



Versionamento e colaboração

Profa: Karllenh Ribeiro



Karllenh Ribeiro


- Especialista em UX design - PUC Campinas
- LinkedIn: Karllenh Ribeiro



Autogestão/ Definição/ Histórico.

Profa: Karllenh Ribeiro

17/05/24



Atualmente, saber trabalhar em equipe é algo muito importante. No entanto, muito além das competências sociais, é importante também possuir as competências técnicas para conseguir colaborar com seus colegas de trabalho na entrega de bons projetos.



O que é versionamento?

O versionamento é o gerenciamento de versões diferentes de um documento de texto qualquer. **Perceba que não precisa ser código.**

- Controlado por **sistema de controle de versões.**

Esses sistemas são utilizados no desenvolvimento de software para controlar as diferentes versões e histórico de desenvolvimento do código.

Por que versionar?



- há a criação de várias versões de um mesmo sistema, assim como de diversos desenvolvedores atuando em uma mesma tarefa;
- Precisa ser registrado e controlado de forma a garantir que todos os membros do time entendam o status de andamento do projeto;
- Esse processo também evita a perda de dados e facilita a gestão de TI da empresa.

Ela é implementada para registrar toda a evolução do projeto, facilitando o controle do status e andamento das atividades.

Quais as vantagens de versionar?



Controle de histórico

É possível visualizar todo o histórico de desenvolvimento e voltar para versões anteriores. Além disso, temos a possibilidade de avaliar no detalhe o que foi mudado de uma versão para outra.

Marcações e resgate de versões estáveis

Através de padrões de nomenclatura para as ramificações do código no versionamento, é possível identificar de forma fácil versões específicas e quais oferecem código estável e quais ainda estão sob avaliação ou desenvolvimento.

Quais as vantagens de versionar?



Trabalho em equipe

Versionamento permite que várias pessoas trabalhem no mesmo conjunto de arquivos (repositório) ao mesmo tempo em que evita conflitos entre as alterações.

Cada membro do time de desenvolvimento tem sua “cópia” dos arquivos que ao final das alterações é colocada junto das versões alteradas dos demais.

Quais as vantagens de versionar?



Ramificações

Esta é a base para o versionamento. A possibilidade de ter várias linhas de desenvolvimento paralelas sem que uma interfira na outra. Cada dev tem sua versão do código e pode alterá-la da forma como achar necessário sem receio de interferir no trabalho do seu time.

Segurança

Os softwares de controle de versão em geral possuem recursos para evitar invasões de agentes infecciosos nos arquivos. Somente usuários aos quais foi concedida permissão para edição do código conseguirão alterá-lo.

Quais as vantagens de versionar?



Confiança

Embora a finalidade das ferramentas de versionamento não seja a de oferecer um backup do código, mas sim seu histórico e fácil navegação entre versões, ele pode acabar servindo para isto também.

Normalmente os código versionados ficam em serviços hospedados na web que nos garantem que eles não serão perdidos. Assim, temos mais confiança de que nenhuma eventualidade poderá nos fazer perder o que já foi construído.

Ferramentas de versionamento



CVS

- uma das ferramentas de controle de software mais antigas no mercado.
- A primeira versão dela foi desenvolvida em 1968.
- Essa ferramenta possui como maior desvantagem o fato de ser considerada como uma tecnologia antiga.
- É muito simples de ser operada.

Ferramentas de versionamento



TFS

- Team Foundation Server -
- Ele traz uma série de características interessantes, principalmente se você utiliza metodologias agile no setor de TI da sua empresa.
- Possibilita a gestão de projetos por meio de SCRUM
- Adequado tanto para equipes que compartilham o mesmo espaço físico quanto aquelas que trabalham à distância.
- Apresentada por esse sistema é o fato de não possuir limitações de crescimento e ter integração direta com o Microsoft Office.

Ferramentas de versionamento



SubVersion

- É uma ferramenta de controle de versão de software bastante utilizada.
- Ela é bastante rápida na execução das funcionalidades do sistema e ainda se mostra como uma das mais simples de ser empregada.
- A aprendizagem da equipe também é rápida nesse aspecto.
- Apresenta problemas na hora de executar as principais funções de um controle de versão de software eficiente.
- não é indicada para todas as equipes de TI, apenas para aquelas que são menores.

Ferramentas de versionamento



Mercurial

- Ferramenta bastante rápida na execução dos comandos e ainda funciona muito bem para equipes grandes, nas quais os desenvolvedores não estão todos trabalhando no mesmo local. Isso porque ela é uma ferramenta de controle de versão distribuída.
- Ela é um pouco mais complexa de ser utilizada em comparação com a Subversion, por exemplo.

Ferramentas de versionamento



GIT

- Ferramentas de controle de versão de software mais populares, principalmente em projetos open source.
- As principais vantagens dessa ferramenta são o design interno e interface, a eficácia e o desempenho do software. Isso significa que ele é agradável de ser utilizado, consegue atingir todos os objetivos de um bom controle de software e é rápido.
- Possui controles um pouco mais complexos quando comparado a outros softwares.
- Adequado para a utilização em grandes equipes.



Qual ferramenta iremos usar?





GIT

O Git é uma **ferramenta de versionamento não centralizado** muito poderosa que permite que desenvolvedores colaborem entre si de forma organizada na construção de um projeto que envolva código.


Termos importantes



Commit

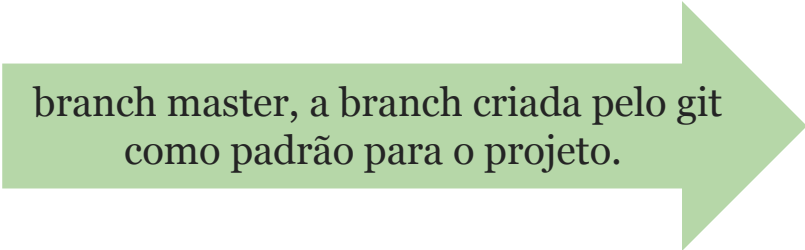
- Se trata de agrupar alterações realizadas em seu código sob um contexto. Podemos pensar nos commits como sendo cópias instantâneas da versão atual do nosso código.
- Quando realizamos um commit, o Git armazena um objeto de commit que possui um ponteiro para a versão atual do código. Além disso, este mesmo objeto de commit armazena outro ponteiro que aponta para o commit anterior a ele, possibilitando ao Git saber a ordem em que as alterações aconteceram.
- você consegue acessar qualquer versão do código simplesmente indicando para o Git o identificador do commit desejado.

Termos importantes



Branches


- Como diretórios que estão sempre apontando para o último commit de que possuem conhecimento.



branch master, a branch criada pelo git como padrão para o projeto.



nova branch chamada
minha_funcionalidade



branch master, a branch criada pelo git
como padrão para o projeto.

nova branch chamada
minha_funcionalidade

- Estaremos criando uma “cópia” da branch master que tem conhecimento dos mesmos commits que ela.alterarmos algum código na branch *minha_funcionalidade* e realizarmos um commit com estas alterações;
- Ela passa a apontar para este último commit enquanto que a branch master continua apontando para o commit anterior.
- A master só terá conhecimento das alterações feitas nesta outra branch caso solicitamos ao git para realizar a mescla do conteúdo da nossa nova branch com ela.

Termos importantes



Origin

- É o nome dado por padrão ao repositório remoto ao qual nosso repositório local está vinculado.

`git remote`

Teremos o retorno
`origin`

Termos importantes



Origin

- É o nome dado por padrão ao repositório remoto ao qual nosso repositório local está vinculado.

`git remote show origin`

Poderemos ver os detalhes do repositório remoto como seu link de acesso, as branches remotas e mais outras configurações



Configuração e instalação

Profa: Karllenh Ribeiro

23/05/24



Instalando o Git

github.com



Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

Welcome to GitHub!
Let's begin the adventure

Enter your email*

→ santoskarllenh@gmail.com

Continue

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ santoskarllenh@gmail.com

Create a password*

→ Senai@1234



Continue



Password is strong

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ santoskarllenh@gmail.com

Create a password*

✓ Senai@1234

Enter a username*

→ karllenh2024

Continue

karllenh2024 is available.

Welcome to GitHub!
Let's begin the adventure

Verify your account

Protegendo sua conta

Resolva este enigma para sabermos que você é uma pessoa de verdade

Verificar

14117d021ab6fa4e4.1391401201



You're almost done!

We sent a launch code to `santoskarllenh@gmail.com`

→ Enter code*

Didn't get your email? [Resend the code](#) or [update your email address](#).



Sign in to GitHub

Your account was created successfully. Please sign in to continue



Username or email address

Password

[Forgot password?](#)

Sign in

[Sign in with a passkey](#)

New to GitHub? [Create an account](#)



GitHub

É um repositório de um projeto.

Local onde vamos guardar os códigos do nosso projeto.



Dashboard

Q Type to search



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository

Import repository

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Home

Send feedback

Filter 8

Updates to your homepage feed

We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design. ✨

Learn more

<> Start writing code



Start a new repository for karllenh2024

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

name your new repository...



Public

Anyone on the internet can see this repository



Private

You choose who can see and commit to this

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

karllenh2024 / README.md

Create

```
1 - 🌟 Hi, I'm @karllenh2024
2 - 🤖 I'm interested in ...
3 - 🌱 I'm currently learning ...
4 - 💖 I'm looking to collaborate on ...
5 - 📫 How to reach me
```



Convince your boss to send you to GitHub Universe in SF on October 29-30.

Use our ready-to-send template today. Tickets to our global developer event are 35% off — only for a limited time.

Learn more

Latest changes

- 7 hours ago
The GitHub Enterprise Server 3.13 Release Candidate is available
- 11 hours ago
New dates for Actions larger runner multi-label deprecation
- Yesterday
Deprecation — security advisories in private repositories



New repository

Q Type ↗ to search



Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *



karllenh2024



Repository name *



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

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:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore



karllenh2024 / Senai

Q Type ↗ to search

| >_

+ ▾



<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings



Senai

Public

Pin

Unwatch 1 ▾

Fork 0 ▾

Star 0 ▾

main ▾

1 Branch 0 Tags

Q Go to file

t

Add file ▾

<> Code ▾

About



No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

karllenh2024 Initial commit

85693bd · now

1 Commits

README.md

Initial commit

now

README



Senai



Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

[Get started with GitHub Copilot](#)



Add collaborators to this repository

Search for people using their GitHub username or email address.

[Invite collaborators](#)

Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

<https://github.com/karllenh2024/Teste.git>



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Teste" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/karllenh2024/Teste.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/karllenh2024/Teste.git
git branch -M main
git push -u origin main
```








PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

 powershell + ▾   ... ^ X

```
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> git init
Reinitialized existing Git repository in C:/Users/Karllenh/Documents/3386-git-github-projeto_inicial/.git/
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> █
```



Em caso de ERRO

Precisamos Instalar o git em nosso computador



GitHub

É onde vamos hospedar nosso código


Git

É a ferramenta onde conseguimos fazer o controle da versão do nosso código, registrar quais mudanças foram feitas ao longo do tempo.



Instalando o Git

“git download”

- 
- Selecciona o sistema operacional
 - 64 bits
 - Portable (portátil): não instala no computador e sim cria um executável.
 - Standalone (sozinho) - windows 64 bits
 - Executa o arquivo .exe e instalar
 - Next ... Next... Next.. Finish

[About](#)[Documentation](#)[Downloads](#)[GUI Clients](#)[Logos](#)[Community](#)

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

[macOS](#)[Windows](#)[Linux/Unix](#)

Older releases are available and the [Git source repository](#) is on GitHub.



GUI Clients

Git comes with built-in GUI tools ([git-gui](#), [gitk](#)), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (**2.42.0**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **19 days ago**, on 2023-08-30.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.42.0**. If you want the newer version, you can build it from [the source code](#).



Information

Please read the following important information before continuing.



When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change

<https://gitforwindows.org/>

Next

Cancel



Git 2.42.0.2 Setup



Configuring experimental options

These features are developed actively. Would you like to try them?



☐ **Enable experimental support for pseudo consoles.**

This allows running native console programs like Node or Python in a Git Bash window without using winpty, but is unfortunately not quite stable yet.

☐ **Enable experimental built-in file system monitor**

(NEW!) Automatically run a [built-in file system watcher](#), to speed up common operations such as ``git status``, ``git add``, ``git commit``, etc in worktrees containing many files.

<https://gitforwindows.org/>

Back

Install

Cancel



Git 2.42.0.2 Setup



Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.



Launch Git Bash



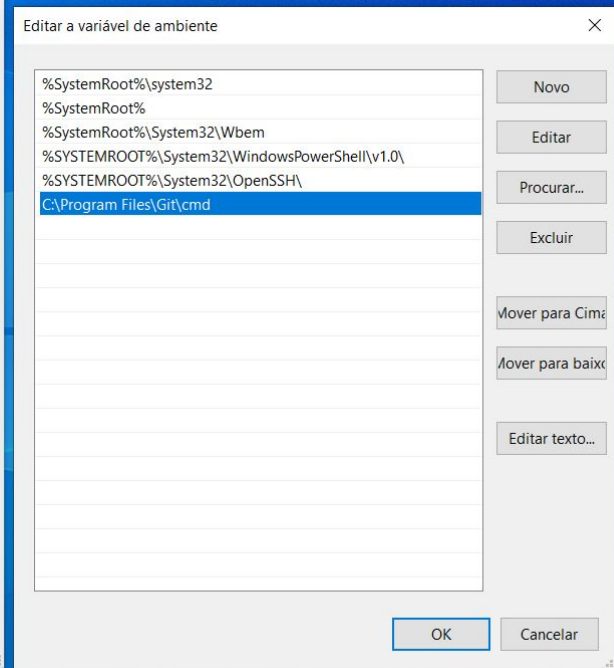
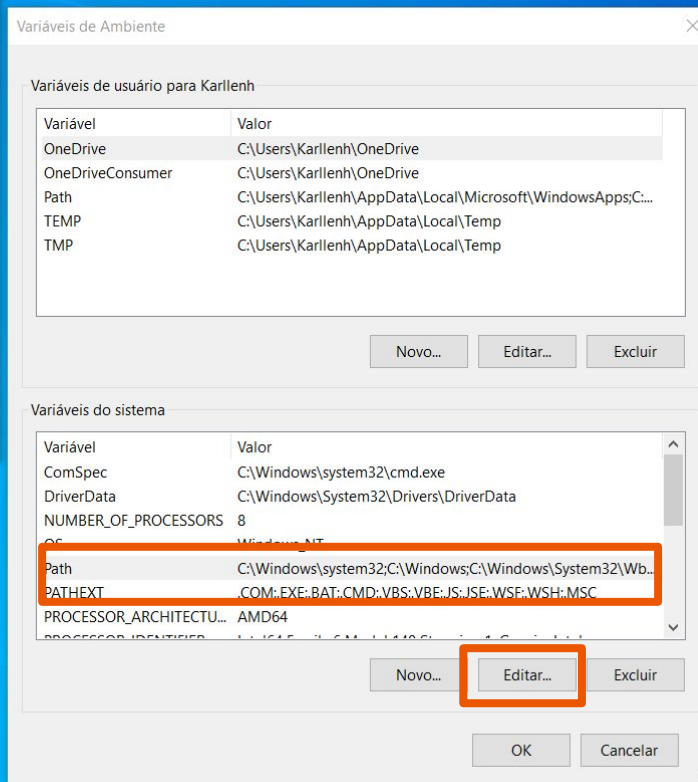
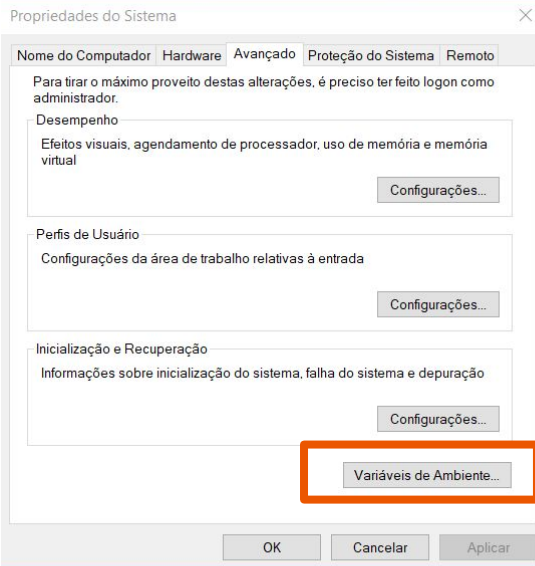
View Release Notes



Finish



Configurando variáveis no windows





Voltando ao Vs code

...



```
git init
```

```
git add README.md
```

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

```
git branch -M main
```

```
git remote add origin https://github.com/karllenh2024/Teste.git
```

```
git push -u origin main
```





PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS


 powershell + ▾   ... ^ X

```
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> git init
Reinitialized existing Git repository in C:/Users/Karllenh/Documents/3386-git-github-projeto_inicial/.git/
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> █
```



Git init

- converter um diretório do computador em um repositório GIT;
- executado apenas uma vez: configura o diretório atual para ser rastreado pelo GIT, iniciando um repositório vazio.



Quando criamos o repositório no **GitHub**, o repositório que se encontra lá é denominado repositório remoto. No entanto, em nosso **computador**, também precisamos criar um repositório, que será o local.



git init


git add .

git commit -m "first commit"

git branch -M main

git remote add origin <https://github.com/karllenh2024/Teste.git>

git push -u origin main



```
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> git init
Reinitialized existing Git repository in C:/Users/Karllenh/Documents/3386-git-github-projeto_inicial/.git/
PS C:\Users\Karllenh\Documents\3386-git-github-projeto_inicial> git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
```



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin <https://github.com/karllenh2024/Teste.git>

git push -u origin main


Author identity unkown



```
git config --global user.email "your@example.com"  
git config --global user.name "Your Name"
```

GitHub sabe quem somos, porque estamos logados no navegador

Git não sabe quem somos, pois não possui vínculo direto com o GitHub.



```
git config --global user.email "your@example.com"  
git config --global user.name "Your Name"
```

[you@example.com](#) = email cadastrado no github

[your name](#) = seu nome e sobrenome



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin <https://github.com/karllenh2024/Teste.git>

git push -u origin main



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin

git push -u origin main



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin

git push -u origin main

Mudar de HTTPS para SSH



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin

git push -u origin main

The authenticity of host 'GitHub.com (20.201.28.151)' can't be established



Are you sure you want to continue connecting (yes/no/[fingerprint])?

Em inglês, ele pergunta: quer continuar a conexão? Respondemos com um `yes` (sim). Feito isso, ele manda uma mensagem dizendo que não reconhece esse computador, portanto, não está autorizado para se conectar no GitHub.



Voltando ao GitHub

...



Q Type to search



Karllenhribeiro



Set status



Your profile



Add account



Your repositories



Your projects



Your Copilot



Your organizations



Your enterprises



Your stars



Your sponsors



Your gists



Upgrade



Try Enterprise

Free







Feature preview








Settings

**Karllenhribeiro (Karllenhribeiro)**

Your personal account

[Go to your personal profile](#) Public profile Account Appearance Accessibility Notifications

Access

 Billing and plans ▾ Emails Password and authentication Sessions SSH and GPG keys Organizations Enterprises Moderation ▾

Code, planning, and automation

 Repositories

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys



SSH

ribeirokarlenh@gmail.com

SHA256:8knRdtldNoE0rv7MKm0hVKdooU6T8B85Re1hEq+Kh110

Added on May 5, 2024

Last used within the last week — Read/write

[Delete](#)

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Vigilant mode

[New SSH key](#)[New GPG key](#)



Settings

Q Type [/](#) to search



Karllenhribeiro (Karllenhribeiro)

Your personal account

[Go to your personal profile](#)

Public profile

Account

Appearance

Accessibility

Notifications

Access

Billing and plans

Emails

Password and authentication

Sessions

SSH and GPG keys

Organizations

Enterprises

Moderation

Code, planning, and automation

Repositories

Add new SSH Key

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'


Add SSH key

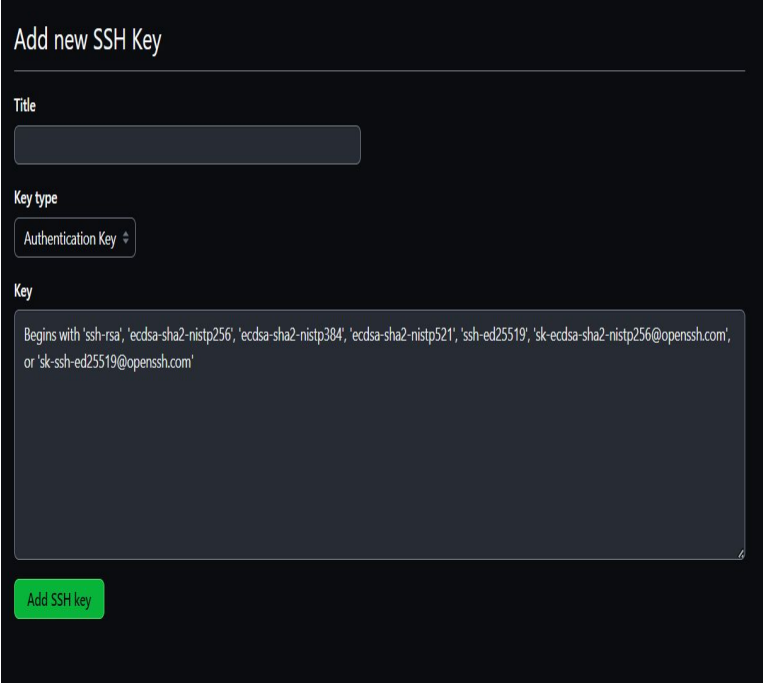


Gerando a chave de autenticação

Cole o texto abaixo, substituindo o email usado no exemplo pelo seu endereço de email GitHub.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

- 
- Pasta "usuário> .ssh". Agora, contém dois arquivo chamamos `id_ed25519`, um com a chave privada, outro com a chave pública.
 - Código da chave começando com `ssh-ed-25519` que precisamos copiar e colar no site do GitHub.



The screenshot shows the 'Add new SSH Key' form on a dark-themed GitHub interface. It includes a 'Title' text input field, a 'Key type' dropdown menu currently set to 'Authentication Key', and a 'Key' text area. The 'Key' field contains a hint: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com''. A green 'Add SSH key' button is at the bottom.

Add new SSH Key

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key



Voltando ao VS code

...



git init

git add .

git commit -m "projeto inicial"

git branch -M main

git remote add origin

git push -u origin main



Git remote

Faz o link entre o repositório local, que está em nosso computador, com o repositório remoto, que está no GitHub;

1 - Adicionar um repositório remoto



```
git remote add apelido url
```

- 'apelido': Este é um nome que você atribui ao repositório remoto.
- 'url': Aqui, você insere a URL do repositório remoto. Esta URL é única para cada repositório remoto e serve como o endereço para acessar e interagir com ele pela internet. Certifique-se de copiar a URL correta do repositório que deseja adicionar.

2 - Listar os repositórios remotos



```
git remote -v
```

- Para listar os repositórios remotos associados ao seu projeto local, você pode utilizar o comando **git remote -v**.
- Isso exibirá uma lista de todos os repositórios remotos vinculados ao seu projeto, juntamente com suas URLs.

3 - Remover um repositório remoto



```
git remote remove origin
```

- Caso você não precise mais de um repositório remoto específico, pode removê-lo com o comando **git remote remove apelido**.
- Substitua 'apelido' pelo nome do repositório remoto que deseja remover.


4 - Alterar a URL de um repositório remoto



```
git remote set-url origin https://github.com/seu-usuari
```

- Às vezes, é necessário atualizar a URL de um repositório remoto, como quando a URL do servidor do GitHub é modificada ou quando você copiou a URL incorreta ao adicionar o repositório remoto.
- Use o comando **git remote set-url apelido nova_url** para realizar essa atualização.
- Substitua 'apelido' pelo nome do repositório remoto e 'nova_url' pela nova URL que você deseja associar a ele.

5 - Alterar o apelido de um repositório remoto



```
git remote rename origin novo-origin
```

- Se você deseja renomear um repositório remoto, use o comando **git remote rename apelido novo_apelido**.
- Substitua 'apelido' pelo nome atual do repositório remoto e 'novo_apelido' pelo novo nome que você deseja atribuir a ele.



Você está desenvolvendo um projeto para um Supermercado Online. Após desenvolver algumas funcionalidades, testar e fazer ajustes, você percebeu que precisa compartilhar o projeto com sua equipe. Para isso, você decide utilizar o GitHub. No momento, o código do projeto existe apenas no seu computador e você deseja conectá-lo a um repositório remoto criado no GitHub.

Qual comando você poderia utilizar para conectar seu projeto local do Git com o repositório remoto no GitHub?

Utilizando o comando `git remote add origin url-do-repositorio-no-github` no terminal.


Utilizando o comando `git push origin main` no terminal.




Você está desenvolvendo um projeto para um Supermercado Online. Após desenvolver algumas funcionalidades, testar e fazer ajustes, você percebeu que precisa compartilhar o projeto com sua equipe. Para isso, você decide utilizar o GitHub. No momento, o código do projeto existe apenas no seu computador e você deseja conectá-lo a um repositório remoto criado no GitHub.

Qual comando você poderia utilizar para conectar seu projeto local do Git com o repositório remoto no GitHub?

O comando `git remote add origin url-do-repositorio-no-github` é o utilizado para conectar seu repositório local ao repositório remoto no Github.



O Git é uma ferramenta excelente para acompanhar a mudança entre versões de um mesmo projeto e, dentre vários benefícios, nos ajuda a observar de perto o desenvolvimento do seu aprendizado. Tudo isso de uma forma organizada através dos commits.



Algo que é essencial no universo da tecnologia é apresentar o seu progresso para a comunidade e montar um portfólio dos seus projetos para demonstrar suas habilidades. Dessa forma, o GitHub é essencial para compartilhar e colaborar em projetos de programação de todo o mundo.

Desafio



1. Crie uma conta no GitHub
2. Crie um repositório para um projeto pessoal.
3. Faça a instalação do Git
4. Crie um repositório localmente para o seu projeto pessoal
5. Adicione alguns arquivos no seu repositório local
6. Faça o commit das alterações
7. Configure a identidade do autor do commit.
8. Crie a branch **Main**
9. Realize a conexão do seu repositório local com o remoto
10. Envie as alterações no repositório local para o remoto
11. Utilize o comando `git status`



Repositório remoto e boas práticas

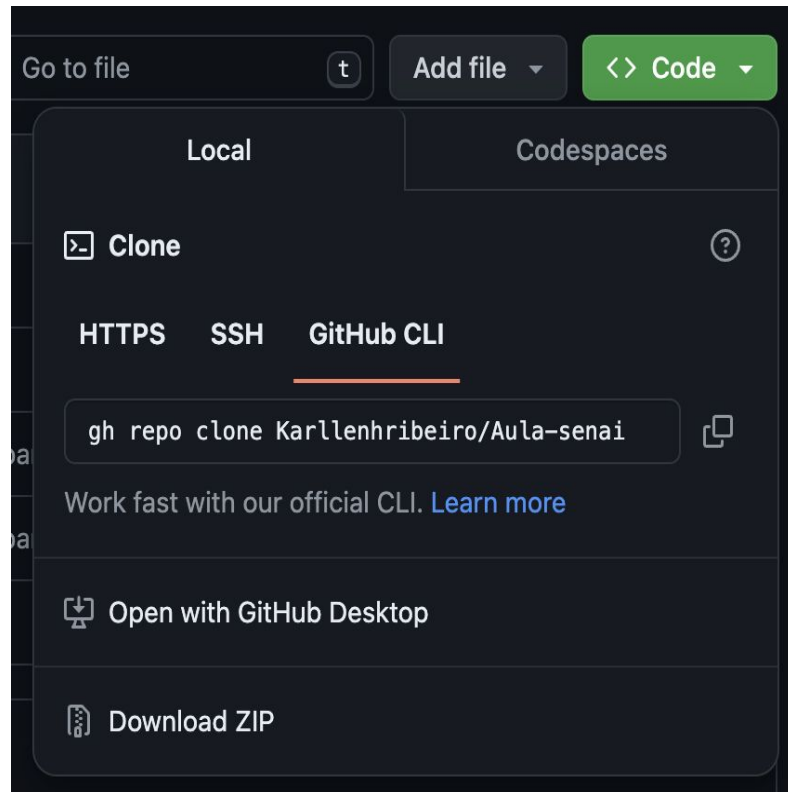
Profa: Karllenh Ribeiro

24/05/24

Compartilhando arquivos

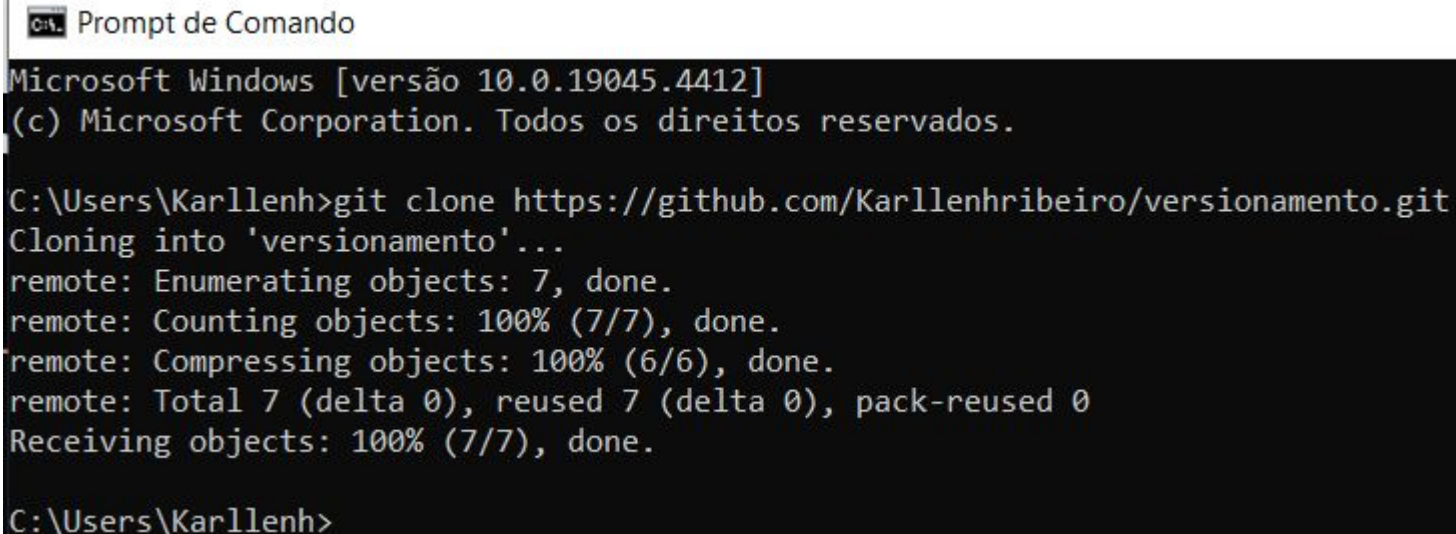
A princípio realizar o Download ZIP poderia ser uma boa opção, pois o GitHub reúne todos os arquivos do repositório, cria o arquivo *zip* e faz o download para o seu computador.

Como nesse caso é realizado apenas o download dos arquivos e não uma cópia do repositório com o commit e todo histórico do projeto. Então, vamos escolher a opção de **clonar**.



Clonando repositório

1. Copiar a URL;
2. Abrir terminal;
3. **Git clone** colar a URL;
4. A pasta já estará em seu computador



```

C:\> Prompt de Comando

Microsoft Windows [versão 10.0.19045.4412]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Karllenh>git clone https://github.com/Karllenhribeiro/versionamento.git
Cloning into 'versionamento'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.

C:\Users\Karllenh>
```





Abrindo o VS code

...

- New Text File Ctrl+N
- New File... Ctrl+Alt+Windows+N
- New Window Ctrl+Shift+N
- Open File... Ctrl+O
- Open Folder... Ctrl+K Ctrl+O
- Open Workspace from File...
- Open Recent >
- Add Folder to Workspace...
- Save Workspace As...
- Duplicate Workspace
- Save Ctrl+S
- Save As... Ctrl+Shift+S
- Save All Ctrl+K S

Code

Walkthroughs

-  **Get Started with VS Code**
Customize your editor, learn the basics, and start coding
-  **Learn the Fundamentals**

Realizando um commit




```
index.html
```

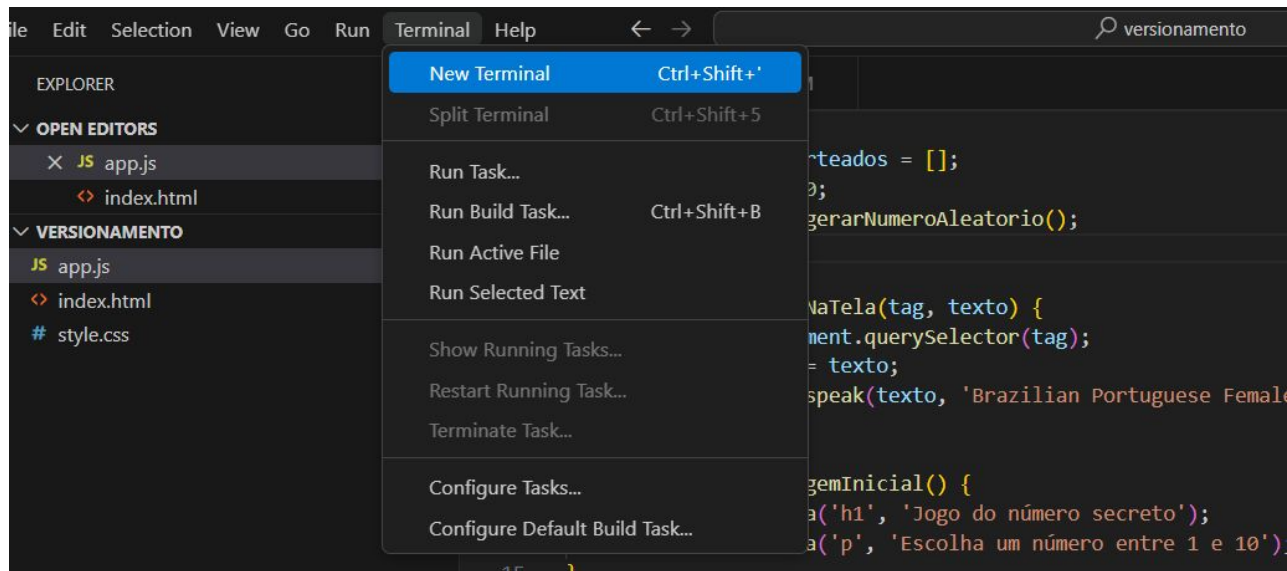
- Começamos abrindo o arquivo `index.html`.
- Feito isso, na linha 23, no texto Escolha um número entre 1 a 10, iremos mudamos para 10 a 300.

```
app.js
```

- Começamos abrindo o arquivo `app.js`.
- Feito isso, na linha 4, na variável tentativas, iremos mudamos para 7.



Realizamos as alterações, porém ainda não estão registradas no repositório local do computador.



Supondo que não lembra onde parou no projeto.



```
git status
```

Então, se esquecermos qual arquivo foi modificado ou qual precisa ser adicionado, basta executar esse comando **git status** que será exibido o status atual do repositório local.



```
PS C:\Users\Karllenh\versionamento> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app.js
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\Karllenh\versionamento> 
```

Repare que o próprio **git status** nos fornece uma dica para usar o comando **git add** seguido do nome do arquivo para fazer essa adição.



```
git add .
```

```
git status
```

Feito isso, provavelmente os arquivos foram adicionados. Para conferir, rodamos novamente o **git status**.

```
PS C:\Users\Karllenh\versionamento> git add .
PS C:\Users\Karllenh\versionamento> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app.js
        modified:   index.html

PS C:\Users\Karllenh\versionamento> █
```



Agora que já adicionamos os arquivos, podemos fazer o commit, que é o comando responsável por registrar uma modificação no código.

Para isso, passamos o comando **git commit**. Lembrando que todo commit sempre terá uma mensagem que informará qual alteração foi feita no projeto.

```
PS C:\Users\Karllenh\versionamento> git commit -m "alterado para 300"
[main 0ee57a6] alterado para 300
 2 files changed, 2 insertions(+), 2 deletions(-)
PS C:\Users\Karllenh\versionamento> █
```

- Quando fazemos um commit é como se estivéssemos registrando uma versão do nosso sistema.
- Se estávamos na versão 1, passamos para a versão 2, depois para a versão 3, 4, 5, e assim por diante.
- Porém, precisamos saber o que é a versão 1, o que mudou na versão 2, 3 e 4.
- É justamente a mensagem que proporciona esse indicativo.

E como identificar quais commits foram realizados no projeto e quem fez?

git log

```
PS C:\Users\Karllenh\versionamento> git log
commit 0ee57a6458197f23a4613e0ef77d356eee7c45f8 (HEAD -> main)
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 19:44:43 2024 -0300


    alterado para 300

commit 025c26d1d28ed3edc90091046c81a88787c86906 (origin/main, origin/HEAD)
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 16:31:50 2024 -0300

    aula versionamento

commit b99f3cefa6059fbac3a9ee844c302ace45d03f1b
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 16:23:34 2024 -0300

    aula versionamento
```



Observe que o `git log` mostra o log, ou seja, o histórico de commits realizados no projeto.

Assim, podemos identificar quem foi que fez cada registro, a mensagem, o significado de cada registro, a data e toda essa sequência de alterações no repositório.



Como enviar essas alterações para o repositório?

**Precisaremos vincular o nosso repositório local
como o repositório remoto?**




git remote

O comando **clone** já realiza automaticamente essa conexão entre o repositório remoto e o repositório local.

verificar isso rodando o comando **git remote**.

Ao fazer isso, será listado os repositórios remotos no seu repositório local.

```
aula versionamento
PS C:\Users\Karllenh\versionamento> git remote
origin
PS C:\Users\Karllenh\versionamento> 
```



```
git push origin main
```

Como você já temos o repositório remoto adicionado, agora podemos executar o comando **git push** para tentar enviar esse commit para o repositório no GitHub.

Passamos **git push** e em sequência precisamos informar para onde enviaremos o commit.

Escrevemos **origin** que é o apelido do repositório remoto, seguido de **main** que é a branch

```
PS C:\Users\Karllenh\versionamento> git push origin main  
info: please complete authentication in your browser...
```

- Embora você tenha conseguido baixar o repositório, criado como público, esse é um material público apenas para leitura.
- Isso significa que as pessoas podem acessá-lo, baixar o código, fazer mudanças e commits, porém apenas localmente. Quando alguém tentar fazer um **push**, o próprio GitHub irá barrar.
- Se quisermos que outras pessoas colaborem nesse projeto, é preciso adicioná-las manualmente no projeto.

```
git push origin main
```



versionamento

Public



Pin



Unwatch 1



main



1 Branch



0 Tags



Go to file



Add file



Code



Karllenhribeiro alterado para 300

0ee57a6 · 52 minutes ago



3 Commits



app.js

alterado para 300

52 minutes ago



index.html

alterado para 300

52 minutes ago



style.css

aula versionamento

4 hours ago

main

Karllenhribeiro committed 53 minutes ago

1 parent 025c26d commit 0ee57a6

Showing 2 changed files with 2 additions and 2 deletions.

Whitespace Ignore whitespace Split Unified

Filter changed files

app.js

index.html

2 app.js

```
... -1,7 +1,7 @@
1 1 let listaDeNumerosSorteados = [];
2 2 let numeroLimite = 10;
3 3 let numeroSecreto = gerarNumeroAleatorio();
4 - let tentativas = 0;
+ 4 + let tentativas = 0;
5 5
6 6 function exibirTextoNaTela(tag, texto) {
7 7 let campo = document.querySelector(tag);
+ ...
+ ↓
```

2 index.html

```
+ ...
+ @@ -20,7 +20,7 @@
20 20 <div class="container_informacoes">
21 21 <div class="container_texto">
22 22 <h1>Adivinhe o <span class="container_texto-azul">numero secreto</span></h1>
23 - <p class="texto_paragrafo">Escolha um número entre 1 a 10</p>
+ 23 + <p class="texto_paragrafo">Escolha um número entre 1 a 200</p>
24 24 </div>
25 25 <input type="number" min="1" max="10" class="container_input">
26 26 <div class="chute container_botoes">
+ ...
+ ↓
```

0 comments on commit 0ee57a6

Lock conversation

Sincronizando as atualizações do repositório remoto com o repositório local.

O repositório local não atualizou automaticamente o commit feito

Para isso, existe o comando **git pull** especificamente para isso.

Ele funciona como o oposto do **push**, já que puxa os commits do remoto para o local.

```
git pull origin main
```

```
PS C:\Users\Karllenh\Documents\Aula de versionamento> git pull origin main
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 368 bytes | 26.00 KiB/s, done.
From github.com:Karllenhribeiro/versionamento
 * branch                main          -> FETCH_HEAD
    025c26d..0ee57a6      main          -> origin/main
Updating 025c26d..0ee57a6
Fast-forward
 app.js      | 2 +-
 index.html  | 2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)
```


git pull

Tem o objetivo de baixar os novos commits que outras pessoas colaboradoras do seu repositório enviaram para o GitHub.

Com isso, temos um fluxo de trabalho.


```
PS C:\Users\Karllenh\Documents\Aula de versionamento> git log
commit 0ee57a6458197f23a4613e0ef77d356eee7c45f8 (HEAD -> main, origin/main)
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 19:44:43 2024 -0300

    alterado para 300

commit 025c26d1d28ed3edc90091046c81a88787c86906
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 16:31:50 2024 -0300

    aula versionamento

commit b99f3cefa6059fbac3a9ee844c302ace45d03f1b
Author: karllenh ribeiro <ribeirokarllenh@gmail.com>
Date: Sun May 19 16:23:34 2024 -0300
```

- 
- Quando estivermos trabalhando em um projeto e precisarmos realizar mudanças, usaremos o **git status** para verificar os arquivos modificados.
 - Adicionaremos essas mudanças com o comando **git add**, depois, realizaremos um commit com o **git commit**.
 - Subiremos essas mudanças para o repositório com o **git push** e eventualmente, conforme outras pessoas forem colaborando com o projeto, traremos essas mudanças novamente para o computador com o **git pull**.

Commit




As mensagens dos commits devem ser **simples e objetivas**:

Mantenha a mensagem curta e concisa: A primeira linha da mensagem deve conter, no máximo, 72 caracteres.

Uso de verbo no infinitivo: É comum que a mensagem do commit inicie com um verbo no infinitivo que descreva a alteração feita, como “adicionar”, “corrigir” ou “atualizar”. Em sequência, são adicionados detalhes concisos da mudança. Por exemplo: “Atualizar texto do título da página”.

Evite detalhes técnicos: Não inclua detalhes técnicos complexos na mensagem de commit.



É importante ter em mente que a mensagem do commit é uma forma de documentação do histórico das mudanças que ocorreram ao longo do código. A pessoa que ler essa mensagem pode não ter conhecimento do contexto original. Assim, garanta que sua mensagem de commit tenham clareza e sejam suficientemente descritivas.

E quando realizar um commit?



- Um commit deve ser realizado sempre que você **finalizar uma tarefa** específica ou **resolver algum bug**. Isso mantém o histórico de commits claro e rastreável, de modo que seja possível entender o que foi feito em cada commit.
- Assim, é importante realizar commits frequentemente.
- Porém, evite realizar commits muito pequenos ou muito grandes, pois isso pode tornar difícil o seu entendimento.



O commit é uma ação muito importante do Git para nos ajudar no controle da versão do nosso projeto. Um commit pode ser considerado como um marco ao longo do cronograma de um projeto.

Sabendo disso, quando realizamos um commit, estamos:

Enviando as alterações do projeto para o repositório remoto.

Adicionando as alterações mais recentes do projeto.

Desafio



1. Crie um novo repositório remoto no Github e insira um arquivo.
2. Faça um clone do seu repositório remoto para o local.
3. Faça uma nova modificação no repositório remoto.
4. Atualize seu repositório local a partir do Remoto.
5. Utilize o comando `git remote -v` no terminal.
6. Confira as mudanças nos arquivos.



Obrigada!





<https://medium.com/biblioteca-dos-devs/como-versionar-utilizando-o-git-1f5d8fe2afcd>