github.com/

Conceitos GIT
Repositório



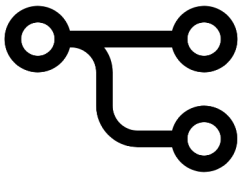
É a “pasta” principal de um projeto, nela ficam armazenadas as subpastas e arquivos de um projeto. Exemplo:
https://github.com/googleapis/googleapis

Repositórios podem ser Públicos ou Privados.
Em repositórios públicos é possível visualizar e fazer download dos arquivos.
Repositórios privados estão acessíveis somente para seus proprietários e pessoas autorizadas.

Repositórios remotos, são hospedados em servidores remotos, públicos ou privados, sendo utilizados geralmente para armazenar as versões já trabalhadas dos arquivos.

Repositórios locais, são em geral “clonados” de servidores remotos para máquinas locais, onde o desenvolvedor pode trabalhar no código e posteriormente “subir” uma nova versão.

Conceitos GIT
Branch



Uma branch é uma “ramificação” do repositório principal chamado de “main”.

Essas ramificações permitem manter cópias do projeto que podem ser alteradas sem afetar o projeto principal.

Essas branches posteriormente pode ser “mescladas” ao projeto principal, adicionando novas funcionalidades, corrigindo erros ou gerando novas versões.

Exemplo:

Projeto meu aplicativo.

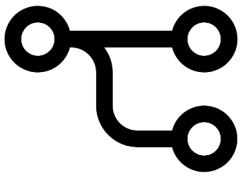
branches:

main	- Ramo principal, aplicação geralmente em uso.
recuso-beta	- Ramo utilizado para criar e testar um recurso novo.
fix-bug-y	- Ramo utilizado para criar e testar a correção um problema.

Vide:

https://github.com/googleapis/googleapis/branches

Conceitos GIT
Branch



Para listar as branches e ver em qual delas está no momento, usa-se o comando `git branch`.

Para criar ou navegar entre as branches locais usa-se o comando `git checkout`.

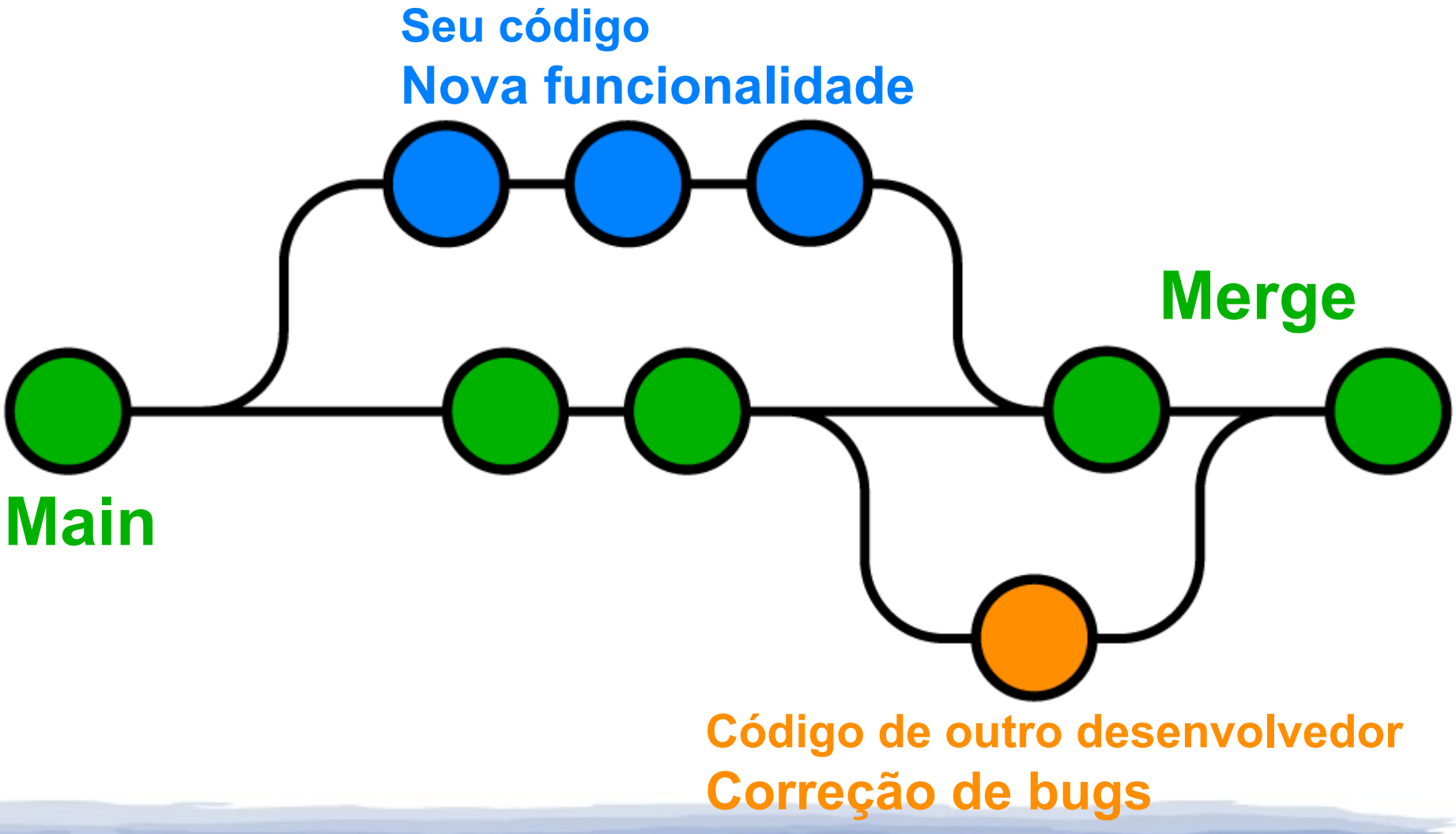
Exemplo:

```
git branches                        #Lista branches e aponta a atual
git checkout -b recurso-beta        #Cria a branch e muda para ela (a opção -b cria)
git checkout main                   #Volta para branch main (sem a opção -b, apenas navega entre as branches)
```

Conceitos GIT

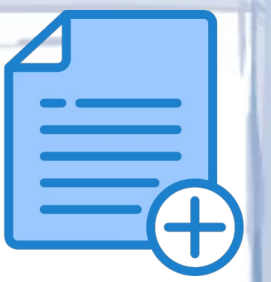
Branch

Fluxo de trabalho



Conceitos GIT

Adicionar arquivos - `git add`



Para que o GIT consiga rastrear as modificações realizadas em um arquivo no repositório local, não basta criar o arquivo na pasta do repositório.

É necessário informar ao GIT que o arquivo dever ser rastreado, isso é feito "Adicionando" o arquivo.

Para verificar os arquivos da pasta que **não** estão sendo "rastreados" pode se usar o comando `git status,` para depois adicionar com o comando `git add`.

Exemplo:

```
git status              #Mostra status do repositório
git add nome_arquivo    #Para adicionar arquivos únicos
git add .               #Para adicionar todos os arquivos de uma vez
```

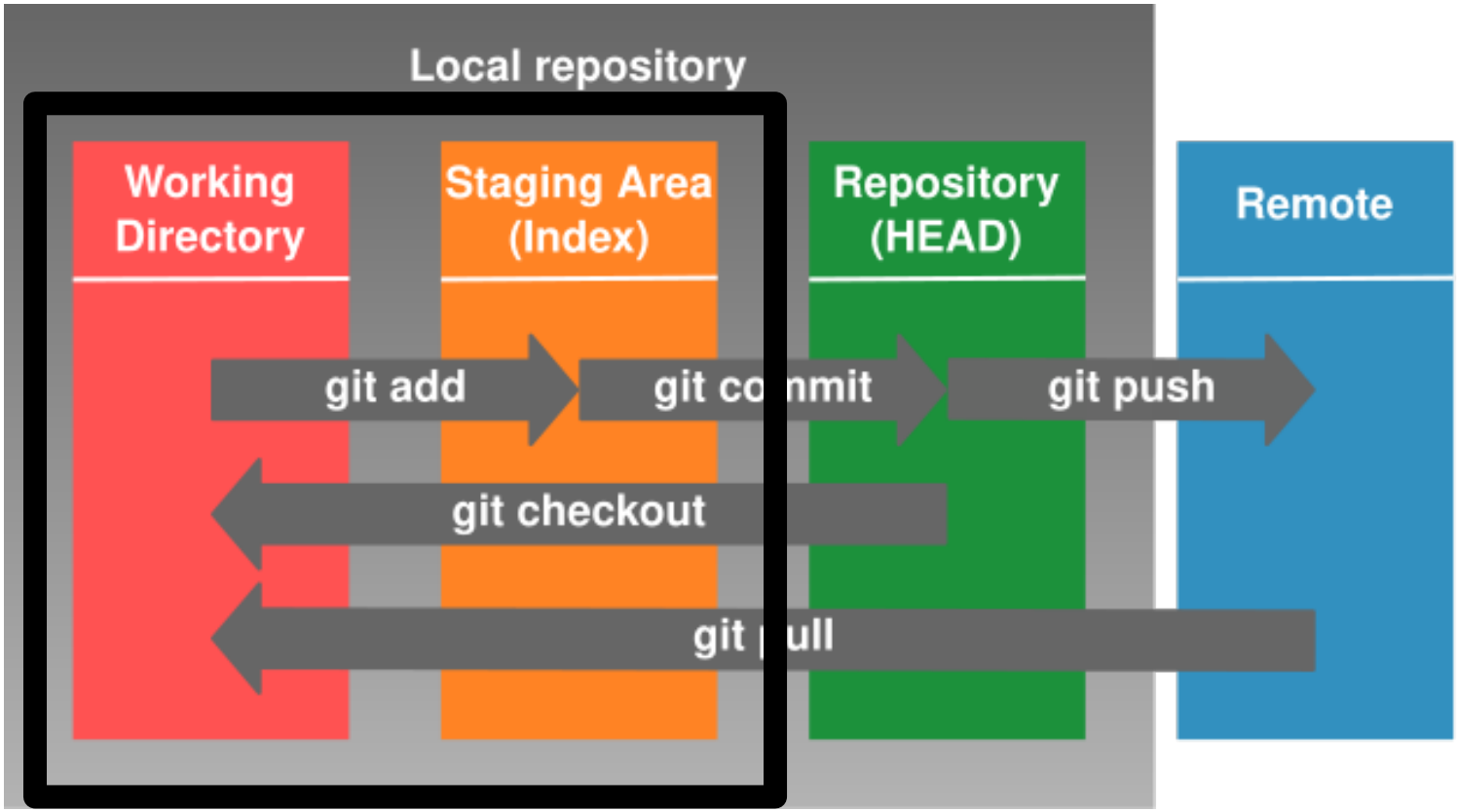
Sempre que um arquivo for modificado, ele precisar ser adicionado novamente, pois se trata de uma nova versão do arquivo.

Conceitos GIT

Adicionar arquivos - `git add`

Arquivos não adicionados estão na área chamada Working Diretory (diretório de trabalho).

Arquivos adicionados passam para a área chamada Staging Area (área de preparação).



Conceitos GIT
Commit

O commit é utilizado para gerar o versionamento dos arquivos adicionados/rastreados do repositório.

- Geralmente, quando termina de trabalhar os códigos, o desenvolvedor adiciona os aquivos e em seguida faz o commit.
- O commit é sempre acompanhado por um comentário que pode ajudar a identificar os detalhes da versão que está sendo criada.

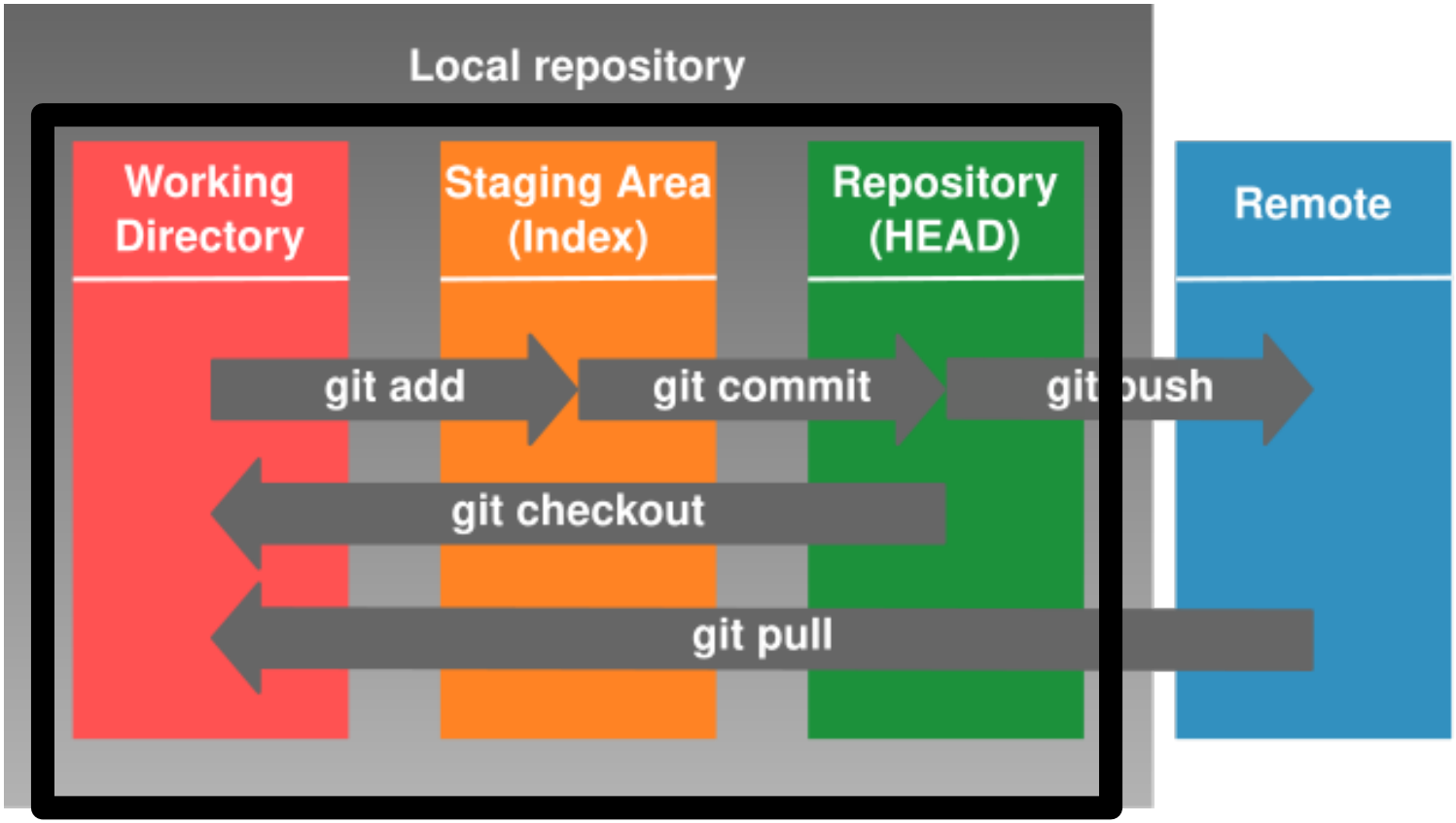
  Exemplo:

```
git status                                  #Mostra status do repositório
git add .                                   #adiciona todos os arquivos alterados
git commit -m “Versão inicial do código”    #faz o commit das modificações
```

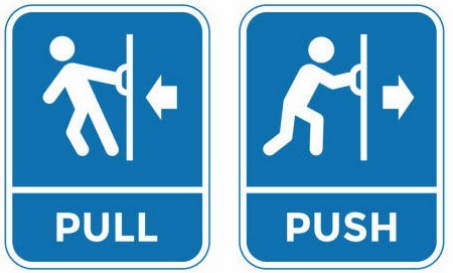
Conceitos GIT
Commit

Arquivos não adicionados **não** farão parte do commit.

Arquivos commitados passam para área chamada Local Repository (repositório local)



Conceitos GIT
Pull / Push



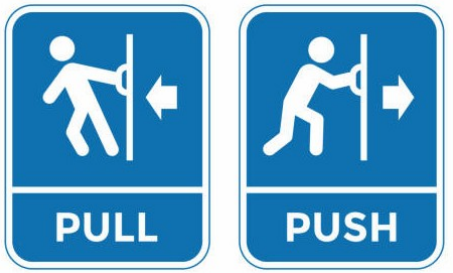
Pull é o processo de fazer o download do conteúdo atual de uma branch de um repositório remoto.

- Usado para atualizar uma branch local com uma branch remota.
- Geralmente antes de começar a trabalhar nos arquivos localmente, por exemplo.
- Caso mais de um desenvolvedor esteja trabalhando na mesma banch, pode ser usado para se manter atualizado.

Push é o processo de fazer o upload do conteúdo local para um repositório remoto.

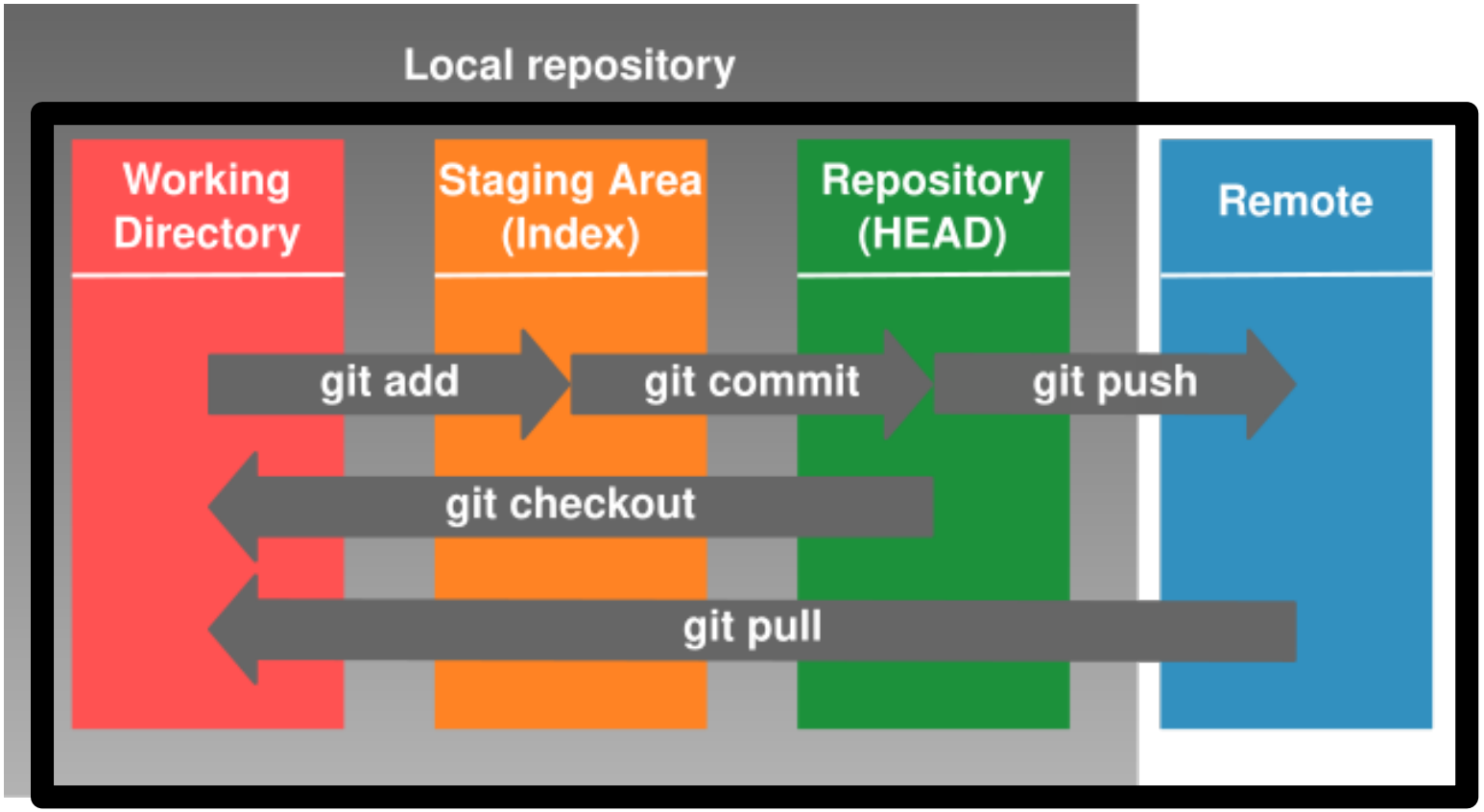
- Somente arquivos **adicionados** fazem parte de um commit; e
- Somente os arquivos **commitados** serão enviados pelo push.

Conceitos GIT
Pull / Push



Pull baixa conteúdo do **repositório remoto para o local.**

Push enviar conteúdo do **repositório local para o remoto.**



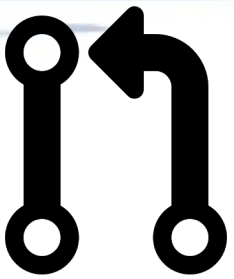
Conceitos GIT
Exemplo de fluxo de trabalho

```
git pull                                                    #Baixa o conteúdo da branch remota para a local

Criado arquivo_x_novo                                       #Criando novo arquivo

Editar arquivo_y_editado                                    #Atualizando arquivo existente

git status                                                  #Verificando arquivos alterados

git add .                                                   #Adicionando todos os arquivos alterados

git commit -m "Criação do arquivo x e edição do arquivo y"  #Commit das alterações (criando versionamento)

git push                                                    #Enviando para o servidor remoto
```

Quando se está trabalhando em equipe, **não se deve fazer push na branch main (principal)**.

O push deve ser feito somente na branch especifica onde se está trabalhando e **posteriormente fazer o merge com autorização do líder de projeto se for o caso.**

Conceitos GIT
Pull Request



Pull request é uma solicitação para mesclagem “merge” de uma branch específica para branch principal main.

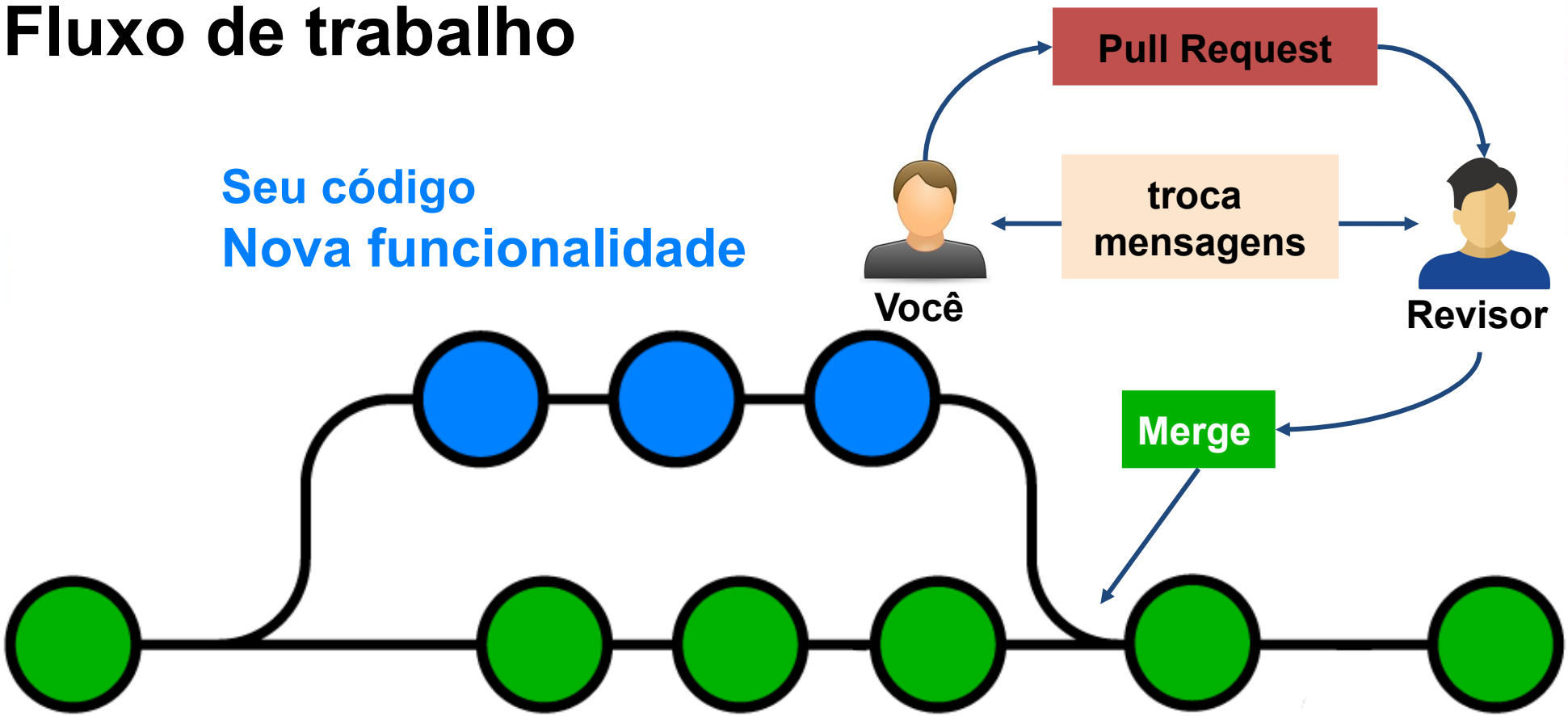
Quando o desenvolvedor finaliza a funcionalidade trabalhada na sua branch, é necessário adicionar esses códigos ao software principal, que está na branch main, para isso é criado um pull request na interface web do GitHub.

Code reviewer é a figura responsável por analisar os pedidos de pull request, ele pode aceitar e realizar o merge ou pedir que o desenvolvedor realize correções antes de fazer o merge. Tudo feito por troca de mensagens via GitHub.

Conceitos GIT

Pull Request

Fluxo de trabalho



GitHub

Finalizamos os conceitos principais

Agora vamos para a PRÁTICA.