

Git e GitHub

SENAC



GIT é a ferramenta de controle de versão;

Usamos para trabalhar com versionamento de código;

- Trabalhamos com CI/CD, ou Integração e Entrega/Implantação Contínuas, é uma prática de desenvolvimento de software que visa automatizar o processo de construção, teste e implantação de aplicações;
 - CI → Integração de alterações;
 - CD → Entrega e implementação das alterações.
- Branch é o conceito de duplicar;
- pull requests é aplicar no original a alteração da duplicata;



Git e GitHub

SENAC



Comandos Git:

Configuração inicial :

- **git config --global user.name "Seu Nome"**
 - Define seu nome
- **git config --global user.email "seu@email.com"**
 - Define seu email
- **git config --list**
 - Mostra as configs atuais

Início do projeto

- **git init**
 - Cria um repositório Git na pasta atual
- **git clone URL**
 - Clona um repositório existente do GitHub



Git e GitHub

SENAC



Comandos Git:

Trabalhando com arquivos

- **git status**
 - Mostra arquivos modificados e não rastreados
- **git add arquivo.py**
 - Adiciona arquivo para o commit
- **git add .**
 - Adiciona todos os arquivos modificados
- **git commit -m "mensagem"**
 - Salva as alterações localmente



Git e GitHub

SENAC



Comandos Git:

Envio e atualizaço com repositrio remoto

- **git remote add origin URL**
 - Liga o repositório local ao GitHub
- **git push -u origin main**
 - Envia o projeto para o GitHub (primeira vez)
- **git push**
 - Envia os commits locais para o GitHub
- **git pull**
 - Baixa e integra mudanças do GitHub
- **git fetch**
 - Só baixa (sem integrar ainda)



Git e GitHub

Comandos Git:

Branches

- **git branch**
 - Lista as branches locais
- **git branch nova-branch**
 - Cria uma nova branch
- **git checkout nome-da-branch**
 - Troca de branch
- **git switch nome**
 - Troca de branch (forma moderna)
- **git merge nome-da-branch**
 - Mescla uma branch com a atual
- **git branch -d nome**
 - Apaga uma branch local
- **git push origin --delete nome**
 - Apaga uma branch remota

SENAC



Git e GitHub

Comandos Git:

Histórico e versões

- **git log**
 - Mostra o histórico de commits
- **git log --oneline**
 - Histórico resumido
- **git diff**
 - Mostra diferenças entre versões

Desfazendo

- **git restore arquivo**
 - Desfaz mudanças não adicionadas
- **git reset arquivo**
 - Remove do staging (git add)
- **git reset --hard HEAD**
 - Volta tudo pro último commit (perigoso!)

SENAC



Git e GitHub

Outros teís

- **git stash**
 - Guarda alterações temporariamente
- **git stash pop**
 - Recupera alterações guardadas
- **git show**
 - Mostra detalhes do último commit

SENAC



Regras do GitHub

Para ter uma segurança ideal, marque estas opções:

SENAC

Protect matching branches

☒ Require a pull request before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ Require approvals

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1 ▼

☐ Dismiss stale pull request approvals when new commits are pushed

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☒ Require review from Code Owners

Require an approved review in pull requests including files with a designated code owner.

☒ Require approval of the most recent reviewable push

Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☒ Require signed commits

Commits pushed to matching branches must have verified signatures.

☐ Require linear history

Prevent merge commits from being pushed to matching branches.

☐ Require deployments to succeed before merging

Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.

☒ Lock branch

Branch is read-only. Users cannot push to the branch.

☒ Do not allow bypassing the above settings

The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

Rules applied to everyone including administrators

☐ Allow force pushes

Permit force pushes for all users with push access.

☐ Allow deletions

Allow users with push access to delete matching branches.

Save changes

Git e GitHub

SENAC



Explicando regras:

- **Lock branch**

- Torna a branch main somente leitura.
- Ninguém pode fazer push direto nela — nem o dono do repositório, nem administradores, nem ninguém.

- **Require signed commits**

- Exige que todos os commits na branch main tenham uma assinatura GPG verificada.
- Garante a autenticidade e integridade dos commits; evita autores falsos ou alterações anônimas.

- **Do not allow bypassing the above settings**

- Impede que administradores ou pessoas com permissões especiais ignorem as regras de proteção.
- Nem com privilégio total alguém pode burlar as regras (ex: push direto, merge sem aprovação, etc).



Git e GitHub

Explicando regras:

- **Rules applied to everyone including administrators**
 - Aplica todas as regras de proteção a todos os usuários, inclusive administradores.
 - Garante que ninguém tem tratamento especial, todos seguem o mesmo fluxo de proteção.
- **Allow force pushes (desmarcada)**
 - Permitiria o uso de git push --force.
 - Ninguém pode sobrescrever histórico da branch com push forçado.
- **Allow deletions (desmarcada)**
 - Função (se estivesse marcada): Permitiria que usuários com acesso pudessem deletar a branch.
 - Efeito (como está desmarcada): Ninguém pode deletar a branch main.



Fluxo básico Git

SENAC

Para conectarmos o git local ao github vamos seguir os seguintes passos:

1: Usar o comando **Git clone** "url", onde vamos colocar o link do git hub.

2: Navegar para dentro da pasta que baixou os arquivos do repositório, para navegar usamos o comando: **cd** "diretório"

3: Iniciar o git com o comando **git init**

4: **git config --global user.name "nome"**
git config --global user.email "email@outlook.com"
(cria as credenciais)

5: Usar o comando **git remote add origin** "url" que adiciona ao link do repositório.

6: Digitar o comando **git add .** (adiciona os arquivos do diretório para enviar);

7: **git commit -m "mensagem"** para colocar um comentário no envio.

8: **git commit -m "mensagem"** para colocar um comentário no envio.

9: **git branch** (para ver a branch)

10: **git push -u origin** "nome_da_branch"

EXTRA: Caso o repositório já exista aí vamos baixar ele usando o comando: **git clone "url"** que aí ele baixa todos os arquivos.

Existem vários métodos e jeitos de fazer, este modo apresentado é o básico. É de **EXTREMA IMPORTÂNCIA** pesquisar novos métodos.

