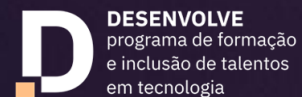


KORU



O QUE VAMOS APRENDER HOJE?

- **Por que Controle de Versão?**
- **Git** vs **GitHub**
- **Fluxo de Trabalho Git:** (workspace, staging, repository).
- **Comandos Essenciais** (no terminal):
 - git init: Começando do zero.
 - git status: Onde estamos no fluxo?
 - git add: Preparando as mudanças.
 - git commit: Salvando uma "foto" do projeto.
 - git log: Vendo a história.
- **Git no VSCode:** Interface visual.
- **GitHub na Prática:** Criando sua conta e 1º repositório.
- **SSH:** Conectando seu PC ao GitHub
- **.gitignore:** O que o Git deve ignorar?















**POR QUE
CONTROLE DE
VERSÃO?**

Name



Date Modified

 meu_projeto_final_2024.docx	Today at 11:25
 meu_projeto_final_analise.docx	Today at 11:25
 meu_projeto_final_apresentacao.docx	Today at 11:25
 meu_projeto_final_ESSE_SIM_FINAL.docx	Today at 11:23
 meu_projeto_final_final_agora_vai.docx	Today at 11:23
 meu_projeto_final_final.docx	Today at 11:23
 meu_projeto_final_joao_alterou_v2_com_css.docx	Today at 11:23
 meu_projeto_final_joao_alterou.docx	Today at 11:23
 meu_projeto_final_plano.docx	Today at 11:25
 meu_projeto_final_relatorio.docx	Today at 11:25
 meu_projeto_final_v2.docx	Today at 11:23
 meu_projeto_final.docx	Today at 11:23

POR QUE CONTROLE DE VERSÃO?

- **Histórico** Completo
- Podemos **voltar no tempo**
- Trabalho em equipe
- **Backup**
- Experimentação Segura


A group of diverse, smiling people, overlaid with a purple gradient. The text "GIT E GITHUB" is centered over the image in a bold, white, sans-serif font.

GIT E GITHUB

O QUE É GITHUB?

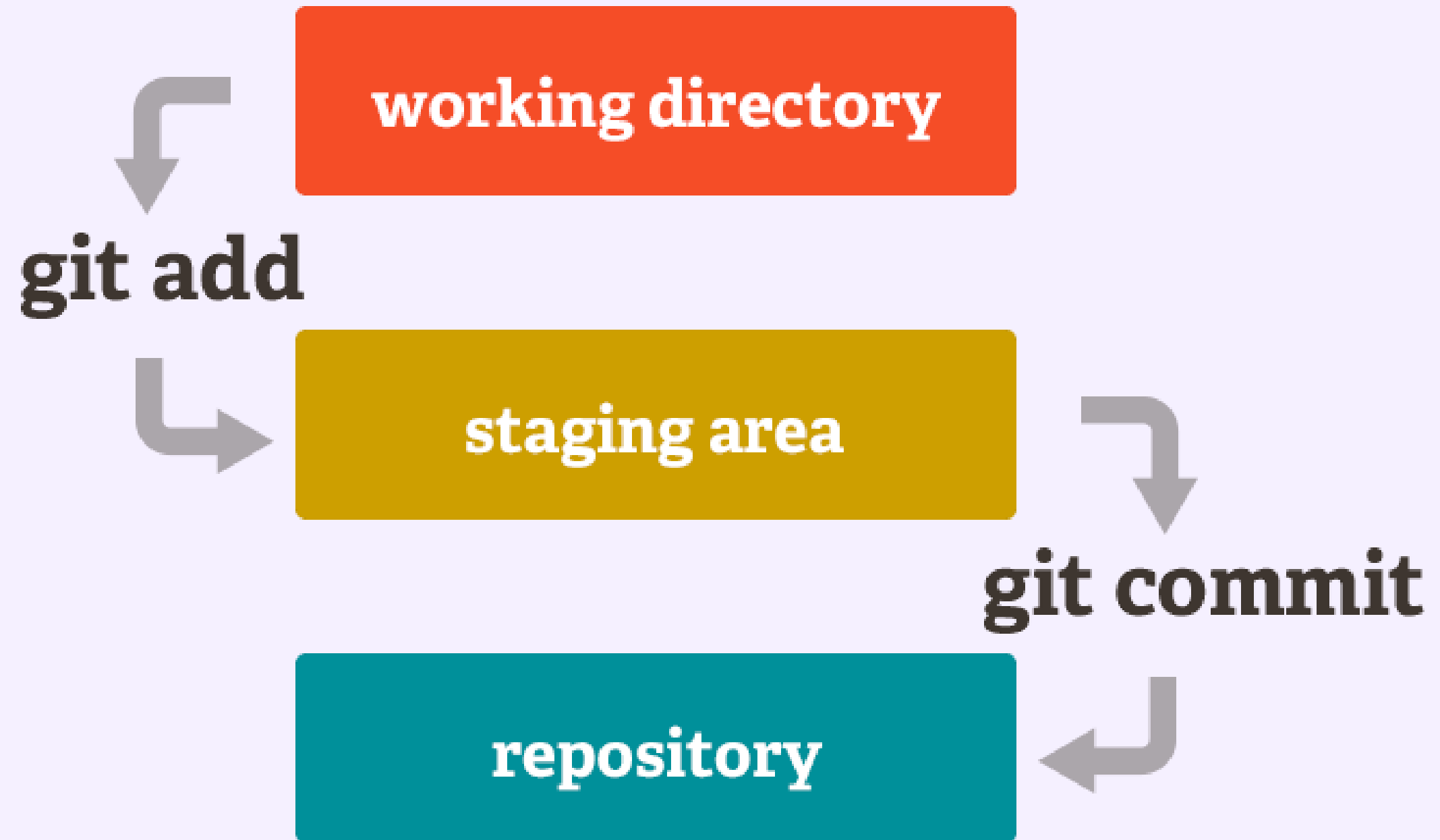
- Git é um **software** que você instala na sua máquina.
- Ele é o **mecanismo que rastreia suas alterações.**
- Seus repositórios Git ficam no seu computador.

O QUE É GIT

- **GitHub** é uma das (várias) plataformas de hospedagem de repositórios online.
- **Repositório** nada mais é do que seu código (sua pasta de arquivos de código) de alguma coisa.
- Permite colaboração, já que é na **nuvem**
-  Como se fosse um Google Drive para código



FLUXO BÁSICO



FLUXO BÁSICO

Workspace

- Pasta do projeto no computador. Os arquivos existem mas nada está sendo rastreado

Staging

- Aqui é onde você adiciona especificamente quais **alterações** quer incluir no próximo "salvamento" (commit)
- É como se fosse um "carrinho de compras" 🛒

Repositório

- É onde o Git salva o histórico oficial das versões (commits). Quando você commita, o Git pega o que está na staging area e salva como se fosse uma "fotografia" permanente

A diverse group of smiling people, including a man with a beard and glasses, a woman with glasses, and a man with glasses, are shown in a close-up shot. The image is overlaid with a semi-transparent purple filter. The text "COMANDOS ESSENCIAIS" is written in large, bold, white capital letters across the center-right of the image.

COMANDOS ESSENCIAIS

COMANDOS ESSENCIAIS

`git init`

- Use em uma pasta que ainda não é um repositório Git.
- Transforma a pasta atual em um novo **repositório Git local**.
- Cria uma pasta oculta chamada .git dentro da sua pasta. É lá que o Git armazena todo o histórico e informações do seu projeto.

COMANDOS ESSENCIAIS

`git status`

- Seu melhor amigo! Use sempre!
- Mostra o estado atual do seu repositório:
 - o Quais arquivos foram modificados (Modified).
 - o Quais arquivos são novos e não estão sendo rastreados (Untracked).
 - o Quais alterações estão na Staging Area (Changes to be committed).
 - o Qual branch você está (veremos branches depois).

COMANDOS ESSENCIAIS

`git add`

- Move as alterações (de arquivos modificados ou novos) do Workspace para a Staging Area.
- **Você escolhe** quais mudanças quer incluir no próximo commit.

COMANDOS ESSENCIAIS

`git commit`

- Pega tudo que está na Staging Area e salva como uma nova versão permanente no Local Repository.
- Cada commit é um **ponto na história** do seu projeto.
- Todo commit **precisa de uma mensagem** descrevendo as mudanças!

COMANDOS ESSENCIAIS

``git log``

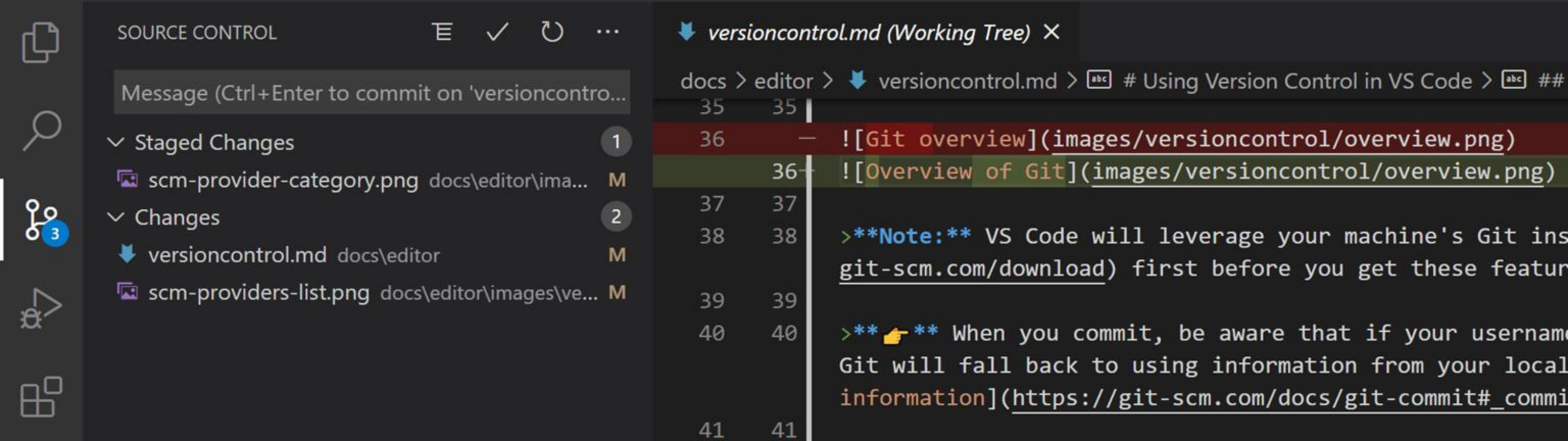
- Mostra o histórico dos seus commits no repositório local.
- Exibe cada commit com seu identificador único (o "hash"), autor, data e a mensagem.
- versão mais simples: ``git log --oneline``.



1. PRÁTICA GIT LOCAL

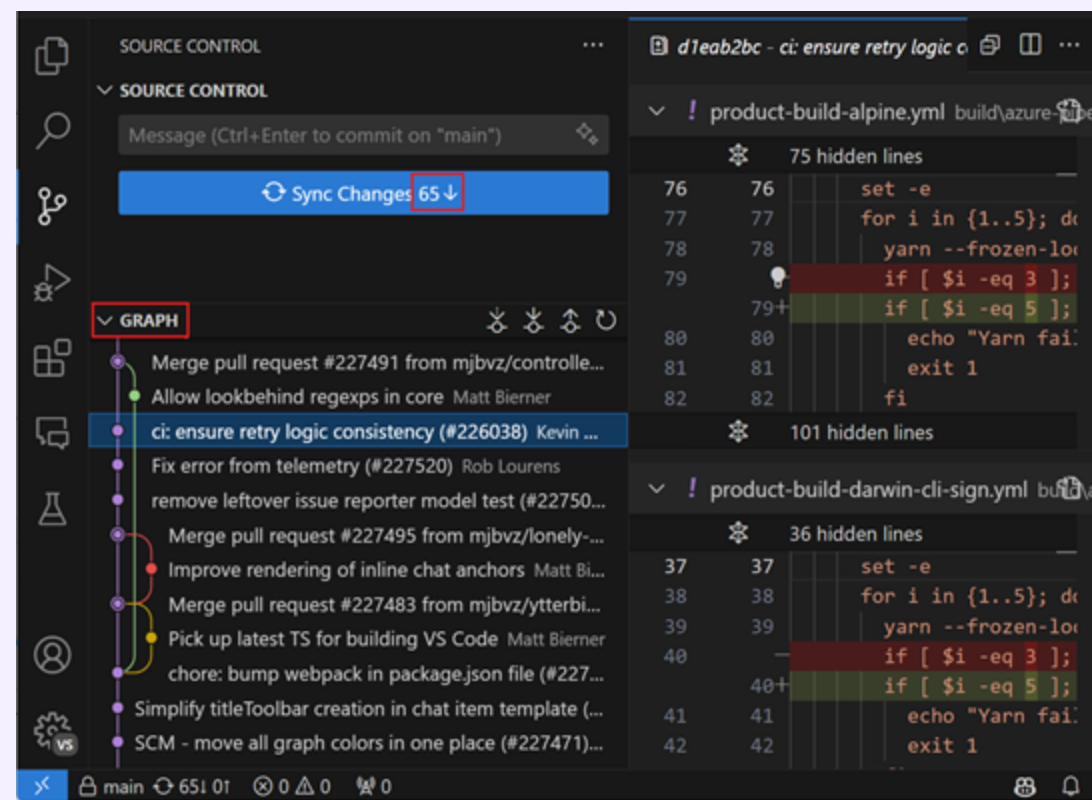
A group of diverse, smiling people, overlaid with a purple gradient and the text "GIT NO VSCODE". The image features a group of seven people of various ethnicities and ages, all smiling and looking towards the camera. The image is overlaid with a semi-transparent purple gradient. The text "GIT NO VSCODE" is written in a bold, white, sans-serif font, centered horizontally across the middle of the image.

GIT NO VSCODE



Você pode controlar tudo no editor visual do VSCode se quiser.

Possui também uma visualização gráfica dos commits





2. PRÁTICA GIT NO VSCODE

A group of seven diverse people are shown from the chest up, smiling and looking towards the camera. The image has a strong purple color overlay. The people are of various ethnicities and ages, creating a sense of community and inclusivity. One person in the foreground is waving their hand.

GITHUB

CRIANDO UMA CONTA

- Abra o navegador e vá para: **github.com**
- Clique em "**Sign up**" (Cadastrar).
- Siga os passos para criar sua conta gratuita.
Escolha um bom username! Será a sua identidade no GitHub - e dificilmente irá mudar ao longo da sua carreira 🧸

CRIANDO REPOSITÓRIO REMOTO

- **Logado no GitHub**, clique no sinal + no canto superior direito.
- Selecione "New repository".
- **Owner**: Será seu username.
- **Repository name**: Dê o mesmo nome da sua pasta local (ex: meu_projeto_web). Isso ajuda a organizar.
- **Description**: (Opcional) Adicione uma breve descrição.
- **Public vs Private**: Escolha Public por enquanto (seu código fica visível para todos - ótimo para portfólio!). Private é pago (ou gratuito com algumas limitações para indivíduos/equipes pequenas).
- **Initialize this repository with**: não selecione nenhuma dessas opções (README, .gitignore, Choose a license). Vamos conectar um repositório existente (o que acabamos de criar localmente).
- Clique em "Create repository"

CHAVE SSH COMO AUTENTICAÇÃO

Funcionamento

- Par de chaves: pública e privada.
- Chave privada fica no seu computador, enquanto a pública você entrega a quem quer acessar os serviços (exemplo: GitHub).
- **Ao fazer o login,** o servidor do GitHub usa a chave pública para "desafiar" o computador
- O computador usa a chave privada para responder ao desafio provando que é você.




3. PRÁTICA CRIANDO UMA CHAVE SSH PARA AUTENTICAÇÃO NO GITHUB

PRÁTICA

- Crie par de chaves SSH.
- Copie a chave pública para o GitHub
- Adicione a chave pública no GitHub
- Teste no seu terminal o login através do comando ``ssh -T git@github.com``

CONECTANDO REPOSITÓRIO LOCAL E REMOTO

- Volte para a **página do repositório** que você criou no GitHub (Ex: *github.com/SEU_USERNAME/meu_projeto_web*).
- Copie a **URL SSH** que aparece (será algo como *git@github.com:SEU_USERNAME/meu_projeto_web.git*).
- Adicione o repositório remoto (terminal):
 - ``git remote add origin git@github.com:SEU_USERNAME/meu_projeto_web.git``
- **origin** é apenas um nome padrão que damos para o repositório remoto principal. Poderia ser github ou outro nome, mas origin é universal.
- Verifique se o remoto foi adicionado: ``git remote -v`` (Deve mostrar as URLs do origin).



COMANDOS ESSENCIAIS (CONTINUAÇÃO)

COMANDOS ESSENCIAIS

`git push`

- Envia seus commits locais (que estão no seu Local Repository) para o repositório remoto (no GitHub).
- É assim que suas alterações aparecem no GitHub para backup e colaboração.

Primeiro Push

- ``git push -u origin main`` (ou `git push -u origin master`)
- **origin**: O nome do remoto (que definimos antes).
- **main**: O nome do branch padrão que você está enviando.
- **-u** (ou `--set-upstream`): Diz ao Git para lembrar que o branch local main está conectado ao branch remoto main em origin. Você só precisa fazer isso no primeiro push desse branch.
- Após o primeiro push, todos os outros podem ser feitos só com ``git push``



.GITIGNORE

.GITIGNORE

Quais arquivos ignorar?

- Arquivos gerados automaticamente (ex: pasta *node_modules* de dependências, arquivos de *build*).
- Arquivos de *configuração local* que contêm chaves secretas ou senhas (ex: *.env*).
- Arquivos *temporários* ou específicos do seu sistema operacional (ex: *.DS_Store* no macOS).
- *Arquivos grandes* que não precisam de versionamento (ex: datasets brutos).

O que é o **.gitignore**?


- O arquivo *.gitignore* é um arquivo de texto simples onde você lista os arquivos/pastas que o Git DEVE IGNORAR.
- Você cria esse arquivo na raiz do seu repositório.
- Importante: O arquivo *.gitignore* ele mesmo deve ser adicionado e commitado no Git!



DICAS RÁPIDAS

DICAS RÁPIDAS


- **Commits pequenos e focados:** Salve alterações frequentemente, cada commit deve resolver uma coisa específica.
- **Mensagens Claras:** Descreva o que foi feito.
- Use **git status** sempre! Saiba o estado do seu trabalho.
- Configure o **.gitignore** cedo: Evite commitar coisas desnecessárias.
- **SSH** é seu amigo: Autenticação mais fluida.
- GitHub é seu Backup: Faça **git push** regularmente!











PRÓXIMOS PASSOS

DICAS RÁPIDAS

- Clonar repositórios (**git clone**).
- Receber alterações de outros (**git pull**).
- **Branches** (trabalhar em paralelo em novas funcionalidades).
- Colaboração (**Pull Requests**).



Classbook da Turma

 600 × 400 Aluno 1	 600 × 400 Aluno 2	 600 × 400 Aluno 2	 600 × 400 Aluno 1
 600 × 400 Aluno 2	 600 × 400 Aluno 2	 600 × 400 Aluno 1	 600 × 400 Aluno 2

4. VAMOS BRINCAR DE COLABORAÇÃO COM GITHUB NA AULA?

4. VAMOS BRINCAR DE COLABORAÇÃO COM GITHUB NA AULA?

Vamos **criar um arquivo simples** no repositório do instrutor, onde **cada um de vocês adicionará seu nome como "contribuinte"**.

Para fazer isso, vocês precisam:

- Criar uma cópia do repositório do professor no GitHub (um Fork).
- Clonar sua cópia (Fork) para o seu computador.
- Fazer uma pequena alteração (adicionar seu nome) no seu repositório local.
- Salvar essa alteração com Git (add + commit).
- Enviar a alteração para sua cópia (Fork) no GitHub (push).
- No GitHub, abrir um Pull Request da sua cópia para o repositório original do professor.



Vamos
avaliar o
encontro?