A.
Git

AULA EXTRA

Prof. Esp. William C. Augustonelli (Billy)
william.augustonelli@fho.edu.br

07/2025

Objetivo

- Entender o que é Git e por que usar
- Instalação e Configuração do Git
- Comandos para para subir e baixar
- Integração com o VS Code

Na aula de hoje...

- Conceitos Fundamentais
- Instalação Passo a Passo
- Comandos Explicados
- Integração com VS Code
- Workflows Práticos
- Soluções para problemas comuns

Introdução ao Git

O que é Git?

- Git é um **sistema de controle de versão distribuído** criado por Linux Torvalds em 2005

Características

- ❖Rastrear mudanças no código
- ❖Trabalhar em equipe de forma organizada
- ❖Voltar para versões anteriores do projeto
- ❖Criar ramificações (branches) para testar funcionalidades

O que é GitHub/ GitLab/ Bitbucket?

- São **plataformas online** que hospedam repositórios Git, facilitando colaboração e backup

Instalação do Git

Windows

- https://git-scm.com/download/win

Linux (Ubuntu/ Debian)

- sudo apt update
- sudo apt install git

Verificar instalação

- git --version

Configuração Inicial (obrigatório)

Configure o seu nome (que irá aparecer no s commits)

- git config --global user.name “Seu Nome”

Configure seu e-mail

- git config --global user.email seuemail@exemplo.com

Verificar configurações

- git config --list

Explicações

- --global → aplica asa configurações para todos os projetos
- Sem --global → aplica apena ao projeto atual

Conceitos Fundamentais

Repositório

- Local onde o Git armazena todo histórico do projeto

Estados dos Arquivos

- Working Directory: Arquivos que você está editando
- Staging Area: Arquivos preparados para commit
- Repository: Histórico permanente de commits

Commit

- Um *snapshot* (foto) do seu projeto em um momento específico

Branch

- Uma linha independente de desenvolvimento

```
Working Directory → Staging Area → Repository
   (modificado)      (preparado)     (commitado)
```

Comandos Básicos

Criar um Novo Repositório

- **mkdir meu-projeto**
- cd meu-projeto

Inicializar o Git

- git init

Explicação

- git init → cria uma pasta oculta .git que armazena todo o histório

Status do Repositório

- git status → mostra arquivos modificados, não rastreados e preparados para commit

Comandos Básicos

Adicionar arquivos

- Para adicionar arquivo específico: git add arquivo.txt
- Para adicionar todos os arquivos modificados ou novos: git add .
- Para adicionar todos os arquivos de um tipo: git add *.js
- Este comando move arquivos do Working Directory para Staging Area

Fazer Commit

- Com mensagem: git commit –m “Adiciona funcionalidade de log”
- Com mensagem mais longa (abre o editor): git commit

Ver histórico

- Histórico completo: git log
- Histórico resumido: git log --online --graph --all
- Histórico dos X últimos: git log –N (onde N é número de históricos desejados)

Comandos Básicos

Ver diferenças

- Diferenças não preparadas (working directory): `git diff`
- Diferenças preparadas (staging área): `git diff --staged`
- Diferenças de um arquivo específico: `git diff arquivo.txt`

Trabalhando com VS Code

Interface Git no VS Code

Barra Lateral Esquerda – Ícone de Controle de Código
- U (Untracked): arquivos novos
- M (Modified): arquivos modificados
- D (Deleted): arquivos deletados
- A (Added): arquivos no staging

Fazer Commit no VS Code
- Abra a aba de Source Controle (Ctrl + Shift + G)
- Veja os arquivos modificados
- Cliente no + ao lado dos arquivos para adicionar ao staging
- Digite mensagem de commit no campo superior
- Clique em ✔ Confirmar ou pressione Ctrl+Enter

Terminal Integrado
- Todos os comandos Git funcionam aqui

Extensões Úteis

- GitLens: visualização avançada de histórico
- Git Graph: Visualiza branches graficamente
- Git History: Explorar histórico de arquivos

Workflows Práticos

Trabalhando com Branches

- Criar uma nova Branch: git Branch feature-nova
- Mudar para a Branch: git checkout feature-nova
- Criar e mudar em um comando: git checkout –b feature-nova
- Listar Branches: git branch
- Ver Branch atual: git branch --show-current
- Branches permitem trabalhar em funcionalidades sem afetar o código principal

Merge (Mesclar)

- Voltar para Branch Principal: git checkout main
- Mesclar Branch: git merge feature-nova

Workflows Práticos

Trabalhando com GitHub

- Conectar repositório local ao GitHub: git remote add origin https://github.com/usuario/repositório.git
- Enviar código para GitHub (primeira vez): git push –u origin main
- Enviar código (próximas vezes): git push
- Baixar código do GitHub: git pull
- Clonar repositório existente: git clone https://github.com/usuario/repositório.git

Explicações

- origin: nome padrão do repositório remoto
- push: envia commits locais para remoto
- pull: baixa commits remotos para local
- clone: copia repositório completo

Workflows Práticos

Desfazer Mudanças

- Descartar mudanças em arquivos (cuidado): `git checkout -- arquivo.txt`
- Remover arquivo doo staging: `git reset arquivo.txt`
- Voltar último commit (mantém as mudanças): `git reset --soft HEAD~1`
- Voltar último commit (descarta mudanças): `git reset --hard HEAD~1`

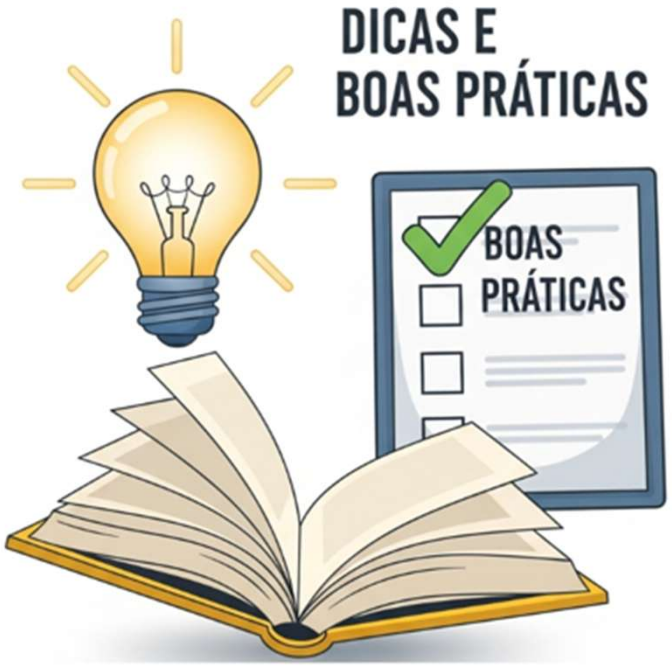
Dicas importantes

.gitignore → arquivo que lista o que o Git deve ignorar

```
#Exemplo de .gitignore
        node_modules/
        *.log
        .env
        .DS_Store
```

Boas práticas

- Faça commits pequenos e frequentes
- Escreva mensagens descritivas
- Sempre faça git pull antes de git push
- Use branches para novas funcionalidades
- Não comitê arquivos sensíveis (senhas, tokens)



Troubleshooting Comum

Permission denied (publickey)

- Configure SSH ou use HTTPS: `git remote set-url origin https://github.com/usuario/repo.git`

Conflitos de Merge

- Git marca arquivos com conflito, edite manualmente os arquivos:

```
git add arquivo-resolvido.txt
git commit –m “Resolver conflito”
```

Esqueceu de adicionar arquivo

- Adicione e faça commit:

```
git add arquivo-esquecido.txt git commit --amend –no-edit
```

Perguntas?! Dúvidas?!?