

Github na prática

*Autores: Bruno Augusto Nassif Travençolo e Jocival Dantas Dias Junior**Data: 11/03/2021*

1.1 Glossário

- *add* – adicionar; indica quais arquivos irão para o próximo commit
- *branch* – ramificação; permite criar cópias paralelas do projeto
- *change* – modificação, alteração;
- *clone* – clonar; fazer uma cópia fiel de um repositório
- *commit* – efetivar, registrar; registra no repositório as modificações efetuadas
- *diff* – diff; permite visualizar a diferença entre arquivos
- *fetch* – buscar; busca as alterações de um repositório - é chamado pelo comando *pull*
- *git - the stupid content tracker*; sistema de controle de revisões (rastrear e controlar alterações)
- *ignore* – ignorar; não incluir na verificação do git
- *merge* – mesclar; incorpora as alterações de dois ou mais históricos - é chamado pelo comando *pull*
- *origin* – origem; apelido do repositório remoto que o projeto foi clonado
- *private* – privado; os dados do repositório são visíveis para quem escolhemos
- *public* – público; os dados do repositório são visíveis ao público
- *pull* – baixar (puxar); incorpora as alterações de um repositório remoto
- *push* – enviar (empurrar); atualiza o repositório remoto
- *remote* – remoto;
- *repository* – repositório; local onde o git guarda todo o histórico de alterações dos arquivos do seu projeto (pasta *.git* local)
- *stage* – indicar um arquivo para ir para a *staging area*
- *staging area* – área intermediária entre o diretório de trabalho local e o histórico do projeto no repositório (como se fosse um buffer)
- *status* – estado; indica qual é o estado (como está) o repositório
- *untracked* – não rastreável; indica um arquivo que está na pasta do projeto mas não foi adicionado ao git (não foi feito *add*)

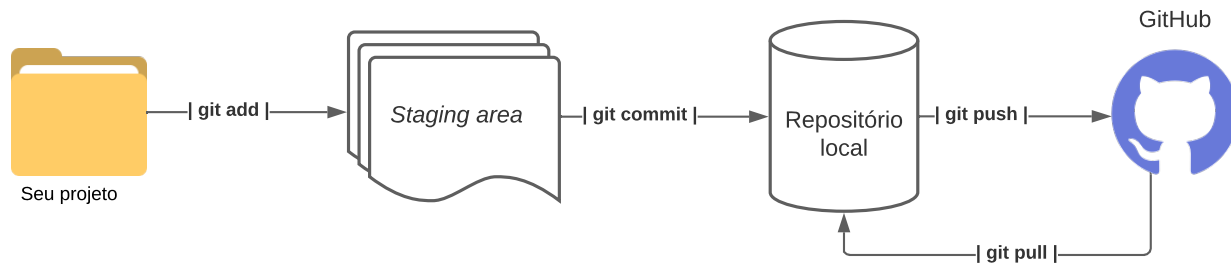


Figura 1.1: Esquema simplificado de trabalho com git e github

1.2 Comandos

1.2.1 git clone

O comando **git clone** é responsável por criar uma cópia local de um determinado repositório.

Exemplo de Execução:

```
git clone https://github.com/travencolo/gsi006-labs.git
```

1.2.2 git commit

O comando **git commit** é responsável por salvar um histórico contendo todas as modificações de um repositório em um determinado momento. Deve-se realizar o *commit* antes de enviar as alterações do projeto para outros repositórios.

Exemplo de Execução:

```
git commit -m "Aqui vai uma mensagem - coloque uma mensagem que descreva as modificações"
```

1.2.3 git push

O comando **git push** envia todas as *commits* feitos localmente para o repositório remoto.

Exemplo de Execução:

```
git push
```

1.2.4 git pull

O comando **git pull** baixa todas as modificações do repositório remoto para o repositório local. Ele também realiza o *merge* do repositório remoto com o repositório local.

Exemplo de Execução:

```
git pull
```

1.2.5 git status

O comando **git status** mostra o estado das atuais modificações efetuadas em um repositório em relação ao repositório remoto. Antes de executar um **push** ou **pull** é sempre recomendado executar comando *status* para verificar a situação do repositório local.

Exemplo de Execução:

```
git status
```

1.2.6 git add

O comando **git add** adiciona arquivos novos ou modificados para a *staging area*, local intermediário em que os arquivos ficam até que seja feito o *commit*

Exemplo de Execução:

```
git add fileName
git add readme.md
git add . // Este comando adicionará todos os arquivos pendentes.
```

1.3 Usando o Github nas Práticas

A nossa utilização do github cobrirá todos esses comandos, abaixo é demonstrado como criar o seu ambiente para trabalhar nas práticas da disciplina.

1.3.1 Criando uma conta no Github

É necessário que o aluno crie uma conta no Github com seu e-mail (preferencialmente o e-mail institucional, porém caso o aluno já tenha pode utilizar a sua conta). Para criar uma conta no github basta acessar e seguir os passos necessários:

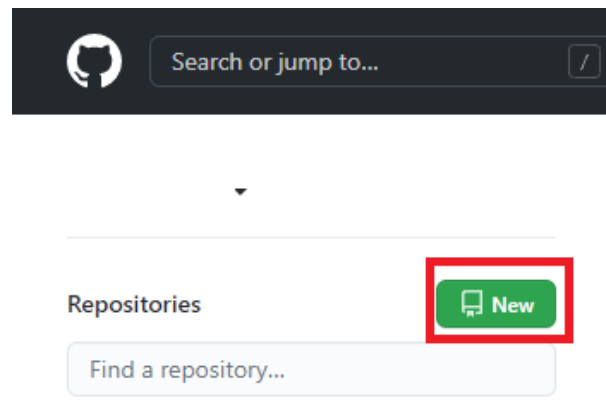
Criar uma Conta: https://github.com/join?ref_cta=Sign+up

1.3.2 Criando um repositório local

Para armazenar suas práticas será necessário criar um repositório em sua conta. O acesso a esse repositório deverá ser compartilhado com o professor para que as práticas sejam corrigidas. O professor irá criar um repositório e fará o compartilhamento com o seu usuário git. No entanto, caso queria criar um repositório basta acessar o github e clicar em "New". Observe imagem abaixo:

Posteriormente basta preencher o nome do repositório e colocar sua visibilidade como PRIVATE. Observe imagem abaixo:

Ao realizar esse passo o repositório estará criado e pronto para utilizar.



Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

/ gsilabs-006 ✓

Great repository names are short and memorable. Need inspiration? How about [didactic-octo-eureka?](#)

Description (optional)

- ☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)
- ☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

1.3.3 Baixando os dados das práticas

Com o repositório criado, precisamos alimentá-lo com os dados das práticas. Para realizar isso basta acessar o terminal do seu gosto (cmd, bash, powershell, git bash), com o git instalado, e executar os seguintes comandos:

```
cd DIRETORIO_ONDE_DESEJA_SALVAR_AS_PRATICAS
git clone <endereço do seu repositório>
cd poo1_t1_<seu-id-ufu-do-email>
git remote add base https://github.com/btravencolo/gsi015-labs-2022-01.git
git pull base main --allow-unrelated-histories
```

Ao fazer isso, acesse novamente o seu repositório e você verá que todos os arquivos das práticas estão lá. O flag “--allow-unrelated-histories” foi necessário pois os dois repositórios não possuem um ancestral comum (são independentes).

1.3.4 Enviando uma resposta para o seu repositório

Depois de responder uma ou todas as questões você pode enviar suas alterações ao seu repositório com os seguintes comandos no terminal:

```
cd DIRETORIO_ONDE_ESTA_O_PROJETO
git status // Apenas para verificar as modificações
git add .
git commit -m "AQUI VAI SUA MENSAGEM QUE SERÁ SALVA NO COMMIT"
git push
```

1.3.5 Baixando uma atualização de exercício do professor

Sempre que o professor atualizar o repositório dele (com mais exercícios ou casos de testes) você deverá atualizar a sua branch local executando os seguintes comandos:

```
cd DIRETORIO_ONDE_ESTA_O_PROJETO
git status
```

O git status não poderá conter nenhuma atualização pendente de commit, caso tenha realize o passo previsto na Seção 1.3.4. Posteriormente continue com o seguinte comando:

```
git pull base main
```

Em quase todos os casos abrirá um arquivo dizendo que o merge foi realizado de forma recursiva. Em raros casos você terá um problema de compatibilidade. Caso tenha, basta observar no código os problemas e resolver os conflitos.

1.4 Como o professor criou o repositório

```
mkdir PASTA_COM_USERNAME_DO_EMAIL_INSTITUCIONAL_DO_ALUNO
echo "# USERNAMEALUNO" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin ENDereco_DO_ALUNO
git push -u origin main
git status // Apenas para verificar as modificações
git add .
git commit -m "AQUI VAI SUA MENSAGEM QUE SERÁ SALVA NO COMMIT"
git push
```

1.5 Leituras avançada & Instalação

Caso você tenha interesse em aprender mais sobre git, github e seus comandos, recomendo os seguintes links:

[1] <https://www.atlassian.com/br/git/tutorials/what-is-version-control>

[2] <https://github.com/git-guides/>

[3] https://learngitbranching.js.org/?locale=pt_BR

[4] Install Git with GitBash <https://gitforwindows.org/>

1.6 Resumo

1.6.1 Baixar novos enunciados do professor

```
git pull base main
```

1.6.2 Enviar as respostas para o github

```
git add .
git commit -m "descrever o que vc fez"
git push
```

1.6.3 Baixar arquivos do seu github

```
git pull
```

1.7 Erro comum de autenticação

```
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations for more information.
fatal: Authentication failed for 'https://github.com:XXXXX@github.com:XXXXX/XXXXX.git'
```

tem que criar um "Personal access token" PAT

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

Passos:

Settings

In the left sidebar, click Developer settings.

In the left sidebar, click Personal access tokens.

Click Generate new token.

tente sincronizar um repositório e use o token no lugar da senha ou use a interface gráfica para digitar o token

e estará configurado